STAT238
Bayesian Statistics

Reece D. Huff
rdhuff@berkeley.edu

Spring 2025

Homeworks, Labs, Projects, Quiz Practice, & Lectures
Alexander Strang
Department of Statistics
University of California, Berkeley



Contents

Homeworks 1

Homework 1 – The Beta, Binomial Model 2
- Notation . . . 2
  - Beta-Binomial Model . . . 2
- Problems . . . 3
  - 1. Chapter 1, Problem 3 [BDA3] . . . 3
  - 2. Chapter 1, Problem 7 [BDA3] . . . 6
  - 3. Chapter 2, Problem 1 [BDA3] . . . 7
  - 4. Chapter 2, Problem 9 [BDA3] . . . 8
  - 5. Moments of the Beta Binomial . . . 10
  - 6. Regularization – Bias in Posterior Point Inference . . . 12
  - 7. Principles for Selecting an Estimator . . . 15
  - 8. Regularization – Reducing Variance . . . 17
  - 9. Information . . . 19

Homework 2 – Asymptotics, Conjugacy, and Decision Theory 22
- Problems . . . 22
  - 1. Asymptotic Abnormality . . . 22
  - Python Code for Problem 1 . . . 29
  - 2. Does Preference Imply Belief? . . . 34
  - 3. Agnostics Hold Opinions . . . 36
  - 4. Conjugacy Practice . . . 39
  - 5. Maximizing Mutual Information . . . 40

Homework 3 – Bias-Variance Tradeoff & Regularization via Priors 44
- Problems . . . 44
  - 1. Instability of the MLE . . . 44
  - 2. Regularization via Prior Information . . . 48
  - 3. Bias in Regularized Estimators . . . 50
  - 4. Variance-Bias Trade-offs . . . 54
  - 5. Robust Inference via Fat-Tailed Priors . . . 57
  - Python Code for Problem 5 . . . 63

Homework 4 – Expectation Maximization, Sampling, & MCMC 65
- Problems . . . 65
  - 1. Chapter 13, Problem 5 [BDA3] . . . 65
  - 2. Chapter 13, Problem 9 [BDA3] . . . 67
  - 3. Chapter 10, Problem 1 [BDA3] . . . 69
  - 4. Chapter 10, Problem 5 [BDA3] . . . 70
  - 5. Chapter 10, Problem 6 [BDA3] . . . 73
  - 6. Chapter 10, Problem 7 [BDA3] . . . 76
  - 7. Chapter 11, Problem 3 [BDA3] . . . 78
  - 8. Chapter 11, Problem 5 [BDA3] . . . 79
  - 9. Chapter 11, Problem 6 [BDA3] . . . 80

Homework 5 – Tempering, HMC, & Model Checking 81
- Problems . . . 81
  - 1. MCMC Sampling in Mixture Models [BDA3] . . . 81
  - 2. Chapter 12, Problem 2: Simulated Tempering [BDA3] . . . 82
  - 3. Chapter 12, Problem 3: Hamiltonian Monte Carlo [BDA3] . . . 88
  - 4. Chapter 6, Problem 1: Posterior Predictive Checking [BDA3] . . . 94
  - 5. Chapter 6, Problem 4: Model Checking and Sensivity Analysis [BDA3] . . . 97
  - 6. Chapter 6, Problem 7: Prior vs. Posterior Predictive Checks [BDA3] . . . 100

Labs 102

Lab 1: The Beta, Binomial Model **103**
Instructions . . . 103
The (Beta, Binomial) Model . . . 103
Explore the Data . . . 104
The Observables . . . 104
The Unknown . . . 104
The Inferential Pipeline . . . 104
Your aim: . . . 104
More on interval estimation . . . 105
1. Directly from PPF $F^{-1}(\cdot)$. . . . 105
2. **Chebyshev Inequality**: . . . 105
Gathering Information . . . 108
Fitting the Prior . . . 115
Estimating α and β . . . 115
Approach 1: Prior maximization (via estimating θ) . . . 115
Approach 2: Evidence maximization (aka Empirical Bayes aka type II MLE) . . . 115
Differences in the approaches: . . . 116
Model Checking . . . 119
The Evidence . . . 119
Posterior Predictive Distribution . . . 120
Post-Script (Peaking at the Answer) . . . 122

Lab 2: Bayesian Pipeline for Kidney Cancer **123**
Instructions . . . 123
The Setting . . . 123
The Model . . . 124
Model Assumptions (A Warning) . . . 125
The Inference Pipeline . . . 125
Evidence maximization (aka Empirical Bayes aka type II MLE) . . . 126
Fit for the Prior Parameters . . . 129
Infer Death Rates . . . 132
Decision Theory . . . 140
Our approach . . . 141

Lab 3: Kalman Filtering and Online Bayesian Inference **147**
Instructions . . . 147
[PART I] Conjugate Gaussian Models . . . 147
The Model: . . . 147
Solving for the Posterior (in 1 Dimension) . . . 148
Solving for the Posterior (in d Dimensions) . . . 149
Inspecting the Update Rule . . . 150
Visualization . . . 151
[PART II - Preview] Kalman Filters . . . 153
An overview. . . . 154
Link to Lab 4: . . . 154

Lab 4: Kalman Filtering and Online Bayesian Inference **155**
Instructions . . . 155
[PART I - Reference] Conjugate Gaussian Models . . . 156
[PART II] Kalman Filters . . . 156
An overview. . . . 156
Example: Simple Harmonic Oscillators . . . 157
Step 1: Define the State-Space Model . . . 157
Step 2: Compute the Prediction Step . . . 158
Step 3: Compute the Kalman Gain . . . 158
Step 4: Compute the Update Step . . . 158
Step 5: Implement the Kalman Filter . . . 158

Step 6: Visualizing the Results . . . 159
Experiments To Run . . . 159
[PART III] Application: Lorenz System Equations . . . 167
Discretize the System . . . 167
Step 1: Generate Noisy Observations . . . 167
Step 2: Compute the Prediction Step . . . 168
Step 3: Compute the Kalman Gain . . . 168
Step 4: Compute the Update Step . . . 168
Step 5: Implement the Kalman Filter and Visualize Results . . . 169
Outro: A note on revolutions . . . 178

Lab 5: Bayesian Optimization and Surrogate Modeling via Gaussian Processes **179**
Instructions . . . 179
Part I: Gaussian Process Regression . . . 179
Part II: Surrogate Modeling and Bayesian Optimization . . . 186

Lab 6: Bayesian Logistic Regression **190**
Introduction . . . 190
Data . . . 190
Bayesian Logistic Regression Models . . . 190
Likelihoods: . . . 190
Priors: . . . 191
Posterior Inference . . . 192
Sampling: . . . 192
Posterior Summary: . . . 195
Posterior Visualization: . . . 196
Instructions for Experimentation: . . . 197
Uniform . . . 197
Expert . . . 200
Regularized . . . 202
Robust . . . 205
Discussion . . . 207

Lab 7: MAP Estimation in Bayesian Hierarchical Models **208**
Instructions . . . 208
Set Up: . . . 208
Introduction . . . 209
Part I: Model Construction . . . 209
Construct the Low-Pass Filter . . . 212
Simulate the Observations . . . 213
Part II: Point Estimation . . . 213
Coordinate-Ascent . . . 214
Gradient-Based Optimization (L-BFGS-B) . . . 217
Part III: Laplace Approximation for Uncertainty Estimation . . . 218
Part IV: Visualization and Comparison . . . 219

Lab 8: Sampling with Metropolis–Hastings on the Ising Model **222**
Instructions . . . 222
Set Up . . . 222
Model . . . 222
Initialize the Ising Grid . . . 223
Compute the Energy . . . 224
Markov Chain Monte Carlo . . . 225
Proposals: . . . 225
Updates: . . . 228
Simulation . . . 230
Visual Diagnostics and Burn-In Detection . . . 231
Mixing Times . . . 233

Lab 9: Hamiltonian Monte Carlo, Posterior Predictive Checks, and Empirical Bayes with PyMC **235**
Instructions . . . 235
Problem Setup: Bayesian Logistic Regression . . . 235
Prior Distributions . . . 235
1. Import Required Packages and Define Helper Functions . . . 235
2. Simulate the Dataset . . . 236
3. Build and Sample the Model in PyMC . . . 236
4. Coverage . . . 238
5. Posterior Predictive Checks (PPC) and Plotting the Fit . . . 241

Mini Projects **245**

Mini Project 1 – Dead Horse Talk **246**

Mini Project 2 – Consistency, Inconsistency, "Ish"-consistency **251**

Mini Project 3 – Bayes Optimal Coding, Search, and Experimental Design **254**

Mini Project 4 – Model Critique **261**

Mini Project 5 – Bayesian Optimization **263**

Mini Project 6 – Structured Models **266**

Mini Project 7 – Exponential Family Proofs **268**

Mini Project 8 – Advanced MCMC Methods **271**
Prompts . . . 271
Model . . . 271
Background & Notation . . . 271
Example . . . 272
Metropolis-Hastings Sampling . . . 272
(a) Mixing Times for MH in the Ising Model . . . 273
Experimental Results . . . 273
Discussion . . . 273
(b) Simulated Tempering . . . 277
(c) Ensemble Methods . . . 284

Mini Project 9 – Mixing Times in Sampling **288**

Mini Project 10 – Variational Inference **290**
Prompts . . . 290
Notation . . . 291
(a) Variational Inference vs. Markov Chain Monte Carlo . . . 291
(b) Why use $\mathrm{KL}(q \,||\, p_*)$ instead of $\mathrm{KL}(p_* \,||\, q)$ for variational inference? . . . 291
Variational inference to Bayesian models . . . 292
(c) Coordinate ascent variational inference (CAVI) algorithm in conditionally conjugate models . . . 293
Complete Conditionals in the Exponential Family . . . 293
Conditional Conjugacy and Bayesian Models . . . 294
Relation to another algorithm in the class . . . 295
(d) Biases induced by the mean-field assumption . . . 296
(e) + (f) Stein variational gradient descent (SVGD) . . . 296
Appendix A. Stein's Identify and Reproducing Kernel Hilbert Spaces . . . 299
Appendix B. Functional Gradient . . . 299
Appendix C. Ensemble Markov Chain Monte Carlo with Teleporting Walkers . . . 300

Quiz Practice **301**

Bayes' Central Limit Theorem **301**

Conjugate Models via Exponential Families **303**
Building Conjugate Models for/from Exponential Families . . . 304

Sampling Procedures **305**
Markov Chain Monte Carlo (MCMC) . . . 305
Metropolis Algorithm . . . 305
Metropolis-Hastings Algorithm . . . 305
Gradient Based Sampling . . . 306
Metropolis-Adjusted Langevin Algorithm . . . 306
Hamiltonian Monte Carlo Algorithm . . . 307

Model Checking **308**
Coverage Checking . . . 308
Posterior Predictive Checking . . . 308
Leave-One-Out Cross-Validation for Predictive Skill . . . 309

Quizzes with Solutions **310**
Quiz 1: Definitions & Beta Binomial Model . . . 310
Quiz 2: Consistency & Decision Theory . . . 312
Quiz 3: Normal Normal Model & Exponential Families . . . 314
Quiz 4: Conjugate Models, de Finetti's Theorem, & Conditional Independence . . . 316
Quiz 5: Laplace Approximation & Coordinate Ascent . . . 318
Quiz 6: Importance & Rejection Sampling + MCMC . . . 320
Quiz 7: Metropolis-Adjusted & Hamiltonian Monte Carlo . . . 322
Quiz 8: Coverage Check, Posterior Predictive Check, & Cross-Validation . . . 325

Lectures **328**
Lecture 1: Introduction . . . 328
Lecture 2: Beta Binomial Model (1/2) . . . 330
Lecture 3: Beta Binomial Model (2/2) . . . 332
Lecture 4: Bayesian Central Limit Theorem . . . 336
Lecture 5: Decision Theory (1/3) . . . 338
Lecture 6: Decision Theory (2/3) . . . 342
Lecture 7: Decision Theory (3/3) . . . 347
Lecture 8: Conjugate Models . . . 352
Lecture 9: Normal Normal Model . . . 356
Lecture 10: M-estimation and Exponential Families . . . 367
Lecture 12: Weakly Informative Priors . . . 373
Lecture 13: Hierarchical Models (1/2) . . . 377
Lecture 14: Hierarchical Models (2/2) . . . 381
Lecture 15: Bayesian Point Estimation . . . 384
Lecture 16: Bernstein Von Mises Theorem . . . 386
Lecture 17: MAP Estimation in Structured Models . . . 389
Lecture 18: Distribution Approximation . . . 396
Lecture 19: Introduction to Sampling . . . 401
Lecture 20: Markov Chain Monte Carlo . . . 409
Lecture 21: Towards Efficient MCMC . . . 415
Lecture 22: Gradient Based Samplers . . . 446
Lecture 23: Sampling & Distribution Approximation . . . 455
Lecture 24: Distribution Estimation & Introduction to Validation . . . 464
Lecture 25: Posterior Summary & Validation . . . 474
Lecture 26: Cross-Validation & Sensitivity Analysis . . . 478
Lecture 27: Ensemble MCMC . . . 482
Lecture 28: Variational Inference . . . 488

Figures

1 Difference between the posterior variance and the prior variance ($\mathrm{Var}[\Theta|S] - \mathrm{Var}[\Theta]$). . . . 21
2 Posterior and Marginal Distributions for $\epsilon = 0.5$, 0.1, and 0.01 . . . 27
3 Example of the two posteriors, $p(\sigma \mid y)$ and $p(\sigma^2 \mid y)$, for $n = 10$ and $v = 1/10$. . . . 38
4 Absolute error for MAP estimator, $\hat{x}_{\mathrm{MAP}}(Y^{(n)}) - x^\star$ for $x^\star = 1, 2, 4, 8, 16$ for the three priors. . . . 60
5 Approximate error for MAP estimator, $-(nI[x^\star])^{-1}\nabla_x \log(p_X(x))\big|_{x=x^\star}$ for $x^\star = 1, 2, 4, 8, 16$ for the three priors. 61
8 An example configuration of the Ising model. We let ✓ and ✗ represent when s_i and its neighbor are aligned and misaligned, respectively. . . . 272
9 Metropolis-Hastings acceptance probability as a function of change in energy $\Delta E(s, s')$ for varied temperatures. 273
10 Energy $M(\mathbf{s})$ and magnetization $M(\mathbf{s})$ mixing time calculated as total number of samples divided by effective number of samples for grid sizes 4, 16, 64, and 256 . . . 274
11 Average absolute magnetization, $\mathbb{E}[|M|]$, for grid sizes 4, 16, 64, and 256. The vertical black dashed lines are the estimated critical temperature τ_{crit} . . . 275
12 Energy and magnetization samples after burn in for grid sizes 4, 16, 64, and 256 . . . 276
13 Effective sample size for energy and magnetization using Metropolis-Hastings and simulated tempering after 100 runs. . . . 280
14 Energy and magnetization samples for Metropolis-Hastings and simulated tempering for 100 runs with $\tau_{\mathrm{hard}} = 0.1$. . . . 281
15 Ensemble MCMC with Teleporting Walkers . . . 286
16 Example target two-dimensional Gaussian with a non-zero correlation between z_1 and z_2. Here, $q_1(\mathbf{z})$ on the left and $q_2(\mathbf{z})$ on the right represent optimal densities under objectives $\mathrm{KL}(q \,||\, p_*)$ and $\mathrm{KL}(p_* \,||\, q)$, respectively. . . . 292
17 Hierarchical model $p(\vec{\beta}, \mathbf{z}, \mathbf{x})$ over global and latent variables $\vec{\beta}$, $\mathbf{z}$ and observable $\mathbf{x}$. . . . 294
18 Flow map $\boldsymbol{\chi} : \mathcal{Z}_0 \to \mathcal{Z}$ and infinitesimal flow $\boldsymbol{\psi}_t : \mathcal{Z}_t \to \mathcal{Z}_{t+dt}$. The dashed gray line indicates the infinitesimal flow $\boldsymbol{\psi}_t$ at time t. . . . 297

Tables

1 Quality control measurements from 6 machines in a factory. . . . 78
2 Bioassay data from Racine et al. (1986) . . . 88
3 Effective sample size for energy and magnetization using Metropolis-Hastings and simulated tempering after 100 runs. . . . 280

Algorithms

1 Rejection Sampling . . . 70
2 Metropolis Algorithm . . . 83
3 Metropolis Algorithm from Lab 6 . . . 97
4 Simulated Tempering . . . 277
5 Importance Resampling . . . 282
6 Rejection Sampling . . . 283
7 Ensemble Markov Chain Monte Carlo with Teleporting Walkers [LWZ22] . . . 284
8 Ising Model + Ensemble MCMC with Teleporting Walkers [LWZ22] . . . 287
9 Coordinate Ascent Variational Inference (CAVI) [BKM17] . . . 293
10 Coordinate Ascent Variational Inference (CAVI) in Conditionally Conjugate Bayesian Models [BKM17] . . . 295
11 Stein Variational Gradient Descent (SVGD) [LW16] . . . 297
12 Ensemble Markov Chain Monte Carlo with Teleporting Walkers [LWZ22] . . . 300
13 Metropolis Algorithm . . . 305
14 Metropolis-Hastings Algorithm . . . 305
15 Metropolis-Adjusted Langevin Algorithm . . . 306
16 Hamiltonian Monte Carlo Algorithm . . . 307
17 Coverage Checking . . . 308
18 Posterior Predictive Checking . . . 308
19 Leave-One-Out Cross-Validation . . . 309

Homeworks

STAT238: Bayesian Statistics February 11th, 2025

Homework 1: The Beta, Binomial Model

Reece D. Huff

Notation

Beta-Binomial Model

Throughout this homework, we working with the Beta, Binomial Model. Here, we have

$$\Theta \sim \text{Beta}(\alpha, \beta) \quad \text{and} \quad S \mid \Theta = \theta \sim \text{Binomial}(n, \theta)$$

where the densities are given by

$$f_\Theta(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\text{B}(\alpha,\beta)} \quad \text{for all } \theta \in [0,1] \quad \text{and} \quad f_{S|\Theta=\theta}(s) = \binom{n}{s}\theta^s(1-\theta)^{n-s} \quad \text{for all } s \in \{0,1,2,\ldots,n\}$$

Here, $\text{B}(\alpha,\beta)$ represents the Beta function which can be written as a product of Gamma functions, i.e.,

$$\text{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \quad \text{where} \quad \Gamma(z) = \begin{cases} (z-1)! & \text{when } z \text{ is a positive integer} \\ \int_0^\infty t^{z-1}e^{-t}\,\mathrm{d}t & \text{all other positive reals} \end{cases}$$

Recall some summary statistics of the Beta distribution. For $\Theta \sim \text{Beta}(\alpha,\beta)$, we have

$$\mathbb{E}[\Theta] = \frac{\alpha}{\alpha+\beta} \qquad \text{Var}[\Theta] = \frac{\mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\alpha+\beta+1} = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \qquad \text{mode}[\Theta] = \frac{\alpha-1}{\alpha+\beta-2}$$

Given this model, the posterior also follows a Beta distribution, $\Theta|S=s;n,\alpha,\beta \sim \text{Beta}(s+\alpha, n-s+\beta)$. Therefore, we have

$$\mathbb{E}[\Theta \mid S=s] = \frac{s+\alpha}{n+\alpha+\beta} \qquad \text{Var}[\Theta \mid S=s] = \frac{\mathbb{E}[\Theta \mid S=s](1-\mathbb{E}[\Theta \mid S=s])}{n+\alpha+\beta+1} \qquad \text{mode}[\Theta \mid S=s] = \frac{s+\alpha-1}{n+\alpha+\beta-2}$$

Problems

Problem 1: Chapter 1, Problem 3 [BDA3]

Probability calculation for genetics (from Lindley, 1965): suppose that in each individual of a large population there is a pair of genes, each of which can be either x or X, that controls eye color: those with xx have blue eyes, while heterozygotes (those with Xx or xX) and those with XX have brown eyes. The proportion of blue-eyed individuals is p^2 and of heterozygotes is $2p(1-p)$, where $0 < p < 1$. Each parent transmits one of its own genes to the child; if a parent is a heterozygote, the probability that it transmits the gene of type X is $\frac{1}{2}$. Assuming random mating, show that among brown-eyed children of brown-eyed parents, the expected proportion of heterozygotes is $2p/(1+2p)$. Suppose Judy, a brown-eyed child of brown-eyed parents, marries a heterozygote, and they have n children, all brown-eyed. Find the posterior probability that Judy is a heterozygote and the probability that her first grandchild has blue eyes.

1. **Show that among brown-eyed children of brown-eyed parents, the expected proportion of heterozygotes is $\frac{2p}{1+2p}$.**

   Let C represent the child of parents P_1 and P_2. Let P be the shorthand for both parents, e.g., $\mathbb{P}[P = XX] = \mathbb{P}[P_1 = XX, P_2 = XX]$ represents the probability that both parents have XX genes. Let Het represent the event that an individual is a heterozygote (those with Xx or xX). Finally, let Br represent the event that an individual has brown eyes, e.g., $\mathbb{P}[C = Br] = \mathbb{P}[C \in \{xX, Xx, XX\}]$ represents that probability that child C has brown eyes.

P_1	P_2	$\mathbb{P}[C = i \mid P_1, P_2]$		
		xx	Het	XX
xx	xx	1	0	0
	Het	1/2	1/2	0
	XX	0	1	0

P_1	P_2	$\mathbb{P}[C = i \mid P_1, P_2]$		
		xx	Het	XX
Het	xx	1/2	1/2	0
	Het	1/4	1/2	1/4
	XX	0	1/2	1/2

P_1	P_2	$\mathbb{P}[C = i \mid P_1, P_2]$		
		xx	Het	XX
XX	xx	0	1	0
	Het	0	1/2	1/2
	XX	0	0	1

   We start with

$$\mathbb{P}[C = Het \mid C = Br, P = Br] = \frac{\mathbb{P}[C = Het, C = Br, P = Br]}{\mathbb{P}[C = Br, P = Br]}$$

   We note that if a child has brown eyes they are a heterozygote, i.e.,

$$\mathbb{P}[C = Het, C = Br] = \mathbb{P}[C \in \{xX, Xx\}, C = \in \{xX, Xx, XX\}] = \mathbb{P}[C \in \{xX, Xx\}] = \mathbb{P}[C = Het],$$

   so the numerator simplifies to $\mathbb{P}[C = Het, C = Br, P = Br] = \mathbb{P}[C = Het, P = Br]$. Next, it follows from the law of total probability that

$$\begin{aligned}
\mathbb{P}[C = Het, P = Br] &= \mathbb{P}[C = Het | P = XX]\,\mathbb{P}[P = XX] + \mathbb{P}[C = Het | P_1 = Het, P_2 = XX]\,\mathbb{P}[P_1 = Het, P_2 = XX] \\
&\quad + \mathbb{P}[C = Het | P_1 = XX, P_2 = Het]\,\mathbb{P}[P_1 = XX, P_2 = Het] + \mathbb{P}[C = Het | P = Het]\,\mathbb{P}[P = Het] \\
\mathbb{P}[C = Het, P = Br] &= \left(0\right)\mathbb{P}[P = XX] + 2 \times \left(\frac{1}{2}\right)\left(2p(1-p)\right)\left((1-p)^2\right) + \left(\frac{1}{2}\right)\left(2p(1-p)\right)^2 \\
\mathbb{P}[C = Het, P = Br] &= (1-p)^2\left(2p(1-p) + 2p^2\right)
\end{aligned}$$

   We apply the the law of total probability to the denominator. We have

$$\begin{aligned}
\mathbb{P}[C = Br, P = Br] &= \mathbb{P}[C = Br | P = XX]\,\mathbb{P}[P = XX] + \mathbb{P}[C = Br | P_1 = Het, P_2 = XX]\,\mathbb{P}[P_1 = Het, P_2 = XX] \\
&\quad + \mathbb{P}[C = Br | P_1 = XX, P_2 = Het]\,\mathbb{P}[P_1 = XX, P_2 = Het] + \mathbb{P}[C = Br | P = Het]\,\mathbb{P}[P = Het] \\
\mathbb{P}[C = Br, P = Br] &= \left(1\right)\left((1-p)^2\right)^2 + 2 \times \left(1\right)\left(2p(1-p)\right)\left((1-p)^2\right) + \left(\frac{3}{4}\right)\left(2p(1-p)\right)^2 \\
\mathbb{P}[C = Br, P = Br] &= (1-p)^2\left((1-p)^2 + 4p(1-p) + 3p^2\right)
\end{aligned}$$

   Substituting the results for the numerator and denominator back into the original expression results in

$$\mathbb{P}[C = Het \mid C = Br, P = Br] = \frac{\mathbb{P}[C = Het, P = Br]}{\mathbb{P}[C = Br, P = Br]} = \frac{2p(1-p) + 2p^2}{(1-p)^2 + 4p(1-p) + 3p^2} = \frac{2p}{1+2p} \quad \checkmark$$

2. Find the posterior probability that Judy is a heterozygote.

Let J represent Judy. Given Judy is a brown-eyed child of brown-eyed parents, we have that

$$\mathbb{P}[J = Het \mid J = Br, P = Br] = \frac{2p}{1+2p} \quad \text{and} \quad \mathbb{P}[J = XX \mid J = Br, P = Br] = \frac{1}{1+2p}.$$

Now given that Judy is a brown-eyed child of brown-eyed parents, marries a heterozygote, and they have n children, all brown-eyed, we ask the probability that J is a heterozygote. We wrap all prior information in the term "prior" and the drop the term entirely as the posterior now represents our prior. Then we have

$$\mathbb{P}[J = Het \mid n \text{ children} = Br, J = Br, P = Br] = \mathbb{P}[J = Het \mid n \text{ children} = Br, \text{prior}] = \mathbb{P}[J = Het \mid n \text{ children} = Br]$$

It then follows that

$$\begin{aligned}
\mathbb{P}[J = Het \mid n \text{ children} = Br] &= \frac{\mathbb{P}[J = Het, n \text{ children} = Br]}{\mathbb{P}[n \text{ children} = Br]} \\
\mathbb{P}[J = Het \mid n \text{ children} = Br] &= \frac{\mathbb{P}[n \text{ children} = Br \mid J = Het]\,\mathbb{P}[J = Het]}{\mathbb{P}[n \text{ children} = Br \mid J = Het]\,\mathbb{P}[J = Het] + \mathbb{P}[n \text{ children} = Br \mid J = XX]\,\mathbb{P}[J = XX]} \\
\mathbb{P}[J = Het \mid n \text{ children} = Br] &= \frac{\left(\frac{3}{4}\right)^n \frac{2p}{1+2p}}{\left(\frac{3}{4}\right)^n \frac{2p}{1+2p} + (1)^n \frac{1}{1+2p}} = \frac{\left(\frac{3}{4}\right)^n \frac{2p}{1+2p}}{\left(\frac{3}{4}\right)^n \frac{2p}{1+2p} + \frac{1}{1+2p}} = \frac{\left(\frac{3}{4}\right)^n 2p}{1 + \left(\frac{3}{4}\right)^n 2p}.
\end{aligned}$$

Following the same steps, we have that

$$\mathbb{P}[J = XX \mid n \text{ children} = Br] = \frac{\frac{1}{1+2p}}{\left(\frac{3}{4}\right)^n \frac{2p}{1+2p} + \frac{1}{1+2p}} = \frac{1}{1 + \left(\frac{3}{4}\right)^n 2p}$$

3. Find the probability that her first grandchild has blue eyes.

Let C_i represent the i-th child of Judy. Then we have that

$$\begin{aligned}
\mathbb{P}[C_i = Het \mid n \text{ children} = Br] &= \mathbb{P}[C_i = Het \mid n \text{ children} = Br, J = Het]\,\mathbb{P}[J = Het \mid n \text{ children} = Br] \\
&\quad + \mathbb{P}[C_i = Het \mid n \text{ children} = Br, J = XX]\,\mathbb{P}[J = XX \mid n \text{ children} = Br] \\
\mathbb{P}[C_i = Het \mid n \text{ children} = Br] &= \mathbb{P}[C_i = Het \mid C_i = Br, J = Het]\,\mathbb{P}[J = Het \mid n \text{ children} = Br] \\
&\quad + \mathbb{P}[C_i = Het \mid C_i = Br, J = XX]\,\mathbb{P}[J = XX \mid n \text{ children} = Br]
\end{aligned}$$

where

$$\begin{aligned}
\mathbb{P}[C_i = Het \mid C_i = Br, J = Het] &= \frac{\mathbb{P}[C_i = Het, J = Het]}{\mathbb{P}[C_i = Br, J = Het]} \\
\mathbb{P}[C_i = Het \mid C_i = Br, J = Het] &= \frac{\mathbb{P}[C_i = Het \mid J = Het]\,\mathbb{P}[J = Het]}{\mathbb{P}[C_i = Het \mid J = Het]\,\mathbb{P}[J = Het] + \mathbb{P}[C_i = XX \mid J = Het]\,\mathbb{P}[J = Het]} \\
\mathbb{P}[C_i = Het \mid C_i = Br, J = Het] &= \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}} = \frac{2}{3}
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{P}[C_i = Het \mid C_i = Br, J = XX] &= \frac{\mathbb{P}[C_i = Het, J = XX]}{\mathbb{P}[C_i = Br, J = XX]} \\
\mathbb{P}[C_i = Het \mid C_i = Br, J = XX] &= \frac{\mathbb{P}[C_i = Het \mid J = XX]\,\mathbb{P}[J = XX]}{\mathbb{P}[C_i = Het \mid J = XX]\,\mathbb{P}[J = XX] + \mathbb{P}[C_i = XX \mid J = XX]\,\mathbb{P}[J = XX]} = \frac{1}{2}.
\end{aligned}$$

Then, we have that

$$\mathbb{P}[C_i = Het \mid n \text{ children} = Br] = \left(\frac{2}{3}\right) \frac{\left(\frac{3}{4}\right)^n 2p}{1 + \left(\frac{3}{4}\right)^n 2p} + \left(\frac{1}{2}\right) \frac{1}{1 + \left(\frac{3}{4}\right)^n 2p} = \frac{\frac{1}{2} + \left(\frac{3}{4}\right)^n \frac{4p}{3}}{1 + \left(\frac{3}{4}\right)^n 2p}$$

To finish, we let G be the first grandchild of Judy. We have that

$$\mathbb{P}[G = xx \mid C_i = Het, n \text{ children} = Br] = \mathbb{P}[G = xx \mid P_2 = xx]\,\mathbb{P}[P_2 = xx] + \mathbb{P}[G = xx \mid P_2 = Het]\,\mathbb{P}[P_2 = Het]$$

$$\mathbb{P}[G = xx \mid C_i = Het, n \text{ children} = Br] = \left(\left(\frac{1}{2}\right)(p^2) + \left(\frac{1}{4}\right)(2p(1-p))\right)\left(\frac{\frac{1}{2} + \left(\frac{3}{4}\right)^n \frac{4p}{3}}{1 + \left(\frac{3}{4}\right)^n 2p}\right)$$

$$\mathbb{P}[G = xx \mid C_i = Het, n \text{ children} = Br] = \left(\frac{1}{2}p\right)\left(\frac{\frac{1}{2} + \left(\frac{3}{4}\right)^n \frac{4p}{3}}{1 + \left(\frac{3}{4}\right)^n 2p}\right)$$

$$\mathbb{P}[\text{Judy's first grandchild has blue eyes} \mid \text{all prior information}] = \frac{\frac{1}{4}p + \left(\frac{3}{4}\right)^n \frac{2p^2}{3}}{1 + \left(\frac{3}{4}\right)^n 2p}$$

Problem 2: Chapter 1, Problem 7 [BDA3]

Conditional probability: the following problem is loosely based on the television game show *Let's Make a Deal*. At the end of the show, a contestant is asked to choose one of three large boxes, where one box contains a fabulous prize and the other two boxes contain lesser prizes. After the contestant chooses a box, Monty Hall, the host of the show, opens one of the two boxes containing smaller prizes. (In order to keep the conclusion suspenseful, Monty does not open the box selected by the contestant.) Monty offers the contestant the opportunity to switch from the chosen box to the remaining unopened box. Should the contestant switch or stay with the original choice? Calculate the probability that the contestant wins under each strategy. This is an exercise in being clear about the information that should be conditioned on when constructing a probability judgment. See Selvin (1975) and Morgan et al. (1991) for further discussion of this problem.

Let A_i be the event that the fabulous prize is behind box i, and let O_j be the event Monty opens box j. To start, we have that $\mathbb{P}[A_1] = \mathbb{P}[A_2] = \mathbb{P}[A_3] = \frac{1}{3}$.

Without loss of generality, suppose the contestant initially picks box 1. Monty then opens one box that does *not* contain the prize. Again, without loss of generality, suppose Monty opens box 2, i.e., O_2. Then the probability that the prize is in box 1 (staying) and box 3 (switching) is given by

$$\mathbb{P}[\text{win by staying}] = \mathbb{P}[A_1 \mid O_2] = \frac{\mathbb{P}[O_2 \mid A_1]\,\mathbb{P}[A_1]}{\mathbb{P}[O_2]} \quad \text{and} \quad \mathbb{P}[\text{win by switching}] = \mathbb{P}[A_3 \mid O_2] = \frac{\mathbb{P}[O_2 \mid A_3]\,\mathbb{P}[A_3]}{\mathbb{P}[O_2]}$$

This is the crux of the problem from which we make two key observations:

- *When the prize is in the box that the contestant chose*: Monty knows this, so Monty can reveal either box 2 or box 3. We assume that he will do so with equal probability, i.e., $\mathbb{P}[O_2 \mid A_1] = \mathbb{P}[O_3 \mid A_1] = \frac{1}{2}$.
- *When the prize is <u>not</u> in the box that the contestant chose*: Monty knows this, so he is forced to reveal the box that does not have the prize of the remaining two, i.e., $\mathbb{P}[O_2 \mid A_2] = 0$ and $\mathbb{P}[O_2 \mid A_3] = 1$

Therefore, we have

$$\mathbb{P}[O_2] = \mathbb{P}[O_2 \mid A_1]\,\mathbb{P}[A_1] + \mathbb{P}[O_2 \mid A_2]\,\mathbb{P}[A_2] + \mathbb{P}[O_2 \mid A_3]\,\mathbb{P}[A_3] = \left(\tfrac{1}{2}\right)\tfrac{1}{3} + (0)\tfrac{1}{3} + (1)\tfrac{1}{3} = \tfrac{1}{2}.$$

To conclude, we have

$$\mathbb{P}[\text{win by staying}] = \frac{\mathbb{P}[O_2 \mid A_1]\,\mathbb{P}[A_1]}{\mathbb{P}[O_2]} = \frac{\left(\frac{1}{2}\right)\left(\frac{1}{3}\right)}{\frac{1}{2}} = \frac{1}{3} \quad \text{and} \quad \mathbb{P}[\text{win by switching}] = \frac{\mathbb{P}[O_2 \mid A_3]\,\mathbb{P}[A_3]}{\mathbb{P}[O_2]} = \frac{(1)\left(\frac{1}{3}\right)}{\frac{1}{2}} = \frac{2}{3}$$

Since $\mathbb{P}[\text{win by switching}]$ is double that of $\mathbb{P}[\text{win by staying}]$, <u>the contestant should switch</u>.

Problem 3: Chapter 2, Problem 1 [BDA3]

Posterior inference: suppose you have a Beta(4, 4) prior distribution on the probability θ that a coin will yield a 'head' when spun in a specified manner. The coin is independently spun ten times, and 'heads' appear fewer than 3 times. You are not told how many heads were seen, only that the number is less than 3. Calculate your exact posterior density (up to a proportionality constant) for θ and sketch it.

This problem utilizes the Beta-Binomial Model (see Beta-Binomial Model) with $\alpha = \beta = 4$ and $S < 3$. Then by Bayes' we have

$$\mathbb{P}[\Theta \mid S < 3] = \frac{\mathbb{P}(S < 3 \mid \Theta)\,\mathbb{P}(\Theta)}{\mathbb{P}(S < 3)} \propto \mathbb{P}(S < 3 \mid \Theta)\,\mathbb{P}(\Theta)$$

Evaluating the likelihood, we have

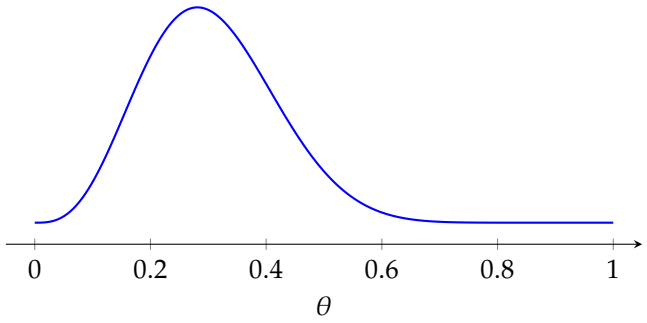
$$\mathbb{P}(S < 3 \mid \Theta) = \sum_{k=0}^{2} \binom{10}{k} \theta^k (1-\theta)^{10-k} = \binom{10}{0}\theta^0(1-\theta)^{10-0} + \binom{10}{1}\theta^1(1-\theta)^{10-1} + \binom{10}{2}\theta^2(1-\theta)^{10-2}$$
$$\mathbb{P}(S < 3 \mid \Theta) = (1-\theta)^{10} + 10\theta(1-\theta)^9 + 45\theta^2(1-\theta)^8.$$

Evaluating the prior, we have

$$\mathbb{P}(\Theta) = \frac{\theta^3(1-\theta)^3}{\mathrm{B}(4,4)} = \frac{\Gamma(8)}{\Gamma(4)\Gamma(4)}\theta^3(1-\theta)^3 = \frac{7!}{3!3!}\theta^3(1-\theta)^3 = 140\theta^3(1-\theta)^3.$$

Thus, our posterior up to proportionality is

$$\mathbb{P}[\Theta \mid S < 3] \propto \Big((1-\theta)^{10} + 10\theta(1-\theta)^9 + 45\theta^2(1-\theta)^8\Big)\Big(140\theta^3(1-\theta)^3\Big)$$
$$\boxed{\mathbb{P}[\Theta \mid S < 3] \propto \theta^3(1-\theta)^{13} + 10\theta^4(1-\theta)^{12} + 45\theta^5(1-\theta)^{11}}$$



Problem 4: Chapter 2, Problem 9 [BDA3]

Setting parameters for a beta prior distribution: suppose your prior distribution for θ, the proportion of Californians who support the death penalty, is beta with mean 0.6 and standard deviation 0.3.

(a) Determine the parameters α and β of your prior distribution. Sketch the prior density function.

We can determine the parameters α and β by matching the moments of the Beta distribution. We have

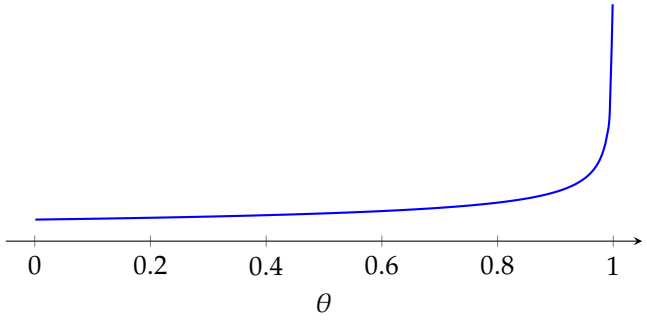
$$\alpha = \left(\frac{\mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\text{Var}[\Theta]} - 1\right)\mathbb{E}[\Theta] \quad \text{and} \quad \beta = \left(\frac{\mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\text{Var}[\Theta]} - 1\right)(1-\mathbb{E}[\Theta]).$$

Setting $\mathbb{E}[\Theta] = 0.6$ and $\text{Var}[\Theta] = 0.3^2 = 0.09$, we have

$$\alpha = \left(\frac{0.6(1-0.6)}{0.09} - 1\right)0.6 = 1 \quad \text{and} \quad \beta = \left(\frac{0.6(1-0.6)}{0.09} - 1\right)(1-0.6) = \frac{2}{3}.$$

For $\Theta \sim \text{Beta}(1, \frac{2}{3})$, we have

$$\mathbb{P}[\Theta] \propto \theta^0(1-\theta)^{-1/3}$$



(b) A random sample of 1000 Californians is taken, and 65% support the death penalty. What are your posterior mean and variance for θ? Draw the posterior density function.

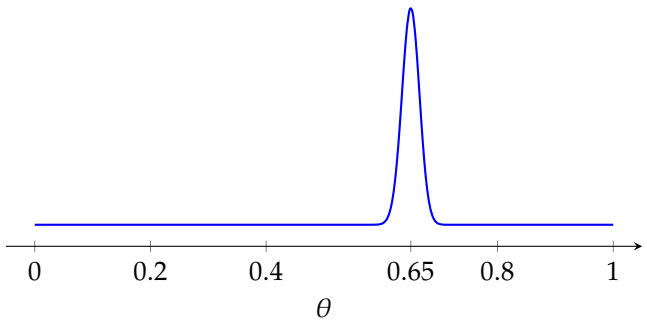
By our Beta, Binomial model, we have that $\Theta \mid S = s \sim \text{Beta}(s+\alpha, n-s+\beta)$. Here, $s = 650$ and $n = 1000$. It follows that

$$\mathbb{E}[\Theta \mid S = s] = \frac{s+\alpha}{n+\alpha+\beta} = \frac{650+1}{1000+1+\frac{2}{3}} \approx 0.65$$

$$\text{Var}[\Theta \mid S = s] = \frac{\mathbb{E}[\Theta \mid S = s](1-\mathbb{E}[\Theta \mid S = s])}{n+\alpha+\beta+1} \approx \frac{6.5(1-6.5)}{1000+1+\frac{2}{3}+1} \approx 0.0002$$

For $\Theta \mid S = s \sim \text{Beta}(s+\alpha, n-s+\beta)$ which will approximate by $\text{Beta}(651, 351)$ we have

$$\mathbb{P}[\Theta \mid S = s] \propto \theta^{650}(1-\theta)^{350}$$

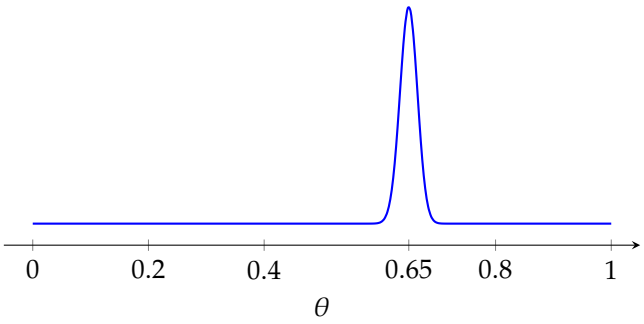


(c) Examine the sensitivity of the posterior distribution to different prior means and widths including a non-informative prior.

For this problem, we try a few different priors:

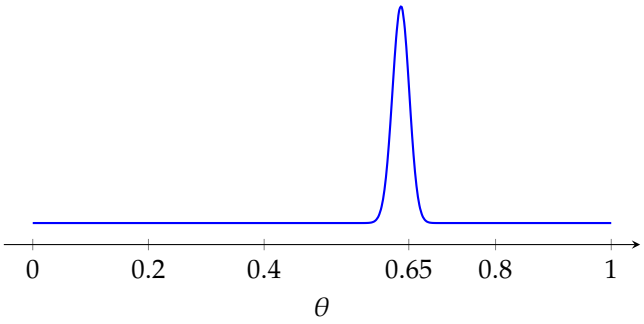
- Prior 1: $\alpha = 1$ and $\beta = 1$ (i.e., uninformative prior):
  For $\Theta \mid S = s \sim \text{Beta}(s + \alpha, n - s + \beta) = \text{Beta}(651, 351)$ we have

$$\mathbb{P}[\Theta \mid S = s] \propto \theta^{650}(1 - \theta)^{350}$$



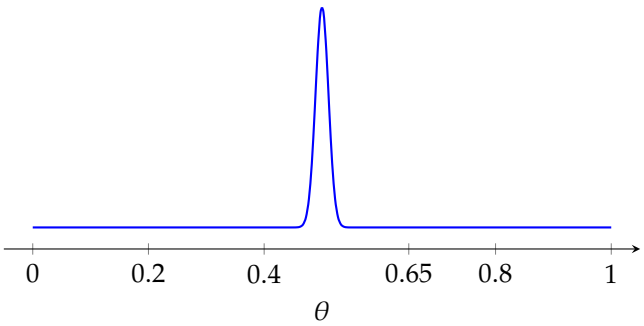
- Prior 2: $\alpha = 50$ and $\beta = 50$ (i.e., moderately strong bad prior):
  For $\Theta \mid S = s \sim \text{Beta}(s + \alpha, n - s + \beta)$ which will approximate by $\text{Beta}(401, 701)$ we have

$$\mathbb{P}[\Theta \mid S = s] \propto \theta^{700}(1 - \theta)^{400}$$



- Prior 3: $\alpha = 350$ and $\beta = 650$ (i.e., strong bad prior):
  For $\Theta \mid S = s \sim \text{Beta}(s + \alpha, n - s + \beta)$ which will approximate by $\text{Beta}(1001, 1001)$ we have

$$\mathbb{P}[\Theta \mid S = s] \propto \theta^{1000}(1 - \theta)^{1000}$$



Problem 5: Moments of the Beta Binomial

Suppose that $\Theta \sim \text{Beta}(\alpha, \beta)$ and $S \mid \Theta = \theta \sim \text{Binomial}(n, \theta)$. That is, S is binomially distributed with n trials and success probability Θ drawn from a Beta distribution with parameters α and β.

If we sample first Θ, then sample S given Θ, the number of successes S is drawn from a Beta – Binomial distribution (see lab 1). In this problem we will learn how to compute its moments without direct integration. Rather we will chain together moments of familiar distributions (the Beta and the Binomial). This is a good strategy that is worth remembering. It is included as a warm up for the rest of this homework.

(a) Recall that the n^{th} raw moment of S is defined, $\mathbb{E}_S[S^n]$. Write down (don't solve) the explicit integral form for the n^{th} raw moment. Comment on the feasibility of direct integration.

To not confuse n^{th} moment with the n number of trials, let m represent the number of trials such that $S \mid \Theta = \theta \sim \text{Binomial}(m, \theta)$. Then, we have

$$\mathop{\mathbb{E}}_{S}[S^n] = \sum_{s=0}^{m} s^n f_S(s)$$

$$\mathop{\mathbb{E}}_{S}[S^n] = \sum_{s=0}^{m} s^n \int_0^1 f_{S|\Theta=\theta}(s) f_\Theta(\theta) d\theta$$

$$\mathop{\mathbb{E}}_{S}[S^n] = \sum_{s=0}^{m} s^n \int_0^1 \binom{m}{s} \theta^m (1-\theta)^{m-s} \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\text{B}(\alpha,\beta)} d\theta$$

$$\mathop{\mathbb{E}}_{S}[S^n] = \frac{1}{\text{B}(\alpha,\beta)} \sum_{s=0}^{m} s^n \binom{m}{s} \int_0^1 \theta^{m+\alpha-1}(1-\theta)^{m-s+\beta-1} d\theta$$

$$\mathop{\mathbb{E}}_{S}[S^n] = \frac{1}{\text{B}(\alpha,\beta)} \sum_{s=0}^{m} s^n \binom{m}{s} \text{B}(s+\alpha, m-s+\beta) \int_0^1 \frac{\theta^{m+\alpha-1}(1-\theta)^{m-s+\beta-1}}{\text{B}(s+\alpha, m-s+\beta)} d\theta \to 1$$

$$\boxed{\mathop{\mathbb{E}}_{S}[S^n] = \frac{1}{\text{B}(\alpha,\beta)} \sum_{s=0}^{m} s^n \binom{m}{s} \text{B}(s+\alpha, m-s+\beta)} \tag{1}$$

Equation (1) is of course extremely difficult to evaluate as it involves both a binomial and a Beta function.

(b) In this case, it is easier to use the rules of conditional expectation, and the law of total variance, to compute the moments. Expand $\mathbb{E}[S]$ and $\text{Var}[S]$ by first conditioning on Θ. Show that:

$$\mathbb{E}[S] = n\,\mathbb{E}[\Theta], \qquad \text{Var}[S] = n(n+\alpha+\beta)\,\text{Var}[\Theta] \tag{2}$$

We first show that $\mathbb{E}[S] = n\,\mathbb{E}[\Theta]$ by the law of total expectation. We have

$$\mathbb{E}[S] = \mathbb{E}[\mathbb{E}[S \mid \Theta]] = \mathbb{E}[n\Theta] = n\,\mathbb{E}[\Theta] \quad \checkmark$$

Next, we show that $\text{Var}[S] = n(n+\alpha+\beta)\,\text{Var}[\Theta]$ via the law of total variance. We have

$$\text{Var}[S] = \mathbb{E}[\text{Var}[S \mid \Theta]] + \text{Var}[\mathbb{E}[S \mid \Theta]] = \mathbb{E}[n\Theta(1-\Theta)] + \text{Var}[n\Theta] = n\Big(\underbrace{\mathbb{E}[\Theta] - \mathbb{E}[\Theta^2]}_{=(\alpha+\beta)\,\text{Var}[\Theta]?} + n\,\text{Var}[\Theta]\Big)$$

At this point, we aim to rewrite the term $\mathbb{E}[\Theta] - \mathbb{E}[\Theta^2]$ in terms of $\text{Var}[\Theta]$. We have

$$\mathbb{E}[\Theta] - \mathbb{E}[\Theta^2] = \mathbb{E}[\Theta] - \text{Var}[\Theta] - \mathbb{E}[\Theta]^2 = \mathbb{E}[\Theta](1 - \mathbb{E}[\Theta]) - \text{Var}[\Theta]$$

Recall that $\text{Var}[\Theta] = \frac{\mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\alpha+\beta+1}$, which implies that $\mathbb{E}[\Theta](1-\mathbb{E}[\Theta]) = (\alpha+\beta+1)\,\text{Var}[\Theta]$. Then we have

$$\mathbb{E}[\Theta] - \mathbb{E}[\Theta^2] = (\alpha+\beta+1)\,\text{Var}[\Theta] - \text{Var}[\Theta] = (\alpha+\beta)\,\text{Var}[\Theta]$$

Substituting this result back into our original expression, we have

$$\text{Var}[S] = n\Big((\alpha+\beta)\,\text{Var}[\Theta] + n\,\text{Var}[\Theta]\Big) = n(n+\alpha+\beta)\,\text{Var}[\Theta] \quad \checkmark$$

(c) Use this result to compute the marginal variance in the MLE estimator for Θ. That is, for $\theta_{\text{MLE}}(S; n, \alpha, \beta) = S/n$. We will use this answer later.

To calculate the marginal variance in the MLE estimator, $\theta_{\text{MLE}}(S; n, \alpha, \beta) = S/n$, we use our result from part (b). We have

$$\text{Var}[\theta_{\text{MLE}}(S; n, \alpha, \beta)] = \text{Var}[S/n] = \frac{\cancel{n}(n+\alpha+\beta)}{n\cancel{^2}} \text{Var}[\Theta] = \frac{n+\alpha+\beta}{n} \text{Var}[\Theta] = \left(\frac{\alpha\beta(n+\alpha+\beta)}{n(\alpha+\beta)^2(\alpha+\beta+1)} \right).$$

Problem 6: Regularization – Bias in Posterior Point Inference

Consider the same joint Θ, S model established in problem 5. Under this model:

(a) Write down the posterior distribution for $\Theta \mid S = s$ as a function of s, the number of trials, n, and the prior parameters α, β. Then, write down the formulas for the posterior expectation, $\hat{\Theta}_{\text{mean}}(s; n, \alpha, \beta) = \mathbb{E}[\Theta \mid S = s]$, and MAP estimator, $\hat{\Theta}_{\text{MAP}}(s; n, \alpha, \beta)$, as functions of s, n, α, and β. You do not need to rederive the equations for the mean and mode of the beta distribution.

In order to write down the posterior distribution for $\Theta \mid S = s$ as a function of s, the number of trials, n, and the prior parameters α, β, we start with Bayes'. We have

$$\mathbb{P}[\Theta \mid S = s] = \frac{\mathbb{P}[S = s \mid \Theta]\,\mathbb{P}[\Theta]}{\mathbb{P}[S = s]} = \frac{\left(\binom{n}{s}\theta^s(1-\theta)^{n-s}\right)\left(\frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}\right)}{\int_\Theta f_{S=s|\Theta}(\theta) f_\Theta(\theta) d\theta}$$

$$\mathbb{P}[\Theta \mid S = s] = \left(\frac{\cancel{\binom{n}{s}}}{\cancel{\mathrm{B}(\alpha,\beta)}}\right)\left(\frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}}{\int_\Theta \cancel{\binom{n}{s}}\theta^s(1-\theta)^{n-s}\cancel{\left(\frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}\right)} d\theta}\right)$$

$$\mathbb{P}[\Theta \mid S = s] = \frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}}{\int_\Theta \theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1} d\theta}$$

$$\mathbb{P}[\Theta \mid S = s] = \frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}}{\mathrm{B}(s+\alpha, n-s+\beta)\int_\Theta \cancel{\frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}}{\mathrm{B}(s+\alpha, n-s+\beta)}} d\theta} \to 1$$

$$\mathbb{P}[\Theta \mid S = s] = \frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}}{\mathrm{B}(s+\alpha, n-s+\beta)} \implies \boxed{\Theta \mid S = s \sim \text{Beta}(s+\alpha, n-s+\beta)}$$

Next we calculate the $\hat{\Theta}_{\text{mean}}(s; n, \alpha, \beta)$ and $\hat{\Theta}_{\text{MAP}}(s; n, \alpha, \beta)$ by recalling that for $\Theta \sim \text{Beta}(\alpha, \beta)$, we have $\mathbb{E}[\Theta] = \frac{\alpha}{\alpha+\beta}$ and $\text{mode}[\Theta] = \frac{\alpha-1}{\alpha+\beta-2}$. It follows that

$$\boxed{\hat{\Theta}_{\text{mean}}(s; n, \alpha, \beta) = \mathbb{E}[\Theta \mid S = s] = \frac{s+\alpha}{n+\alpha+\beta}} \quad \text{and} \quad \boxed{\hat{\Theta}_{\text{MAP}}(s; n, \alpha, \beta) = \text{mode}[\Theta \mid S = s] = \frac{s+\alpha-1}{n+\alpha+\beta-2}}$$

For convenience, we can rewrite $\hat{\Theta}_{\text{mean}}(S)$ in terms of $\hat{\Theta}_{\text{MLE}}(S)$. We have that

$$\hat{\Theta}_{\text{mean}}(S) = \frac{S+\alpha}{n+\alpha+\beta} = \left(\frac{n}{n+\alpha+\beta}\right)\underbrace{\left(\frac{S}{n}\right)}_{\hat{\Theta}_{\text{MLE}}(S)} + \left(\frac{\alpha+\beta}{n+\alpha+\beta}\right)\underbrace{\left(\frac{\alpha}{\alpha+\beta}\right)}_{\mathbb{E}[\Theta]}$$

$$\boxed{\hat{\Theta}_{\text{mean}}(S) = \left(\frac{n}{n+\alpha+\beta}\right)\hat{\Theta}_{\text{MLE}}(S) + \left(1 - \frac{n}{n+\alpha+\beta}\right)\mathbb{E}[\Theta]} \tag{3}$$

We can also rewrite $\hat{\Theta}_{\text{MAP}}(S)$ in terms of $\hat{\Theta}_{\text{MLE}}(S)$. We have that

$$\hat{\Theta}_{\text{MAP}}(S) = \frac{S+\alpha-1}{n+\alpha+\beta-2} = \left(\frac{n}{n+\alpha+\beta-2}\right)\underbrace{\left(\frac{S}{n}\right)}_{\hat{\Theta}_{\text{MLE}}(S)} + \left(\frac{\alpha+\beta-2}{n+\alpha+\beta-2}\right)\underbrace{\left(\frac{\alpha-1}{\alpha+\beta-2}\right)}_{\text{mode}[\Theta]}$$

$$\boxed{\hat{\Theta}_{\text{MAP}}(S) = \left(\frac{n}{n+\alpha+\beta-2}\right)\hat{\Theta}_{\text{MLE}}(S) + \left(1 - \frac{n}{n+\alpha+\beta-2}\right)\text{mode}[\Theta]} \tag{4}$$

(b) Compute the bias in $\hat{\Theta}_{\text{MLE}}(S)$, $\hat{\Theta}_{\text{mean}}(S)$, $\hat{\Theta}_{\text{MAP}}(S)$ conditional on $\Theta = \theta$. Recall that the bias in an estimator is the expected error, $\mathbb{E}[\hat{\Theta} - \theta]$. In this case, we want $\mathbb{E}_{S|\Theta=\theta}[\hat{\Theta}(S) - \Theta]$.

We first calculate the bias of the maximum likelihood estimator, $\hat{\Theta}_{\text{MLE}}(S) = S/n$. We have

$$\mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MLE}}(S) - \Theta] = \mathop{\mathbb{E}}_{S|\Theta=\theta}[S/n - \Theta] = \frac{\mathbb{E}[S \mid \Theta = \theta]}{n} - \mathbb{E}[\Theta \mid \Theta = \theta] = \frac{n\theta}{n} - \theta = \theta - \theta = 0$$

Since the MLE estimate is unbiased, we will have $\hat{\Theta}_{\text{mean}}(S)$ in terms of $\hat{\Theta}_{\text{MLE}}(S)$ for the bias calculation (Equation (3)). It follows that

$$\begin{aligned}
\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\Theta}_{\text{mean}}(S)-\Theta\right] &= \mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\left(\frac{n}{n+\alpha+\beta}\right)\hat{\Theta}_{\text{MLE}}(S)+\left(1-\frac{n}{n+\alpha+\beta}\right)\mathbb{E}[\Theta]-\Theta\right] \\
\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\Theta}_{\text{mean}}(S)-\Theta\right] &= \left(\frac{n}{n+\alpha+\beta}\right)\theta+\left(1-\frac{n}{n+\alpha+\beta}\right)\mathbb{E}[\Theta]-\theta \\
\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\Theta}_{\text{mean}}(S)-\Theta\right] &= \left(\frac{n}{n+\alpha+\beta}-1\right)\theta+\left(1-\frac{n}{n+\alpha+\beta}\right)\mathbb{E}[\Theta]
\end{aligned}$$

We do the same for $\hat{\Theta}_{\text{MAP}}(S)$ using Equation (4). We have that

$$\begin{aligned}
\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\Theta}_{\text{MAP}}(S)-\Theta\right] &= \mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\left(\frac{n}{n+\alpha+\beta-2}\right)\hat{\Theta}_{\text{MLE}}(S)+\left(1-\frac{n}{n+\alpha+\beta-2}\right)\text{mode}[\Theta]-\Theta\right] \\
\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\Theta}_{\text{MAP}}(S)-\Theta\right] &= \left(\frac{n}{n+\alpha+\beta-2}\right)\theta+\left(1-\frac{n}{n+\alpha+\beta-2}\right)\text{mode}[\Theta]-\theta \\
\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\Theta}_{\text{MAP}}(S)-\Theta\right] &= \left(\frac{n}{n+\alpha+\beta-2}-1\right)\theta+\left(1-\frac{n}{n+\alpha+\beta-2}\right)\text{mode}[\Theta]
\end{aligned}$$

(c) Which estimators are conditionally unbiased given θ? Which estimator is most biased?

In summary, we have

$$\boxed{\mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MLE}}(S)-\Theta]=0} \tag{5}$$

$$\boxed{\mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{mean}}(S)-\Theta]=\left(\frac{n}{n+\alpha+\beta}-1\right)\theta+\left(1-\frac{n}{n+\alpha+\beta}\right)\mathbb{E}[\Theta]} \tag{6}$$

$$\boxed{\mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MAP}}(S)-\Theta]=\left(\frac{n}{n+\alpha+\beta-2}-1\right)\theta+\left(1-\frac{n}{n+\alpha+\beta-2}\right)\text{mode}[\Theta]} \tag{7}$$

The only unbiased estimator is $\hat{\Theta}_{\text{MLE}}(S)$. Which of the other two estimators ($\hat{\Theta}_{\text{mean}}(S)$ and $\hat{\Theta}_{\text{MAP}}(S)$) is more biased depends on α and β however it is generally the case that $\hat{\Theta}_{\text{mean}}(S)$ is more biased than $\hat{\Theta}_{\text{MAP}}(S)$.

Conceptually, we see this by noticing that the biases would be the same if we add one more ficticious success and one more ficticious trial to the $\hat{\Theta}_{\text{MAP}}(S)$ estimator, we recover the $\hat{\Theta}_{\text{mean}}(S)$ estimator (i.e., $\alpha=\alpha+1$ and $\beta=\beta+1$) . Adding the ficticious success and trials increases the weight on the prior and therefore we can conclude that. . .

- $\hat{\Theta}_{\text{mean}}(S)$ estimator is more sensitive to the prior and less sensitive to the likelihood compared to the $\hat{\Theta}_{\text{MAP}}(S)$, and conversely,
- $\hat{\Theta}_{\text{MAP}}(S)$ estimator is more sensitive to the likelihood and less sensitive to the prior compared to the $\hat{\Theta}_{\text{mean}}(S)$.

Since our expectation is over the likelihood distribution $\mathbb{E}_{S|\Theta=\theta}[\cdot]$, the estimator that more closely follows the likelihood is less biased. Therefore, $\hat{\Theta}_{\text{mean}}(S)$ is more biased than $\hat{\Theta}_{\text{MAP}}(S)$ in general.

(d) In what direction are the posterior estimators biased relative to the MLE estimate? Rationalize this effect in terms of the prior.

Since we wrote the biases of $\hat{\Theta}_{\text{mean}}(S)$ and $\hat{\Theta}_{\text{MAP}}(S)$ estimators in terms of the mean and mode of the prior respectively, it becomes clear which direction they are biased. Their biases are summarized in two cases: 1. when $n \gg \alpha+\beta$ and 2. when $n \ll \alpha+\beta$.

Case 1. when $n \gg \alpha+\beta$, i.e., when there is much more information coming from the likelihood than the prior.

The bias of the mean estimator $\hat{\Theta}_{\text{mean}}(S)$: $\quad \frac{n}{n+\alpha+\beta} \xrightarrow{n\gg\alpha+\beta} 1 \implies \mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MAP}}(S)-\Theta] \xrightarrow{n\gg\alpha+\beta} 0$

The bias of the MAP estimator $\hat{\Theta}_{\text{MAP}}(S)$: $\quad \frac{n}{n+\alpha+\beta-2} \xrightarrow{n\gg\alpha+\beta} 1 \implies \mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MAP}}(S)-\Theta] \xrightarrow{n\gg\alpha+\beta} 0$

Case 2. when $n \ll \alpha + \beta$, i.e., when there is much more information coming from the prior than the likelihood.

The bias of the mean estimator $\hat{\Theta}_{\text{mean}}(S)$: $\frac{n}{n+\alpha+\beta} \xrightarrow{n\ll\alpha+\beta} 1 \implies \mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MAP}}(S) - \Theta] \xrightarrow{n\ll\alpha+\beta} \mathbb{E}[\Theta]$

The bias of the MAP estimator $\hat{\Theta}_{\text{MAP}}(S)$: $\frac{n}{n+\alpha+\beta-2} \xrightarrow{n\ll\alpha+\beta} 1 \implies \mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}_{\text{MAP}}(S) - \Theta] \xrightarrow{n\ll\alpha+\beta} \text{mode}[\Theta]$

Therefore, the posterior estimators $\hat{\Theta}_{\text{mean}}(S)$ and $\hat{\Theta}_{\text{MAP}}(S)$ are biased toward $\mathbb{E}[\Theta]$ and mode$[\Theta]$, respectively.

(e) Repeat the same calculation, but now marginalize over Θ as well. That is, compute the joint bias, $\mathbb{E}_{\Theta,S}[\hat{\Theta}(S) - \Theta]$. Which estimators are unbiased with respect to the full data generating process (joint model)?

Note that $\mathbb{E}_{\Theta,S}[\Theta] = \mathbb{E}_{\Theta}[\mathbb{E}_{S|\Theta}[\Theta]] = \mathbb{E}_{\Theta}[\Theta] = \frac{\alpha}{\alpha+\beta}$ and $\mathbb{E}_{\Theta,S}[S] = \mathbb{E}_{\Theta}[\mathbb{E}_{S|\Theta}[S]] = \mathbb{E}_{\Theta}[n\Theta] = \frac{n\alpha}{\alpha+\beta}$. Then the bias of the maximum likelihood estimator, $\hat{\Theta}_{\text{MLE}}(S) = S/n$, is

$$\mathop{\mathbb{E}}_{\Theta,S}[\hat{\Theta}_{\text{MLE}}(S) - \Theta] = \mathop{\mathbb{E}}_{\Theta,S}[\hat{\Theta}_{\text{MLE}}(S)] - \mathop{\mathbb{E}}_{\Theta,S}[\Theta] = \frac{1}{n}\left(\frac{n\alpha}{\alpha+\beta}\right) - \frac{\alpha}{\alpha+\beta} = 0.$$

Next, we calculate the bias of the mean estimator, $\hat{\Theta}_{\text{mean}}(S)$. We have

$$\mathop{\mathbb{E}}_{\Theta,S}[\hat{\Theta}_{\text{mean}}(S) - \Theta] = \frac{\frac{n\alpha}{\alpha+\beta} + \alpha}{n+\alpha+\beta} - \frac{\alpha}{\alpha+\beta} = \frac{n\alpha + \alpha(\alpha+\beta) - \alpha(n+\alpha+\beta)}{(\alpha+\beta)(n+\alpha+\beta)} = 0.$$

Finally, we calculate the bias of the maximum a posteriori estimator, $\hat{\Theta}_{\text{MAP}}(S)$. We have

$$\mathop{\mathbb{E}}_{\Theta,S}[\hat{\Theta}_{\text{MAP}}(S) - \Theta] = \frac{\frac{n\alpha}{\alpha+\beta} + \alpha - 1}{n+\alpha+\beta-2} - \frac{\alpha}{\alpha+\beta} = \frac{n\alpha + (\alpha-1)(\alpha+\beta) - \alpha(n+\alpha+\beta-2)}{(\alpha+\beta)(n+\alpha+\beta-2)} = \frac{\alpha-\beta}{(\alpha+\beta)(n+\alpha+\beta-2)}.$$

Therefore, both the MLE and mean estimators ($\hat{\Theta}_{\text{MLE}}(S)$ and $\hat{\Theta}_{\text{mean}}(S)$) are unbiased whereas the MAP estimator ($\hat{\Theta}_{\text{MAP}}(S)$) is biased.

(f) Interpret your conclusions: (i) suggest a scenario where you would prefer the conditional notion of bias, and a situation where you would prefer the joint notion of bias. (ii) give a reasonable standard for the largest bias you would accept (e.g., the bias should be ... relative to the posterior standard deviation, or the standard deviation in the sampling distribution of the estimators).

(i) Interpreting Conditional vs. Joint Bias.

- **Conditional notion of bias.** We treat θ as fixed but unknown and look at $\mathop{\mathbb{E}}_{S|\Theta=\theta}[\hat{\Theta}(S) - \theta]$. This is the standard frequentist notion of bias: given a true parameter value, does the estimator systematically overshoot or undershoot?
  - *When to prefer:* If you want guarantees that your estimate has no systematic error *for each* possible θ.
- **Joint notion of bias.** We treat Θ itself as random from a prior, so the bias is $\mathop{\mathbb{E}}_{\Theta,S}[\hat{\Theta}(S) - \Theta]$. We average over both Θ and S. This is a more Bayesian notion of bias: how well does the estimator perform *on average* across the entire prior distribution of Θ?
  - *When to prefer:* If you genuinely believe Θ is random and care about average performance over the prior.

(ii) A Reasonable Standard for "Acceptable" Bias.

- A common rule of thumb is that the *magnitude* of bias should be small relative to the relevant scale of uncertainty. Formally, we have

$$\left|\text{Bias}[\hat{\Theta}(S)]\right| < c \times \text{SD}[\hat{\Theta}(S)] \implies \begin{cases} \left|\mathbb{E}_{S|\Theta=\theta}[\hat{\Theta}(S) - \Theta]\right| < c \times \sqrt{\text{Var}_{S|\Theta=\theta}[\hat{\Theta}(S) - \Theta]} & \text{Frequentist} \\ \left|\mathbb{E}_{\Theta,S}[\hat{\Theta}(S) - \Theta]\right| < c \times \sqrt{\text{Var}_{\Theta|S=s}[\hat{\Theta}(S) - \Theta]} & \text{Bayesian} \end{cases}$$

where c is a small constant (e.g., 0.1). Notice that in the frequentist setting, one compares the bias to the standard deviation of the *sampling distribution* of the estimator whereas in the Bayesian setting, one compares to the *posterior* standard deviation of Θ.

For more on these variances of the sampling distribution and the posterior see Problem 8.

Problem 7: Principles for Selecting an Estimator

Given an observation, say $S = s$, we usually don't return the full posterior for an unknown, $\Theta \mid S = s$. Instead, we return inferential summaries, at minimum, the posterior mean, $\hat{\theta}_{\text{mean}}(s) = \mathbb{E}[\Theta \mid S = s]$, or posterior mode, $\hat{\theta}_{\text{MAP}}(s)$. Why adopt these points as summaries?

One way to justify an estimation procedure is to select the optimizer to control properties of the conditional error. Prove each statement below regarding a Bayesian inference problem where S denotes a generic observable that is drawn jointly with an unknown Θ taking values on the real line.

(a) The estimator $\hat{\theta}(s)$ that minimizes the expected square error $L_2(\hat{\theta}(s), \Theta) = (\hat{\theta}(s) - \Theta)^2$, for each possible observed s, is the posterior mean:

$$\hat{\theta}_{\text{mean}}(s) = \underset{\theta}{\text{argmin}} \left\{ \underset{\Theta|S=s}{\mathbb{E}} [L_2(\theta, \Theta)] \right\}. \tag{8}$$

To solve for the $\theta^\star$ that minimizes $\mathbb{E}_{\Theta|S=s}[L_2(\theta, \Theta)]$, we set its derivative with respect to θ to zero. We have that

$$\frac{\partial}{\partial \theta} \left\{ \underset{\Theta|S=s}{\mathbb{E}} [L_2(\theta, \Theta)] \right\} = \underset{\Theta|S=s}{\mathbb{E}} \left[\frac{\partial}{\partial \theta} (\theta - \Theta)^2 \right] = \underset{\Theta|S=s}{\mathbb{E}} [2\theta - 2\Theta] = 0 \quad \Longrightarrow \quad \theta^\star = \mathbb{E}[\Theta \mid S = s] = \hat{\theta}_{\text{mean}}(s) \quad \checkmark$$

(b) Let $t(s)$ denote an arbitrary, real-valued, function of the observable. We will consider t a "test" function. Let $\text{Corr}_{\Theta,S}[t(S), \hat{\theta}(S) - \Theta]$ denote the correlation between the test function value and the estimation error $\hat{\theta}(S) - \Theta$. Here the correlation is evaluated jointly over the unknown and the observable. Note that, if the correlation is nonzero for any test function, then that test function carries information about the estimation error. So, it could be used to try and improve the estimator.

The only estimator whose estimation error has zero correlation with all test functions is, up to a constant, the posterior mean:

$$\text{Corr}_{\Theta,S}[t(S), \hat{\theta}(S) - \Theta] = 0 \quad \text{for all } t \text{ if and only if} \quad \hat{\theta}(\cdot) = \hat{\theta}_{\text{mean}}(\cdot). \tag{9}$$

We start by rewriting the expression $\text{Corr}_{\Theta,S}[t(S), \hat{\theta}(S) - \Theta] = 0$ as

$$\text{Corr}_{\Theta,S}[t(S), \hat{\theta}(S) - \Theta] = \frac{\text{Cov}_{\Theta,S}[t(S), \hat{\theta}(S) - \Theta]}{\sqrt{\text{Var}_{\Theta,S}[t(S)] \, \text{Var}_{\Theta,S}[\hat{\theta}(S) - \Theta]}} = 0 \quad \Longrightarrow \quad \underset{\Theta,S}{\text{Cov}}[t(S), \hat{\theta}(S) - \Theta] = 0.$$

Simplifying further, we have that

$$\underset{\Theta,S}{\text{Cov}}[t(S), \hat{\theta}(S) - \Theta] = 0 \quad \Longrightarrow \quad \underset{\Theta,S}{\mathbb{E}} \left[t(S)(\hat{\theta}(S) - \Theta) \right] - \underset{\Theta,S}{\mathbb{E}} \left[t(S) \right] \underset{\Theta,S}{\mathbb{E}} \left[\hat{\theta}(S) - \Theta \right] = 0.$$

Finally by the law of total expectation, we have

$$\begin{aligned}
&\underset{S}{\mathbb{E}} \left[\underset{\Theta}{\mathbb{E}} \left[t(S)(\hat{\theta}(S) - \Theta) \mid S \right] \right] - \underset{S}{\mathbb{E}} \left[\underset{\Theta}{\mathbb{E}} \left[t(S) \mid S \right] \right] \underset{S}{\mathbb{E}} \left[\underset{\Theta}{\mathbb{E}} \left[\hat{\theta}(S) - \Theta \mid S \right] \right] = 0 \\
&\underset{S}{\mathbb{E}} \left[t(S)(\hat{\theta}(S) - \underset{\Theta}{\mathbb{E}} \left[\Theta \mid S \right]) \right] - \underset{S}{\mathbb{E}} \left[t(S) \right] \underset{S}{\mathbb{E}} \left[\hat{\theta}(S) - \underset{\Theta}{\mathbb{E}} \left[\Theta \mid S \right] \right] = 0
\end{aligned}$$

Therefore, we conclude that the only estimator whose estimation error has zero correlation with all test functions is the posterior mean, i.e.,

$$\boxed{\hat{\theta}(S) = \hat{\Theta}_{\text{mean}}(S)}$$

(c) The estimator $\hat{\theta}(s)$ that minimizes the expected absolute error $L_1(\hat{\theta}(s), \Theta) = \left| \hat{\theta}(s) - \Theta \right|$, for each possible observed s, is the posterior median:

$$\hat{\theta}_{\text{median}}(s) = \text{Median}(\Theta \mid S = s) = \underset{\theta}{\text{argmin}} \left\{ \underset{\Theta|S=s}{\mathbb{E}} [L_1(\theta, \Theta)] \right\}. \tag{10}$$

To solve for the $\theta^\star$ that minimizes $\mathbb{E}_{\Theta|S=s}[L_1(\theta,\Theta)]$, we set its derivative with respect to θ to zero. We have that

$$\frac{\partial}{\partial\theta}\Big\{\mathop{\mathbb{E}}_{\Theta|S=s}[L_1(\theta,\Theta)]\Big\} = \mathop{\mathbb{E}}_{\Theta|S=s}\left[\frac{\partial}{\partial\theta}|\theta-\Theta|\right] = \mathop{\mathbb{E}}_{\Theta|S=s}\big[\text{sign}(\theta-\Theta)\big] = 0 \quad \text{where} \quad \text{sign}(\theta-\Theta) = \begin{cases} +1 & \text{when } \theta-\Theta>0 \\ -1 & \text{when } \theta-\Theta<0 \end{cases}.$$

It follows that

$$\int_0^1 \text{sign}(\theta-\Theta)\, p_{\Theta|S}\, d\Theta = \int_0^\theta p_{\Theta|S}\, d\Theta - \int_\theta^1 p_{\Theta|S}\, d\Theta = 0 \quad \Longrightarrow \quad \int_0^{\theta^\star} p_{\Theta|S}\, d\Theta = \int_{\theta^\star}^1 p_{\Theta|S}\, d\Theta.$$

The expression above shows clearly that $\theta^\star = \text{Median}(\Theta \mid S=s)$ because

$$\underbrace{\int_0^{\theta^\star=\text{Median}(\Theta|S=s)} p_{\Theta|S}\, d\Theta}_{\frac{1}{2}} = \underbrace{\int_{\theta^\star=\text{Median}(\Theta|S=s)}^1 p_{\Theta|S}\, d\Theta}_{\frac{1}{2}}$$

We conclude that

$$\boxed{\hat\theta_{\text{median}}(s) = \text{Median}(\Theta \mid S=s) = \mathop{\text{argmin}}_\theta \Big\{\mathop{\mathbb{E}}_{\Theta|S=s}[L_1(\theta,\Theta)]\Big\}.}$$

Problem 8: Regularization – Reducing Variance

Consider the same joint Θ, S model established in problem 5. A common reason to introduce a prior is to regularize inference. Regularization typically aims to reduce the sampling variance in an estimator by biasing it towards answers that are more likely, a priori.

We've already studied the biasing effects of the prior. In this problem, we will study its role in reducing sample variance relative to Frequentist estimators.

(a) Compute the variance in the estimation error $\hat{\theta}_{\text{MLE}}(S) - \theta$ given a fixed value for θ. That is $\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S) - \theta]$. This variance measures the expected (squared) magnitude of the error due to sampling variability.

We have

$$\underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{MLE}}(S) - \theta] = \underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{MLE}}(S)] = \underset{S|\Theta=\theta}{\text{Var}}\left[\frac{S}{n}\right] = \frac{\text{Var}_{S|\Theta=\theta}[S]}{n^2} = \frac{n\theta(1-\theta)}{n^2} \implies \boxed{\underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{MLE}}(S) - \theta] = \frac{\theta(1-\theta)}{n}.}$$

(b) Compute the expectation over Θ of the variance in the estimation error $\hat{\theta}_{\text{MLE}}(S) - \Theta$ over S. That is, $\mathbb{E}_\Theta[\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S) - \theta]]$. Interpret this quantity as a measure of the accuracy of the maximum likelihood estimator.

Using our result from part (a), we have

$$\underset{\Theta}{\mathbb{E}}[\underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{MLE}}(S) - \theta]] = \underset{\Theta}{\mathbb{E}}\left[\frac{\Theta(1-\Theta)}{n}\right] = \frac{\mathbb{E}_\Theta[\Theta(1-\Theta)]}{n}.$$

It then follows that

$$\underset{\Theta}{\mathbb{E}}[\Theta(1-\Theta)] = \mathbb{E}[\Theta] - \mathbb{E}[\Theta^2] = \mathbb{E}[\Theta] - \text{Var}[\Theta] - \mathbb{E}[\Theta]^2 = \mathbb{E}[\Theta](1 - \mathbb{E}[\Theta]) - \frac{\mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\alpha+\beta+1}$$

$$\underset{\Theta}{\mathbb{E}}[\Theta(1-\Theta)] = \frac{(\alpha+\beta+1)\,\mathbb{E}[\Theta](1-\mathbb{E}[\Theta]) - \mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\alpha+\beta+1} = \frac{(\alpha+\beta)\,\mathbb{E}[\Theta](1-\mathbb{E}[\Theta])}{\alpha+\beta+1} = \frac{\alpha+\beta}{\alpha+\beta+1}\left(\frac{\alpha}{\alpha+\beta}\right)\left(\frac{\beta}{\alpha+\beta}\right)$$

$$\underset{\Theta}{\mathbb{E}}[\Theta(1-\Theta)] = \frac{\alpha\beta}{(\alpha+\beta)(\alpha+\beta+1)} = (\alpha+\beta)\,\text{Var}[\Theta] \tag{11}$$

So we conclude that

$$\boxed{\underset{\Theta}{\mathbb{E}}\left[\underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{MLE}}(S) - \theta]\right] = \frac{\alpha\beta}{n(\alpha+\beta)(\alpha+\beta+1)} = \frac{(\alpha+\beta)\,\text{Var}[\Theta]}{n}.}$$

(c) Repeat this same pair of calculations for the posterior mean estimator, $\hat{\theta}_{\text{mean}}(S)$. Compute the ratios:

$$\frac{\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{mean}}(S) - \theta]}{\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S) - \theta]} \quad \text{and} \quad \frac{\mathbb{E}_\Theta[\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{mean}}(S) - \Theta]]}{\mathbb{E}_\Theta[\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S) - \Theta]]}. \tag{12}$$

Then, use your answer to compare the sampling variability in the estimators. Is the variance/expected variance for the posterior mean estimator always less than the variance/expected variance in the maximum likelihood estimator?

We repeat the same procedure for $\hat{\theta}_{\text{mean}}(S) = \frac{S+\alpha}{n+\alpha+\beta}$. We have

$$\underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{mean}}(S) - \theta] = \underset{S|\Theta=\theta}{\text{Var}}\left[\frac{S+\alpha}{n+\alpha+\beta} - \theta\right] = \underset{S|\Theta=\theta}{\text{Var}}\left[\frac{S}{n+\alpha+\beta} + \frac{\alpha}{n+\alpha+\beta}\right] = \frac{\text{Var}_{S|\Theta=\theta}[S]}{(n+\alpha+\beta)^2} = \frac{n\theta(1-\theta)}{(n+\alpha+\beta)^2}.$$

Similarly, we use Equation (11) to show that

$$\underset{\Theta}{\mathbb{E}}\left[\underset{S|\Theta=\theta}{\text{Var}}[\hat{\theta}_{\text{mean}}(S) - \theta]\right] = \underset{\Theta}{\mathbb{E}}\left[\frac{n\Theta(1-\Theta)}{(n+\alpha+\beta)^2}\right] = \frac{n\,\mathbb{E}_\Theta[\Theta(1-\Theta)]}{(n+\alpha+\beta)^2} \overset{(11)}{=} \frac{n}{(n+\alpha+\beta)^2}\left(\frac{\alpha\beta}{(\alpha+\beta)(\alpha+\beta+1)}\right) = \frac{n(\alpha+\beta)\,\text{Var}[\Theta]}{(n+\alpha+\beta)^2}. \tag{13}$$

Now we calculate the first ratio of Equation (12). We have

$$\frac{\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{mean}}(S) - \theta]}{\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S) - \theta]} = \left(\frac{n\theta(1-\theta)}{(n+\alpha+\beta)^2}\right)\left(\frac{n}{\theta(1-\theta)}\right) = \frac{n^2}{(n+\alpha+\beta)^2} < 1.$$

Similarly, the second ratio of Equation (12) is

$$\frac{\mathbb{E}_\Theta[\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{mean}}(S)-\Theta]]}{\mathbb{E}_\Theta[\text{Var}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S)-\Theta]]} = \left(\frac{n(\alpha+\beta)\,\text{Var}[\Theta]}{(n+\alpha+\beta)^2}\right)\left(\frac{n}{(\alpha+\beta)\,\text{Var}[\Theta]}\right) = \frac{n^2}{(n+\alpha+\beta)^2} < 1.$$

Given both ratios are less than 1, we conclude that the variance of $\hat{\theta}_{\text{mean}}(S)$ is less than $\hat{\theta}_{\text{MLE}}(S)$ which is what we would hope to gain by introducing bias to our estimator.

(d) Use the law of total variance to show that the *expected sampling* variance in the maximum likelihood estimator's error equals the *joint* variance in the maximum likelihood estimator error. That is, show that your answer to (b) equals $\text{Var}_{\Theta,S}[\hat{\theta}(S)-\Theta]$.

Our goal is to show that

$$\underbrace{\mathop{\mathbb{E}}_{\Theta}\left[\mathop{\text{Var}}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S)-\theta]\right]}_{\textit{expected sampling} \text{ variance in the MLE estimator's error}} = \underbrace{\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{MLE}}(S)-\Theta]}_{\textit{joint} \text{ variance in the MLE estimator error}}.$$

By the law of total variance, we write the joint variance of the MLE estimator error as

$$\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{MLE}}(S)-\Theta] = \mathop{\mathbb{E}}_{\Theta}\left[\mathop{\text{Var}}_{S|\Theta=\theta}\left[\hat{\theta}_{\text{MLE}}(S)-\Theta\right]\right] + \mathop{\text{Var}}_{\Theta}\left[\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\theta}_{\text{MLE}}(S)-\Theta\right]\right].$$

Recall that our MLE estimator is unbiased, therefore the second term $\mathbb{E}_{S|\Theta}\left[\hat{\theta}_{\text{MLE}}(S)-\Theta\right]$ vanishes (see Problem 6 part (b)). Therefore, we conclude that

$$\boxed{\mathop{\mathbb{E}}_{\Theta}\left[\mathop{\text{Var}}_{S|\Theta=\theta}[\hat{\theta}_{\text{MLE}}(S)-\theta]\right] = \mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{MLE}}(S)-\Theta]}$$

(e) Use the law of total variance to show that the joint variance in the posterior mean estimation error is:

$$\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{mean}}(S)-\Theta] = \frac{\alpha+\beta}{n+\alpha+\beta}\,\text{Var}[\Theta] \tag{14}$$

Starting with the law of total variance, we have

$$\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{mean}}(S)-\Theta] = \underbrace{\mathop{\mathbb{E}}_{\Theta}\left[\mathop{\text{Var}}_{S|\Theta=\theta}\left[\hat{\theta}_{\text{mean}}(S)-\Theta\right]\right]}_{\text{see Equation (13)}} + \mathop{\text{Var}}_{\Theta}\left[\underbrace{\mathop{\mathbb{E}}_{S|\Theta=\theta}\left[\hat{\theta}_{\text{mean}}(S)-\Theta\right]}_{\text{see Problem 6 part (b)}}\right] = \frac{n(\alpha+\beta)\,\text{Var}[\Theta]}{(n+\alpha+\beta)^2} + \mathop{\text{Var}}_{\Theta}\left[\frac{\alpha-\Theta(\alpha+\beta)}{n+\alpha+\beta}\right]$$

$$\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{mean}}(S)-\Theta] = \left(\frac{n(\alpha+\beta)}{(n+\alpha+\beta)^2} + \frac{(\alpha+\beta)^2}{(n+\alpha+\beta)^2}\right)\text{Var}[\Theta] = \left(\frac{(\alpha+\beta)(n+\alpha+\beta)}{(n+\alpha+\beta)^2}\right)\text{Var}[\Theta] = \frac{\alpha+\beta}{n+\alpha+\beta}\,\text{Var}[\Theta] \quad \checkmark$$

(f) Show that the joint variance in the posterior mean estimator's error is strictly less than the joint variance in the maximum likelihood estimator's error if $\alpha+\beta>1$, and that the ratio of variances is $\frac{n}{n+\alpha+\beta}$.

From our work in the previous parts, we have that

$$\boxed{\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{MLE}}(S)-\Theta] = \frac{\alpha+\beta}{n}\,\text{Var}[\Theta]} \quad \text{and} \quad \boxed{\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{mean}}(S)-\Theta] = \frac{\alpha+\beta}{n+\alpha+\beta}\,\text{Var}[\Theta]}$$

From these results, it follows that

$$\frac{\alpha+\beta}{n} > \frac{\alpha+\beta}{n+\alpha+\beta} \quad \text{for all } \alpha+\beta>1 \quad \Longrightarrow \quad \boxed{\mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{MLE}}(S)-\Theta] > \mathop{\text{Var}}_{\Theta,S}[\hat{\theta}_{\text{mean}}(S)-\Theta] \quad \text{for all } \alpha+\beta>1,}$$

and

$$\frac{\text{Var}_{\Theta,S}[\hat{\theta}_{\text{MLE}}(S)-\Theta]}{\text{Var}_{\Theta,S}[\hat{\theta}_{\text{mean}}(S)-\Theta]} = \left(\frac{(\alpha+\beta)\,\text{Var}[\Theta]}{n+\alpha+\beta}\right)\left(\frac{n\,\text{Var}[\Theta]}{\alpha+\beta}\right) = \frac{n}{n+\alpha+\beta} \quad \checkmark$$

Problem 9: Information

In the Bayesian paradigm, we learn from observations by conditioning on them. Before observing $S = s$, we believe $\Theta \sim p_\Theta$. After observing $S = s$, we believe that $\Theta \sim p_{\Theta|S=s}$. Observations provide information through the likelihood, which pushes the prior to the posterior. In this problem, we will compare the prior and posterior variance after observing $S = s$. As usual, we restrict our attention to the (beta, binomial) model established in question 5.

(a) Write down the expressions for the prior and posterior variance as functions of $\mathbb{E}[\Theta]$, $\mathbb{E}[\Theta|S = s]$, and the parameters α, β, n.

Since both the prior and the posterior follow Beta distributions, we have

$$\boxed{\text{Var}[\Theta] = \frac{\mathbb{E}[\Theta](1 - \mathbb{E}[\Theta])}{\alpha + \beta + 1}} \quad \text{and} \quad \boxed{\text{Var}[\Theta \mid S = s] = \frac{\mathbb{E}[\Theta \mid S = s](1 - \mathbb{E}[\Theta \mid S = s])}{n + \alpha + \beta + 1}}.$$

(b) Use the law of total variance to show that, in expectation, observation always reduces the variance in the unknown. That is:

$$\mathop{\mathbb{E}}_{S}\Big[\text{Var}[\Theta] - \text{Var}[\Theta|S]\Big] > 0. \tag{15}$$

Starting with the law of total variance, we have

$$\text{Var}[\Theta] = \mathop{\mathbb{E}}_{S}\Big[\text{Var}[\Theta|S]\Big] + \mathop{\text{Var}}_{S}\Big[\mathbb{E}[\Theta|S]\Big] \quad \Longrightarrow \quad \mathop{\mathbb{E}}_{S}\Big[\text{Var}[\Theta] - \text{Var}[\Theta|S]\Big] = \mathop{\text{Var}}_{S}\Big[\mathbb{E}[\Theta|S]\Big] > 0 \quad \checkmark$$

as Var$[\cdot]$ is strictly positive (for non-degenerate r.v.'s as we have here).

(c) Show that the expected percent reduction in variance due to observation approaches 1 as n diverges. In particular, show that:

$$\mathop{\mathbb{E}}_{S}\left[\frac{\text{Var}[\Theta] - \text{Var}[\Theta|S]}{\text{Var}[\Theta]}\right] = \frac{n}{n + \alpha + \beta}. \tag{16}$$

We start with the result from the previous part that $\mathbb{E}_S\Big[\text{Var}[\Theta] - \text{Var}[\Theta|S]\Big] = \text{Var}_S\Big[\mathbb{E}[\Theta|S]\Big]$. So we have

$$\mathop{\mathbb{E}}_{S}\left[\frac{\text{Var}[\Theta] - \text{Var}[\Theta|S]}{\text{Var}[\Theta]}\right] = \mathop{\mathbb{E}}_{S}\left[\frac{\text{Var}_S\Big[\mathbb{E}[\Theta|S]\Big]}{\text{Var}[\Theta]}\right] = \frac{\text{Var}_S\Big[\mathbb{E}[\Theta|S]\Big]}{\text{Var}[\Theta]}$$

where we drop the expectation over S as both the numerator and the denominator are deterministic quantities.

Recalling from Problem 5 that $\text{Var}[S] = n(n + \alpha + \beta)\,\text{Var}[\Theta]$, we have

$$\mathop{\text{Var}}_{S}\Big[\mathbb{E}[\Theta|S]\Big] = \mathop{\text{Var}}_{S}\left[\frac{S + \alpha}{n + \alpha + \beta}\right] = \frac{\text{Var}[S]}{(n + \alpha + \beta)^2} = \frac{n(n + \alpha + \beta)\,\text{Var}[\Theta]}{(n + \alpha + \beta)^2} = \frac{n}{n + \alpha + \beta}\,\text{Var}[\Theta]. \tag{17}$$

Therefore, we have

$$\mathop{\mathbb{E}}_{S}\left[\frac{\text{Var}[\Theta] - \text{Var}[\Theta|S]}{\text{Var}[\Theta]}\right] = \frac{\frac{n}{n+\alpha+\beta}\,\text{Var}[\Theta]}{\text{Var}[\Theta]} = \frac{n}{n + \alpha + \beta} \quad \checkmark$$

(d) While observation reduces the variance in expectation, some observations may increase the variance in the unknown. Let's show these are atypical. First, use Markov's inequality to show that:

$$\mathbb{P}(\text{Var}[\Theta|S] > \text{Var}[\Theta]) \leq \frac{\alpha + \beta}{n}. \tag{18}$$

Interpret this inequality. What does it imply as n diverges?

Recall Markov's inequality that for a nonnegative random variable X, we have $\mathbb{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$ for all $a > 0$. Then we have

$$\mathbb{P}[\text{Var}[\Theta|S] > \text{Var}[\Theta]] \leq \frac{\mathbb{E}_S[\text{Var}[\Theta|S]]}{\text{Var}[\Theta]}$$

Again by the law of total variance, we have that

$$\text{Var}[\Theta] = \mathop{\mathbb{E}}_{S}[\text{Var}[\Theta|S]] + \mathop{\text{Var}}_{S}[\mathbb{E}[\Theta|S]] \quad \Longrightarrow \quad \mathop{\mathbb{E}}_{S}[\text{Var}[\Theta|S]] = \text{Var}[\Theta] - \mathop{\text{Var}}_{S}[\mathbb{E}[\Theta|S]] \overset{(17)}{=} \text{Var}[\Theta] - \frac{n}{n+\alpha+\beta}\text{Var}[\Theta].$$

Substituting this result into our Markov's inequality, we have that

$$\mathbb{P}[\text{Var}[\Theta|S] > \text{Var}[\Theta]] \leq \frac{\text{Var}[\Theta] - \frac{n}{n+\alpha+\beta}\text{Var}[\Theta]}{\text{Var}[\Theta]} = 1 - \frac{n}{n+\alpha+\beta} = \frac{\alpha+\beta}{n+\alpha+\beta} \leq \frac{\alpha+\beta}{n} \quad \checkmark$$

Taking the limit of our inequality as $n \to \infty$ leads to

$$\lim_{n\to\infty}\left\{\mathbb{P}[\text{Var}[\Theta|S] > \text{Var}[\Theta]]\right\} = 0$$

which means that as n diverges the probability that our posterior variance is greater than our prior variance goes to zero.

(e) Solve for the region of s given n, α and β such that the posterior variance is greater than the prior variance. Don't worry about simplifying (your answer will be expressed in terms of an ugly quadratic equation). Write a code to illustrate this region for $\alpha = \beta = 0.1$, $\alpha = \beta = 1$, $\alpha = \beta = 3$, and for $\alpha = 8$, $\beta = 2$ as a function of n (sweep n from 1 to 30). I'd suggest plotting n on the vertical axis, and s/n on the horizontal. Attach your plots.

We start by solving for the roots of the expression $\text{Var}[\Theta|S] - \text{Var}[\Theta] = 0$. Let $d = (n+\alpha+\beta)^2(n+\alpha+\beta+1)$. Then we have

$$\frac{(s+\alpha)(n-s+\beta)}{(n+\alpha+\beta)^2(n+\alpha+\beta+1)} - \text{Var}[\Theta] = 0 \quad \Longrightarrow \quad -s^2 + (n+\beta-\alpha)s + \alpha(n+\beta) - d\,\text{Var}[\Theta] = 0.$$

Applying the quadratic formula, it follows that roots are equal to

$$s_i = \frac{-(n+\beta-\alpha) \pm \sqrt{(n+\beta-\alpha)^2 - 4(\alpha(n+\beta) - d\,\text{Var}[\Theta])}}{2} \quad \text{for } i \in \{-,+\}.$$

Since the quadratic $\text{Var}[\Theta|S] - \text{Var}[\Theta] = 0$ is concave down, the region of s such that the posterior variance is greater than the prior variance is given by $\min\{s_-, s_+\} < s < \max\{s_-, s_+\}$.

See Figure 1 for an illustration of this region for a variety of prior parameters.

(f) What do you notice about the observations that increase our uncertainty about the unknown? What prerequisites do we need to place on the prior, and the number of data points collected, such that an observation can increase our uncertainty?

We notice that only when we have a strong prior (e.g., $\alpha = 8$, $\beta = 2$) and limited imformation from our likeliood (e.g., $s < 2$, $n < 5$) do we see this phenomenon where $\text{Var}[\Theta|S] > \text{Var}[\Theta]$.

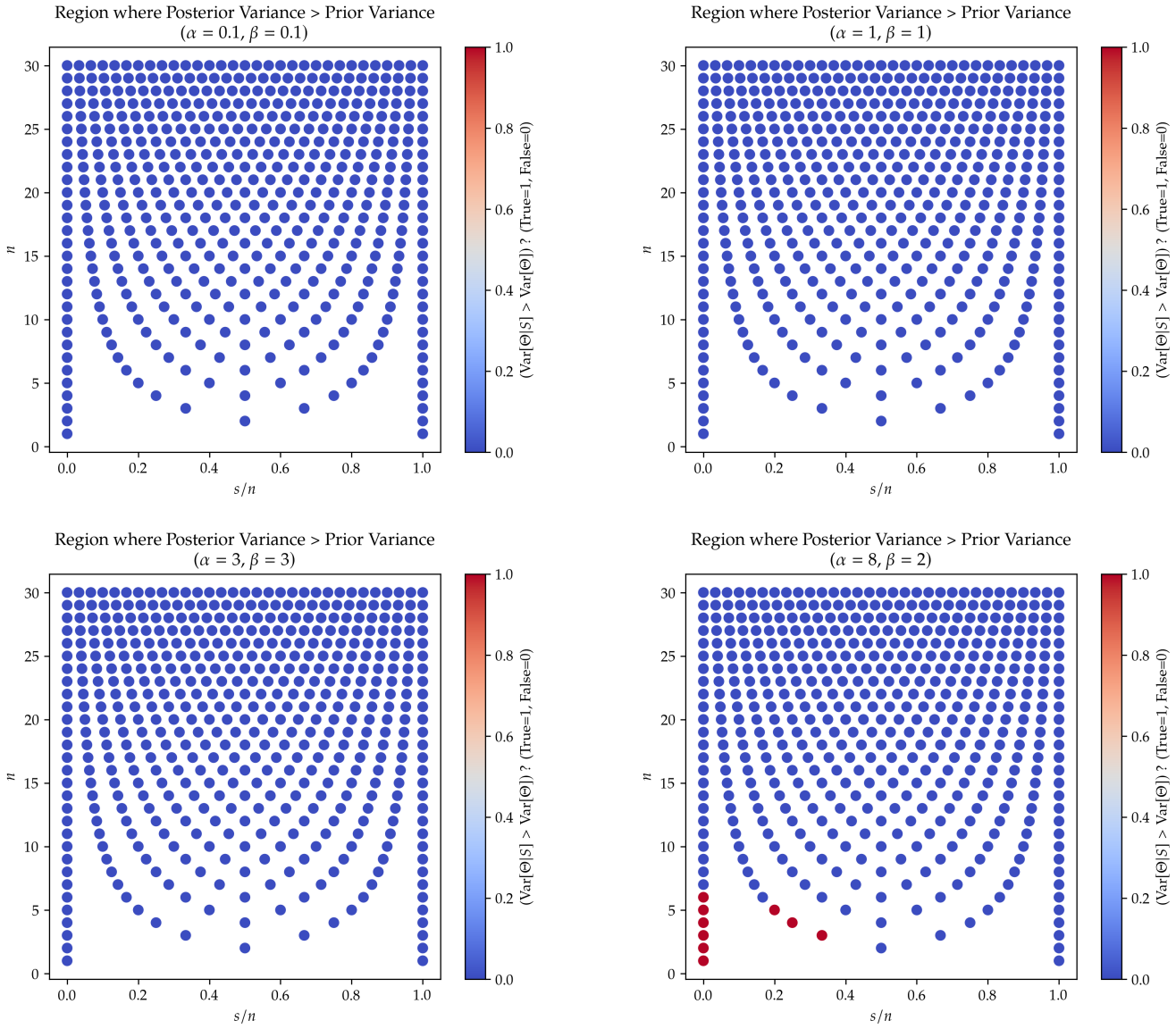


Figure 1: Difference between the posterior variance and the prior variance ($\text{Var}[\Theta|S] - \text{Var}[\Theta]$).

STAT238: Bayesian Statistics February 28th, 2025

Homework 2: Asymptotics, Conjugacy, and Decision Theory

Reece D. Huff

Problems

Problem 1: Asymptotic Abnormality

We argued in class that, under "mild" regularity conditions, the posterior distribution for a continuous-valued unknown should converge to a normal distribution, if we make many, i.i.d. measurements. Anytime someone asserts that their conditions are "mild", remember that, as is true with dining and spicy food, mild is in the eye (tongue?) of the beholder. Here, you will explore one of the common failure modes of the Bayesian central limit theorem. It highlights a key distinction - mathematically special cases may be the general case for specific applications. In our case, for "small sample" problems where the likelihood struggles to uniquely pin down the unknown, even with a very large number of observations.

For this problem, we will consider the following model motivated by tomography problems. In a tomography problem, we shine an intense particle source (the emitter) at an object, and record the shadows it casts when the beam of particles passes through it. For an example of how this might work, see this video on electron microscopes.[1] The three-dimensional distribution of density and absorbtivity in the object is then reconstructed by moving the beam around the object and recording how its shadow changes as we change the direction of the beam.

It is common, especially in microscopy settings, that we want to minimize the amount of radiation we expose our targets to. For this reason, we use as weak a beam as we can, and as sensitive a detector (in essence, the film recording the shadow) as we can. Engineers and physicists are remarkably good at building detectors. Using appropriate amplification methods, it is common to build detectors that can count single particle impacts. For example, if electromagnetic waves (light) is used, then a detector may count single photon strikes. If an electron beam is used, then it might count electron strikes.

Consider the following model. Let $[Y_1, Y_2]$ be the number of observed electron strikes in two adjacent pixels of the detector over the imaging period. Since the beam emitter releases particles continuously, the particles largely don't interact, and the number of incident strikes in any pixel is independent and identically distributed across disjoint time intervals. The number of expected strikes per time period is proportional to the length of the time period, τ. This implies that the counting process must have exponential waiting times, so the number of observed strikes in any time period, in any pixel, is a Poisson random variable with some characteristic rate. Our goal is to estimate these rates. So, consider the likelihood model:

$$\textbf{Likelihood:} \quad Y_1 \sim \text{Poiss}(\lambda_1 \tau), \quad Y_2 \sim \text{Poiss}(\lambda_2 \tau) \tag{19}$$

Here, τ plays the role that n usually does in our asymptotic theory (remember, the sum of two indpendent Poisson random variables is Poisson).

Usually, the goal is to infer the rates λ, to reconstruct the shadow the image should have cast if we shone the beam on it forever. This is, up to first order physical models, the intensity of the original beam reduced by the integral of the density of the tissue along the line passing from the emitter to the detector. Repeating this measurement for many lines through the object (many rotations of the emitter) produces a large linear system that can be solved to resolve the distribution of density in the target object. Like many imaging techniques, this is a technological, mathematical, physical, and engineering marvel. It allows us to peer, with surprising precision, through objects, and to create a 3D reconstruction of those objects!

However, life is never quite that easy. It is extremely common that we don't see $Y_1 \sim \text{Poiss}(\lambda_1)$ and $Y_2 \sim \text{Poiss}(\lambda_2)$ where λ is the vector of rates we would see under our first order physical model. Instead, for nearby pixels, we see:

$$\begin{aligned} \textbf{Blurred Likelihood:} \quad & Y_1 \sim \text{Poiss}(\nu_1 \tau), Y_2 \sim \text{Poiss}(\nu_2 \tau) \\ & \nu_1 = 1/2(1+\epsilon)\lambda_1 + 1/2(1-\epsilon)\lambda_2 \\ & \nu_2 = 1/2(1-\epsilon)\lambda_1 + 1/2(1+\epsilon)\lambda_2 \end{aligned} \tag{20}$$

for some small ϵ. In other words, the ideal rates get blurred. This happens for a variety of reasons. First, scattering particles may make more than one bounce, as passing through the image, or may bounce off at an angle and strike a detector that is out of line with the emitter. Second, since we collect counts over time, the objects within the CT will usually move about over the course of imaging. Under thermal fluctuations, these motions are mostly Gaussian jitter, which blurs the image in nearby pixels. Finally, our beam may not be perfectly focus, introducing blur and aberration.

The fact that ϵ is small introduces an interesting wrinkle into the asymptotic theory we developed during our unit on foundations. Let's see what it does to illustrate the divide between "generic" mathematical and application.

[1] This problem is motivated by a Bayesian statistics problem I am currently working on in Cryo-electron microscopy, where the goal is to reconstruct an image of a cell at atomic resolution.

(a) Let $B(\epsilon)$ denote the matrix such that $\nu = B(\epsilon)\lambda$. Solve for the Fisher information as a function of ϵ, λ, and τ, setting $\tau = 1$.

The Fisher Information is given by $J(\lambda) := \mathbb{E}_{Y\sim f_Y}\left[\nabla^2_\lambda \log(\mathcal{L}(\lambda))\right]$. Here is the Likelihood is given by

$$\mathcal{L}(\lambda) := \mathbb{P}[Y = y \mid \lambda] = \mathbb{P}[Y_1 = y_1, Y_2 = y_2 \mid \lambda] = \mathbb{P}[Y_1 = y_1 \mid \lambda]\,\mathbb{P}[Y_2 = y_2 \mid \lambda] = \left(\frac{e^{-\nu_1\tau}(\nu_1\tau)^{y_1}}{y_1!}\right)\left(\frac{e^{-\nu_2\tau}(\nu_2\tau)^{y_2}}{y_2!}\right)$$

and the Log-Likelihood is given by

$$\log\big(\mathcal{L}(\lambda)\big) := \log\left[\left(\frac{e^{-\nu_1\tau}(\nu_1\tau)^{y_1}}{y_1!}\right)\left(\frac{e^{-\nu_2\tau}(\nu_2\tau)^{y_2}}{y_2!}\right)\right] = y_1\log(\nu_1\tau) + y_2\log(\nu_2\tau) - (\nu_1+\nu_2)\tau - \log\big(y_1!y_2!\big)$$
$$\log\big(\mathcal{L}(\lambda)\big) = y_1\log(\nu_1) + y_2\log(\nu_2) - (\lambda_1+\lambda_2)\tau + (y_1+y_2)\log(\tau) - \log\big(y_1!y_2!\big).$$

We note that all terms that are either linear in or independent of on λ will not contribute to the Fisher Information (as they will not contribute to the Hessian of the Log-Likelihood). Therefore, the only term we care about is $y_k\log(\nu_k)$ for $k \in \{1,2\}$. We have

$$\frac{\partial^2}{\partial\lambda_i\partial\lambda_j}\left\{y_k\log(\nu_k)\right\} = \begin{cases} -\frac{y_k}{4\nu_k^2}(1+\epsilon)^2 & \text{when } i=j=k,\\ -\frac{y_k}{4\nu_k^2}(1-\epsilon)^2 & \text{when } i=j\neq k,\\ -\frac{y_k}{4\nu_k^2}(1+\epsilon)(1-\epsilon) & \text{otherwise.}\end{cases}$$

Now we take the negative expectation of the derivatives recalling that for $X \sim \text{Poiss}(\mu)$, we have $\mathbb{E}[X] = \mu$.

$$-\mathop{\mathbb{E}}_{y_k\sim\text{Poiss}(\nu_k\tau)}\left[\frac{\partial^2}{\partial\lambda_i\partial\lambda_j}\left\{y_k\log(\nu_k)\right\}\right] = \frac{\tau}{4\nu_k}\cdot\begin{cases}(1+\epsilon)^2 & \text{when } i=j=k,\\ (1-\epsilon)^2 & \text{when } i=j\neq k,\\ (1+\epsilon)(1-\epsilon) & \text{otherwise.}\end{cases}$$

Taken together, the Fisher Information is given by

$$J(\lambda) = -\mathop{\mathbb{E}}_{Y\sim\text{Poiss}(\nu\tau)}\left[\nabla^2_\lambda\log\big(\mathcal{L}(\lambda)\big)\right] = -\mathop{\mathbb{E}}_{Y\sim\text{Poiss}(\nu\tau)}\begin{bmatrix}\frac{\partial^2\log\left(\mathcal{L}(\lambda)\right)}{\partial\lambda_1^2} & \frac{\partial^2\log\left(\mathcal{L}(\lambda)\right)}{\partial\lambda_1\partial\lambda_2}\\ \frac{\partial^2\log\left(\mathcal{L}(\lambda)\right)}{\partial\lambda_2\partial\lambda_1} & \frac{\partial^2\log\left(\mathcal{L}(\lambda)\right)}{\partial\lambda_2^2}\end{bmatrix}$$
$$J(\lambda) = \frac{\tau}{4}\begin{bmatrix}\frac{(1+\epsilon)^2}{\nu_1}+\frac{(1-\epsilon)^2}{\nu_2} & \frac{(1+\epsilon)(1-\epsilon)}{\nu_1}+\frac{(1+\epsilon)(1-\epsilon)}{\nu_2}\\ \frac{(1+\epsilon)(1-\epsilon)}{\nu_1}+\frac{(1+\epsilon)(1-\epsilon)}{\nu_2} & \frac{(1-\epsilon)^2}{\nu_1}+\frac{(1+\epsilon)^2}{\nu_2}\end{bmatrix}$$
$$J(\lambda) = \frac{\tau}{4\nu_1}\begin{bmatrix}(1+\epsilon)^2 & 1-\epsilon^2\\ 1-\epsilon^2 & (1-\epsilon)^2\end{bmatrix} + \frac{\tau}{4\nu_2}\begin{bmatrix}(1-\epsilon)^2 & 1-\epsilon^2\\ 1-\epsilon^2 & (1+\epsilon)^2\end{bmatrix}$$

For $\tau = 1$, we have

$$\boxed{J(\lambda) = \frac{1}{2\big[(1+\epsilon)\lambda_1 + (1-\epsilon)\lambda_2\big]}\begin{bmatrix}(1+\epsilon)^2 & 1-\epsilon^2\\ 1-\epsilon^2 & (1-\epsilon)^2\end{bmatrix} + \frac{1}{2\big[(1-\epsilon)\lambda_1 + (1+\epsilon)\lambda_2\big]}\begin{bmatrix}(1-\epsilon)^2 & 1-\epsilon^2\\ 1-\epsilon^2 & (1+\epsilon)^2\end{bmatrix}} \tag{21}$$

(b) Use the Fisher information to write down the asymptotic normal approximation to the posterior. To do this, first solve for the Fisher information with $\tau = 1$, then replace n with τ in the asymptotic normal theorem.[2] *You do not need to solve for the inverse Fisher information explicitly.*

Under the asymptotic normal approximation to the posterior, we have

$$\lambda \mid Y = y \sim \mathcal{N}\left(\hat\lambda_{\text{MLE}}, \big(\tau J(\lambda)\big)^{-1}\Big|_{\lambda=\hat\lambda_{\text{MLE}}}\right)$$

[2] This is valid since, you could imagine subdividing a time period of length τ into τ time periods of length 1 if τ is an integer. Then, the total number of observed counts over all time periods is sufficient, Poisson distributed, and the total number of intervals is τ. Notice that this arguments works whether or not τ is integer-valued since it should work for any choice of time unit, and, since it gives back the same answer no matter the length of the time intervals we use to subdivide τ.

Now we solve for the $\hat{\lambda}_{\text{MLE}}$ where we have

$$\hat{\lambda}_{\text{MLE}} = \underset{\lambda}{\text{argmax}} \Big(\mathcal{L}(\lambda) \Big) = \underset{\lambda}{\text{argmax}} \Big(\log(\mathcal{L}(\lambda)) \Big).$$

It follows that

$$\nabla_\lambda \Big(\log(\mathcal{L}(\lambda)) \Big) = \begin{bmatrix} \frac{\partial}{\partial \lambda_1} \Big\{ y_1 \log(\nu_1) + y_2 \log(\nu_2) - (\lambda_1 + \lambda_2)\tau + f(y, \tau) \Big\} \\ \frac{\partial}{\partial \lambda_2} \Big\{ y_1 \log(\nu_1) + y_2 \log(\nu_2) - (\lambda_1 + \lambda_2)\tau + f(y, \tau) \Big\} \end{bmatrix} = \begin{bmatrix} \frac{y_1}{2\nu_1}(1+\epsilon) + \frac{y_2}{2\nu_2}(1-\epsilon) - \tau \\ \frac{y_1}{2\nu_1}(1-\epsilon) + \frac{y_2}{2\nu_2}(1+\epsilon) - \tau \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Let $a = 1/2(1+\epsilon)$ and $b = 1/2(1-\epsilon)$ such that $a + b = 1$. Then, solving for $\nu_1^\star$ and $\nu_2^\star$, we have

$$\begin{bmatrix} \frac{y_1}{\nu_1} a + \frac{y_2}{\nu_2} b \\ \frac{y_1}{\nu_1} b + \frac{y_2}{\nu_2} a \end{bmatrix} = \begin{bmatrix} \tau \\ \tau \end{bmatrix} \implies \begin{matrix} (a+b)\frac{y_1}{\nu_1} + (a+b)\frac{y_2}{\nu_2} = \frac{y_1}{\nu_1} + \frac{y_2}{\nu_2} = 2\tau \\ (a-b)\frac{y_1}{\nu_1} + (b-a)\frac{y_2}{\nu_2} = \epsilon \frac{y_1}{\nu_1} - \epsilon \frac{y_2}{\nu_2} = 0 \end{matrix} \implies \begin{matrix} \frac{y_1}{\nu_1} + \frac{y_2}{\nu_2} = 2\tau \\ \frac{y_1}{\nu_1} - \frac{y_2}{\nu_2} = 0 \end{matrix} \implies \begin{matrix} \nu_1^\star = \frac{y_1}{\tau} \\ \nu_2^\star = \frac{y_2}{\tau} \end{matrix}$$

Now solving for $\lambda_1^\star$ and $\lambda_2^\star$, we have

$$\begin{aligned} \nu_1 &= 1/2(1+\epsilon)\lambda_1 + 1/2(1-\epsilon)\lambda_2 \\ \nu_2 &= 1/2(1-\epsilon)\lambda_1 + 1/2(1+\epsilon)\lambda_2 \end{aligned}$$

$$\begin{bmatrix} a\lambda_1 + b\lambda_2 \\ b\lambda_1 + a\lambda_2 \end{bmatrix} = \begin{bmatrix} \nu_1^\star \\ \nu_2^\star \end{bmatrix} \implies \begin{matrix} \lambda_1 + \lambda_2 = \frac{y_1 + y_2}{\tau} \\ \lambda_1 - \lambda_2 = \frac{y_1 - y_2}{\epsilon\tau} \end{matrix} \implies \begin{matrix} \lambda_1^\star = \frac{1/2(1+\epsilon)y_1 - 1/2(1-\epsilon)y_2}{\epsilon\tau} \\ \lambda_2^\star = \frac{1/2(1+\epsilon)y_2 - 1/2(1-\epsilon)y_1}{\epsilon\tau} \end{matrix}$$

Therefore, the MLE estimate for λ is

$$\boxed{\hat{\lambda}_{\text{MLE}} = \frac{1}{2\epsilon\tau} \begin{bmatrix} (1+\epsilon)y_1 - (1-\epsilon)y_2 \\ (1+\epsilon)y_2 - (1-\epsilon)y_1 \end{bmatrix}} \tag{22}$$

Taking this result and our result from part (i), we have that the asymptotic normal approximation to the posterior is

$$\boxed{\lambda \mid Y = y \sim \mathcal{N}\left(\hat{\lambda}_{\text{MLE}}, \big(\tau J(\lambda)\big)^{-1} \Big|_{\lambda = \hat{\lambda}_{\text{MLE}}} \right) \quad \text{where } \hat{\lambda}_{\text{MLE}} \text{ and } J(\lambda) \text{ are provided in Equations (21) and (22), respectively.}}$$

(c) Find the eigenvalue decomposition of the covariance of the asymptotic normal approximation to the posterior, assuming $\lambda_1 = \lambda_2$. Identify the eigenvector corresponding to the largest eigenvalue, and the standard deviation along that direction. Identify the eigenvector corresponding to the smallest eigenvalue, and the standard deviation along that direction. Express your answers as functions of ϵ. *It will help to group the entries of the Fisher information into pairs of matching terms, and to find the eigenvalue decomposition for the generic form before plugging in. Your answer here should be simple. If it isn't, back up.*

We note that when $\lambda = \lambda_1 = \lambda_2$, we have

$$\nu_1 = 1/2(1+\epsilon)\lambda + 1/2(1-\epsilon)\lambda = \lambda \quad \text{and} \quad \nu_2 = 1/2(1-\epsilon)\lambda + 1/2(1+\epsilon)\lambda = \lambda$$

It follows that the Fisher Information (for $\tau = 1$) is given by

$$J(\lambda) = \frac{1}{4\lambda} \begin{bmatrix} (1+\epsilon)^2 & 1-\epsilon^2 \\ 1-\epsilon^2 & (1-\epsilon)^2 \end{bmatrix} + \frac{1}{4\lambda} \begin{bmatrix} (1-\epsilon)^2 & 1-\epsilon^2 \\ 1-\epsilon^2 & (1+\epsilon)^2 \end{bmatrix} = \frac{1}{4\lambda} \begin{bmatrix} 2+2\epsilon^2 & 2-2\epsilon^2 \\ 2-2\epsilon^2 & 2+2\epsilon^2 \end{bmatrix} = \frac{1}{2\lambda} \left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \epsilon^2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right)$$

Letting $\alpha = \frac{1}{2\lambda}$ and $\beta = \frac{\epsilon^2}{2\lambda}$, we solve for the eigenvalues of $J(\lambda)$ by solving for the roots of the characteristic equation,

$$\det\big(J(\lambda) - \gamma I\big) = 0 \implies \det\left(\begin{bmatrix} \alpha+\beta & \alpha-\beta \\ \alpha-\beta & \alpha+\beta \end{bmatrix} - \gamma \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = \det\left(\begin{bmatrix} \alpha+\beta-\gamma & \alpha-\beta \\ \alpha-\beta & \alpha+\beta-\gamma \end{bmatrix} \right) = (\alpha+\beta-\gamma)^2 - (\alpha-\beta)^2 = 0.$$

It follows that

$$(\alpha+\beta-\gamma)^2-(\alpha-\beta)^2=\gamma^2-2(\alpha+\beta)\gamma+(\alpha+\beta)^2-(\alpha-\beta)^2=\gamma^2-2(\alpha+\beta)\gamma+4\alpha\beta=(\gamma-2\alpha)(\gamma-2\beta)=0$$

which implies that the eigenvalues of $J(\lambda)$ are $\frac{1}{\lambda}$ and $\frac{\epsilon^2}{\lambda}$. The eigenvectors corresponding to these eigenvalues are

$$\begin{bmatrix}\alpha+\beta & \alpha-\beta\\ \alpha-\beta & \alpha+\beta\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}=2\alpha\begin{bmatrix}u_1\\u_2\end{bmatrix} \qquad \begin{bmatrix}\alpha+\beta & \alpha-\beta\\ \alpha-\beta & \alpha+\beta\end{bmatrix}\begin{bmatrix}v_1\\v_2\end{bmatrix}=2\beta\begin{bmatrix}v_1\\v_2\end{bmatrix}$$

$$\begin{bmatrix}-\alpha+\beta & \alpha-\beta\\ \alpha-\beta & -\alpha+\beta\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix} \qquad \begin{bmatrix}\alpha-\beta & \alpha-\beta\\ \alpha-\beta & \alpha-\beta\end{bmatrix}\begin{bmatrix}v_1\\v_2\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$$

$$\begin{bmatrix}u_1\\u_2\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \qquad \begin{bmatrix}v_1\\v_2\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$

Therefore, the eigendecomposition of $J(\lambda)$ is given by

$$J(\lambda)=D\Lambda D^T=\begin{bmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}\begin{bmatrix}\frac{1}{\lambda} & 0\\ 0 & \frac{\epsilon^2}{\lambda}\end{bmatrix}\begin{bmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}=\frac{1}{2\lambda}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1 & 0\\0 & \epsilon^2\end{bmatrix}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}.$$

Taking the inverse of $J(\lambda)$, we have

$$J(\lambda)^{-1}=\left(D\Lambda D^\top\right)^{-1}=D\Lambda^{-1}D^\top=\begin{bmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}\begin{bmatrix}\lambda & 0\\ 0 & \frac{\lambda}{\epsilon^2}\end{bmatrix}\begin{bmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}=\frac{\lambda}{2}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1 & 0\\0 & \frac{1}{\epsilon^2}\end{bmatrix}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}. \tag{23}$$

Taken together, we have that the...

- the largest eigenvalue of the covariance of the asymptotic normal approximation to the posterior is $\frac{\lambda}{\epsilon^2}$ and the standard deviation along that direction is $\frac{\sqrt{\lambda}}{\epsilon}$. The corresponding eigenvector is $\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$.
- the smallest eigenvalue of the covariance of the asymptotic normal approximation to the posterior is λ and the standard deviation along that direction is $\sqrt{\lambda}$. The corresponding eigenvector is $\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$.

(d) What happens as ϵ approaches zero, assuming $\lambda_1=\lambda_2$? Identify the direction of maximal uncertainty in λ and the proposed standard deviation along that direction. Why must the effect you observe be true?[3] Would this effect change if $\lambda_1\neq\lambda_2$?

As $\epsilon\to 0$, the normal approximation has maximal uncertainty along the direction $\lambda=\begin{bmatrix}1\\-1\end{bmatrix}$ with an infinite standard deviation (i.e., flat uninformative improper prior between $(-\infty,\infty)$). This suggests that as $\epsilon\to 0$, any pair where $\lambda_1=-\lambda_2$ are equally probable. This makes sense as we note that

$$\begin{bmatrix}\nu_1\\\nu_2\end{bmatrix}=\frac{1}{2}\begin{bmatrix}1+\epsilon & 1-\epsilon\\1-\epsilon & 1+\epsilon\end{bmatrix}\begin{bmatrix}\lambda_1\\\lambda_2\end{bmatrix}\quad\xrightarrow{\epsilon\to 0}\quad\begin{bmatrix}\nu_1\\\nu_2\end{bmatrix}=\frac{1}{2}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}\lambda_1\\\lambda_2\end{bmatrix}\quad\Longrightarrow\quad\text{the system is not invertible}$$

Given the system is not invertible, we have that the posterior is uninformative and flat. This has to be true as the system is not invertible and the data does not provide any information about the parameters. This effect would not change if $\lambda_1\neq\lambda_2$ as the system would still be non-invertible.

(e) Derive the form (up to normalization) for the posterior distribution of λ given an observation $Y=y$ using independent, identical Gamma priors on the rates λ_1 and λ_2. You do not need to simplify your answer. You will use it numerically in parts (f)–(i).

Our likelihood is given by $\mathbb{P}[Y_i=y_i\mid\lambda]=\frac{e^{-\nu_i\tau}(\nu_i\tau)^{y_i}}{y_i!}$ for $i\in\{1,2\}$ and our prior $\lambda_i\overset{\text{i.i.d.}}{\sim}\text{Gamma}(\alpha,\beta)$ has a density of $f_{\lambda_i}(\lambda_i)=\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda_i^{\alpha-1}e^{-\beta\lambda_i}$ for shape $\alpha>0$, rate $\beta>0$, and $i\in\{1,2\}$. It then follows that the posterior is proportional to

$$\mathbb{P}[\lambda\mid Y=y]\propto\text{Lik}(Y=y\mid\lambda)\,\text{Prior}(\lambda)=\mathbb{P}[Y_1=y_1,Y_2=y_2\mid\lambda]\,\mathbb{P}[\lambda_1,\lambda_2]=\mathbb{P}[Y_1=y_1\mid\lambda]\,\mathbb{P}[Y_2=y_2\mid\lambda]\,\mathbb{P}[\lambda_1]\,\mathbb{P}[\lambda_2]$$

$$\mathbb{P}[\lambda\mid Y=y]\propto\left(\frac{e^{-\nu_1\tau}(\nu_1\tau)^{y_1}}{y_1!}\right)\left(\frac{e^{-\nu_2\tau}(\nu_2\tau)^{y_2}}{y_2!}\right)\left(\lambda_1^{\alpha-1}e^{-\beta\lambda_1}\right)\left(\lambda_2^{\alpha-1}e^{-\beta\lambda_2}\right)$$

[3] Think about the linear system defining ν as a function of λ. Is it invertible when $\epsilon=0$?

$$\mathbb{P}[\lambda \mid Y = y] \propto \left(\frac{(\nu_1\tau)^{y_1}(\nu_2\tau)^{y_2}(\lambda_1\lambda_2)^{\alpha-1}}{y_1!y_2!}\right)\exp\big(-\tau(\nu_1+\nu_2)-\beta(\lambda_1+\lambda_2)\big)$$

$$\boxed{\mathbb{P}[\lambda \mid Y = y] \overset{4}{\propto} \left(\frac{(\nu_1\tau)^{y_1}(\nu_2\tau)^{y_2}(\lambda_1\lambda_2)^{\alpha-1}}{y_1!y_2!}\right)\exp\big(-(\tau+\beta)(\lambda_1+\lambda_2)\big)}$$

(f) Write a code that can plot the normal approximation to the posterior and the true posterior over $\lambda_1 \geq 0$, $\lambda_2 \geq 0$. You should display the densities as colored surfaces whose colors indicate the height of the surfaces. You should be able to borrow much of this code from Lab 3.

The code is provided in Python Code for Problem 1.

(g) Write a code that plots the marginal distributions of these densities along the directions of maximal and minimal variance (w.r.t. the normal approximation).

In order to derive the marginal of the posterior distribution along the direction $\begin{bmatrix}1 & 1\end{bmatrix}^\top$ and $\begin{bmatrix}1 & -1\end{bmatrix}^\top$ that passes through the point $\lambda^\star = \begin{bmatrix}\lambda_1^\star & \lambda_2^\star\end{bmatrix}^\top$, we let

$$\mathbb{P}[\lambda \mid Y = y] \propto \left(\frac{(\nu_1\tau)^{y_1}(\nu_2\tau)^{y_2}(\lambda_1\lambda_2)^{\alpha-1}}{y_1!y_2!}\right)\exp\big(-(\tau+\beta)(\lambda_1+\lambda_2)\big) := f_{\lambda_1,\lambda_2}(\lambda_1,\lambda_2).$$

It then follows that we define the linear transformation $\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix} := \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}s\\ t\end{bmatrix} + \begin{bmatrix}\lambda_1^\star\\ \lambda_2^\star\end{bmatrix}$ such that the marginals are given by

$$f_S(s) \propto \int_{-\infty}^{\infty} f_{\lambda_1,\lambda_2}\big(\lambda_1^\star + s + t, \lambda_2^\star + s - t\big)dt \quad \text{and} \quad f_T(t) \propto \int_{-\infty}^{\infty} f_{\lambda_1,\lambda_2}\big(\lambda_1^\star + s + t, \lambda_2^\star + s - t\big)ds.$$

Here, the marginal distributions $f_S(s)$ and $f_T(t)$ represents the distributions along the directions $\begin{bmatrix}1 & 1\end{bmatrix}^\top$ and $\begin{bmatrix}1 & -1\end{bmatrix}^\top$, respectively, both of which pass through the point $\lambda^\star = \begin{bmatrix}\lambda_1^\star & \lambda_2^\star\end{bmatrix}^\top$.[5] We solve these integrals numerically. Next the normal approximation are straight forward to infer from our eigendecomposition decomposition in Equation (23).

The code is provided in Python Code for Problem 1.

(h) Plot the results for $\lambda = [0.2, 0.25]$ and $y = \text{round}(\tau\lambda)$.[6] Use a Gamma prior with scale parameter 0.15 and shape parameter 2. Make plots for $\epsilon = 0.5$, 0.1, and 0.01. For each ϵ make your initial plot using $\tau = 10$. Then, try increasing τ until the posterior normal approximation is decent.

See Figure 2 for the results.

(i) Comment on how close the posterior is to its normal approximation in each case. Comment on how the match breaks down as ϵ decreases, and how large you needed to make τ in each case. Comment on which marginal distribution best illustrates the failure of the asymptotic theory, and whether the posterior does appear normal along any marginal.

In Figure 2, we see that the posterior is very close to its normal approximation when $\epsilon = 0.5$. As ϵ decreases, the match breaks down and we need to make τ larger to get a decent approximation. The marginal distribution along the direction of maximal variance best illustrates the failure of the asymptotic theory. As ϵ decreases, the posterior does not appear normal along the marginal in the direction $\begin{bmatrix}-1 & 1\end{bmatrix}^\top$ as the normal approximation breaks down. The posterior remain normal looking along the marginal in the direction $\begin{bmatrix}1 & 1\end{bmatrix}^\top$.

Outro: Always beware an asymptotic guarantee. Whenever you are given an asymptotic guarantee you should ask, but how far in to the limit do I need to be before this approximation is any good? It is often true that the asymptotic guarantees hold for all but some special cases, but the associated approximations fail spectacularly away from the limit when we are close to the special case. This moral will come back to haunt us when we build samplers.

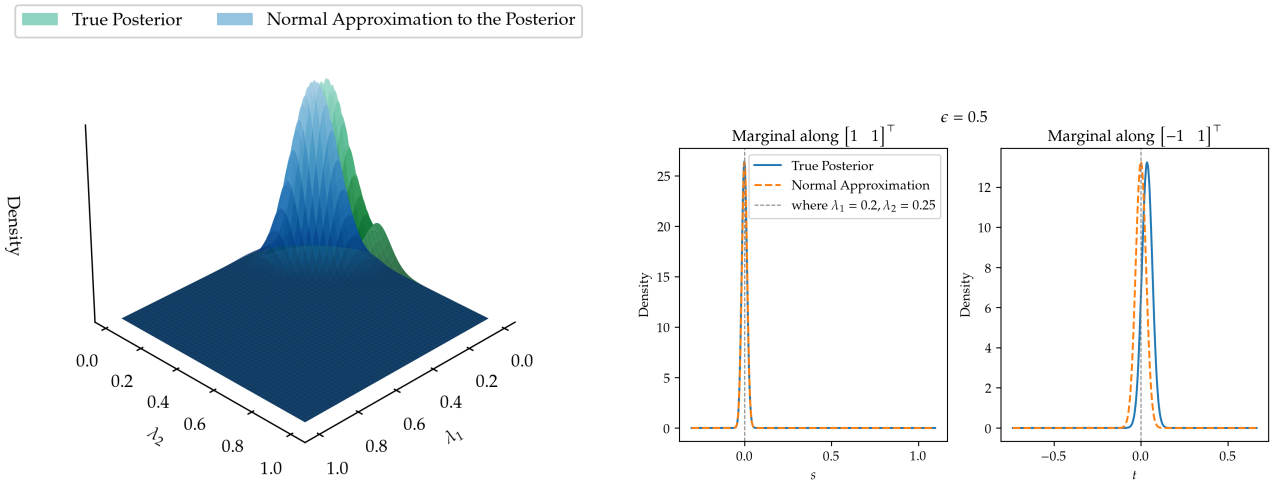
[4] We note that $\nu_1 + \nu_2 = \left(\frac{1}{2}(1+\epsilon)\lambda_1 + \frac{1}{2}(1-\epsilon)\lambda_2\right) + \left(\frac{1}{2}(1-\epsilon)\lambda_1 + \frac{1}{2}(1+\epsilon)\lambda_2\right) = \lambda_1 + \lambda_2$

[5] Also, note that technically we should be including the Jacobian in the transformation, i.e., for any linear transformation of a joint over random variables X and Y, $\begin{bmatrix}X\\ Y\end{bmatrix} := \begin{bmatrix}T_1(U,V)\\ T_2(U,V)\end{bmatrix} := \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}\begin{bmatrix}U\\ V\end{bmatrix} + \begin{bmatrix}c_1\\ c_2\end{bmatrix} = \mathbf{A}\begin{bmatrix}U & V\end{bmatrix}^\top + \mathbf{c}$ is given by

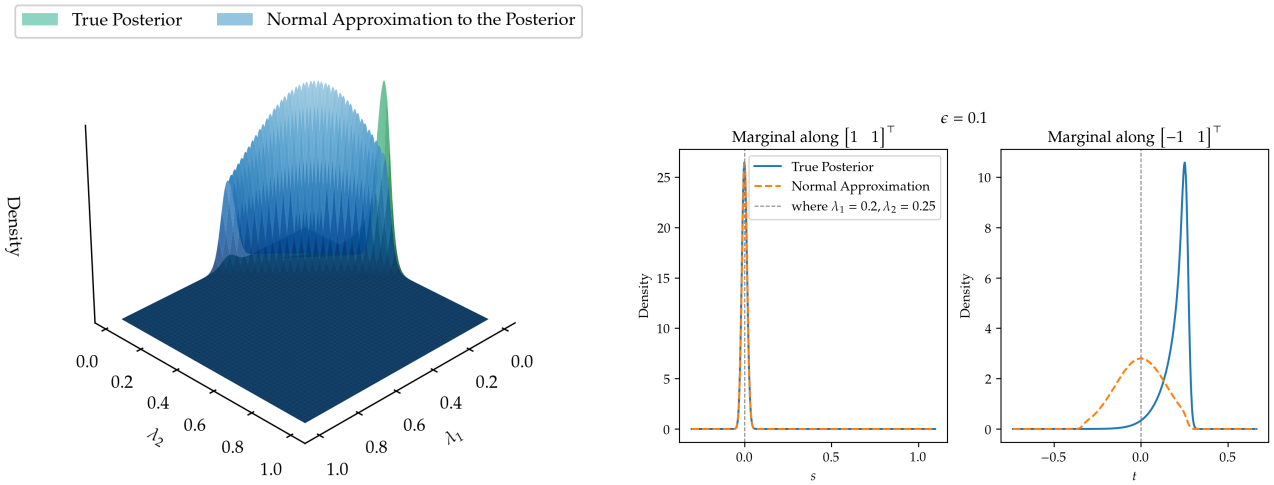
$$f_{U,V}(u,v) := f_{X,Y}\Big(T_1(u,v), T_2(u,v)\Big)\big|\det(A)\big| = f_{X,Y}\Big(A_{11}u + A_{12}v + c_1, A_{21}u + A_{22}v + c_2\Big)\left|\det\left(\begin{bmatrix}\frac{\partial A_{11}u}{\partial u} & \frac{\partial A_{12}v}{\partial v}\\ \frac{\partial A_{21}u}{\partial u} & \frac{\partial A_{22}v}{\partial v}\end{bmatrix}\right)\right|$$

[6] This is the best measurement we could hope for, in that it will lead to the posterior that it centered closest to the truth.

$\tau = 100,\ \epsilon = 0.5,\ \lambda^\star = \begin{bmatrix}0.20\\0.25\end{bmatrix},\ \Sigma = \left(\tau J(\lambda^\star)\right)^{-1} = \begin{bmatrix}0.005 & -0.003\\-0.003 & 0.006\end{bmatrix}$



$\tau = 300,\ \epsilon = 0.1,\ \lambda^\star = \begin{bmatrix}0.20\\0.25\end{bmatrix},\ \Sigma = \left(\tau J(\lambda^\star)\right)^{-1} = \begin{bmatrix}0.038 & -0.037\\-0.037 & 0.038\end{bmatrix}$



$\tau = 500,\ \epsilon = 0.01,\ \lambda^\star = \begin{bmatrix}0.20\\0.25\end{bmatrix},\ \Sigma = \left(\tau J(\lambda^\star)\right)^{-1} = \begin{bmatrix}2.250 & -2.250\\-2.250 & 2.250\end{bmatrix}$

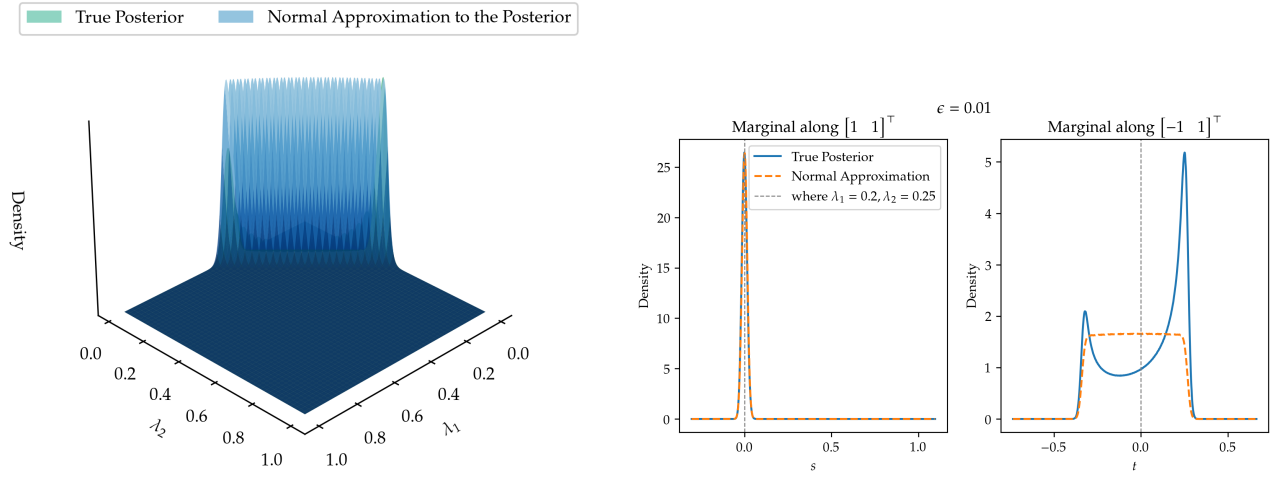


FIGURE 2: Posterior and Marginal Distributions for $\epsilon = 0.5$, 0.1, and 0.01

Python Code for Problem 1

```python
def true_posterior(pos, epsilon, tau, y_1, y_2, alpha, beta, normalize=True, **kwargs):
    L_1 = pos[:, :, 0]
    L_2 = pos[:, :, 1]

    # Compute nu_1 and nu_2 as before
    nu_1 = 0.5 * (1 + epsilon) * L_1 + 0.5 * (1 - epsilon) * L_2
    nu_2 = 0.5 * (1 - epsilon) * L_1 + 0.5 * (1 + epsilon) * L_2

    # Compute the log of each term to avoid large intermediate values
    # Use gammaln(y+1) as a robust alternative to np.log(math.factorial(y))
    log_term1 = y_1 * np.log(nu_1 * tau) + y_2 * np.log(nu_2 * tau)
    log_term2 = (alpha - 1) * np.log(L_1 * L_2)
    log_term3 = - (tau + beta) * (L_1 + L_2)
    log_term4 = gammaln(y_1 + 1) + gammaln(y_2 + 1)

    # Combine the log terms
    log_posterior = log_term1 + log_term2 + log_term3 - log_term4

    # Normalize in log-space for numerical stability
    if normalize:
        log_posterior -= np.max(log_posterior)

    # Exponentiate to get back to the original scale
    output = np.exp(log_posterior)
    return output

def mle(epsilon, tau, y_1, y_2, **kwargs):
    return (1/(2*epsilon*tau)) * np.array([
        (1+epsilon)*y_1-(1-epsilon)*y_2,
        (1+epsilon)*y_2-(1-epsilon)*y_1,
        ])

def fisher_information(lambda_1, lambda_2, epsilon, **kwargs):
    nu_1 = (1/2)*(1+epsilon)*lambda_1 + (1/2)*(1-epsilon)*lambda_2
    nu_2 = (1/2)*(1-epsilon)*lambda_1 + (1/2)*(1+epsilon)*lambda_2
    mat1 = np.array([
                    [(1+epsilon)**2,  1-epsilon **2],
                    [ 1-epsilon **2, (1-epsilon)**2]
                    ])
    mat2 = np.array([
                    [(1-epsilon)**2,  1-epsilon **2],
                    [ 1-epsilon **2, (1+epsilon)**2]
                    ])
    return (1/(4*nu_1))*mat1 + (1/(4*nu_2))*mat2

def get_normal_approximation(lambda_1, lambda_2, epsilon, tau, y_1, y_2, use_MLE=False, **kwargs):
    mu        = mle(epsilon, tau, y_1, y_2) if use_MLE else np.array([lambda_1,lambda_2])
    J         = fisher_information(mu[0], mu[1], epsilon)
    sigma     = (1/tau)*np.linalg.pinv(J)
    sigma_inv = tau*J
    return mu, sigma, sigma_inv

# function that can plot the normal approximation to the posterior and the true posterior over $\lambda_1
 ↪  \geq 0$, $\lambda_2 \geq 0$
def plot_true_and_normal_approximation(params: dict,
    filename: str = None,
    title: str = None,
    color_schemes: list = ['BuGn', 'Blues'],
    grid_points: int = 200,
    xlim: tuple = (0.01, 1),
    ylim: tuple = (0.01, 1),
) -> None:
```

```
    # Create the lambda meshgrid
    x = np.linspace(xlim[0], xlim[1], grid_points)
    y = np.linspace(ylim[0], ylim[1], grid_points)
    X, Y = np.meshgrid(x, y)
    pos = np.empty(X.shape + (2,))
    pos[:, :, 0] = X
    pos[:, :, 1] = Y

    # Compute the true posterior surface
    true_Z = true_posterior(pos, **params)

    # Compute the normal approximation
    mu, Sigma, Sigma_inv = get_normal_approximation(**params)
    diff = pos - mu
    approx_Z = np.exp(-0.5 * np.einsum('...i,ij,...j', diff, Sigma_inv, diff))
    if params.get('normalize', False):
        approx_Z /= np.max(approx_Z)

    # Create a figure and a 3D axes
    fig = plt.figure()
    ax  = fig.add_subplot(111, projection='3d')
    elev = 30
    azim = 45
    ax.view_init(elev=elev, azim=azim)
    ax.grid(False)
    ax.xaxis.pane.fill = False
    ax.yaxis.pane.fill = False
    ax.zaxis.pane.fill = False
    ax.xaxis.pane.set_edgecolor('none')
    ax.yaxis.pane.set_edgecolor('none')
    ax.zaxis.pane.set_edgecolor('none')

    # Prepare colormaps (reverse the given colormaps)
    first_cmap  = plt.get_cmap(color_schemes[0] + '_r')
    second_cmap = plt.get_cmap(color_schemes[1] + '_r')

    # Plot the true posterior surface (with slight transparency)
    ax.plot_surface(X, Y, true_Z, cmap=first_cmap,
                            alpha=0.8, edgecolor='none', vmin=0, vmax=1.5)
    # Plot the normal approximation surface
    ax.plot_surface(X, Y, approx_Z, cmap=second_cmap,
                            alpha=0.8, edgecolor='none', vmin=0, vmax=1.5)

    # Create proxy artists for legend since plot_surface doesn't support labels directly
    proxy1 = Patch(facecolor=first_cmap(0.6), edgecolor='none', label='True Posterior')
    proxy2 = Patch(facecolor=second_cmap(0.6), edgecolor='none', label='Normal Approximation to the
    ↪  Posterior')
    ax.legend(handles=[proxy1, proxy2], loc='upper center', framealpha=0.9, ncol=2)

    # Build a custom title with LaTeX formatting
    custom_title = [f"$",
                    f"\\tau={params['tau']}, \enspace " ,
                    f"\epsilon={params['epsilon']}, \enspace " ,
                    # f"p(\lambda ; \mu, \Sigma), \enspace " ,
                    f"\lambda^{{\star}} = \\begin{{bmatrix}}{mu[0]:0.2f} \\\ {mu[1]:0.2f} \end{{bmatrix}},
                    ↪  \enspace " ,
                    f"\Sigma= \\big(\\tau J(\lambda^{{\star}})\\big)^{{-1}} = \\begin{{bmatrix}}
                    ↪  {Sigma[0,0]:0.3f} & {Sigma[0,1]:0.3f} \\\ {Sigma[1,0]:0.3f} & {Sigma[1,1]:0.3f}
                    ↪  \end{{bmatrix}}",
                f"$"]
    if title:
        custom_title.insert(1, r"\text{%s}: \quad " % title)
    ax.set_title(''.join(custom_title), pad=20)
```

```python

    # Set the axis labels
    ax.set_xlabel(r"$\lambda_1$")
    ax.set_ylabel(r"$\lambda_2$")
    ax.set_zlabel("Density")
    ax.set_zticks([])

    if filename:
        plt.savefig(os.path.join(FIG_DIR, filename), dpi=300, bbox_inches='tight')
    plt.tight_layout()
    plt.show()

def marginal_along_direction(X, Y, F, v1, v2, p1, p2, ns=1000, nt=1000):
    # Create a unit vector along the given direction.
    u = np.array([v1, v2])
    norm = np.linalg.norm(u)
    if norm == 0:
        raise ValueError("Direction vector (v1, v2) must be non-zero.")
    u = u / norm

    # Compute the perpendicular unit vector (rotated 90°)
    u_perp = np.array([-u[1], u[0]])

    # Rotate coordinates so that s is along the specified direction.
    s = (X - p1) * u[0] + (Y - p2) * u[1]
    t = (X - p1) * u_perp[0] + (Y - p2) * u_perp[1]

    # Flatten the arrays for interpolation.
    s_flat = s.ravel()
    t_flat = t.ravel()
    F_flat = F.ravel()

    # Create a LinearNDInterpolator with the (s, t) points.
    interp_func = LinearNDInterpolator(list(zip(s_flat, t_flat)), F_flat)

    # Create a regular grid in the (s, t) coordinate system.
    s_lin = np.linspace(s_flat.min(), s_flat.max(), ns)
    t_lin = np.linspace(t_flat.min(), t_flat.max(), nt)
    S_grid, T_grid = np.meshgrid(s_lin, t_lin)

    # Evaluate the interpolator on the regular grid.
    F_interp = interp_func(S_grid, T_grid)
    # Replace any NaN values (outside convex hull) with zeros.
    F_interp = np.nan_to_num(F_interp)

    # Sum F along the t (perpendicular) direction.
    marginal = np.sum(F_interp, axis=0)
    marginal /= np.trapz(marginal, s_lin)

    return s_lin, marginal

def plot_true_and_normal_approximation_marginal(params: dict,
    filename: str = None,
    grid_points: int = 200,
    xlim: tuple = (0.01, 1),
    ylim: tuple = (0.01, 1),
) -> None:

    # Create the lambda meshgrid
    x = np.linspace(xlim[0], xlim[1], grid_points)
    y = np.linspace(ylim[0], ylim[1], grid_points)
    X, Y = np.meshgrid(x, y)
    pos = np.empty(X.shape + (2,))
    pos[:, :, 0] = X
```

```
    pos[:, :, 1] = Y

    # Compute the true posterior surface
    true_Z = true_posterior(pos, **params)

    # Compute the normal approximation
    mu, Sigma, Sigma_inv = get_normal_approximation(**params)
    diff = pos - mu
    approx_Z = np.exp(-0.5 * np.einsum('...i,ij,...j', diff, Sigma_inv, diff))
    if params.get('normalize', False):
        approx_Z /= np.max(approx_Z)

    # Get the marginal along the direction (1, 1)
    s, marginal = marginal_along_direction(X, Y, true_Z, 1, 1, params['lambda_1'], params['lambda_2'])
    s_approx, marginal_approx = marginal_along_direction(X, Y, approx_Z, 1, 1, mu[0], mu[1])

    # Get the marginal along the direction (-1, 1)
    s_perp, marginal_perp = marginal_along_direction(X, Y, true_Z, -1, 1, params['lambda_1'],
    ↪  params['lambda_2'])
    s_perp_approx, marginal_perp_approx = marginal_along_direction(X, Y, approx_Z, -1, 1, mu[0], mu[1])

    # Create a figure with two subplots
    fig, axes = plt.subplots(1, 2, figsize=(8, 4))

    # Plot the marginal along the direction (1, 1)
    axes[0].plot(s, marginal, label='True Posterior', color='tab:blue')
    axes[0].plot(s_approx, marginal_approx, '--', label='Normal Approximation', color='tab:orange')
    axes[0].axvline(0, color='gray', linestyle='--', linewidth=0.8, label=r'where $\lambda_1 = 0.2, \lambda_2
    ↪  = 0.25$')
    axes[0].set_title(r'Marginal along $\begin{bmatrix} 1 & 1 \end{bmatrix}^\top$')
    axes[0].set_xlabel('$s$')
    axes[0].set_ylabel('Density')
    axes[0].legend()

    # Plot the marginal along the direction (-1, 1)
    axes[1].plot(s_perp, marginal_perp, label='True Posterior', color='tab:blue')
    axes[1].plot(s_perp_approx, marginal_perp_approx, '--', label='Normal Approximation', color='tab:orange')
    axes[1].axvline(0, color='gray', linestyle='--', linewidth=0.8, label=r'where $\lambda_1 = 0.2, \lambda_2
    ↪  = 0.25$')
    axes[1].set_title(r'Marginal along $\begin{bmatrix} -1 & 1 \end{bmatrix}^\top$')
    axes[1].set_xlabel('$t$')
    axes[1].set_ylabel('Density')

    # Specify the title as the epsilon value
    fig.suptitle(rf'$\epsilon = {params["epsilon"]}$')

    if filename:
        plt.savefig(os.path.join(FIG_DIR, filename), dpi=300, bbox_inches='tight')
    plt.tight_layout()
    plt.show()

# Example usage:
lambda_1 = 0.2
lambda_2 = 0.25
epsilons = [0.5, 0.1, 0.01]
taus     = [100, 300, 500]
alpha    = 0.15
beta     = 2
model_params = dict(
    lambda_1=lambda_1,
    lambda_2=lambda_2,
    alpha=alpha,
    beta=beta,
    use_MLE=False,
```

```
    normalize=True
)

for tau, epsilon in zip(taus, epsilons):
    y_1      = int(np.round(tau*lambda_1))
    y_2      = int(np.round(tau*lambda_2))
    model_params['tau']     = tau
    model_params['epsilon'] = epsilon
    model_params['y_1']     = y_1
    model_params['y_2']     = y_2
    fn = f"epsilon_{epsilon}.png"
    plot_true_and_normal_approximation(model_params, filename=f"epsilon_{epsilon}.png")
    plot_true_and_normal_approximation_marginal(model_params, filename=f"marginal_{epsilon}.png")
```

Problem 2: Does Preference Imply Belief?

Introducing a joint model over both the unknown and the data can make certain decision problems easier to solve, and offers stronger, more consistent solutions. In class we posed a joint statistical model over the unknown and the data, a loss function, then showed that we could select a decision rule by minimizing the posterior loss conditional on the observation, for all possible observations. There is a deeper representation theorem here. It is largely credited to Leonard Savage (The Foundations of Statistics, 1954), who built on the works of Von Neumann and Morgenstern.

Savage argued that statistics, in particular, prior distributions over unknowns, can sensibly reflect unstated beliefs about the state of an unknown inherent in a loss function and set of preferences for different decision rules. In particular, he argued that, if a decision maker adopts a sufficiently self-consistent set of preferences over decision rules given a loss, then the preference over decision rules is equivalent to the preference order induced by computing the expected posterior loss for some uniquely specified a prior distribution. This essentially runs our argument in reverse. Instead of deriving a particular decision rule (our most preferred rule) from a likelihood, prior, and loss, Savage showed that every likelihood, loss, and preference relation over decisions implies a prior. The axioms needed to ensure the preference set could correspond to a prior are frustratingly hard to find laid out in clear language. The Stanford Encyclopedia of Philosophy makes a reasonable attempt.

Savage's argument is the foundational pillar for most of subjective Bayesian statistics. It ensures that, if a user has reasonably self consistent preferences over decision rules given a loss, then they are behaving as if they are Bayesian with respect to some prior. The prior may or may not match their actual beliefs about the world, but they behave as if it did. In this sense, the prior is really an expression of the user's preferences, and the mathematics of Bayesian statistics are a convenient formal system for performing computation. This framework is especially popular in economics, since it expresses revealed beliefs as the necessary consequence of some notion of utility or preference. From the "objective" statistician's perspective it is profoundly disturbing. If your prior is just an expression of your preferences, then any "inference" with respect to it is entirely subjective.

Let's work a bit with this idea. The two problems below ask you to walk between the three components of expected utility theory to show that fixing two (e.g. a statistical model and loss) implies the third (e.g. a preference relation over decision rules).

(a) Suppose that I am performing a simple binary hypothesis test. I adopt a likelihood ratio test with the threshold $k = 15$ (that is, I decide for hypothesis 1 if it would make the data 15 times more likely than under hypothesis 0). I state that the loss incurred by a false positive (decide 1, reality 0) is three times worse than a loss under a false negative (decide 0, reality 1). You ask why I chose $k = 15$, and I say that I selected $k = 15$ since it is my favorite among all other possible choices of thresholds. Assuming the rest of my preference relations satisfy Savage's axioms (unstated), what is my implied prior distribution over the two hypotheses?

Recall the set-up of the Binary Hypothesis Test. We defined the model as

$$\textbf{Prior: } \Theta \sim \mathop{\mathbb{P}}_{\Theta} \quad \text{and} \quad \textbf{Likelihood: } (-)\ \theta = 0 \ :\ H_0 \ :\ Y \sim \mathop{\mathbb{P}}_{Y|\Theta=0}$$
$$(+)\ \theta = 1 \ :\ H_1 \ :\ Y \sim \mathop{\mathbb{P}}_{Y|\Theta=1},$$

the decision rule and loss as

$$d(Y) := \begin{cases} 0 & \text{if } \mathrm{LR}(y) < k \\ 1 & \text{if } \mathrm{LR}(y) \geq k \end{cases} \quad \text{and} \quad \ell(\Theta, d(Y)) := \begin{array}{cc|cc} & & \multicolumn{2}{c}{d(Y)} \\ & & 0 & 1 \\ \hline \Theta & 0 & 0 & \ell_{FP} \\ & 1 & \ell_{FN} & 0 \end{array}$$

where the Likelihood-Ratio, $\mathrm{LR}(y)$, is defined as $\mathrm{LR}(y) = \frac{\mathbb{P}[Y=y|\Theta=1]}{\mathbb{P}[Y=y|\Theta=0]}$. Let $\mathcal{D}$ represent the set of possible decision rules and recall that the Bayes risk is $R_B(d) = \mathbb{E}_{\Theta,Y}[\ell(\Theta, d(Y))]$. In class, we showed that threshold of our decision rule $k^\star$ that minimizes the Bayes risk is given by

$$k^\star = \left(\frac{\ell_{FP}}{\ell_{FN}}\right)\left(\frac{\mathbb{P}_\Theta[\Theta = 0]}{\mathbb{P}_\Theta[\Theta = 1]}\right).$$

In our problem, we have $\frac{\ell_{FP}}{\ell_{FN}} = 3$ and $k^\star = 15$ which implies that our prior is given by

$$15 = 3\left(\frac{\mathbb{P}_\Theta[\Theta = 0]}{\mathbb{P}_\Theta[\Theta = 1]}\right) \quad \Longrightarrow \quad \boxed{\mathop{\mathbb{P}}_{\Theta}[\Theta] = \begin{cases} \frac{5}{6} & \text{when } \Theta = 0 \\ \frac{1}{6} & \text{when } \Theta = 1 \end{cases}}$$

(b) It is common, in a variety of on-line learning problems (problems where you collect data as you make decisions, and your decisions influence what data you see), to select options based on a posterior upper credible bound. A version of this rule would say, "If, given my current knowledge about the world, option j out of my n options could credibly have the largest value, then I will pick j." To implement this rule, we look at the posterior probability distribution over the value of each option, select a probability $p \approx 1$ with which we hope to certify our bounds (the larger p the more credible the bound), and, for each option, compute the pth percentile of its possible value. If V_j is the unknown value of option j, then this is an upper bound b_j such that $\mathbb{P}(V_j \leq b_j) = p$. You used a rule of this kind to optimize your research allocation in Lab 2.

Find a loss function over the estimated value of an unknown that would justify this procedure. That is, find a loss function $l(V, \hat{v}(y))$ for which the Bayes optimal decision rule, for all choices of p, is to set $\hat{v}(Y)$ equal to the p-upper credible bound on $V \mid Y = y$.

Hint: check the suggested reading from weeks 3 and 4, or the rule in HW 1 that suggested the median as an optimal posterior estimator.

Recall that our Bayes optimal decision rule is given by

$$\hat{v}^\star(Y) := \underset{\hat{v}}{\operatorname{argmin}} \left\{ R_B(\hat{v}) \right\} = \underset{\hat{v}}{\operatorname{argmin}} \left\{ \mathop{\mathbb{E}}_{V,Y} \left[\ell(V, \hat{v}(Y)) \right] \right\} = \underset{\hat{v}}{\operatorname{argmin}} \left\{ \mathop{\mathbb{E}}_{Y} \left[\mathop{\mathbb{E}}_{V|Y=y} \left[\ell(V, \hat{v}(y)) \right] \right] \right\}.$$

In other words, we can say that the optimal decision rule is found by setting the gradient of the Bayes risk with respect to $\hat{v}$ to 0 at $\hat{v}^\star$. In class, we showed that this is equivalent to the rule that optimizes the posterior risk for all $Y = y$. We have

$$\left. \frac{\partial R_B(\hat{v})}{\partial \hat{v}} \right|_{\hat{v}=\hat{v}^\star} = 0 \quad \text{or equivalently} \quad \left. \frac{\partial R_P(\hat{v})}{\partial \hat{v}} \right|_{\hat{v}=\hat{v}^\star} = 0 \quad \text{for all } Y = y.$$

We seek to find the loss $\ell(V, \hat{v}(y))$ that results in $\hat{v}^\star$ being the p-upper credible bound on $V \mid Y = y$, i.e.,

$$\mathop{\mathbb{P}}_{V|Y=y} \left[V \leq \hat{v}^\star(y) \right] = p = \int_{-\infty}^{\hat{v}^\star} f_{V|Y=y}(v) dv.$$

It follows that

$$\left. \frac{\partial R_P(\hat{v})}{\partial \hat{v}} \right|_{\hat{v}=\hat{v}^\star} = \frac{\partial}{\partial \hat{v}} \mathop{\mathbb{E}}_{V|Y=y} \left[\ell(V, \hat{v}(y)) \right] = \frac{\partial}{\partial \hat{v}} \int_{-\infty}^{\infty} \ell(v, \hat{v}(y)) f_{V|Y=y}(v) dv = \int_{-\infty}^{\infty} \frac{\partial \ell(v, \hat{v}(y))}{\partial \hat{v}} f_{V|Y=y}(v) dv = 0.$$

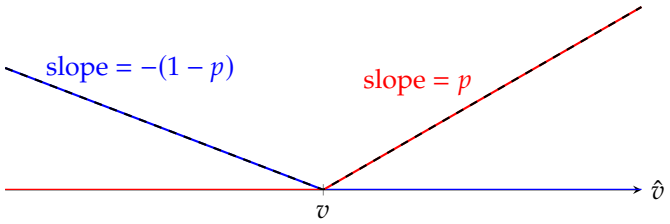
Let α and β be the gradient of the loss w.r.t. $\hat{v}$ in the domain from $-\infty$ to $\hat{v}^\star$ and the domain from $\hat{v}^\star$ to ∞, respectively. Here, we suppose that the gradient in the two domains are distinct and constant such that we can pull it out of the integral. We have

$$\frac{\partial \ell}{\partial \hat{v}} = \begin{cases} \alpha & \text{when } v \in (-\infty, \hat{v}^\star) \\ \beta & \text{when } v \in (\hat{v}^\star, \infty) \end{cases} \implies \alpha \underbrace{\int_{-\infty}^{\hat{v}^\star} f_{V|Y=y}(v) dv}_{p} + \beta \underbrace{\int_{\hat{v}^\star}^{\infty} f_{V|Y=y}(v) dv}_{1-p} = 0 \implies \alpha p + \beta(1-p) = 0.$$

Thus, we seek to define a loss such that its gradient w.r.t. $\hat{v}$ satisfies the constraint $\alpha p + \beta(1-p) = 0$. One such loss is where $\alpha = -(1-p)$ and $\beta = p$. We can achieve this with a sum of rectified linear units or ReLU's. We have

$$\boxed{\ell(v, \hat{v}) = p \max\left\{0, \hat{v} - v\right\} + (1-p) \max\left\{0, v - \hat{v}\right\}} \implies \frac{\partial \ell}{\partial \hat{v}} = \begin{cases} -(1-p) & \text{when } v \in (-\infty, \hat{v}^\star) \\ p & \text{when } v \in (\hat{v}^\star, \infty) \end{cases}$$

It is easiest to see this result by plotting the ReLU's seperately.



Problem 3: Agnostics Hold Opinions

It's surprisingly hard to pose an uninformative prior over continuous valued unknowns (see BDA chapter 2.8). Any continuous random variable must admit a density function. Density functions are not invariant under transformations of the underlying space, since the density is defined with respect to that space. Remember that a probability density is the expected number of samples in a small region of space, in the limit as the volume of the region vanishes. In other words, its units are probability mass per volume. Volumes depend on the units used to measure the unknown (i.e. the length of an interval changes if you change units), and may change differently in different parts of a parameter space if we change coordinates nonlinearly. Thus, a uniform distribution under one choice of coordinates for the unknowns may not be uniform under another.

As a result, even our best attempts at picking an agnostic prior, which express no prior belief, still imply a specific set of beliefs, namely, uniformity in a specific set of coordinates. This is an important lesson to remember. If you choose to use a Bayesian approach in a continuous space, you must always express some belief.

In this problem, you will practice using our best tool for finding "uninformative" priors.

(a) BDA 2.12: Suppose $Y \mid \theta \sim \text{Poiss}(\Theta)$. Apply the invariance principle to find Jeffreys' prior density for Θ.

Recall that $X \sim \text{Poiss}(\mu)$ has a density of $\mathbb{P}[X = k] = \frac{\mu^k e^{-\mu}}{k!}$ for $\mu > 0, k \in \mathbb{N}_0$ and an expectation and variance of $\mathbb{E}[X] = \text{Var}[X] = \mu$. The Jeffreys' prior density is given by $\mathbb{P}_\Theta(\theta) = [J(\theta)]^{1/2}$ where $J(\theta)$ is the Fisher Information. We have that

$$\mathop{\mathbb{P}}_{\Theta}(\theta) \propto \sqrt{\mathbb{E}\left[\left(\nabla_\theta \log\left(\text{Lik}(Y \mid \theta)\right)\right)^2\right]} = \sqrt{\mathbb{E}\left[\left(\nabla_\theta \{Y \log(\theta) - \theta - \log(Y!)\}\right)^2\right]} = \sqrt{\mathbb{E}\left[\left(\frac{Y}{\theta} - 1\right)^2\right]} = \sqrt{\mathbb{E}\left[\frac{Y^2}{\theta^2} - 2\frac{Y}{\theta} + 1\right]}.$$

Note that $\mathbb{E}[Y^2] = \text{Var}[Y] + \mathbb{E}[Y]^2 = \theta + \theta^2$. It then follows that

$$\mathop{\mathbb{P}}_{\Theta}(\theta) \propto \sqrt{\frac{\theta + \theta^2}{\theta^2} - 2\frac{\theta}{\theta} + 1} \implies \boxed{\mathop{\mathbb{P}}_{\Theta}(\theta) \propto \sqrt{\frac{1}{\theta}}}$$

(b) Use the invariance principle to find Jeffrey's prior for the standard deviation of a 1-dimensional Gaussian with known mean.

In this problem, we define the Likelihood as $\text{Lik}(Y \mid \sigma) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$. The Log-Likelihood is given by $\log(\text{Lik}(Y \mid \sigma)) := -\log\left(\sqrt{2\pi}\right) - \log(\sigma) - \frac{(y-\mu)^2}{2\sigma^2}$. The Fisher Information is then given by

$$J(\sigma) = \mathbb{E}\left[\left(\frac{\partial}{\partial\sigma}\log(\text{Lik}(Y \mid \sigma))\right)^2\right] = \mathbb{E}\left[\left(-\frac{1}{\sigma} + \frac{(y-\mu)^2}{\sigma^3}\right)^2\right] = \mathbb{E}\left[\frac{1}{\sigma^2} - 2\frac{(y-\mu)^2}{\sigma^4} + \frac{(y-\mu)^4}{\sigma^6}\right]$$

$$J(\sigma) = \frac{1}{\sigma^2} - \frac{2}{\sigma^4}\mathbb{E}\left[(y-\mu)^2\right] + \frac{1}{\sigma^6}\mathbb{E}\left[(y-\mu)^4\right]$$

Recalling that $\mathbb{E}\left[(X-\mu)^p\right] = \begin{cases} 0 & \text{if } p \text{ is odd,} \\ \sigma^p(p-1)!! & \text{if } p \text{ is even.} \end{cases}$ where $n!!$ is the double factorial, we have

$$J(\sigma) = \frac{1}{\sigma^2} - \frac{2\sigma^2}{\sigma^4} + \frac{3\sigma^4}{\sigma^6} = \frac{2}{\sigma^2}$$

Therefore, the Jeffrey's prior is given by

$$\boxed{\mathop{\mathbb{P}}_{\sigma}(\sigma^2) \propto \frac{1}{\sigma^2}}$$

How does your prior compare to:

- The choice recommended for scale parameters in BDA 2.9 (see *Pivotal Quantities*)?
  The Jeffrey prior is the same as in BDA 2.9 where they suggest $p(\sigma^2) \propto \frac{1}{\sigma^2}$.
- The conjugate prior family used for the variances of a standard normal? Is Jeffrey's prior an instance of the conjugate family? If so, for what parameters?
  In BDA 2.6, it is shown that the conjugate prior to a *Normal distribution with known mean but unknown variance* is given by a scaled inverse-χ^2 distribution with scale σ_0^2 and ν_0 degrees of freedom. We have that $\sigma^2 \sim \text{Inv}-\chi^2(\nu_0, \sigma_0^2)$ with density

$$p(\sigma^2) \propto \left(\frac{\sigma_0^2}{\sigma^2}\right)^{\nu_0/2+1} \exp\left(-\frac{\nu_0\sigma_0^2}{2\sigma^2}\right)$$

  If we set $\nu_0 = 0$ and $\sigma_0^2 = 1$, we get

$$p(\sigma^2) \propto \left(\frac{1}{\sigma^2}\right)^{0+1} \exp\left(-\frac{0 \cdot 1}{2\sigma^2}\right) = \frac{1}{\sigma^2}$$

  which matches the Jeffrey prior distribution.

(c) BDA 2.17: The fact that uninformative densities are hard to find follows from the fact that density functions change shape under nonlinear transformations. Since the endpoint of most Bayesian inference is a posterior interval estimate, this fact also has consequences for the way we define intervals. In particular, since the highest posterior density interval (HPDI) depends on selecting regions of high density, it is not preserved under transformations.

Suppose that, given σ^2, the quantity nv/σ^2 is χ_n^2 distributed, and that σ has the (improper) non-informative prior density $p(\sigma) \propto \sigma^{-1}$ for all $\sigma > 0$.

In BDA 2.6, we formulate the conjugate model for a *Normal distribution with known mean but unknown variance*. We had that Likelihood for a vector y of n independent and identically distributed observations is given by

$$p(y \mid \sigma^2) = \prod_{i=1}^{n} \mathbb{P}[Y^{(n)} = y^{(n)} \mid \sigma^2] \propto \sigma^{-n} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2\right) = \left(\sigma^2\right)^{-n/2} \exp\left(-\frac{nv}{2\sigma^2}\right)$$

where we define $v = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2$ as the *sufficient statistic*, i.e., the quantity that controls the likelihood for the unknown σ^2 that depends strictly on the data y.

- Find the corresponding prior density for σ^2.
  We find the corresponding prior density for σ^2 via change of variables. We have that if $Y = g(X)$ and $X = g^{-1}(Y)$ then $f_Y(y) = f_X(g^{-1}(Y))\left|\frac{dg^{-1}(Y)}{dy}\right|$.[7] Let $Y = \sigma^2$ such that $\sigma = \sqrt{Y}$. Then we have

$$f_Y(y) = f_\sigma\left(\sqrt{Y}\right)\left|\frac{d}{dY}\left\{\sqrt{Y}\right\}\right| = \frac{1}{\sqrt{Y}} \cdot \frac{1}{2\sqrt{Y}} = \frac{1}{2Y} \quad \Longrightarrow \quad \boxed{p(\sigma^2) \propto \frac{1}{\sigma^2}}$$

- Show that the 95% HPDI for σ^2 is not the same as the region obtained by squaring the endpoints of the 95% HPDI for σ. *Here, the "P" in HPDI means prior.*
  Depending on the prior we use, we have that the posterior density would be proportional to

$$p(\sigma \mid y) \propto p(y \mid \sigma^2)p(\sigma) \qquad p(\sigma^2 \mid y) \propto p(y \mid \sigma^2)p(\sigma^2)$$
$$p(\sigma \mid y) \propto \left(\sigma^2\right)^{-n/2} \exp\left(-\frac{nv}{2\sigma^2}\right) \cdot \frac{1}{\sigma} \qquad p(\sigma^2 \mid y) \propto \left(\sigma^2\right)^{-n/2} \exp\left(-\frac{nv}{2\sigma^2}\right) \cdot \frac{1}{\sigma^2}$$
$$p(\sigma \mid y) \propto \left(\sigma^2\right)^{-n/2-1/2} \exp\left(-\frac{nv}{2\sigma^2}\right) \qquad p(\sigma^2 \mid y) \propto \left(\sigma^2\right)^{-n/2-1} \exp\left(-\frac{nv}{2\sigma^2}\right)$$

  First, we note that both posteriors are unimodal. This can be shown via calculus or we can simply plot the two posteriors (Figure 3). Given that both posteriors are unimodal, their 95% HPDI's will only be one interval.

[7] This follows directly from the definition of the cummulative density function and chain rule. We have that

$$f_Y(y) = \frac{\mathrm{d}}{\mathrm{d}y}\left\{F_Y(y)\right\} = \frac{\mathrm{d}}{\mathrm{d}y}\left\{\mathbb{P}[Y \le y]\right\} = \frac{\mathrm{d}}{\mathrm{d}y}\left\{\mathbb{P}[g(X) \le y]\right\} = \frac{\mathrm{d}}{\mathrm{d}y}\left\{\mathbb{P}[X \le g^{-1}(y)]\right\} = \frac{\mathrm{d}}{\mathrm{d}y}\left\{F_X(g^{-1}(y))\right\} = \frac{\mathrm{d}F_X(g^{-1}(y))}{\mathrm{d}g^{-1}(y)}\left|\frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y}\right| = f_X(g^{-1}(Y))\left|\frac{dg^{-1}(Y)}{dy}\right|$$

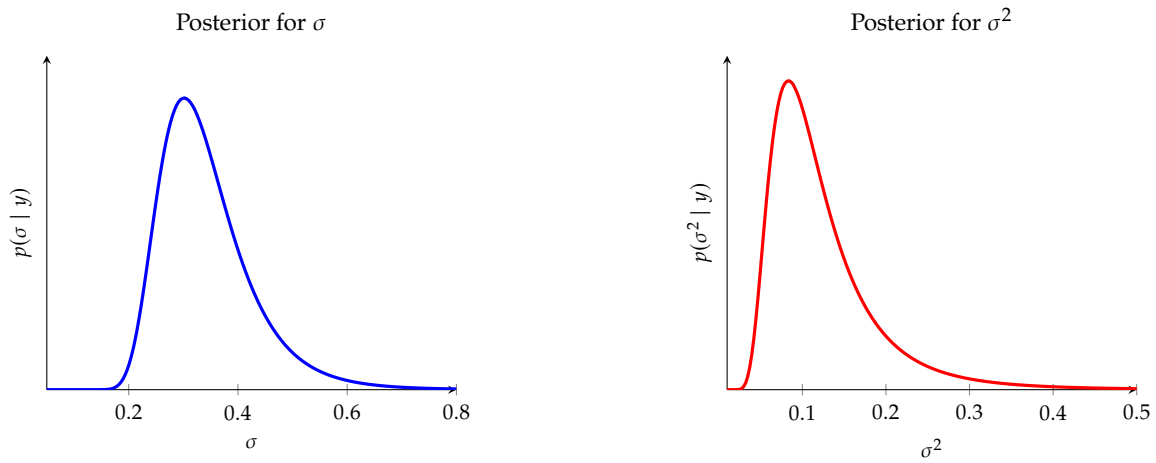


FIGURE 3: Example of the two posteriors, $p(\sigma \mid y)$ and $p(\sigma^2 \mid y)$, for $n = 10$ and $v = 1/10$.

Our goal is to show that the 95% HPDI for σ^2 is not the same as the region obtained by squaring the endpoints of the 95% HPDI for σ. In other words, if $[\sqrt{a}, \sqrt{b}]$ is the 95% HPDI for σ, then the 95% HPDI for σ^2 is not simply $[a, b]$.

We prove this via contradiction. Let $[\sqrt{a}, \sqrt{b}]$ be the 95% HPDI for σ. We have that

$$(a)^{-n/2-1/2} \exp\left(-\frac{nv}{2a}\right) = (b)^{-n/2-1/2} \exp\left(-\frac{nv}{2b}\right) \quad \Longrightarrow \quad (-n/2-1/2)\log(a) - \frac{nv}{2a} = (-n/2-1/2)\log(b) - \frac{nv}{2b}$$

because in a unimodal density, the endpoints of the HPDI occur at the same density level. Similarly, let $[a, b]$ be the 95% HPDI for σ^2. Then we have that

$$(a)^{-n/2-1} \exp\left(-\frac{nv}{2a}\right) = (b)^{-n/2-1} \exp\left(-\frac{nv}{2b}\right) \quad \Longrightarrow \quad (-n/2-1)\log(a) - \frac{nv}{2a} = (-n/2-1)\log(b) - \frac{nv}{2b}$$

Combining our two equations, we find that

$$(-n/2-1/2)\log(a) - (-n/2-1)\log(a) = (-n/2-1/2)\log(b) - (-n/2-1)\log(b) \quad \Longrightarrow \quad \frac{1}{2}\log(a) = \frac{1}{2}\log(b)$$

which implies that $a = b$, which is not a valid interval. Therefore, the 95% HPDI for σ^2 is not the same as the region obtained by squaring the endpoints of the 95% HPDI for σ.

Problem 4: Conjugacy Practice

(a) Show that if $Y \mid \theta$ is exponentially distributed with rate θ, then the gamma distribution forms a conjugate prior for θ.

Recall that for $Y \mid \theta \sim \text{Exp}(\theta)$, we have $p_{Y=y|\Theta}(\theta) = \theta e^{-\theta y}$ for $\theta > 0, y \in [0, \infty)$ and for $\Theta \sim \text{Gamma}(\alpha, \lambda)$, we have $p_\Theta(\theta) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} e^{-\lambda\theta}$ for $\alpha > 0, \lambda > 0, x \in (0, \infty)$. It follows that

$$p_{\Theta|Y=y}(\theta) \propto p_{Y=y|\Theta}(\theta) p_\Theta(\theta) = \theta e^{-\theta y} \frac{\lambda^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} e^{-\lambda\theta} = \frac{\lambda^\alpha}{\Gamma(\alpha)} \theta^\alpha e^{-(\lambda+y)\theta} \quad \Longrightarrow \quad \Theta \mid Y = y \sim \text{Gamma}(\alpha + 1, \lambda + y).$$

(b) Show that a gamma prior is also conjugate if, given θ, the sequence $Y^{(n)} = \{Y_j\}_{j=1}^n$ are drawn i.i.d. from an exponential distribution with rate θ.

Our likelihood can now be written as

$$p_{Y^{(n)}=y^{(n)}|\Theta}(\theta) = \mathbb{P}[Y^{(n)} = y^{(n)} \mid \Theta] \overset{\text{(i.i.d.)}}{=} \prod_{j=1}^n \mathbb{P}[Y_j = y_j \mid \Theta] = \prod_{j=1}^n \theta e^{-\theta y_j} = \theta^n e^{-\theta \sum_{j=1}^n y_j}.$$

It then follows that

$$p_{\Theta|y^{(n)}}(\theta) \propto p_{y^{(n)}|\Theta}(\theta) p_\Theta(\theta) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \theta^{\alpha+n-1} e^{-(\lambda+\sum_{j=1}^n y_j)\theta} \quad \Longrightarrow \quad \Theta \mid Y^{(n)} = y^{(n)} \sim \text{Gamma}\left(\alpha + n, \lambda + \sum_{j=1}^n y_j\right).$$

(c) Find all sufficient statistics in this scenario (a set of functions of $Y^{(n)}$ that uniquely specify the posterior distribution). How many do you need? Do they correspond to any standard statistics?

Recall that the *sufficient statistic* is the quantity $t(y)$ for which the likelihood for the unknown θ depends on the data y only through the value of $t(y)$. Only <u>one sufficient statistic</u> is required.

For part (a), we have that the sufficient statistic $t_a(y)$ is given by $\boxed{t_a(y) = y}$ which corresponds to the sample mean. Using the sufficient statistic $t_a(y)$, we can write the posterior as

$$\Theta \mid Y = y \sim \text{Gamma}\left(\alpha + 1, \lambda + t_a(y)\right).$$

For part (b), we have that the sufficient statistic $t_b\left(y^{(n)}\right)$ is given by $\boxed{t_b\left(y^{(n)}\right) = \frac{1}{n} \sum_{j=1}^n y_j}$ which corresponds to the sample mean. Using the sufficient statistic $t_b(y)$, we can write the posterior as

$$\Theta \mid Y^{(n)} = y^{(n)} \sim \text{Gamma}\left(\alpha + n, \lambda + n t_b(y^{(n)})\right).$$

Problem 5: Maximizing Mutual Information

Given a prior distribution for some unknowns, and a set of possible measurements, each with its own likelihood, what measurement should we pick to gain the most information about the unknown?

This is a special form of a Bayesian decision theory problem. It is widely used to optimize experimental design, and if iterated, can construct optimal codes, compression algorithms, and can maximize channel capacity (communication rate). It can also be used to construct optimal search procedures that iteratively select the measurement which is expected to provide the most information about the unknown given our current knowledge about the unknown.

You've seen versions of this question in Labs 2 and 3. Now, we have enough tools to solve it elegantly for a moderately general class of models. Our main tool here will be conjugate normal, normal models. If you need a refresher, check your work from Lab 3, or Chapters 2.5 and 3.3 in BDA.

Let $X \sim p_X$ denote an unknown, real-valued state vector taking values in $\mathbb{R}^d$. Let $\mathcal{M}$ be a set of possible measurements we could make (questions we could ask about the unknown), and let $m \in \mathcal{M}$ denote a particular measurement. After measurement m, we observe $Y \sim p_{Y|X,m}$. Note that, since different measurements have different likelihoods, they will induce distinct posteriors.

To choose the measurement m we consider the following three strategies:

1. Choose a measure of uncertainty $U[p]$ that can evaluate the degree of uncertainty in a distribution. For example, U might be some measure of the total variance in the distribution (trace of the covariance matrix). Then, select m to minimize the expected uncertainty left over after a measurement: $m^\star = \operatorname{argmin}_{m\in\mathcal{M}} \mathbb{E}_Y[U[p_{X|Y,m}]]$.

2. Choose the measure that maximizes the mutual information between the observed outcome and the unknown. Mutual information is, equivalently, the expected reduction in uncertainty after a measurement, for an appropriate choice of uncertainty measure, and a measure of how easy it is to show, in a hypothesis test, that the unknown and the measurement are coupled (not independent). That is, select $m^\star = \operatorname{argmax}_{m\in\mathcal{M}} \big\{I(X;Y|m)\big\}$, where $I(X;Y)$ is the mutual information between two random quantities.

3. We could, instead, take an adversarial approach, where our goal is to select the measurement that is expected to lead to the largest change in our beliefs. That is, we want the measurement that is expected to teach us the most, by inducing the largest change when moving from prior to posterior. Let $D(p\|q)$ be a divergence (that is, a measure of how distinct two distributions are). Then, select $m^\star = \operatorname{argmin}_{m\in\mathcal{M}} \big\{\mathbb{E}[D(p_{X|Y,m}\|p_X)]\big\}$.

Amazingly, all three approaches are the same if we adopt the right definition of uncertainty, information, and divergence. In particular, if we adopt the Shannon entropy, H, as our measure of uncertainty, and the KL divergence as our measure of divergence, then all three methods are equivalent to maximizing the mutual information between the measurement and the unknown, where information is an expected reduction in entropy. This remarkable result gives a simple aim:

$$\begin{aligned} &\textbf{Find:} \quad m^\star = \operatorname*{argmax}_{m\in\mathcal{M}} \big\{I(X;Y|m)\big\} = \operatorname*{argmax}_{m\in\mathcal{M}} \big\{\mathop{\mathbb{E}}_{Y}[H[p_{X|Y,m}]]\big\}, \\ &\textbf{where:} \quad H[p] = -\mathop{\mathbb{E}}_{X\sim p}\big[\log\big(p(X)\big)\big]. \end{aligned} \tag{24}$$

You ran a version of this algorithm in your second lab, but used standard deviation for U instead of entropy. In this problem, you will show that the information-theoretic framing gives reasonable, simple, and easily implemented answers for the normal-normal model. This answer justifies standard approaches in signal processing and experimental design, namely, matched filtering.

Consider a Bayesian model on a d-dimensional unknown $X \in \mathbb{R}^d$ of the form:

$$\begin{aligned} &\textbf{Prior:} \quad X \sim \mathcal{N}(\hat{x}, C_x) \\ &\textbf{Likelihood:} \quad Y = m^\top X + \zeta, \quad \text{where } m \in \mathcal{M} \subset \mathbb{R}^d \text{ and } \zeta \sim \mathcal{N}(0, \sigma_y^2). \end{aligned} \tag{25}$$

This is a standard measurement model in many applied settings, namely, we make one-dimensional measurements by evaluating some inner-product (projection) of the state vector, and assume that our measurements are corrupted by Gaussian noise. Here $m \in \mathcal{M}$ is the measurement vector. Our freedom to choose different measurements is represented by the freedom to select different projections of the state. It should be intuitive that the best direction will depend on C_x and will aim to constrain the covariance along directions with large variance.

To proceed we will need to know the entropy of a Gaussian distribution. If $Z \sim \mathcal{N}(\hat{z}, C_z)$ then $H[p_Z] = \frac{1}{2}\log\big(\det(C_z)\big) + c_d$, where c_d is a constant that only depends on the dimension of Z, and where $\det(\cdot)$ denotes the determinant.[8]

Using this fact, let's select the information theoretic optimal measurement direction, m, for any normally distributed state vector evaluated with a linear measurement and normal measurement error.

[8] $c_d = \frac{d}{2}(1 + \log(2\pi))$.

(a) Derive the formula for $p_{X|Y=y,m}$ for any observation y and measurement direction m. You may reference your notes from Lab 3. Remember, the normal-normal model is conjugate, so the posterior will remain normal. This means you only need to recall the formulas that update the mean and covariance in X after observing $Y = y$ for a measurement m.

Since the likelihood is given by $Y = m^\top X + \zeta$, it follows a normal distribution with mean $m^\top x$ and variance σ_y^2. Then, the densities of our prior and likelihood are given by

$$\textbf{Prior:} \qquad X \sim \mathcal{N}(\hat{x}, C_x) \qquad f_X(x) = \frac{1}{(2\pi)^{d/2}\det(C_x)^{1/2}} \exp\left(-\frac{1}{2}(x-\hat{x})^{-1}C_x^{-1}(x-\hat{x})\right),$$

$$\textbf{Likelihood:} \qquad Y \mid X = x, m \sim \mathcal{N}(m^\top x, \sigma_y^2) \qquad f_{Y|X=x,m}(y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(y-m^\top x)^2}{2\sigma_y^2}\right).$$

It follows that the posterior $X \mid Y = y, m$ is proportional to

$$f_{X|Y=y,m}(x) \propto f_{Y|X=x,m}(y) f_X(x) \propto \exp\left(-\frac{(y-m^\top x)^2}{2\sigma_y^2}\right)\exp\left(-\frac{1}{2}(x-\hat{x})^{-1}C_x^{-1}(x-\hat{x})\right)$$

$$f_{X|Y=y,m}(x) \propto \exp\left(-\frac{1}{2}\Big[x^\top \underbrace{\left(C_x^{-1} + \sigma_y^{-2}mm^\top\right)}_{A^{-1}} x - 2\underbrace{\left(\hat{x}^\top C_x^{-1} + \sigma_y^{-2}ym^\top\right)}_{b^\top} x\Big]\right)$$

$$f_{X|Y=y,m}(x) \overset{9}{\propto} \exp\left(-\frac{1}{2}\Big[(x-Ab)^\top A^{-1}(x-Ab) - \underbrace{b^\top A^\top A^{-1} Ab}_{\text{does not depend on } x}\Big]\right)$$

$$f_{X|Y=y,m}(x) \propto \exp\left(-\frac{1}{2}\Big[(x-Ab)^\top A^{-1}(x-Ab)\Big]\right)$$

$$\implies \boxed{\mathbb{E}[X \mid Y = y, m] = \left(C_x^{-1} + \sigma_y^{-2}mm^\top\right)^{-1}\left(C_x^{-1}\hat{x} + \sigma_y^{-2}ym\right) \quad \text{and} \quad \mathrm{Cov}[X \mid Y = y, m] = \left(C_x^{-1} + \sigma_y^{-2}mm^\top\right)^{-1}}$$

We can do a bit more work on the expectation of the posterior to write it as

$$\boxed{\mathbb{E}[X \mid Y = y, m] \overset{10}{=} \hat{x} + \sigma_y^{-2}\left(y - m^\top\hat{x}\right)\left(I + \sigma_y^{-2}C_x mm^\top\right)^{-1}C_x m}$$

---

[9] This follows from matching coefficients of the quadratic equation within the exponential with that of a Gaussian. Specifically, we have

$$(x-\mu)^\top\Sigma^{-1}(x-\mu) = x^\top\Sigma^{-1}x + \mu^\top\Sigma^{-1}\mu - 2\mu^\top\Sigma^{-1}x \quad \text{match coefficients} \quad x^\top A^{-1}x - 2b^\top x \implies A = \Sigma \text{ and } b^\top = \mu^\top\Sigma^{-1} \implies b = \Sigma^{-1}\mu \implies \mu = \Sigma b$$

[10]

$$\mathbb{E}[X \mid Y = y, m] = \left(C_x^{-1} + \sigma_y^{-2}mm^\top\right)^{-1}\left(C_x^{-1}\hat{x} + \sigma_y^{-2}ym\right)$$
$$\mathbb{E}[X \mid Y = y, m] = \left(C_x^{-1} + \sigma_y^{-2}mm^\top\right)^{-1}C_x^{-1}\left(\hat{x} + \sigma_y^{-2}yC_x m\right)$$
$$\mathbb{E}[X \mid Y = y, m] = \left(C_x\left(C_x^{-1} + \sigma_y^{-2}mm^\top\right)\right)^{-1}\left(\hat{x} + \sigma_y^{-2}yC_x m\right)$$
$$\mathbb{E}[X \mid Y = y, m] = \left(I + \sigma_y^{-2}C_x mm^\top\right)^{-1}\left(\hat{x} + \sigma_y^{-2}yC_x m\right)$$
$$\mathbb{E}[X \mid Y = y, m] = \left(I + \sigma_y^{-2}C_x mm^\top\right)^{-1}\Big(\underbrace{\Big[\left(I + \sigma_y^{-2}C_x mm^\top\right)\hat{x} - \sigma_y^{-2}C_x mm^\top\hat{x}\Big]}_{\hat{x}} + \sigma_y^{-2}yC_x m\Big)$$
$$\mathbb{E}[X \mid Y = y, m] = \hat{x} + \left(I + \sigma_y^{-2}C_x mm^\top\right)^{-1}\left(\sigma_y^{-2}yC_x m - \sigma_y^{-2}C_x mm^\top\hat{x}\right)$$
$$\mathbb{E}[X \mid Y = y, m] = \hat{x} + \sigma_y^{-2}\left(y - m^\top\hat{x}\right)\left(I + \sigma_y^{-2}C_x mm^\top\right)^{-1}C_x m$$

(b) Show that the change in entropy between the prior p_X and the posterior $p_{X|Y=y,m}$ does not depend on the change in the means of either distribution, or the dimensional constant c_d.

Entropy is given by $H[p_Z] = \frac{1}{2}\log\big(\det(C_z)\big) + c_d$, for $Z \sim \mathcal{N}(\hat{z}, C_z)$ where $c_d = \frac{d}{2}(1 + \log(2\pi))$. It follows that the entropy of the prior and posterior are given by

$$H[p_X] = \frac{1}{2}\log(\det(C_x)) + c_d, \quad \text{and} \quad H[p_{X|Y,m}] = \frac{1}{2}\log\Big(\det\Big(\big(C_x^{-1} + \sigma_y^{-2} m m^\top\big)^{-1}\Big)\Big) + c_d, \quad \text{respectively.}$$

We recall $\det\big(M^{-1}\big) = \frac{1}{\det(M)}$ and the matrix determinant lemma, $\det(A + uv^\top) = (1 + v^\top A^{-1} u)\det(A)$, to show that $\det\Big(\big(C_x^{-1} + \sigma_y^{-2} m m^\top\big)^{-1}\Big) = \frac{1}{\det\left(C_x^{-1} + \sigma_y^{-2} m m^\top\right)} = \frac{1}{(1+\sigma_y^{-2} m^\top C_x m)\det\left(C_x^{-1}\right)} = \frac{\det(C_x)}{1+\sigma_y^{-2} m^\top C_x m}$. Then we have that

$$H[p_{X|Y,m}] = \frac{1}{2}\log\left(\frac{\det(C_x)}{1 + \sigma_y^{-2} m^\top C_x m}\right) + c_d = \frac{1}{2}\Big(\log(\det(C_x)) - \log\Big(1 + \sigma_y^{-2} m^\top C_x m\Big)\Big) + c_d.$$

Taken together, the change in entropy, $H[p_X] - H[p_{X|Y,m}]$, is given by

$$H[p_X] - H[p_{X|Y,m}] = \frac{1}{2}\log(\det(C_x)) + c_d - \frac{1}{2}\log(\det(C_x)) + \frac{1}{2}\log\Big(1 + \sigma_y^{-2} m^\top C_x m\Big) - c_d = \boxed{\frac{1}{2}\log\Big(1 + \sigma_y^{-2} m^\top C_x m\Big)}$$

which is does not depend on the change in the means of either distribution, or the dimensional constant c_d.

(c) Show, in addition, that the change in entropy, $H[p_X] - H[p_{X|Y,m}]$, does not depend on the actual measured value, but is a deterministic function of m alone! Use this fact to give a closed formula for the information acquisition function,

$$I(X; Y \mid m) = \mathop{\mathbb{E}}_{Y|m}\Big[H[p_X] - H[p_{X|Y,m}]\Big]. \tag{26}$$

We showed in part (b) that the change in entropy is $H[p_X] - H[p_{X|Y,m}] = \frac{1}{2}\log\Big(1 + \sigma_y^{-2} m^\top C_x m\Big)$, which does not depend on the actual measured value, and is a deterministic function of m alone. Taking the expectation over $Y \mid m$ to calculate the information acquisition function, we have

$$I(X; Y \mid m) = \mathop{\mathbb{E}}_{Y|m}\Big[H[p_X] - H[p_{X|Y,m}]\Big] = \mathop{\mathbb{E}}_{Y|m}\left[\frac{1}{2}\log\Big(1 + \sigma_y^{-2} m^\top C_x m\Big)\right] \quad \Longrightarrow \quad \boxed{I(X; Y \mid m) = \frac{1}{2}\log\left(1 + \frac{m^\top C_x m}{\sigma_y^2}\right)}$$

(d) Using rules for logarithms and determinants, show that the information acquisition function (i.e., the information gained if we use the measurement m) can be written as

$$\begin{aligned} I(X; Y|m) &= \frac{1}{2}\log\Big(1 + \mathrm{SNR}(m; C_x, \sigma_y^2)\Big), \\ \text{where:} \quad \mathrm{SNR}(m; C_x, \sigma_y^2) &= \frac{m^\top C_x m}{\sigma_y^2}. \end{aligned} \tag{27}$$

We showed in part (c) that the information acquisition function is $I(X; Y|m) = \frac{1}{2}\log\left(1 + \frac{m^\top C_x m}{\sigma_y^2}\right)$, which is the desired result.

(e) Show that maximizing the information acquisition function over possible measurements is equivalent to maximizing the signal-to-noise ratio, $\mathrm{SNR}(m; C_x, \sigma_y^2)$, in the measurement, and that this is a valid signal-to-noise ratio since $m^\top C_x m$ is the variance in Y given m in the absence of noise.

Our first goal is to show that

$$\operatorname*{argmax}_{m \in \mathcal{M}} \Big\{I(X; Y|m)\Big\} = \operatorname*{argmax}_{m \in \mathcal{M}} \Big\{\mathrm{SNR}(m; C_x, \sigma_y^2)\Big\}$$

Noting that

- $\operatorname{argmax}_{x \in \mathcal{X}} \{c f(x) + d\} = \operatorname{argmax}_{x \in \mathcal{X}} \{f(x)\}$ for any postive constant $c > 0$ and any constant $d \in \mathbb{R}$ and

- $\operatorname{argmax}_{x\in\mathcal{X}}\left\{\log\big(f(x)\big)\right\} = \operatorname{argmax}_{x\in\mathcal{X}}\left\{f(x)\right\}$ if $f(x) > 0$ for all $x \in \mathcal{X}$ (because $\log(\cdot)$ is monotonically increasing),

we have that

$$\operatorname*{argmax}_{m\in\mathcal{M}}\left\{I(X;Y|m)\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{\tfrac{1}{2}\log\left(1+\frac{m^\top C_x m}{\sigma_y^2}\right)\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{\log\left(1+\frac{m^\top C_x m}{\sigma_y^2}\right)\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{1+\frac{m^\top C_x m}{\sigma_y^2}\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{\frac{m^\top C_x m}{\sigma_y^2}\right\}$$

$$\operatorname*{argmax}_{m\in\mathcal{M}}\left\{I(X;Y|m)\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{\mathrm{SNR}(m;C_x,\sigma_y^2)\right\}$$

Next, we show that $\mathrm{SNR}(m;C_x,\sigma_y^2)$ is a valid signal-to-noise ratio since $m^\top C_x m$ is the variance in Y given m in the absence of noise. In the noiseless case, $Y = m^\top X$, so we have

$$\operatorname{Var}[Y \mid m] = \operatorname{Var}[m^\top X] = m^\top \operatorname{Var}[X] m = m^\top C_x m,$$

therefore, this is a valid signal-to-noise ratio.

(f) Maximizing the signal-to-noise ratio with respect to m is equivalent to maximizing the quadratic form $m^\top C_x m$ over $\mathcal{M}$. Since covariance matrices are positive semi-definite, if $\mathcal{M}$ is convex then the solution will always lie at the boundary of the set of available measurements. Suppose, for simplicity, that $\mathcal{M} = \{\, m \mid \|m\|_2 \le 1 \,\}$. Show then that $m^\star$ must be parallel to the leading principal component of C_x (i.e. the eigenvector of C_x whose eigenvalue is the largest among all eigenvalues of C_x).

This result follows from the eigendecomposition of the covariance matrix, C_x. Let Λ be a d-by-d matrix with the eigenvalues $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_d \ge 0$ along its diagonal and let D represent the corresponding orthonormal matrix of eigenvectors such that $C_x = D\Lambda D^\top$. Then our maximization becomes

$$m^\star = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{\mathrm{SNR}(m;C_x,\sigma_y^2)\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{\frac{m^\top C_x m}{\sigma_y^2}\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{m^\top C_x m\right\} = \operatorname*{argmax}_{m\in\mathcal{M}}\left\{m^\top D\Lambda D^\top m\right\}$$

We note that since we are searching the space of unit vectors, $m \in \mathcal{M} = \{\, m \mid \|m\|_2 \le 1 \,\}$, and D is an orthonormal matrix (i.e., $\|m\|_2 = \|Dm\|_2 = \sqrt{m^\top D^\top D m} = \sqrt{m^\top m} = \|m\|_2$), an equivalent optimization problem is

$$u^\star = \operatorname*{argmax}_{\substack{u\in\mathcal{U}=\{\,u \mid \|u\|_2\le 1\,\}\\ \text{where } u = D^\top m}}\left\{u^\top \Lambda u\right\} = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^\top \in \mathbb{R}^d$$

as $u^\star$ will pick off the largest eigenvalue of C_x, i.e., $\operatorname{argmax}_{u\in\mathcal{U}}\left\{u^\top\Lambda u\right\} = \lambda_1$. It follows that

$$u^\star = D^\top m^\star \quad\Longrightarrow\quad m^\star = Du^\star = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \dots & v_d \\ | & | & & | \end{bmatrix}\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \quad\Longrightarrow\quad m^\star = v_1$$

showing that $m^\star$ is parallel to the leading principal component of C_x, v_1 (i.e. the eigenvector of C_x whose eigenvalue is the largest among all eigenvalues of C_x).

STAT238: Bayesian Statistics April 1st, 2025

Homework 3: Bias-Variance Tradeoff & Regularization via Priors

Reece D. Huff

Problems

Problem 1: Instability of the MLE

Suppose that $Y|x \sim N(Ax, C_y)$ for some $A \in \mathbb{R}^{m\times d}$, $x \in \mathbb{R}^d$, and C_y a valid $m \times m$ covariance matrix. Assume that C_y is invertible.

Let's study the MLE estimator in this case. This is an essential case since it allows explicit algebraic insight into the behavior of unstable estimators, and is the most common statistical estimator in applied problems, especially large applied problems. Getting good at this algebra is a useful applied skill.

(a) Show that the MLE estimator for x given y is the solution to the least squares problem:

$$\hat{x}_{\text{MLE}}(y) = \operatorname{argmin}\left\{\|Ax - y\|^2_{C_y^{-1}}\right\} \quad \text{where} \quad \|u\|^2_M = u^\top M u. \tag{28}$$

The MLE estimator is given by

$$\hat{x}_{\text{MLE}}(y) = \operatorname*{argmax}_{x\in\mathcal{X}}\left\{\text{Lik}(Y \mid x)\right\} = \operatorname*{argmax}_{x\in\mathcal{X}}\left\{p_{Y|x}(y \mid x)\right\} = \operatorname*{argmax}_{x\in\mathcal{X}}\left\{\frac{1}{(2\pi)^{m/2}\det(C_y)^{1/2}}\exp\left(-\frac{1}{2}(y-Ax)^\top C_y^{-1}(y-Ax)\right)\right\}$$

Noting that $\operatorname{argmax}_x\{cf(x)\} = \operatorname{argmax}_x\{f(x)\}$ and $\operatorname{argmax}_x\{f(x)\} = \operatorname{argmax}_x\{\log(f(x))\}$, we have

$$\hat{x}_{\text{MLE}}(y) = \operatorname*{argmax}_{x\in\mathcal{X}}\left\{-\frac{1}{2}(y-Ax)^\top C_y^{-1}(y-Ax)\right\}$$

Finally, we note that $\operatorname{argmax}_x\{-cf(x)\} = \operatorname{argmin}_x\{f(x)\}$, we have

$$\hat{x}_{\text{MLE}}(y) = \operatorname*{argmin}_{x\in\mathcal{X}}\left\{(y-Ax)^\top C_y^{-1}(y-Ax)\right\} = \operatorname*{argmin}_{x\in\mathcal{X}}\left\{\|Ax-y\|^2_{C_y^{-1}}\right\} \quad \checkmark$$

(b) Show that, there exists a linear change of coordinates, $\tilde{y} = Ty$ for some T invertible, $\tilde{y}|x \sim \mathcal{N}(\tilde{A}x, I_{m\times m})$ where $I_{m\times m}$ is the $m \times m$ identity and where $\tilde{A} = TA$. Use this fact to argue that, we can assume, without loss of generality, that $C_y = \sigma_y^2 I$ for some σ_y^2.[11]

We have $\text{Var}(\tilde{y}) = \text{Var}(Ty) = T\,\text{Var}(y)T^\top = TC_yT^\top = I_{m\times m} \quad \Longrightarrow \quad T = C_y^{-1/2}$.

Note that $C_y^{-1/2}$ is only possible because C_y is both invertible and therefore positive definite (not just positive semi-definite). Consider a scaled linear transformation of the form $\sigma_y C_y^{-1/2}$. It then follows that

$$\text{Var}(\sigma_y C_y^{-1/2}y) = \sigma_y^2 C_y^{-1/2}\,\text{Var}(y)C_y^{-1/2} = \sigma_y^2 C_y^{-1/2}C_yC_y^{-1/2} = \sigma_y^2 I_{m\times m}$$

Therefore, we can assume, without loss of generality, that $C_y = \sigma_y^2 I$ for some σ_y^2 as all we need to do is apply a scaled linear transformation to have $C_y = \sigma_y^2 I$.

(c) Given ill-conditioned A[12], or singular A, maximum likelihood estimation is unstable. Show that, if we write $y = Ax + \zeta$ for $\zeta \sim \mathcal{N}(0, \sigma_y^2 I)$, then:

$$\hat{x}_{\text{MLE}}(y) = x + A^\dagger\zeta \quad \text{where} \quad A^\dagger = (A^\top A)^{-1}A^\top \tag{29}$$

[11] It can be useful to keep some measure of the overall noise level, σ_y^2 to study how answers depend on the noise variance. Adopting this convention changes coordinates to "whiten" the noise, that is to consider i.i.d. Gaussian noise, while retaining a parameter that explicitly models the noise variance.

[12] A admits singular values of very different scales.

is the pseudo-inverse of A.[13] Expand the pseudo-inverse of A using its singular-value decomposition, $A = USV^\top$, to show that components of the noise vector ζ projected onto the singular vectors corresponding to the smallest singular value of A, $s_{\min}(A)$, may be greatly amplified when computing the MLE if $s_{\min}(A)$ is small.

We have

$$\|Ax - y\|^2_{\sigma_y^{-2}I} = (Ax - y)^\top \sigma_y^{-2} I (Ax - y) = \sigma_y^{-2} x^\top A^\top A x - 2\sigma_y^{-2} y^\top A x + \sigma_y^{-2} y^\top C_y^{-1} y$$

Taking the gradient with respect to x and setting it to zero, we arrive at

$$\nabla_x \|Ax - y\|^2_{\sigma_y^{-2}I} = 2\sigma_y^{-2} A^\top A x - 2\sigma_y^{-2} A^\top y = 0 \implies A^\top A x = A^\top y \implies \hat{x}_{\text{MLE}}(y) = (A^\top A)^{-1} A^\top y.$$

Now we substitute $y = Ax + \zeta$ to arrive at

$$\hat{x}_{\text{MLE}}(y) = (A^\top A)^{-1} A^\top y = (A^\top A)^{-1} A^\top (Ax + \zeta) = (A^\top A)^{-1} A^\top A x + (A^\top A)^{-1} A^\top \zeta = x + A^\dagger \zeta$$

We apply SVD to the pseudo-inverse

$$A^\dagger = (A^\top A)^{-1} A^\top = (VS \underbrace{U^\top U}_{I} SV^\top)^{-1} VSU^\top = (VS^2V^\top)^{-1} VSU^\top = VS^{-2} \underbrace{V^\top V}_{I} SU^\top = VS^{-1}U^\top \tag{30}$$

If $s_{\min}(A)$ is small then its entry in S^{-1}, i.e., $1/s_{\min}(A)$ will be large and therefore the component of the noise vector ζ projected onto the singular vectors $U^\top \zeta$ will be scaled by $1/s_{\min}(A)$. Therefore, we can conclude MLE will be unstable when A is ill-conditioned.

(d) Assume $C_y = \sigma_y^2 I$. Then, show that the standard deviation of the error in estimation, is:

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \langle A^\dagger A^{\dagger\top}, C_y \rangle^{1/2} \quad \text{where} \quad \langle B, D \rangle = \sum_{i,j} b_{ij} d_{ij}. \tag{31}$$

Then, use Equation (31) to show that, given $C_y = \sigma_y^2 I$, the standard error in the estimator is:

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \sigma_y \left\| A^\dagger \right\|_{\text{Fro}} \quad \text{where} \quad \|M\|_{\text{Fro}}^2 = \sum_{i,j} m_{ij}^2. \tag{32}$$

Then, use the fact that the Frobenius norm of a matrix is the Euclidean norm of the vector of its singular values (see wiki on matrix norm facts), to show that, the standard error in the estimator is:

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \sigma_y \sqrt{\sum_{i=1}^{\text{rank}(A)} s_i(A)^{-2}} = O\left(s_{\min}(A)^{-1} \right). \tag{33}$$

We start with

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \mathbb{E}_\zeta \left[\left\| x + A^\dagger \zeta - x \right\|_2^2 \right]^{1/2} = \mathbb{E}_\zeta \left[\left\| A^\dagger \zeta \right\|_2^2 \right]^{1/2} = \mathbb{E}_\zeta \left[\zeta^\top (A^\dagger)^\top A^\dagger \zeta \right]^{1/2}$$

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \mathbb{E}_\zeta \left[\text{Tr}\left(\zeta^\top (A^\dagger)^\top A^\dagger \zeta \right) \right]^{1/2} = \mathbb{E}_\zeta \left[\text{Tr}\left(\zeta \zeta^\top (A^\dagger)^\top A^\dagger \right) \right]^{1/2} = \text{Tr}\left(\mathbb{E}_\zeta \left[\zeta \zeta^\top \right] (A^\dagger)^\top A^\dagger \right)^{1/2}$$

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \text{Tr}\left(C_y (A^\dagger)^\top A^\dagger \right)^{1/2}$$

where we use the cyclic property of the $\text{Tr}(\cdot)$ operator and the linearity of $\mathbb{E}[\cdot]$. Note that $\langle B, D \rangle = \sum_{i,j} b_{ij} d_{ij}$ is known as the Frobenius inner product and can be equivalently expressed as $\langle B, D \rangle = \text{Tr}(B^\top D)$. Therefore, we have

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \text{Tr}\left(C_y (A^\dagger)^\top A^\dagger \right)^{1/2} = \text{Tr}\left((A^\dagger)^\top A^\dagger C_y \right)^{1/2} = \left\langle (A^\dagger)^\top A^\dagger, C_y \right\rangle^{1/2} \quad \checkmark$$

Next, we consider when $C_y = \sigma_y^2 I$ to show Equation (32). We have

$$\mathbb{E}_\zeta \left[\left\| \hat{x}_{\text{MLE}}(y) - x \right\|_2^2 \right]^{1/2} = \text{Tr}\left(\sigma_y^2 I (A^\dagger)^\top A^\dagger \right)^{1/2} = \sigma_y \, \text{Tr}\left((A^\dagger)^\top A^\dagger \right)^{1/2} = \sigma_y \left\langle A^\dagger, A^\dagger \right\rangle^{1/2} = \sigma_y \left\| A^\dagger \right\|_{\text{Fro}} \quad \checkmark$$

[13] *Hint*: Recall the normal equations used to solve any unweighted least-squares problem.

Finally, we show Equation (33) by recalling that $A^\dagger = VS^{-1}U^\top$ from Equation (30) in part (c). We have that

$$\mathbb{E}_\zeta\left[\left\|\hat{x}_{\text{MLE}}(y) - x\right\|_2^2\right]^{1/2} = \sigma_y \left\|A^\dagger\right\|_{\text{Fro}} = \sigma_y \operatorname{Tr}\left((A^\dagger)^\top A^\dagger\right)^{1/2} = \sigma_y \operatorname{Tr}\left(US^{-1}V^\top VS^{-1}U^\top\right)^{1/2} = \sigma_y \operatorname{Tr}\left(S^{-2}U^\top U\right)^{1/2}$$

$$\mathbb{E}_\zeta\left[\left\|\hat{x}_{\text{MLE}}(y) - x\right\|_2^2\right]^{1/2} = \sigma_y \operatorname{Tr}\left(S^{-2}\right)^{1/2} = \sigma_y \sqrt{\sum_{i=1}^{\operatorname{rank}(A)} s_i(A)^{-2}} \leq \sigma_y \sqrt{\operatorname{rank}(A) s_{\min}(A)^{-2}} = O\left(s_{\min}(A)^{-1}\right) \quad \checkmark$$

(e) What if we collect multiple i.i.d. samples $Y^{(n)} = \{Y_1, Y_2, \ldots, Y_n\}$ where $Y_j \sim \mathcal{N}(Ax, \sigma^2 I)$ are drawn i.i.d? Show that, this is equivalent to drawing a single sample $y^{(n)} = [y_1, y_2, \ldots, y_n] \sim \mathcal{N}(A^{(n)}x, \sigma^2 I_{nm\times nm})$ where $A^{(n)}$ is the block matrix:

$$A^{(n)} = \begin{bmatrix} A \\ A \\ \vdots \\ A \end{bmatrix}, \quad \text{and} \quad I_{nm\times nm} = \begin{bmatrix} I_{m\times m} & & & \\ & I_{m\times m} & & \\ & & \ddots & \\ & & & I_{m\times m} \end{bmatrix} \tag{34}$$

and, thus, that:

$$\hat{x}_{\text{MLE}}(y^{(n)}) = x + A^\dagger \bar{\zeta}^{(n)} \quad \text{where} \quad \bar{\zeta}^{(n)} = \frac{1}{n}\sum_{j=1}^n \zeta_j. \tag{35}$$

Consider collecting multiple i.i.d. samples. Our likelihood can be written as

$$\text{Lik}\left(x; Y^{(n)}\right) := p\left(Y^{(n)} = y^{(n)} \mid X\right) \overset{\text{i.i.d.}}{=} \prod_{i=1}^n p\left(Y_i = y_i \mid X\right) \propto \prod_{i=1}^n \exp\left(-\frac{(y_i - Ax)^\top I (y_i - Ax)}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{i=1}^n \left\|Ax - y_i\right\|_2^2}{2\sigma^2}\right)$$

Then the MLE estimator is given by

$$\hat{x}_{\text{MLE}}(y^{(n)}) = \underset{x\in\mathcal{X}}{\operatorname{argmax}} \left\{\text{Lik}\left(x; y^{(n)}\right)\right\} = \underset{x\in\mathcal{X}}{\operatorname{argmin}} \left\{-2\sigma^2 \log\left(\text{Lik}\left(x; y^{(n)}\right)\right)\right\} = \underset{x\in\mathcal{X}}{\operatorname{argmin}} \left\{\sum_{i=1}^n \left\|Ax - y_i\right\|_2^2\right\}.$$

Setting the gradient of $\sum_{i=1}^n \left\|Ax - y_i\right\|_2^2$ w.r.t. x to zero gives us our MLE estimator:

$$\nabla_x \left\{\sum_{i=1}^n \left\|Ax - y_i\right\|_2^2\right\} = \sum_{i=1}^n \left(2A^\top Ax - 2A^\top y_i\right) = 0 \implies nA^\top Ax = A^\top \sum_{i=1}^n y_i \implies \hat{x}_{\text{MLE}}(y^{(n)}) = A^\dagger\left(\frac{1}{n}\sum_{i=1}^n y_i\right) \tag{36}$$

Writing our sample mean as $\frac{1}{n}\sum_{i=1}^n y_i = \frac{1}{n}\sum_{i=1}^n Ax + \zeta_i = Ax + \frac{1}{n}\sum_{i=1}^n \zeta_i = Ax + \bar{\zeta}^{(n)}$ gives us the desired result:

$$\hat{x}_{\text{MLE}}(y^{(n)}) = A^\dagger\left(Ax + \bar{\zeta}^{(n)}\right) = x + A^\dagger \bar{\zeta}^{(n)} \quad \checkmark$$

Now we show that this is equivalent to drawing a single sample $y^{(n)} = [y_1, y_2, \ldots, y_n] \sim \mathcal{N}(A^{(n)}x, \sigma^2 I_{nm\times nm})$. The likelihood is given by

$$\text{Lik}\left(x; y^{(n)}\right) := p\left(y^{(n)} \mid X\right) \propto \exp\left(-\frac{(y^{(n)} - A^{(n)}x)^\top I_{nm\times nm}(y^{(n)} - A^{(n)}x)}{2\sigma^2}\right)$$

We note that the numerator can be rewritten as

$$(y^{(n)} - A^{(n)}x)^\top I_{nm\times nm}(y^{(n)} - A^{(n)}x) = \left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} A \\ A \\ \vdots \\ A \end{bmatrix} x\right)^\top \begin{bmatrix} I_{m\times m} & & & \\ & I_{m\times m} & & \\ & & \ddots & \\ & & & I_{m\times m} \end{bmatrix} \left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} A \\ A \\ \vdots \\ A \end{bmatrix} x\right)$$

$$(y^{(n)} - A^{(n)}x)^\top I_{nm\times nm}(y^{(n)} - A^{(n)}x) = \left(\begin{bmatrix} y_1 - Ax \\ y_2 - Ax \\ \vdots \\ y_n - Ax \end{bmatrix}\right)^\top \left(\begin{bmatrix} y_1 - Ax \\ y_2 - Ax \\ \vdots \\ y_n - Ax \end{bmatrix}\right) = \sum_{i=1}^n (y_i - Ax)^\top (y_i - Ax) = \sum_{i=1}^n \left\|Ax - y_i\right\|_2^2$$

Since the likelihoods for pulling n i.i.d. samples is the same the likelihood for pulling a single sample $y^{(n)} = [y_1, y_2, \ldots, y_n] \sim \mathcal{N}(A^{(n)}x, \sigma^2 I_{nm\times nm})$, we can conclude that our MLE estimators are the same.

(f) Use your result from part (e) to show that, given n i.i.d. observations, the standard error in the estimator is:

$$\mathbb{E}_\zeta\left[\left\|\hat{x}_{\text{MLE}}(y^{(n)}) - x\right\|_2^2\right]^{1/2} = \frac{\sigma_y}{n^{1/2}}\left\|A^\dagger\right\|_{\text{Fro}}. \tag{37}$$

Use this conclusion to argue that, to maintain a constant standard error as $s_{\min}(A)$ vanishes, you would need $n = O\left(s_{\min}(A)^{-2}\right)$ samples.

We have that

$$\mathbb{E}_\zeta\left[\left\|\hat{x}_{\text{MLE}}(y^{(n)}) - x\right\|_2^2\right]^{1/2} = \mathbb{E}\left[\left\|A^\dagger\bar{\zeta}^{(n)}\right\|_2^2\right]^{1/2} = \mathbb{E}\left[\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \zeta_i^\top (A^\dagger)^\top A^\dagger \zeta_j\right]^{1/2} = \left(\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \text{Tr}\Big(\mathbb{E}\left[\zeta_j\zeta_i^\top\right](A^\dagger)^\top A^\dagger\Big)\right)^{1/2}$$

$$\mathbb{E}_\zeta\left[\left\|\hat{x}_{\text{MLE}}(y^{(n)}) - x\right\|_2^2\right]^{1/2} = \left(\frac{n\sigma_y^2}{n^2}\text{Tr}\Big((A^\dagger)^\top A^\dagger\Big)\right)^{1/2} = \frac{\sigma_y}{n^{1/2}}\text{Tr}\Big((A^\dagger)^\top A^\dagger\Big)^{1/2} = \frac{\sigma_y}{n^{1/2}}\left\|A^\dagger\right\|_{\text{Fro}} \quad ✓$$

Since $\sigma_y\left\|A^\dagger\right\|_{\text{Fro}} = O\left(s_{\min}(A)^{-1}\right)$, we need to set $n = O\left(s_{\min}(A)^{-2}\right)$ such that

$$\mathbb{E}_\zeta\left[\left\|\hat{x}_{\text{MLE}}(y^{(n)}) - x\right\|_2^2\right]^{1/2} = \frac{\sigma_y}{n^{1/2}}\left\|A^\dagger\right\|_{\text{Fro}} = \frac{O\left(s_{\min}(A)^{-1}\right)}{O\left(s_{\min}(A)^{-2}\right)^{1/2}} = O(1) \quad ✓$$

Problem 2: Regularization via Prior Information

Problem 1 shows that, when the matrix A (or, TA if starting with $C_y \neq \sigma^2 I$), has small singular values, attempting to invert A via a MLE amplifies errors proportional to the reciprocal of the smallest singular value. Since many problems of interest use A with some very small singular values, direct MLE is hopelessly unstable for many real problems, even for reasonably large n.

Regularized estimators attempt to reduce the sample variance in the estimator by introducing a bias. The bias should move the estimator towards solutions we believe, a priori, are more plausible, or, in a direction such that the induced errors are acceptable. Commonly, naive MLE solutions can be rejected on the grounds that they return implausibly large estimated x. This leads to the general regularized estimator:

$$\hat{x}(y;\lambda,R) = \operatorname{argmin}\left\{\|Ax-y\|^2_{C_y^{-1}} + \lambda\mathcal{R}(x)\right\} \tag{38}$$

where $\mathcal{R}: X \to \mathbb{R}$ is the regularizer. In most applications, $C_y \propto I$ (see the whitening argument in Problem 1), and $\mathcal{R}(x)$ takes the form:

$$\mathcal{R}(x) = \|Rx\|_p^p \quad \text{where} \quad \|u\|_p^p = \sum_{j=1}^{d} |u_j|^p \tag{39}$$

for some $p \in [0,\infty)$. If $\mathcal{R}$ takes the form of Equation (39) then we say that we are using L_p regularization.

(a) Show that, if R is invertible, then letting $\tilde{x} = Rx$, we can solve for the regularized estimator by solving the regularized problem:

$$\hat{\tilde{x}}(y;\lambda,\|\cdot\|_p^p) = \operatorname{argmin}\left\{\|\tilde{A}\tilde{x}-y\|^2_{C_y^{-1}} + \lambda\|\tilde{x}\|_p^p\right\} \quad \text{with} \quad \tilde{A} = AR^{-1} \tag{40}$$

then setting $\hat{x} = R^{-1}\hat{\tilde{x}}$. Argue, then, that, without loss of generality[14], we can change coordinates in x and y so that the regularized least-squares problem takes the canonical form:

$$\hat{x}(y;\lambda,\|\cdot\|_p^p) = \operatorname{argmin}\left\{\|Ax-y\|_2^2 + \lambda\|x\|_p^p\right\}. \tag{41}$$

Consider the regularized estimator with L_p regularization after we have whitened our data such that $C_y = I$. Note that $C_y = I$ implies that $\|\cdot\|^2_{C_y^{-1}} = \|\cdot\|_2^2$. We have

$$\hat{x}(y;\lambda,\|\cdot\|_p^p) = \operatorname*{argmin}_{x\in\mathcal{X}}\left\{\Big\| \underbrace{Ax}_{AR^{-1}\tilde{x}=\tilde{A}\tilde{x}} - y\Big\|_2^2 + \lambda\Big\|\underbrace{Rx}_{\tilde{x}}\Big\|_p^p\right\} \quad \Leftrightarrow \quad \hat{\tilde{x}}(y;\lambda,\|\cdot\|_p^p) = \operatorname*{argmin}_{\tilde{x}\in\tilde{\mathcal{X}}}\left\{\|\tilde{A}\tilde{x}-y\|_2^2 + \lambda\|\tilde{x}\|_p^p\right\}$$

Therefore by applying the transformation $\tilde{x} = Rx$, we can optimize for $\tilde{x} \in \tilde{\mathcal{X}}$ in the canonical form instead of optimizing for $x \in \mathcal{X}$ in a non-canonical form. Once we optimize for $\tilde{x} \in \tilde{\mathcal{X}}$, all we need to do is reverse the transformation, i.e., $\hat{x} = R^{-1}\hat{\tilde{x}}$. Therefore, much like the whitening argument in Problem 1, we can change coordinates in x and y so that the regularized least-squares problem takes the canonical form:

$$\hat{x}(y;\lambda,\|\cdot\|_p^p) = \operatorname{argmin}\left\{\|Ax-y\|_2^2 + \lambda\|x\|_p^p\right\} \quad \checkmark$$

(b) Adopting a regularized estimator expresses a preference relation over estimation procedures. In particular, it adopts a notion of a best estimator. As usual, any preference relation over decision procedures, can, with a fixed likelihood, be expressed equivalently as a Bayesian estimation procedure for a matched choice of prior. Show that:

1. Solving for the regularized estimator defined by Equation (41) is the same, for any λ, as solving for:

$$\hat{x}(y;\delta,\|\cdot\|_p^p) = \operatorname{argmin}\left\{\|Ax-y\|_2^2 \text{ subject to } \|x\|_p \leq \delta\right\} \tag{42}$$

[14] Beware what is hidden when we claim WLOG. Reducing problems without loss of generality to canonical forms is useful for analysis, but hides explicit degrees of freedom in the model implicitly in the terms that collect the necessary coordinate transformations. Here, the matrix A is storing all the information about C_y and R that was dropped by changing coordinates.

for some δ that depends on λ. For what choice of prior is $\hat{x}(y;\delta,\|\cdot\|_p^p)$ the MAP estimator? In what sense is this a weakly informative prior? In what sense is it strongly informative?

We let the prior be uniform over an L^p-ball of radius δ:

$$\boxed{p_1(x) \propto \mathbb{1}\left\{\|x\|_p \leq \delta\right\} = \begin{cases} 1 & \text{when } \|x\|_p \leq \delta, \\ 0 & \text{otherwise.} \end{cases}}$$

In this form, we would have that the MAP estimator is

$$\hat{x}(y;\delta,\|\cdot\|_p^p) := \operatorname{argmax}\left\{p_{X|Y}(x \mid y)\right\} = \operatorname{argmin}\left\{-\log\big(p_{X|Y}(x \mid y)\big)\right\} = \operatorname{argmin}\left\{-\log\big(p_{Y|X}(x)\big) - \log\big(p_1(x)\big)\right\}$$
$$\hat{x}(y;\delta,\|\cdot\|_p^p) = \operatorname{argmin}\left\{\|Ax-y\|_2^2 - \log\big(p_1(x)\big)\right\} \Leftrightarrow \operatorname{argmin}\left\{\|Ax-y\|_2^2 \text{ subject to } \|x\|_p \leq \delta\right\}$$

as our the negative log of our prior will be ∞ if x is outside the L^p-ball.

- **Weakly informative:** The prior is weakly informative as it is uniform over the L^p-ball.
- **Strongly informative:** The prior is strongly informative as it only admits solutions within the L^p-ball. Suppose that our MLE estimate is slightly outside of the boundary of the ball. The prior provides a firmly threshold above which we throw out the solution, so we would pick the nearest solution that is on the boundary of the ball (i.e., $\|\hat{x}\|_p = \delta$) for the MAP estimator.

2. The previous equivalence only matches the optimal estimation procedure under two different approaches. If we adopt the objective function $f(x,y) = \|Ax-y\|_2^2 + \lambda\mathcal{R}(x)$ as a measure of the quality of an estimate, x, and prefer any estimate with a smaller objective, then the regularization implicitly expresses a complete prior. Find a prior such that, the preference rule, "given y, prefer x to x' iff $p(x|y) > p(x'|y)$"[15], is the same as adopting the preference rule, "given y, prefer x to x' iff $f(x,y) < f(x',y)$". Show that the MAP estimator under this prior is the solution to the regularized least squares problem.

This prior is of the form:

$$\boxed{p_2(x) = \exp\left(-\frac{\lambda}{2}\mathcal{R}(x)\right) \propto \exp(-\lambda\mathcal{R}(x))}$$

The MAP estimator under this prior is given by

$$\hat{x}_{\text{MAP}}(x,\lambda,\mathcal{R}(x)) := \operatorname{argmax}\left\{p_{X|Y}(x \mid y)\right\} = \operatorname{argmin}\left\{-\log\big(p_{X|Y}(x \mid y)\big)\right\} = \operatorname{argmin}\left\{-\log\big(p_{Y|X}(x)\big) - \log\big(p_2(x)\big)\right\}$$
$$\hat{x}_{\text{MAP}}(x,\lambda,\mathcal{R}(x)) = \operatorname{argmin}\left\{\|Ax-y\|_2^2 + \lambda\mathcal{R}(x)\right\} = \operatorname{argmin}\left\{f(x,y)\right\}$$

Therefore, we see that $\hat{x}_{\text{MAP}}$ is equal to the preference rule of "given y, prefer x to x' iff $p(x|y) > p(x'|y)$" as $\hat{x}_{\text{MAP}} = \operatorname{argmax}\{p(x|y)\}$ which is the same as adopting the preference rule, "given y, prefer x to x' iff $f(x,y) < f(x',y)$" as $\hat{x}_{\text{MAP}} = \operatorname{argmin}\{f(x,y)\}$.

(c) Which of these two prior specifications do you prefer? Which more sensibly represents the regularized estimator as an optimal Bayesian estimator? Which expresses stronger beliefs about the unknown x?

I prefer the prior from part 2., $p_2(x) \propto \exp(-\lambda\mathcal{R}(x))$. I think this is a more natural prior as it admits any possible x where as the prior from part 1., $p_1(x) \propto \mathbb{1}\left\{\|x\|_p \leq \delta\right\}$, only admits solutions within the L^p-ball of radius δ. As such, the prior $p_2(x)$ leads to a *consistent*[16] MAP estimator, whereas $p_1(x)$ does not. Consider the example where $\|x^\star\| > \delta$. Then even as $n \to \infty$, the MAP estimator derived from $p_1(x)$ does not converge to $x^\star$. In this sense, $p_2(x)$ more sensibly represents the regularized estimator as an optimal Bayesian estimator.

A downside to $p_2(x)$ is that it prefers solutions x that are smaller in magnitude and therefore is a non-uniform prior whereas $p_1(x)$ is a uniform prior. In this sense, $p_2(x)$ is expressing stronger beliefs about the unknown x.

[15] that is, prefer estimates with higher posterior probability

[16] By consistent we mean that $\hat{x}_{\text{MAP}} \to \hat{x}_{\text{MLE}} \to x^\star$ as $n \to \infty$.

Problem 3: Bias in Regularized Estimators

How does regularization bias our estimates?

As in Problem 1, we will assume, WLOG (but with some loss of explicit clarity), that $C_y = \sigma^2 I$, and that $R = I$. We will, for this problem, assume that $p = 2$.

(a) Show that the L_2 regularized estimator:

$$\hat{x}(y;\lambda) = \operatorname{argmin}\left\{\|Ax - y\|_2^2 + \lambda \|x\|_2^2\right\} \tag{43}$$

returns the MAP estimator for a normal, normal model, where $x \sim \mathcal{N}(0, C_x)$ for some choice of C_x that depends on λ.

Recall our $\mathcal{N}$-$\mathcal{N}$ model:

$$\textbf{Prior:}\quad X \sim \mathcal{N}(0, C_x) \quad \text{and} \quad \textbf{Likelihood:}\quad Y \mid X = x \sim \mathcal{N}(Ax, C_y) \text{ where } Y = Ax + \zeta \text{ and } \zeta \sim \mathcal{N}(0, C_y)$$

It follows that posterior is proportional to

$$p_{X|Y}(x) \propto p_{Y|X}(x) \cdot p_X(x) = \exp\left(-\frac{1}{2}\left[(y - Ax)^\top C_y^{-1}(y - Ax) + x^\top C_x^{-1} x\right]\right) = \exp\left(-\frac{1}{2}\left[\frac{1}{\sigma^2}\|Ax - y\|_2^2 + x^\top C_x^{-1} x\right]\right)$$

The MAP estimator is the maximizer of the posterior. Recalling that $\operatorname{argmax}_x\{f(x)\} = \operatorname{argmin}_x\{-c\log(f(x))\}$, we have

$$\hat{x}_{\text{MAP}}(y;\lambda) = \operatorname{argmax}\{p_{X|Y}(x)\} = \operatorname{argmin}\{-2\log(p_{X|Y}(x))\} = \operatorname{argmin}\left\{\frac{1}{\sigma^2}\|Ax - y\|_2^2 + x^\top C_x^{-1} x\right\}$$

Therefore, we set $C_x^{-1} = \frac{\lambda}{\sigma^2} I$ such that

$$\hat{x}_{\text{MAP}}(y;\lambda) = \operatorname{argmin}\left\{\frac{1}{\sigma^2}\left(\|Ax - y\|_2^2 + \lambda\|x\|_2^2\right)\right\} = \operatorname{argmin}\left\{\|Ax - y\|_2^2 + \lambda\|x\|_2^2\right\}$$

So we conclude that the MAP estimator of our $\mathcal{N}$-$\mathcal{N}$ model with $\boxed{C_x = \dfrac{\sigma^2}{\lambda} I}$ is equivalent to L_2 regularized estimator.

(b) The L_2 regularized estimator admits an explicit analytic solution. Show that the L_2 regularized estimator is the solution to the augmented least squares problem:

$$\hat{x}(y;\lambda) = \operatorname{argmin}\left\{\|\hat{A}(\lambda)x - \hat{y}\|_2^2\right\} \tag{44}$$

where:

$$\hat{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \quad \text{and} \quad \hat{y} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{45}$$

We start by expanded the augmented norm. We have

$$\|\hat{A}(\lambda)x - \hat{y}\|_2^2 = \left(\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix}\right)^\top \left(\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix}\right) = x^\top \begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - 2\begin{bmatrix} y^\top & 0 \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x + \begin{bmatrix} y^\top & 0 \end{bmatrix} \begin{bmatrix} y \\ 0 \end{bmatrix}$$

$$\|\hat{A}(\lambda)x - \hat{y}\|_2^2 = x^\top A^\top A x + \lambda x^\top x - 2y^\top A x + y^\top y = (Ax - y)^\top (Ax - y) + \lambda\|x\|_2^2 = \|Ax - y\|_2^2 + \lambda\|x\|_2^2.$$

Since the augmented norm is equal to the regularized objective, we have

$$\|\hat{A}(\lambda)x - \hat{y}\|_2^2 = \|Ax - y\|_2^2 + \lambda\|x\|_2^2 \quad \Longrightarrow \quad \hat{x}(y;\lambda) = \operatorname{argmin}\left\{\|\hat{A}(\lambda)x - \hat{y}\|_2^2\right\} = \operatorname{argmin}\left\{\|Ax - y\|_2^2 + \lambda\|x\|_2^2\right\}$$

(c) By converting into a standard least squares problem, we can borrow the standard solution, which follows from the augmented normal equations:

$$\hat{A}^\top \hat{A}\hat{x} = \hat{A}^\top \hat{y} \tag{46}$$

Show that the augmented normal equations are solved by:

$$\hat{x}(y,\lambda) = (A^\top A + \lambda I)^{-1} A^\top y \tag{47}$$

then show that this equation also returns the posterior mean estimator for $x|y$ under an appropriate normal, normal model (see part (a)).

We have that

$$\hat{A}^\top \hat{A}\hat{x} = \hat{A}^\top \hat{y} \quad \Longrightarrow \quad \hat{x}(y,\lambda) = (\hat{A}^\top \hat{A})^{-1}\hat{A}^\top \hat{y} = \left(\begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}\right)^{-1} \begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} y \\ 0 \end{bmatrix} = (A^\top A + \lambda I)^{-1} A^\top y \quad \checkmark$$

In part (a), we showed that the MAP estimator of our $\mathcal{N}$-$\mathcal{N}$ model with $C_x = \frac{\sigma^2}{\lambda} I$ is given by

$$\hat{x}_{\text{MAP}}(y;\lambda) = \operatorname{argmin}\left\{ \|Ax - y\|_2^2 + \lambda \|x\|_2^2 \right\}$$

Setting the gradient of $\|Ax - y\|_2^2 + \lambda\|x\|_2^2$ with respect to x to zero, we have

$$\nabla_x \left\{ \|Ax - y\|_2^2 + \lambda \|x\|_2^2 \right\} = \nabla_x \left\{ x^\top A^\top A x - 2y^\top A x + y^\top y + \lambda x^\top x \right\} = 2A^\top A x - 2A^\top y + 2\lambda x = 0$$
$$\Longrightarrow \quad (A^\top A + \lambda I)x = A^\top y \quad \Longrightarrow \quad \hat{x}_{\text{MAP}}(y;\lambda) = (A^\top A + \lambda I)^{-1} A^\top y. \quad \checkmark$$

(d) Extend this solution to the case when we draw n i.i.d. vectors y using the same arguments from Problem 1 part (e). Show that:

$$\hat{x}(y^{(n)},\lambda) = \left(A^\top A + \frac{\lambda}{n} I\right)^{-1} A^\top \bar{y}^{(n)} \quad \text{where} \quad \bar{y}^{(n)} = \frac{1}{n}\sum_{j=1}^{n} y_j \tag{48}$$

In Problem 1 part (e), we showed that sampling n i.i.d. vectors changed our objective from $\|Ax - y\|_2^2$ to $\sum_{i=1}^n \|Ax - y_i\|_2^2$. Applying this to our MAP objective we have that

$$\hat{x}_{\text{MAP}}(y^{(n)};\lambda) = \operatorname{argmin}\left\{ \sum_{i=1}^{n} \|Ax - y_i\|_2^2 + \lambda \|x\|_2^2 \right\}$$

Setting the gradient of $\sum_{i=1}^n \|Ax - y_i\|_2^2 + \lambda\|x\|_2^2$ w.r.t. x to zero and solving for the MAP estimator results in

$$\nabla_x \left\{ \sum_{i=1}^{n} \|Ax - y_i\|_2^2 + \lambda \|x\|_2^2 \right\} = \sum_{i=1}^{n} \left(2A^\top A x - 2A^\top y_i\right) + 2\lambda x = 0 \quad \Longrightarrow \quad nA^\top A x + \lambda x = A^\top \sum_{i=1}^{n} y_i$$
$$\Longrightarrow \quad \boxed{\hat{x}_{\text{MAP}}(y^{(n)};\lambda) = \left(A^\top A + \frac{\lambda}{n} I\right)^{-1} A^\top \bar{y}^{(n)}}$$

(e) Use Equation (48) to solve for the bias:

$$b(x,\lambda) = \mathbb{E}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)},\lambda) - x\right] \tag{49}$$

Show that the bias takes the form $M(\lambda, n)x$ for some matrix $M(\lambda, n)$ that depends on λ, A, and n. Then, expand using the singular value decomposition of A, $A = USV^\top$ to show the following facts about the bias:[17]

Recall that $\bar{y}^{(n)} = Ax + \bar{\zeta}^{(n)}$. Then the bias of the L_2 regularized estimator in Equation (48) may be written as

$$b(x,\lambda) = \mathbb{E}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)},\lambda) - x\right] = \mathbb{E}_{Y^{(n)}|x}\left[\left(A^\top A + \frac{\lambda}{n} I\right)^{-1} A^\top \left(Ax + \bar{\zeta}^{(n)}\right) - x\right] = \left[\left(A^\top A + \frac{\lambda}{n} I\right)^{-1} A^\top A - I\right] x$$

[17] It may help to rotate your coordinate system for x by working in the coordinate system $\tilde{x} = V^\top x$. This will diagonalize the problem.

as $\mathbb{E}_{Y^{(n)}|x}\left[\bar{\zeta}^{(n)}\right] = 0$. Then we apply the SVD of A, e.g., $A^\top A = VSU^\top USV^\top = VS^2V^\top$ to expand:

$$b(x,\lambda) = \left[\left(VS^2V^\top + \frac{\lambda}{n}I\right)^{-1} VS^2V^\top - I\right]x = \left[\left(V\left[S^2 + \frac{\lambda}{n}I\right]V^\top\right)^{-1} VS^2V^\top - I\right]x = \left[V\left[S^2 + \frac{\lambda}{n}I\right]^{-1}\underbrace{V^\top V}_{I}S^2V^\top - I\right]x$$

$$b(x,\lambda) = \left[V\left[S^2 + \frac{\lambda}{n}I\right]^{-1}S^2V^\top - I\right]x = \left(V\left[\underbrace{\left[S^2 + \frac{\lambda}{n}I\right]^{-1}S^2 - I}_{M(\lambda,n)}\right]V^\top\right)x = VMV^\top x = M\tilde{x} \quad \text{where} \quad \tilde{x} = V^\top x.$$

Let's continue working on the matrix $M(\lambda, n)$. Let $r = \operatorname{rank}(A)$ and $\operatorname{diag}(\cdot) : \mathbb{R}^r \to \mathbb{R}^{r\times r}$ such that

$$S := \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_r \end{bmatrix} = \operatorname{diag}(s_i) \quad \text{for } i \in \{1, 2, \ldots, r\}.$$

Then we have

$$M(\lambda,n) := \left[S^2 + \frac{\lambda}{n}I\right]^{-1}S^2 - I = \operatorname{diag}\left(s_i^2 + \frac{\lambda}{n}\right)^{-1}\operatorname{diag}\left(s_i^2\right) - I = \operatorname{diag}\left(\frac{s_i^2}{s_i^2 + \frac{\lambda}{n}} - 1\right) = \operatorname{diag}\left(\frac{-\frac{\lambda}{n}}{s_i^2 + \frac{\lambda}{n}}\right) = \operatorname{diag}\left(\frac{-\lambda}{ns_i^2 + \lambda}\right)$$

implying that our bias in the rotated coordinate system is

$$b(\tilde{x},\lambda) = \operatorname{diag}\left(\frac{-\lambda}{ns_i^2 + \lambda}\right)\tilde{x} \tag{50}$$

1. The bias is proportional to $\|x\|_2$, and, the relative bias $\mathbb{E}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)},\lambda) - x\right]/\|x\|_2$ is constant in $\|x\|_2$ for fixed $x/\|x\|_2$. That is, the relative bias depends only on the direction of the vector x, but the absolute bias grows with larger x.

   This follows directly from Equation (50). Note that $\|\tilde{x}\|_2 = \|V^\top x\|_2 = \sqrt{x^\top VV^\top x} = \sqrt{x^\top x} = \|x\|_2$. Then it follows

   $$\|b(\tilde{x},\lambda)\|_2 \le r\frac{\lambda}{ns_{\min}^2 + \lambda}\|x\|_2 = O\left(\|x\|_2\right) \quad \Longrightarrow \quad b(x,\lambda) \propto \|x\|_2$$

   Similarly, the relative bias only depends on the direction of x:

   $$\mathbb{E}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)},\lambda) - x\right]/\|x\|_2 = \frac{M\tilde{x}}{\|\tilde{x}\|_2} = \operatorname{diag}\left(\frac{-\lambda}{ns_i^2 + \lambda}\right)\operatorname{direction}(x)$$

   as the direction of x will dictate which diagonal elements of M will be in the relative bias.

2. Find the direction $x/\|x\|_2$ of inputs that produce the largest bias (in magnitude). Show that it aligns to a singular vector of A, and that, the direction associated with the largest bias (strongest regularization effect/shrinkage) is the direction that was expected to contribute the most to the expected magnitude of the error in the MLE estimator (the direction that is most amplified by $A^\dagger$, and, that is least well-resolved by the likelihood alone).

   Note that the largest element of M is associated with the smallest singular value of A, i.e., $\max_i\left\{\frac{-\lambda}{ns_i^2+\lambda}\right\} = \frac{-\lambda}{ns_{\min}^2+\lambda}$. Therefore, we select a direction $x/\|x\|_2$ that picks off this element of M. Let $x^\star = v_{\min}$ where $v_{\min}$ is the singular vector corresponding to the smallest singular value of A. Then we have

   $$b(x,\lambda) = VMV^\top x^\star = VMV^\top v_{\min} = \frac{-\lambda}{ns_{\min}^2 + \lambda}v_{\min} = \frac{-\lambda}{ns_{\min}^2 + \lambda}x^\star$$

   Recall from Problem 1 part (f) that the error in the MLE estimator is given by $\frac{\sigma_y}{n^{1/2}}\left\|A^\dagger\right\|_{\text{Fro}}$. We note that the direction $v_{\min}$ is the same direction contributes most to this error (i.e., maximizes $\left\|\left(A^\dagger\right)^\top x\right\|_2^2$). Recall that $A^\dagger = VS^{-1}U^\top$ from Equation (30). We have that

   $$\max_{x\in\mathcal{X}}\left\{\left\|\left(A^\dagger\right)^\top x\right\|_2^2\right\} = v_{\min}^\top VS^{-1}U^\top US^{-1}V^\top v_{\min} = \frac{1}{s_{\min}^2(A)}.$$

3. The regularized estimator is always conservative in the sense that the bias always has a negative projection onto x. That is, the regularization "shrinks" the estimator. To study the "shrinkage" induced by the regularization, decompose x onto the singular vectors of A. Show that the bias in every component "shrinks" each component, the degree of shrinkage (magnitude) of the bias is monotonically increasing in λ, and is monotonically decreasing in the associated singular value.

   This follows directly from Equation (50). The component-wise bias is given by

$$b(\tilde{x}_i, \lambda) = \frac{-\lambda}{ns_i^2 + \lambda}\tilde{x}_i \tag{51}$$

   which is

   - monotonically increasing in λ (in magnitude for $\lambda > 0$) and
   - monotonically decreasing in the singular value of A, $s_i(A)$.

4. Show that the degree of shrinkage (magnitude of the bias) along each component is monotonically decreasing in n. In particular, show that the estimator is asymptotically unbiased for large n, converges to the MLE, and, thus, must be consistent.

   This follows directly from the component wise bias in Equation (51) which is monotonically decreasing in n. In addition, the limit of the bias is given by:

$$\lim_{n\to\infty}\{b(x,\lambda)\} = V \lim_{n\to\infty}\{M(\lambda, n)\} V^\top x = V \lim_{n\to\infty}\left\{\operatorname{diag}\left(\frac{-\lambda}{ns_i^2 + \lambda}\right)\right\} V^\top x = \begin{bmatrix}0 & 0 & \dots & 0\end{bmatrix}^\top$$

   showing that our regularized estimator is consistent.

5. Argue that, if, as is generally true, the singular values of A span many orders of magnitude, then, for most reasonable λ there exist a set of directions $\mathcal{J}$ along which $\lambda \gg ns_j(A)^2$ if $j \in \mathcal{J}$ and a set of directions j along which $\lambda \ll ns_j(A)^2$ if $j \notin \mathcal{J}$. Show that, along the former directions, the expected regularized estimator is strongly shrunk to zero, while along the latter the expected regularized estimator is essentially unbiased. How are the strongly shrunk directions related to the directions that made the MLE unstable?[18]

   Note that our component-wise bias in Equation (51) is approximately

$$b(\tilde{x}_j, \lambda) = \frac{-\lambda}{ns_j^2 + \lambda}\tilde{x}_j \approx \begin{cases} -\tilde{x}_j & \text{for } j \in \mathcal{J} \text{ as } \lambda \gg ns_j(A)^2, \text{ and} \\ 0 & \text{for } j \notin \mathcal{J} \text{ as } \lambda \ll ns_j(A)^2. \end{cases}$$

   which implies that components of the expected regularized estimator are given by

$$\left(\mathbb{E}_{Y^{(n)}|\tilde{x}}\left[\hat{x}(Y^{(n)}, \lambda)\right]\right)_j = \begin{cases} 0 & \text{for } j \in \mathcal{J} \text{ as } \lambda \gg ns_j(A)^2, \text{ and} \\ \tilde{x}_j & \text{for } j \notin \mathcal{J} \text{ as } \lambda \ll ns_j(A)^2. \end{cases}$$

   The strongly shrunken directions correspond to directions along singular vectors with small corresponding singular values, i.e., $\lambda \ll ns_j(A)^2$. These are the same directions that causes MLE to be unstable in Problem 1.

[18] This is standard for most regularized least squares problems. The estimator is essentially unbiased along a subset of directions where the likelihood is sharply resolved and is highly biased towards zero along a subset of directions where the likelihood is not specific. It also reveals a general phenomena in practical Bayesian inference with weakly informative priors: the likelihood dominates along some well resolved direction, while the posterior is constrained in the remaining directions by the prior.

Problem 4: Variance-Bias Trade-offs

What λ should we use?

Increasing λ stabilizes inference, but expresses stronger beliefs/constraints about/on the unknown, so produces larger biases. This is a bias-variance trade-off.

A variety of heuristic rules are widely used to balance the bias-variance trade-off. For example, Morozov's discrepancy principle suggests that the user should pick λ as large as possible while keeping the likelihood of $\hat{x}(y;\lambda,\mathcal{R})$ sufficiently large. This usually amounts to starting with large λ, and gradually decreasing λ, until the discrepancy between $A\hat{x}(y;\lambda)$ and y is plausibly small (that is, on the scale expected under $y \sim \mathcal{N}(Ax, C_y)$). This approach adopts the heuristic "*regularize as much as possible while retaining fidelity to the data.*"

In this problem, you will show that, if the regularized estimator is derived under a Bayesian approach, then the prior induces a λ that optimally balances the estimation errors introduced by bias and variance. We will analyze the bias-variance trade-off in a L_2 regularized estimator, then show that, if we adopted a normal prior on x, the regularization parameter λ can be selected automatically based on our prior beliefs.

This problem proceeds under the same assumptions as Problem 3.

(a) Compute the sample covariance $\mathrm{Var}_{Y^{(n)}|x}[\hat{x}(Y^{(n)},\lambda)]$.

The sample covariance is given by

$$\underset{Y^{(n)}|x}{\mathrm{Var}}[\hat{x}(Y^{(n)},\lambda)] = \underset{Y^{(n)}|x}{\mathrm{Var}}\left[\left(A^\top A + \frac{\lambda}{n}I\right)^{-1} A^\top \left(Ax + \bar{\zeta}^{(n)}\right)\right] = \underset{Y^{(n)}|x}{\mathrm{Var}}\left[\left(A^\top A + \frac{\lambda}{n}I\right)^{-1} A^\top \bar{\zeta}^{(n)}\right]$$

Let $N := \left(A^\top A + \frac{\lambda}{n}I\right)^{-1} A^\top$. Then we have

$$\underset{Y^{(n)}|x}{\mathrm{Var}}[\hat{x}(Y^{(n)},\lambda)] = N \underset{Y^{(n)}|x}{\mathrm{Var}}\left[\bar{\zeta}^{(n)}\right] N^\top = N\left(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{i=1}^{n}\underset{Y^{(n)}|x}{\mathbb{E}}\left[\zeta_i\zeta_j^\top\right]\right)N^\top = N\left(\frac{n\sigma^2}{n^2}I\right)N^\top = \frac{\sigma^2}{n}NN^\top$$

$$\boxed{\underset{Y^{(n)}|x}{\mathrm{Var}}[\hat{x}(Y^{(n)},\lambda)] = \frac{\sigma^2}{n}\left(A^\top A + \frac{\lambda}{n}I\right)^{-1} A^\top A \left(A^\top A + \frac{\lambda}{n}I\right)^{-1}}$$

(b) Use the singular value decomposition of A to show that the sample covariance is aligned to the singular vectors of A (the principal components, e.g. eigenvectors of the covariance are parallel to the singular vectors of A). Provide a formula for the standard deviation along each principal component of the sample covariance.

Recall that $A^\top A = VS^2V^\top$ and $\left(A^\top A + \frac{\lambda}{n}I\right)^{-1} = V\left(S^2 + \frac{\lambda}{n}I\right)^{-1}V^\top$. It then follows that

$$\underset{Y^{(n)}|x}{\mathrm{Var}}[\hat{x}(Y^{(n)},\lambda)] = \frac{\sigma^2}{n}\left(V\left(S^2+\frac{\lambda}{n}I\right)^{-1}V^\top\right)VS^2V^\top\left(V\left(S^2+\frac{\lambda}{n}I\right)^{-1}V^\top\right) = \frac{\sigma^2}{n}V\left(S^2+\frac{\lambda}{n}I\right)^{-1}S^2\left(S^2+\frac{\lambda}{n}I\right)^{-1}V^\top = V\Sigma V^\top$$

$$\text{where} \quad \Sigma := \mathrm{diag}\left(\frac{\sigma^2 s_i^2}{n(s_i^2+\frac{\lambda}{n})^2}\right)$$

Clearly the eigenvectors of the sample variance are parallel to the singular vectors of A as we can write the sample variance in the form $V\Sigma V^\top$ where V contains the singular vectors of A. The standard deviation along each principal component of the sample covariance is given by

$$\boxed{\mathrm{SD}\left(\underset{Y^{(n)}|x}{\mathrm{Var}}[\hat{x}(Y^{(n)},\lambda)]\right) \text{ along } v_i = \frac{\sigma s_i}{\sqrt{n}(s_i^2+\frac{\lambda}{n})} = \frac{\sqrt{n}\sigma s_i}{ns_i^2+\lambda}}$$

(c) Using your result from part (b), show that:

1. The sample standard deviations are ordered in decreasing order from the smallest singular values of A to the largest.

   Let us order the singular values of A from smallest to largest, i.e., $s_{\min}(A) \le s_2(A) \le \cdots \le s_{\max}(A)$. Then the sample standard deviations along the principal directions $\frac{\sigma s_i}{\sqrt{n}(s_i^2+\frac{\lambda}{n})}$ are in decreasing order:

$$\frac{\sqrt{n}\sigma s_{\min}}{ns_{\min}^2+\lambda} \ge \frac{\sqrt{n}\sigma s_2}{ns_2^2+\lambda} \ge \cdots \ge \frac{\sqrt{n}\sigma s_{\max}}{ns_{\max}^2+\lambda}$$

2. Increasing λ monotonically reduces the sample standard deviation along each principal component. Thus, while larger λ leads to larger bias, it always reduces the sample variance in the estimator.

   This follows directly from our formula for the standard deviations along the principal directions, $\frac{\sqrt{n}\sigma s_i}{ns_i^2+\lambda}$.

   As λ increases, the sample variance decreases monotonically.

3. For large n, the sample standard deviations are all $O(n^{-1/2})$, and, increasing n reduces the regularizing effect of λ on the sample covariance in the estimator.

   In the limit, the sample standard deviations along the principal directions is given by

$$\lim_{n\to\infty}\left\{\frac{\sqrt{n}\sigma s_i}{ns_i^2+\lambda}\right\} = O(n^{-1/2})$$

   As we increase n we notice that the denomator simplifies to $\approx ns_i^2$ as $ns_i^2 \gg \lambda$ as $n \to \infty$. Therefore, we conclude that increasing n reduces the regularizing effect of λ on the sample covariance in the estimator.

(d) Show that, by explicitly expanding the error $\hat{x} - x = \hat{x} - \mathbb{E}[\hat{x}] + \mathbb{E}[\hat{x}] - x$, for any estimator, whose sampling covariance is independent of the true x, the expected standard error in the estimator squared, $\mathbb{E}_{Y^{(n)}|x}[\|\hat{x}(Y^{(n)}) - x\|^2]$, can be expanded:

$$\mathbb{E}_X\left[\mathbb{E}_{Y^{(n)}|x}\left[\left\|\hat{x}(Y^{(n)}) - x\right\|^2\right]\right] = \mathbb{E}_X\left[\|b(X)\|^2\right] + \mathrm{Tr}\left(\underset{Y^{(n)}|x}{\mathrm{Var}}\left[\hat{x}(Y^{(n)})\right]\right). \tag{52}$$

Interpret the left hand side as a standard notion of risk (choose Bayesian, posterior, or frequentist and specify the loss function used), to show that a natural notion of risk can be decomposed into the expected magnitude of the bias plus the sampling variance in the estimator. This establishes the bias-variance trade-off explicitly since increasing λ increases the first term, while decreasing the second.

This follows directly from the decomposition of the ℓ_2 norm. Consider a random vector $Z \in \mathbb{R}^n$. Then the trace of its covariance is given by

$$\mathrm{Var}(Z) = \mathbb{E}[ZZ^\top] - \mathbb{E}[Z]\,\mathbb{E}[Z^\top] \quad\Longrightarrow\quad \mathrm{Tr}(\mathrm{Var}(Z)) = \mathrm{Tr}\big(\mathbb{E}[ZZ^\top] - \mathbb{E}[Z]\,\mathbb{E}[Z^\top]\big) = \mathrm{Tr}\big(\mathbb{E}[ZZ^\top]\big) - \mathrm{Tr}\big(\mathbb{E}[Z]\,\mathbb{E}[Z^\top]\big)$$

as $\mathrm{Tr}(A \pm B) = \mathrm{Tr}(A) \pm \mathrm{Tr}(B)$. Recall that both $\mathrm{Tr}(\cdot)$ and $\mathbb{E}[\cdot]$ are linear operators so we have $\mathrm{Tr}(\mathbb{E}[\cdot]) = \mathbb{E}[\mathrm{Tr}(\cdot)]$. We also have the cyclic property of $\mathrm{Tr}(\cdot)$, e.g., $\mathrm{Tr}(ZZ^\top) = \mathrm{Tr}(Z^\top Z)$. So we have

(i.) $\mathbb{E}[\mathrm{Tr}(ZZ^\top)] = \mathbb{E}[\mathrm{Tr}(Z^\top Z)] = \mathbb{E}[\mathrm{Tr}(\|Z\|_2^2)] = \mathbb{E}[\|Z\|_2^2\,\mathrm{Tr}([1])] = \mathbb{E}[\|Z\|_2^2]$ and

(ii.) $\mathrm{Tr}(\mathbb{E}[Z]\,\mathbb{E}[Z^\top]) = \mathrm{Tr}(\mathbb{E}[Z^\top]\,\mathbb{E}[Z]) = \mathbb{E}[Z^\top]\,\mathbb{E}[Z]\,\mathrm{Tr}([1]) = \mathbb{E}[Z^\top]\,\mathbb{E}[Z] = \|\mathbb{E}[Z]\|_2^2$.

Taken together, we have

$$\mathrm{Tr}(\mathrm{Var}(Z)) = \mathbb{E}\left[\|Z\|_2^2\right] - \|\mathbb{E}[Z]\|_2^2 \quad\Longrightarrow\quad \boxed{\mathbb{E}\left[\|Z\|_2^2\right] = \|\mathbb{E}[Z]\|_2^2 + \mathrm{Tr}(\mathrm{Var}(Z))} \tag{53}$$

Let us apply Equation (53) to the vector $\hat{x}(Y^{(n)}) - x$ over the distribution of the likelihood $Y^{(n)}|x$:

$$\mathbb{E}_{Y^{(n)}|x}\left[\left\|\hat{x}(Y^{(n)}) - x\right\|^2\right] = \Big\|\underbrace{\mathbb{E}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)}) - x\right]}_{b(x)}\Big\|^2 + \mathrm{Tr}\left(\underset{Y^{(n)}|x}{\mathrm{Var}}\left[\hat{x}(Y^{(n)}) - x\right]\right) = \|b(x)\|^2 + \mathrm{Tr}\left(\underset{Y^{(n)}|x}{\mathrm{Var}}\left[\hat{x}(Y^{(n)})\right]\right)$$

We take the expectation of both sides over X and arrive at

$$\mathbb{E}_X\left[\mathbb{E}_{Y^{(n)}|x}\left[\left\|\hat{x}(Y^{(n)}) - x\right\|^2\right]\right] = \mathbb{E}_X\left[\|b(x)\|^2\right] + \mathbb{E}_X\left[\mathrm{Tr}\left(\underset{Y^{(n)}|x}{\mathrm{Var}}\left[\hat{x}(Y^{(n)})\right]\right)\right]$$

and since $\mathrm{Var}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)})\right]$ is independent of the true x, we have $\mathbb{E}_X\left[\mathrm{Tr}\left(\mathrm{Var}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)})\right]\right)\right] = \mathrm{Tr}\left(\mathrm{Var}_{Y^{(n)}|x}\left[\hat{x}(Y^{(n)})\right]\right)$. We conclude with

$$\mathbb{E}_X\left[\mathbb{E}_{Y^{(n)}|x}\left[\left\|\hat{x}(Y^{(n)}) - x\right\|^2\right]\right] = \mathbb{E}_X\left[\|b(x)\|^2\right] + \mathrm{Tr}\left(\underset{Y^{(n)}|x}{\mathrm{Var}}\left[\hat{x}(Y^{(n)})\right]\right) \quad \checkmark$$

Here, the left hand side represents the Bayesian risk of the expected square error. Recall from Homework 1 Problem 7 part (a) that we showed that the estimator that minimizes the posterior risk of the expected square error $L_2(\hat{x}(Y^{(n)}), x) = \|\hat{x}(Y^{(n)}) - x\|^2$ for each possible observed $Y^{(n)}$ is the posterior mean:

$$\hat{x}_{\text{mean}}(Y^{(n)}) = \underset{x \in \mathcal{X}}{\text{argmin}} \left\{ \underset{Y^{(n)}|x}{\mathbb{E}} \left[\|\hat{x}(Y^{(n)}) - x\|^2 \right] \right\} \quad \text{from Homework 1 Problem 7 part (a).}$$

We note that the estimator that minimizes the posterior risk for each possible observed $Y^{(n)}$ is equal to the one that minimizes the Bayesian risk.

(e) Using the decision-theoretic rational for the posterior mean estimator, and the relation between the posterior mean and MAP estimators in a normal, normal model, argue that, if $X \sim \mathcal{N}(0, \sigma_x^2 I)$, then the choice of λ that optimizes the trade-off expressed in Equation (52) can be derived directly from the prior variance.

As stated in the previous part, we know that the estimator that minimizes the Bayesian risk with the expected squared error is the posterior mean. For a normal, normal model the posterior mean is equal to the MAP estimator. We also know that a prior choice of $C_x = \frac{\sigma^2}{\lambda} I$ on the prior leads to the MAP estimator. This implies that

$$\frac{\sigma^2}{\lambda} I = \sigma_x^2 I \quad \Longrightarrow \quad \lambda^\star = \frac{\sigma^2}{\sigma_x^2}$$

This makes sense as if we are unsure about our prior ($\sigma_x^2 \gg \sigma^2$), our optimal λ is small and we reduce the MLE, unregularized estimator. Conversely, if we are very sure about our prior ($\sigma_x^2 \ll \sigma^2$), our optimal λ is large and condenses on the prior mean.

(f) Weak Information: If $X \sim \mathcal{N}(0, \sigma_x^2 I)$, but we use $\hat{x}(Y^{(n)}, \lambda)$ for λ derived from $\sigma' > \sigma_x$ (use a weaker prior than the truth), then will our estimator be more or less biased, and more or less variable, then the optimal estimator?

Let $\lambda^\star$ be the optimal λ derived from the true prior σ_x and let λ' be the regularizer derived from σ'. Then we have

$$\text{when } \sigma' > \sigma_x \implies \lambda' < \lambda^\star \implies \lambda \text{ decreases} \implies \text{the estimator from } \sigma',\ \hat{x}(Y^{(n)}, \lambda) \text{ is } \begin{cases} \underline{\text{less biased}}. \\ \underline{\text{more variable}}. \end{cases}$$

Problem 5: Robust Inference via Fat-Tailed Priors

In Problem 3 you established that, using a normal prior introduced a bias that increased proportional to the truth. This is a natural consequence of adopting a normal prior, whose tails decay very rapidly. The larger the truth, the more evidence it takes to override the fast decaying tails of the normal prior. In this sense, even for large σ_x, a normal prior induces a strong regularizing effect, and, as noted above, is rarely "weakly informative." This is particularly true for the standard setting where A is ill-conditioned, so some directions are well resolved by the likelihood alone, and are largely unaffected by the prior, while others are essentially unresolved by the likelihood, and are entirely resolved by the prior.

In many applications, we want a regularizer whose tails decay more slowly than the tails of a normal distribution. These are more willing to allow the possibility of outliers, so demand less evidence to convince that the true unknown is large. This is useful if, the true prior distribution is heavy-tailed, or, if we want a regularizer that is more amenable to outliers. Adopting a heavy-tailed regularizer produces an estimator that is more sample efficient when attempting to estimate large unknowns, but is more variable.

Throughout this problem we will assume that $Y^{(n)}|x \sim p_{Y|x}$ i.i.d., that the regularity conditions assumed in our unit on consistency and asymptotics apply, and that our model is well-specified. Under those conditions we may assume that, for large n, the likelihood is sharply peaked about the true unknown $x^\star$, the posterior is very small outside of a neighborhood of x near to $x^\star$, and, that the posterior near $x^\star$ admits the second-order Taylor expansion:

$$\frac{1}{n}\log(p_{X|y}(x|Y^{(n)})) \simeq \frac{1}{n}\left(\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star}(x-x^\star) + \frac{1}{2}(x-x^\star)^\top \nabla_x^2 \log\big(p_X(x)\big)\Big|_{x=x^\star}(x-x^\star)\right) - \frac{1}{2}(x-x^\star)^\top I[x^\star](x-x^\star) \tag{54}$$

where $\simeq$ is meant up to an additive constant, $I[x^\star]$ is the Fisher information, and convergence occurs in probability over $Y^{(n)}$ as n diverges.[19]

So, if n is large, then the MAP estimator will converge, in probability to the maximizer of Equation (54), which, provided $I[x^\star]$ is invertible, is, to lowest order in n:

$$(\hat{x}_{\text{MAP}}(Y^{(n)}) - x^\star) \xrightarrow{n\to\infty} -(nI[x^\star])^{-1}\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star} \tag{55}$$

where convergence occurs in probability.[20] This equation provides a more general method for analyzing the consistency of posterior point estimators.

In particular, we can use Equation (55) to approximate the convergence rate in n of posterior estimators to the true unknown, $x^\star$, as a function of $x^\star$, for different likelihoods and priors.

(a) For consistency with the rest of this HW, we will assume a normal likelihood. Show that the Fisher information for $Y|x \sim \mathcal{N}(Ax^\star, C_y)$ is a fixed matrix that is independent of $x^\star$. What is the Fisher information when $C_y = \sigma_y^2 I$? Have you seen this matrix earlier in this HW? If so, what role did it play?

Recall the Fisher information is given by

$$I[x^\star] := \mathop{\mathbb{E}}_{Y|x}\left[\left[\nabla_x \log\big(\text{Lik}(x^\star)\big)\right]\left[\nabla_x \log\big(\text{Lik}(x^\star)\big)\right]^\top\right] = -\mathop{\mathbb{E}}_{Y|x}\left[\nabla_x^2 \log\big(\text{Lik}(x^\star)\big)\right] \quad \text{where}$$

$$\log\big(\text{Lik}(x^\star)\big) := -\frac{1}{2}(y-Ax^\star)^\top C_y^{-1}(y-Ax^\star) + f(C_y)$$

It then follows that

$$\nabla_{x^\star} \log\big(\text{Lik}(x^\star)\big) = -\frac{1}{2}\left(2A^\top C_y^{-1}Ax^\star - 2A^\top C_y^{-1}y\right) \quad \Longrightarrow \quad \nabla_{x^\star}^2 \log\big(\text{Lik}(x^\star)\big) = -A^\top C_y^{-1}A$$

Therefore, the Fisher information is given by

$$I[x^\star] = -\mathop{\mathbb{E}}_{Y|x}\left[-A^\top C_y^{-1}A\right] = A^\top C_y^{-1}A$$

When $C_y = \sigma_y^2 I$, the Fisher information is simply

$$I[x^\star] = \frac{1}{\sigma_y^2}A^\top A$$

[19] Notice that Equation (54) recovers the asymptotic normal result, but now includes a correction associated with the vanishing influence of the prior.

[20] Technically, this statement should be written $n(\hat{x}_{\text{MAP}}(Y^{(n)}) - x^\star) \xrightarrow{n\to\infty} -I[x^\star]^{-1}\nabla_x \log(p_X(x))\big|_{x=x^\star}$.

We see that this is the same matrix from normal equations. The role it played was related x to y for the MLE and MAP estimators.

(b) Adopt a normal prior. That is, $X \sim \mathcal{N}(0, \sigma_x^2 I)$ where, without loss of generality, we assume a coordinate system where the entries of X are independent (see Problem 2). Simplify the right-hand side of Equation (55) in this case. Show that the right-hand side has the form $-\frac{1}{n}Mx$ for some matrix M that depends on A, σ_x, and σ_y. Try to express the entries of M in terms of a signal-to-noise or noise-to-signal ratio (covariance in Ax versus the noise variance).

We start by calculating the gradient of the $\log\big(p_X(x)\big)$ w.r.t. x. We have

$$\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star} = \nabla_x \left\{-\frac{x^\top x}{2\sigma_x^2} + f(\sigma_x^2)\right\}\Big|_{x=x^\star} = -\left\{\frac{x}{\sigma_x^2}\right\}\Big|_{x=x^\star} = -\frac{x^\star}{\sigma_x^2}$$

Then the right-hand side of Equation (55) can be written as

$$-(nI[x^\star])^{-1}\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star} = -\frac{1}{n}\left(\frac{1}{\sigma_y^2}A^\top A\right)^{-1}\left(-\frac{x^\star}{\sigma_x^2}\right) = \frac{1}{n}\frac{\sigma_y^2}{\sigma_x^2}\left(A^\top A\right)^{-1}x^\star = \boxed{\frac{1}{n}\frac{1}{\text{SNR}}\left(A^\top A\right)^{-1}x^\star \quad \text{where} \quad \text{SNR} = \frac{\sigma_x^2}{\sigma_y^2}}$$

(c) Adopt a prior of the form $p_X(x) \propto \exp\left(-\frac{\lambda}{p}\|x\|_p^p\right)$. Show that, for $p = 1$, $\nabla_x \log\big(p_X(x)\big)|_{x=x^\star} = -\lambda \times \text{sign}(x)$ so the convergence rate of the MAP estimator is independent of the magnitude of the true unknown $x^\star$. In this sense the L_1 prior (Laplace prior), is more robust than the normal prior. It demands less information to believe in large $x^\star$.

Given $\nabla_x \log\big(p_X(x)\big)|_{x=x^\star} = -\lambda \times \text{sign}(x^\star)$, the right-hand side of Equation (55) can be written as

$$-(nI[x^\star])^{-1}\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star} = -\frac{\sigma_y^2}{n}\left(A^\top A\right)^{-1}\left(-\lambda \cdot \text{sign}(x^\star)\right) = \boxed{\frac{\lambda\sigma_y^2}{n}\left(A^\top A\right)^{-1}\text{sign}(x^\star)}$$

where we see its convergence is independent of its magnitude. Rather its convergence depends on $\text{sign}(x^\star)$.

(d) Repeat your analysis for a Student's-t prior, $p_X(x) \propto (\nu + \|x\|_2^2)^{-(\nu+1)/2}$. This is a prior with very heavy tails. How does the convergence rate of the MAP estimator to $x^\star$ in n depend on $\|x^\star\|$ now? How does it depend on ν? Try to make sense of your result by reasoning about the tail decay rate of this prior.

We start by taking the gradient of the log-prior with respect to x. We have

$$\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star} = \nabla_{x^\star}\left\{\frac{-(\nu+1)}{2}\log\big(\nu + \|x^\star\|_2^2\big)\right\} = \frac{-(\nu+1)}{2}\left(\frac{1}{\nu + \|x^\star\|_2^2}\right)2x^\star = \frac{-(\nu+1)x^\star}{\nu + \|x^\star\|_2^2}$$

Then the right-hand side of Equation (55) can be written as

$$-(nI[x^\star])^{-1}\nabla_x \log\big(p_X(x)\big)\Big|_{x=x^\star} = -\frac{\sigma_y^2}{n}\left(A^\top A\right)^{-1}\left(\frac{-(\nu+1)x^\star}{\nu + \|x^\star\|_2^2}\right) = \boxed{\frac{\sigma_y^2}{n}\frac{\nu+1}{\nu + \|x^\star\|_2^2}\left(A^\top A\right)^{-1}x^\star}$$

(e) Consider the three examples (Normal, Laplace, Student's-t). Use $\nu = 1$ (Cauchy), and $\nu = 3$ for the Student's-t distribution. For your experiments, use a 1-dimensional example with $A = 1$, $\sigma_y = 1$, $\sigma_x = 1$, and $\lambda = 1$. Fix $x^\star = 1, 2, 4, 8$, and 16. Sample $Y^{(n)}$ from the corresponding likelihood for $n = 10^m$ where $m \in [0, 4]$ are evenly spaced. To sample, first sample a single sequence $Z^{(n)} \sim \mathcal{N}(0, 1)$ with $n = 10^4$, then set $Y^{(n)} = x^\star + Z^{(n)}$ for each desired n and $x^\star$. This fixes the random seed, so that all ensuing simulations only differ in the choice of prior. Make a sequence of comparison plots showing the error in the MAP estimator for the four prior specifications listed above, as a function of m, for the five $x^\star$ values. Do your results match your previous analysis/intuition about the regularizing effects of the priors?

We begin by deriving the 1-dimensional MAP estimator after observing n samples (i.e., $\dim(x) = \dim(y) = 1$) for each of our three priors. We have

$$\hat{x}_{\text{MAP}}(y^{(n)}) = \text{argmax}\left\{p\left(x \mid y^{(n)}\right)\right\} = \text{argmin}\left\{-\log\left(p\left(y^{(n)} \mid x\right)\right) - \log\big(p(x)\big)\right\} = \text{argmin}\left\{\frac{\sum_{j=1}^n (Ax - y_j)^2}{2\sigma_y^2} - \log\big(p(x)\big)\right\}.$$

It follows that

$$\frac{\mathrm{d}}{\mathrm{d}x}\left\{-\log\Big(p\big(x \mid y^{(n)}\big)\Big)\right\} = \sum_{j=1}^{n} \frac{\big(2A^2x - 2Ay_j\big)}{2\sigma_y^2} - \frac{\mathrm{d}\log\big(p(x)\big)}{\mathrm{d}x} = 0 \quad \Longrightarrow \quad A^2x - \left(\frac{\sigma_y^2}{n}\right)\frac{\mathrm{d}\log\big(p(x)\big)}{\mathrm{d}x} = A\bar{y}^{(n)}$$

where $\bar{y}^{(n)} := \frac{1}{n}\sum_{j=1}^{n} y_j$. With $A = \sigma_y = 1$, our expression is simply $x - \left(\frac{1}{n}\right)\frac{\mathrm{d}\log\big(p(x)\big)}{\mathrm{d}x} = \bar{y}^{(n)}$. Then we using our results in the previous parts ($\nabla_x \log\big(p_X(x)\big)$) to solve for $\hat{x}_{\mathrm{MAP}}(y^{(n)})$.

- **Normal Prior:** $\frac{\mathrm{d}\log\big(p(x)\big)}{\mathrm{d}x} = -\frac{x}{\sigma_x^2}\Big|_{\sigma_x=1} = -x \implies x + \frac{x}{n} = \bar{y}^{(n)} \implies \frac{n+1}{n}x = \bar{y}^{(n)} \implies \hat{x}^{\mathcal{N}}_{\mathrm{MAP}}(y^{(n)}) = \frac{n}{n+1}\bar{y}^{(n)}$
- **Laplace Prior:** $\frac{\mathrm{d}\log\big(p(x)\big)}{\mathrm{d}x} = -\lambda \times \mathrm{sign}(x)\big|_{\lambda=1} = -\mathrm{sign}(x) \implies x + \frac{\mathrm{sign}(x)}{n} = \bar{y}^{(n)}$. Recall from class that we showed that the Laplace prior leads to soft thresholding, i.e.,

$$\hat{x}^{\mathrm{Lapl}}_{\mathrm{MAP}}(y^{(n)}) = \begin{cases} \bar{y}^{(n)} - \frac{1}{n} & \text{if } y^{(n)} > \frac{1}{n} \\ \bar{y}^{(n)} + \frac{1}{n} & \text{if } y^{(n)} < -\frac{1}{n} \\ 0 & \text{otherwise} \end{cases}$$

- **Student's-t Prior:** $\frac{\mathrm{d}\log\big(p(x)\big)}{\mathrm{d}x} = \frac{-(\nu+1)x}{\nu+x^2} \implies x + \frac{-(\nu+1)x}{n(\nu+x^2)} = \bar{y}^{(n)} \implies nx^3 - n\bar{y}^{(n)}x^2 + (n\nu+\nu+1)x - n\nu\bar{y}^{(n)} = 0$. Here we see that $\hat{x}^{t}_{\mathrm{MAP}}(y^{(n)})$ is the positive root to this cubic polynomial. We will solve for this numerically.

Additionally, we plot the approximate error of the MAP estimators summarized in the table below.

Prior	$-(nI[x^\star])^{-1}\nabla_x \log\big(p_X(x)\big)\big\|_{x=x^\star}$	With $A=1$, $\sigma_y=1$, $\sigma_x=1$, and $\lambda=1$
Normal	$\frac{1}{n}\frac{\sigma_y^2}{\sigma_x^2}\left(A^\top A\right)^{-1}x^\star$	$\frac{1}{n}x^\star$
Laplace	$\frac{\lambda\sigma_y^2}{n}\left(A^\top A\right)^{-1}\mathrm{sign}(x^\star)$	$\frac{1}{n}\mathrm{sign}(x^\star)$
Student's t ($\nu=1$)	$\frac{\sigma_y^2}{n}\frac{\nu+1}{\nu+\|x^\star\|_2^2}\left(A^\top A\right)^{-1}x^\star$	$\frac{1}{n}\left(\frac{2}{1+\|x^\star\|_2^2}\right)x^\star$
Student's t ($\nu=3$)		$\frac{1}{n}\left(\frac{4}{3+\|x^\star\|_2^2}\right)x^\star$

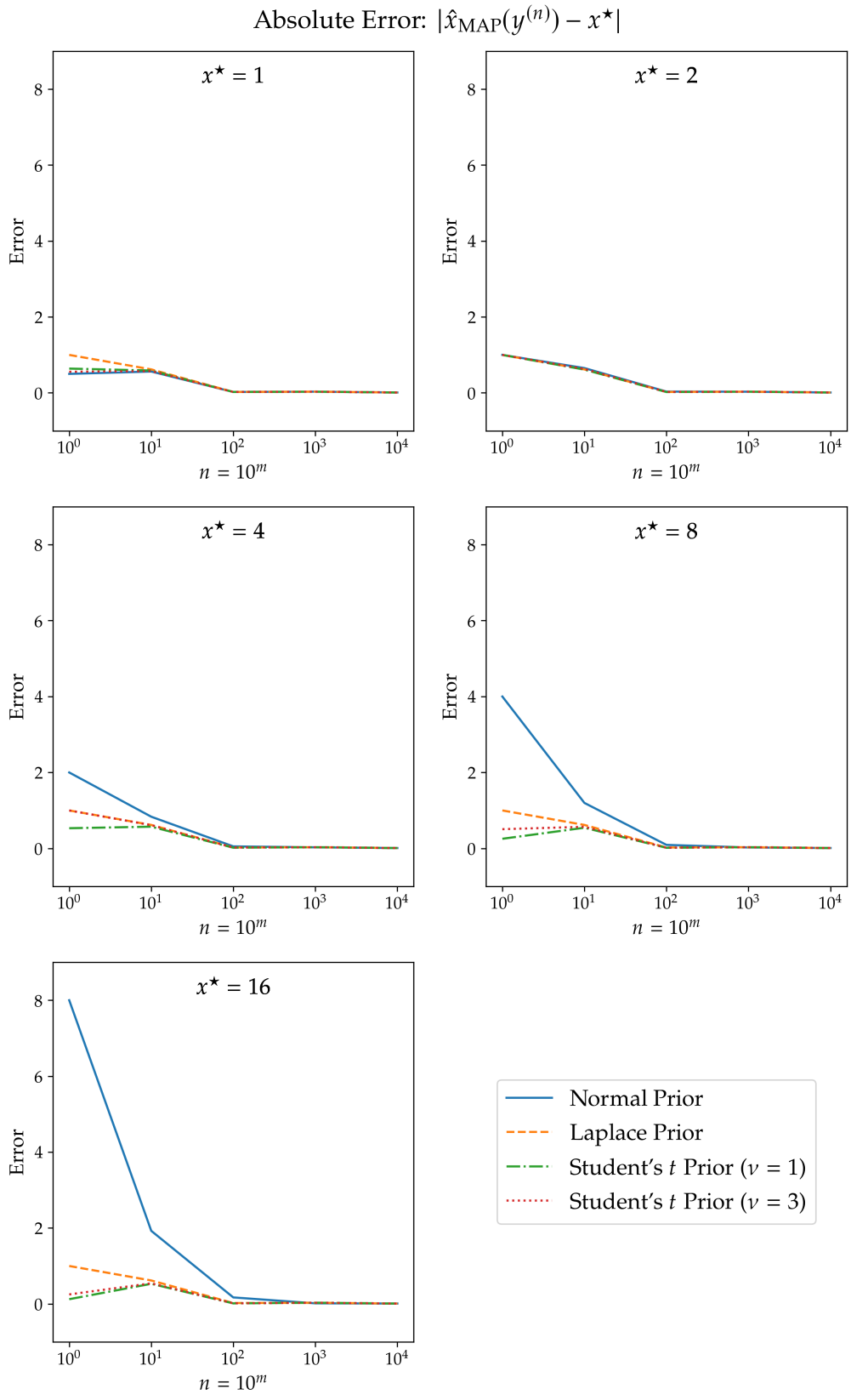


Figure 4: Absolute error for MAP estimator, $\hat{x}_{\text{MAP}}(Y^{(n)}) - x^\star$ for $x^\star = 1, 2, 4, 8, 16$ for the three priors.

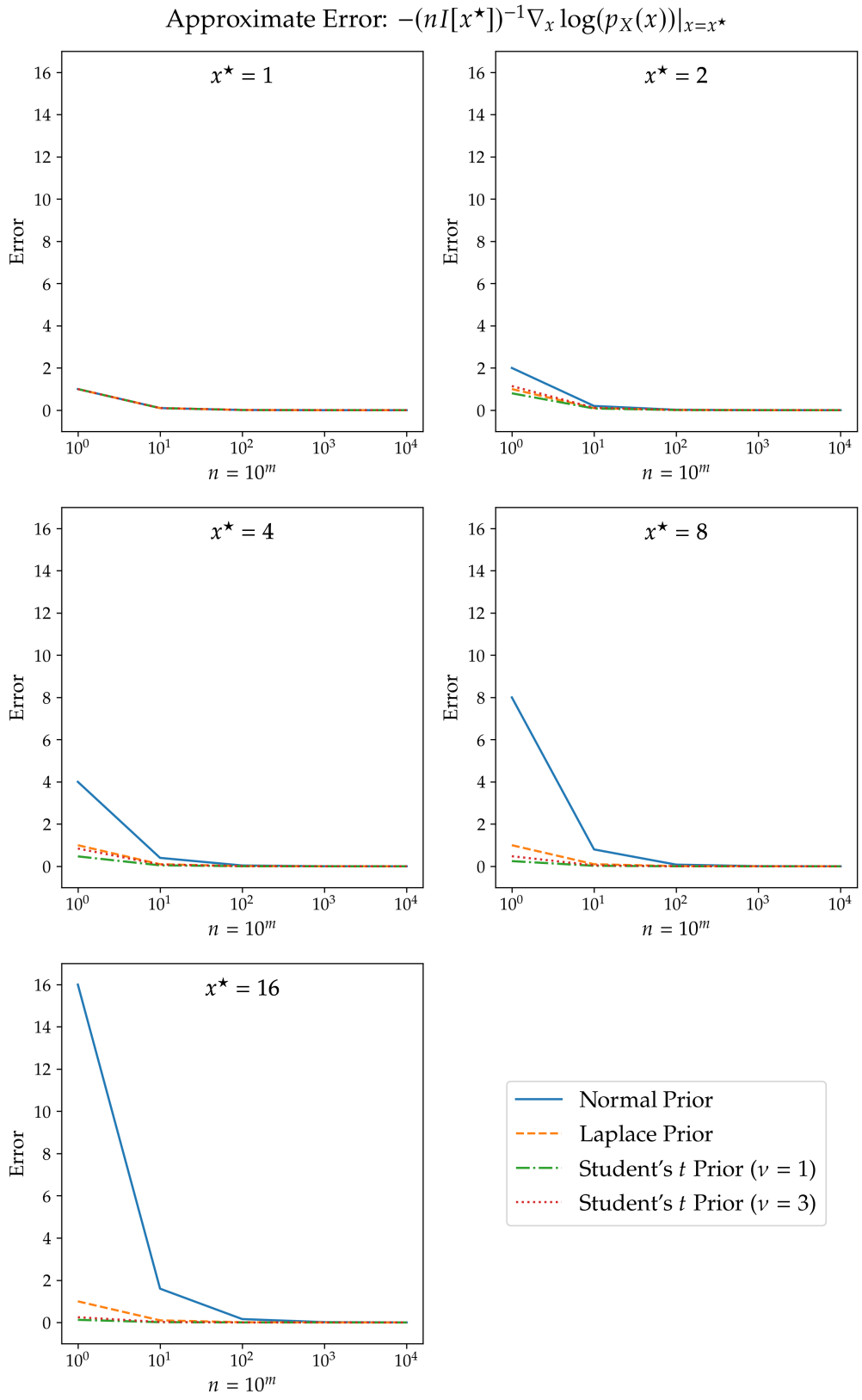


Figure 5: Approximate error for MAP estimator, $-(nI[x^\star])^{-1}\nabla_x \log\big(p_X(x)\big)\big|_{x=x^\star}$ for $x^\star = 1, 2, 4, 8, 16$ for the three priors.

Commentary on plots

- For the Normal prior, both the absolute and approximate error are quite large especially when $x^\star$ is large and n is small.
- For the Laplace prior, the error only really depends on n as we are considering cases where $x^\star$ is positive and therefore $\hat{x}^{\text{Lapl}}_{\text{MAP}}(y^{(n)}) = \bar{y}^{(n)} - \frac{1}{n}$ with high probability especially for large n and $x^\star$. Therefore, the absolute error is $|\hat{x} - x^\star| = |\bar{y}^{(n)} - \frac{1}{n} - x^\star| = |x^\star + \bar{\zeta}^{(n)} - \frac{1}{n} - x^\star| = |\bar{\zeta}^{(n)} - \frac{1}{n}|$ which is why rarely see the error above 1 regardless of how large $x^\star$ is.
- For the Student's-t prior, recall that $\nu = 1$ has fatter tails than $\nu = 3$. This means that $\nu = 1$ prior is more willing to admit large x than the $\nu = 3$ prior. This is why we see that $\nu = 1$ generally performs better than the $\nu = 3$ prior.

Python Code for Problem 5

```python
import numpy as np
np.random.seed(9)
from scipy.optimize import minimize_scalar
import matplotlib.pyplot as plt

approaches = ['approx', 'diff']
noise   = np.random.normal(loc=0,scale=1,size=int(1e4))
x_stars = [1,2,4,8,16]
ms      = [j for j in range(5)]
ns      = [int(10**j) for j in range(5)]
ybars   = {x:{n:np.mean(x+noise[:n]) for n in ns} for x in x_stars}
# ====================================== #
# ========== Approximate error ========== #
# ====================================== #
# Normal Prior (part (b))
# === 1/n x^*
def approx_normal(x_star_, n_):
    return x_star_/n_
# Laplace Prior (part (c))
# === 1/n sign(x^*)
def approx_laplace(x_star_, n_):
    return np.sign(x_star_)/n_
# Student's-t Prior (part (d))
# === 1/n [(v+1)x^*] / [v + (x^*)^2]
def approx_student(x_star_, n_, nu_):
    return (1/n_) * ( (nu_+1) * x_star_ ) / ( nu_ + x_star_**2 )

# ================================= #
# ========== Diff error ========== #
# ================================= #
def map_normal(ybar, n):
    return (n / (n + 1)) * ybar
def map_laplace(ybar, n):
    thres = 1 / n
    if ybar > thres:
        return ybar - thres
    elif ybar < -thres:
        return ybar + thres
    else:
        return 0.0
def map_student(ybar_, n_, nu_):
    def objective(x_):
        return 0.5 * n_ * (x_ - ybar_) ** 2 + ((nu_ + 1) / 2) * np.log(nu_ + x_**2)
    res = minimize_scalar(objective)
    return res.x

# ====================================== #
# ========== Plotting errors ========== #
# ====================================== #
for approach in approaches:
    names = [r"Normal Prior",
             r"Laplace Prior",
             r"Student's $t$ Prior ($\nu = 1$)",
             r"Student's $t$ Prior ($\nu = 3$)"]
    data = {name:{x:[] for x in x_stars} for name in names}
    largest_error = 0
    for x in x_stars:
        for n in ns:
            if approach=='approx':
                data[names[0]][x].append(approx_normal(x,n))
                data[names[1]][x].append(approx_laplace(x,n))
                data[names[2]][x].append(approx_student(x,n,1))
```

```
                data[names[3]][x].append(approx_student(x,n,3))
            elif approach=='diff':
                ybar = ybars[x][n]
                data[names[0]][x].append(abs(map_normal(ybar,n) - x))
                data[names[1]][x].append(abs(map_laplace(ybar,n) - x))
                data[names[2]][x].append(abs(map_student(ybar,n,nu_=1) - x))
                data[names[3]][x].append(abs(map_student(ybar,n,nu_=3) - x))
        largest_error = max(largest_error,  max(data[names[0]][x]),
                                            max(data[names[1]][x]),
                                            max(data[names[2]][x]),
                                            max(data[names[3]][x]))

    fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(10,15))
    axes = axes.flatten()
    axes[-1].remove()
    line_styles = ["-", "--", "-.", ":"]
    for x, ax in zip(x_stars, axes):
        for name, style in zip(names, line_styles):
            ax.plot(ms, data[name][x], style, label=name)
            ax.set_title("")
            ax.text(
                0.5, 0.9,
                f"$x^\star = {x}$",
                transform=ax.transAxes,
                ha="center",
                va="bottom",
                fontsize=16
            )
            ax.set_ylabel(r'Error')
            ax.set_ylim([-1,largest_error+1])
            ax.set_xlabel(r'$n=10^m$')
            ax.set_xticks(
                [0, 1, 2, 3, 4],
                [r'$10^0$', r'$10^1$', r'$10^2$', r'$10^3$', r'$10^4$']
            )
    ax.legend(
        loc="center left",
        bbox_to_anchor=(1.2, 0.5)
    )
    if approach=='approx':
        plt.suptitle(r'Approximate Error: $-(nI[x^\star])^{-1}\nabla_x\log(p_X(x))|_{x=x^\star}$')
    elif approach=='diff':
        plt.suptitle(r'Absolute Error: $|\hat{x}_{\text{MAP}}(y^{(n)})-x^\star|$')
    plt.subplots_adjust(top=0.95)
    plt.savefig(f'5_{approach}.png', bbox_inches='tight', pad_inches=0.1)
    plt.show()
    plt.close()
```

STAT238: Bayesian Statistics April 15th, 2025

Homework 4: Expectation Maximization, Sampling, & MCMC

Reece D. Huff

Problems

Problem 1: Chapter 13, Problem 5 [BDA3]

Estimating the number of unseen species (see Fisher, Corbet, and Williams, 1943, Efron and Thisted, 1976, and Seber, 1992): suppose that during an animal trapping expedition the number of times an animal from species i is caught is $x_i \sim \text{Poisson}(\lambda_i)$. For parts (a)–(d) of this problem, assume a Gamma(α, β) prior distribution for the λ_i's, with a uniform hyperprior distribution on (α, β). The only observed data are y_k, the number of species observed exactly k times during a trapping expedition, for $k = 1, 2, 3, \ldots$

(a) Write the distribution $p(x_i \mid \alpha, \beta)$.

The densities for $\lambda_i \mid \alpha, \beta$ and $x_i \mid \lambda_i$ are given by

$$p(\lambda_i \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} \exp\left(-\beta\lambda_i\right) \quad \text{for } \lambda_i, \alpha, \beta > 0 \quad \text{and} \quad p(x_i \mid \lambda_i) = \frac{\lambda_i^{x_i} \exp(-\lambda_i)}{x_i!} \quad \text{for } x_i \in \{0, 1, 2, \ldots\}.$$

Note x_i is conditionally independent of α, β given λ_i (i.e., $x_i \mid \lambda_i \perp\!\!\!\perp \alpha, \beta \mid \lambda_i$) implying that $p(x_i \mid \lambda_i, \alpha, \beta) = p(x_i \mid \lambda_i)$[21]. It follows that

$$p(x_i \mid \alpha, \beta) = \int_{\text{all } \lambda_i} p(x_i \mid \lambda_i, \alpha, \beta)\, p(\lambda_i \mid \alpha, \beta)\, \mathrm{d}\lambda_i = \int_0^\infty p(x_i \mid \lambda_i)\, p(\lambda_i \mid \alpha, \beta)\, \mathrm{d}\lambda_i$$

$$p(x_i \mid \alpha, \beta) = \int_0^\infty \left(\frac{\lambda_i^{x_i} \exp(-\lambda_i)}{x_i!}\right) \left(\frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} \exp\left(-\beta\lambda_i\right)\right) \mathrm{d}\lambda_i = \left(\frac{\beta^\alpha}{\Gamma(\alpha) x_i!}\right) \int_0^\infty \lambda_i^{(x_i+\alpha)-1} \exp\left(-\lambda_i\,(\beta+1)\right) \mathrm{d}\lambda_i$$

Recall that $\Gamma(z) = \int_0^\infty t^{z-1} \exp(-t)\mathrm{d}t$ and $\frac{1}{c^z}\Gamma(z) = \int_0^\infty t^{z-1} \exp(-ct)\mathrm{d}t$[22]. We have

$$p(x_i \mid \alpha, \beta) = \left(\frac{\beta^\alpha}{\Gamma(\alpha) x_i!}\right) \int_0^\infty \lambda_i^{(x_i+\alpha)-1} \exp\left(-\lambda_i\,(\beta+1)\right) \mathrm{d}\lambda_i = \left(\frac{\beta^\alpha}{\Gamma(\alpha) x_i!}\right) \left(\frac{\Gamma(x_i+\alpha)}{(\beta+1)^{x_i+\alpha}}\right)$$

$$\boxed{p(x_i \mid \alpha, \beta) = \frac{\Gamma(\alpha + x_i)}{\Gamma(\alpha) x_i!} \left(\frac{\beta}{\beta+1}\right)^\alpha \left(\frac{1}{\beta+1}\right)^{x_i}}$$

(b) Use the distribution of x_i to derive a multinomial distribution for y given that there are a total of N species.

We have that $y_k = \sum_{i=1}^N \mathbb{1}\{x_i = k\}$ where $\mathbb{1}\{\cdot\}$ is an indicator function. Let y_1 be the number of species we observed once, y_2 be the number of species we observed twice, and so forth, such that the probability of $y = [y_1, y_2, \ldots]$ be

$$\mathbb{P}\left(y = [y_1, y_2, \ldots]\right) = \frac{\left(\sum_{k=1}^\infty y_k\right)!}{\prod_{k=1}^\infty y_k} \prod_{k=1}^\infty {p_k}^{y_k}$$

We note that $\sum_{k=1}^\infty y_k = \sum_{k=1}^\infty \sum_{i=1}^N \mathbb{1}\{x_i = k\} = \sum_{i=1}^N \sum_{k=1}^\infty \mathbb{1}\{x_i = k\} = \sum_{i=1}^N 1 = N$ and $p_k = \mathbb{P}[x_i = k \mid \alpha, \beta] = \frac{\Gamma(\alpha+k)}{\Gamma(\alpha)k!} \left(\frac{\beta}{\beta+1}\right)^\alpha \left(\frac{1}{\beta+1}\right)^k$ from the previous part. Therefore, we conclude

$$\boxed{p(y \mid \alpha, \beta, N) = \frac{N!}{\prod_{k=1}^\infty y_k!} \prod_{x_i=1}^\infty p(x_i \mid \alpha, \beta)^{y_{x_i}}}$$

[21] This follows from Bayes' rule, i.e., $p(x_i \mid \lambda_i, \alpha, \beta) = \frac{p(x_i, \lambda_i, \alpha, \beta)}{p(\lambda_i, \alpha, \beta)} = \frac{p(x_i, \alpha, \beta \mid \lambda_i)\, p(\lambda_i)}{p(\alpha, \beta \mid \lambda_i)\, p(\lambda_i)} = \frac{p(x_i \mid \lambda_i)\, p(\alpha, \beta \mid \lambda_i)}{p(\alpha, \beta \mid \lambda_i)} = p(x_i \mid \lambda_i)$.

[22] This follow from $\int_0^\infty t^{z-1} \exp(-ct)\mathrm{d}t = \frac{c^{z-1}}{c^{z-1}} \int_0^\infty t^{z-1} \exp(-ct)\mathrm{d}t = \frac{c}{c^z} \int_0^\infty (ct)^{z-1} \exp(-ct)\mathrm{d}t = \frac{1}{c^z} \int_0^\infty (ct)^{z-1} \exp(-ct)(c\mathrm{d}t)$. Let $u = ct$ such that $du = c\mathrm{d}t$. Then we have $\frac{1}{c^z} \int_0^\infty u^{z-1} \exp(-u)\mathrm{d}u = \frac{1}{c^z}\Gamma(z)$.

(c) Suppose that we are given $y = (118, 74, 44, 24, 29, 22, 20, 14, 20, 15, 12, 14, 6, 12, 6, 9, 9, 6, 10, 10, 11, 5, 3, 3)$ so that 118 species were observed only once, 74 species were observed twice, and so forth with a total of 496 species observed and 3266 animals caught. Write down the likelihood for y using the multinomial distribution with 24 cells (ignoring unseen species). Use any method to find the mode of α, β and an approximate second derivative matrix.

We start by writing the likelihood for $y = (118, 74, \ldots, 3)$ where $N = 496$:

$$p(y \mid \alpha, \beta, N = 496) = \frac{496!}{118!74!\ldots3!}\left[\frac{\Gamma(\alpha+1)}{\Gamma(\alpha)\,1!}\left(\frac{\beta^\alpha}{(\beta+1)^{\alpha+1}}\right)\right]^{118}\left[\frac{\Gamma(\alpha+2)}{\Gamma(\alpha)\,2!}\left(\frac{\beta^\alpha}{(\beta+1)^{\alpha+2}}\right)\right]^{74}\cdots\left[\frac{\Gamma(\alpha+24)}{\Gamma(\alpha)\,24!}\left(\frac{\beta^\alpha}{(\beta+1)^{\alpha+24}}\right)\right]^{3}$$

$$p(y \mid \alpha, \beta, N = 496) = \left(\frac{496!}{(118!74!\ldots3!)\cdot(1!2!\ldots24!)}\right)\left(\frac{\Gamma(\alpha+1)^{118}\,\Gamma(\alpha+2)^{74}\ldots\Gamma(\alpha+24)^{3}}{\Gamma(\alpha)^{496}}\right)\left(\frac{\beta}{\beta+1}\right)^{496\alpha}\left(\frac{1}{\beta+1}\right)^{3266}$$

$$\{\hat\alpha_{\text{mode}}, \hat\beta_{\text{mode}}\} = \underset{\alpha,\beta}{\text{argmax}}\left\{\left(\frac{\Gamma(\alpha+1)^{118}\,\Gamma(\alpha+2)^{74}\ldots\Gamma(\alpha+24)^{3}}{\Gamma(\alpha)^{496}}\right)\left(\frac{\beta}{\beta+1}\right)^{496\alpha}\left(\frac{1}{\beta+1}\right)^{3266}\right\}$$

$$\{\hat\alpha_{\text{mode}}, \hat\beta_{\text{mode}}\} = \underset{\alpha,\beta}{\text{argmin}}\left\{-118\log(\Gamma(\alpha+1)) - 74\log(\Gamma(\alpha+2)) - \cdots - 3\log(\Gamma(\alpha+24)) + 496\log(\Gamma(\alpha))\right.$$
$$\left. - 496\alpha\log\left(\frac{\beta}{\beta+1}\right) - 3266\log\left(\frac{1}{\beta+1}\right)\right\}$$

(d) Derive an estimate and approximate 95% posterior interval for the number of additional species that would be observed if 10,000 more animals were caught.

TODO

Problem 2: Chapter 13, Problem 9 [BDA3]

EM algorithm:

(a) For the hierarchical normal model in Section 13.6, derive the expressions (13.14) for μ_{new}, σ_{new}, and τ_{new}.
optional

(b) Pick values for the hyperparameters in this model, then simulate fake data, then apply EM to estimate the model. Compare the EM estimate to the assumed true model.

Recall our hierarchical model, where

- we assume a uniform hyperprior, $p(\mu, \log\sigma, \log\tau) \propto \tau$, we then draw J experiment means,
- $\theta_j \mid \mu, \tau \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mu, \tau^2)$ for $j \in \{1, 2, \ldots, J\}$, and then we draw n_j samples from each of the J experiments,
- $y_{ij} \mid \theta_j \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\theta_j, \sigma^2)$ for $i \in \{1, 2, \ldots, n_j\}$ for a total of $n = \sum_{j=1}^{J} n_j$ total samples.

We have $\theta_j \mid \mu, \sigma, \tau, y \sim \mathcal{N}(\hat{\theta}_j, V_{\theta_j})$ where

$$\hat{\theta}_j = \frac{\frac{1}{\tau^2}\mu + \frac{n_j}{\sigma^2}\bar{y}_{.j}}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}} \quad \text{and} \quad V_{\theta_j} = \frac{1}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}} \quad \text{where} \quad \bar{y}_{.j} = \frac{1}{n_j}\sum_{i=1}^{n_j} y_{ij} \tag{[BDA3]}$$

The maximization step (see *M-step* and expressions (13.14) on page 329 of [BDA3]) leads to

$$\mu^{\text{new}} = \frac{1}{J}\sum_{j=1}^{J} \hat{\theta}_j$$

$$\sigma^{\text{new}} = \left(\frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\left((y_{ij} - \hat{\theta}_j)^2 + V_{\theta_j}\right)\right)^{1/2}$$

$$\tau^{\text{new}} = \left(\frac{1}{J-1}\sum_{j=1}^{J}\left((\hat{\theta}_j - \mu^{\text{new}})^2 + V_{\theta_j}\right)\right)^{1/2}$$

After running the EM algorithm $J = 19$ and $n_j \approx 1000$ for all j, and an initial guess of $(\mu, \tau^2, \sigma^2) = (0, 1, 1)$, we approach the true parameters:

	μ	τ^2	σ^2
true parameters	5	4	16
initial guess	0	1	1
after EM	5.102	3.510	15.980

```python
import numpy as np
np.random.seed(9)
np.set_printoptions(precision=4)
import scipy.stats as stats

mu     = 5
tau2   = 4
tau    = np.sqrt(tau2)
sigma2 = 16
sigma  = np.sqrt(sigma2)

n_js = [1005,1006,1007,1008,1009,1010,1011,1005,1006,1007,1008,1009,1010,1011,1005,1006,1007,1008,1009]
N    = sum(n_js)
J    = len(n_js)

# Generate theta
```

```python
theta_js = stats.norm.rvs(loc=mu, scale=tau, size=J)

# Generate y and ybar
y = {}
for j, (n_j, theta_j) in enumerate(zip(n_js, theta_js)):
    y[j] = stats.norm.rvs(loc=theta_j, scale=sigma, size=n_j)
def get_ybar(y):
    means = [np.mean(v) for _, v in y.items()]
    return np.array(means)
ybar = get_ybar(y)

# theta_hat and V_{\theta}
def get_theta_hat_and_V(_mu, _tau2, _sigma2, _n_js, _ybar):
    T = []
    V = []
    for nj, yb in zip(_n_js, _ybar):
        num = ((1/_tau2)*_mu) + ((nj/_sigma2)*yb)
        den =  (1/_tau2)      +  (nj/_sigma2)
        T.append(num/den)
        V.append( 1 /den)
    return np.array(T), np.array(V)

# mu_new, sigma_new, and tau_new
def get_mu_new(_theta_hat):
    return np.mean(_theta_hat)

def get_mu_sigma_and_tau_new(_y, _theta_hat_js, _V_theta_js, _N):

    # mu_new
    mu_new   = np.mean(_theta_hat_js)

    # tau_new
    _J       = len(_theta_hat_js)
    tau2_new = 1/(_J-1) * np.sum((_theta_hat_js-mu_new)**2 + _V_theta_js)

    # sigma_new
    s_new_sum = 0
    for (j, y_j), T_j, V_j in zip(_y.items(), _theta_hat_js, _V_theta_js):
        s_new_sum += np.sum((y_j-T_j)**2 + V_j)
    sigma2_new = s_new_sum / _N

    # return mu_new, np.sqrt(tau2_new), np.sqrt(sigma2_new)
    return mu_new, tau2_new, sigma2_new

# EM Algorithm
mu_k = 0
t2_k = 1
t_k  = np.sqrt(t2_k)
s2_k = 1
s_k  = np.sqrt(s2_k)

print('True parameters')
print('mu, tau2, sigma2')
print(np.array([mu, tau2, sigma2]))
print('')
print('EM Algorithm')
print('mu, tau2, sigma2')
print(np.array([mu_k, t2_k, s2_k]))

N_iters = 100
for k in range(N_iters):
    theta_hat_k, V_theta_k = get_theta_hat_and_V(mu_k, t2_k, s2_k, n_js, ybar)
    mu_k, t2_k, s2_k       = get_mu_sigma_and_tau_new(y, theta_hat_k, V_theta_k, N)
    print(f"iter={k+1}", np.array([mu_k, t2_k, s2_k]))
```

Problem 3: Chapter 10, Problem 1 [BDA3]

Number of simulation draws: Suppose the scalar variable θ is approximately normally distributed in a posterior distribution that is summarized by n independent simulation draws. How large does n have to be so that the 2.5% and 97.5% quantiles of θ are specified to an accuracy of 0.1 sd$(\theta|y)$?

(a) Figure this out mathematically, without using simulation.

We have that the accuracy of quantile q is given by

$$\text{sd}(q) = \sqrt{\frac{p(1-p)}{nf(x_p)^2}}$$

where $x_p = F^{-1}(p)$. For a normal distribution, $f(F^{-1}(0.975)) = f(F^{-1}(0.025)) \approx 0.0584/\text{sd}(\theta|y)$. Also, we note that $p(1-p) = 0.025 \cdot 0.975$ for both $p = 0.025$ and $p = 0.975$. It then follows that

$$\text{sd}(q) = \sqrt{\frac{0.025 \cdot 0.975}{n(0.0584/\text{sd}(\theta|y))^2}} = \text{sd}(\theta|y)\sqrt{\frac{0.025 \cdot 0.975}{n \cdot 0.0584^2}} = 0.1\text{sd}(\theta|y) \quad \Longrightarrow \quad n = \frac{1}{0.01}\left(\frac{0.025 \cdot 0.975}{0.0584^2}\right) \approx 715$$

Problem 4: Chapter 10, Problem 5 [BDA3]

Rejection sampling and importance sampling: Consider the model, $y_j \sim \text{Binomial}(n_j, \theta_j)$, where $\theta_j = \text{logit}^{-1}(\alpha + \beta x_j)$, for $j = 1, ..., J$, and with independent prior distributions, $\alpha \sim t_4(0, 2^2)$ and $\beta \sim t_4(0, 1)$. Suppose $J = 10$, the x_j values are randomly drawn from a $\mathcal{U}(0, 1)$ distribution, and $n_j \sim \text{Poisson}^+(5)$, where Poisson^+ is the Poisson distribution restricted to positive values.

(a) Sample a dataset at random from the model.

```python
import numpy as np
np.random.seed(10)
np.set_printoptions(precision=4)
import scipy.stats as stats

def logit_inv(v):
    return np.exp(v)/(1+np.exp(v))

def sample_zero_truncated_poisson(lam=5, size=1):
    samples = []
    while len(samples) < size:
        sample = stats.poisson.rvs(lam)
        if sample > 0:
            samples.append(sample)
    return samples[0] if size == 1 else np.array(samples)

J = 10
x_j = stats.uniform.rvs(size=J)
alpha = stats.t.rvs(df=4, loc=0, scale=2)
beta  = stats.t.rvs(df=4, loc=0, scale=1)
theta_j = logit_inv(alpha+beta*x_j)
n_j = sample_zero_truncated_poisson(lam=5,size=J)
y_j = np.array([stats.binom.rvs(n,p) for n,p in zip(n_j,theta_j)])
```

(b) Use rejection sampling to get 1000 independent posterior draws from (α, β).

Rejection Sampling

Let z represent the unknowns of interest (e.g., $z = (\alpha, \beta)$). Recall that

$$\tilde{p}(z) := \frac{\tilde{p}_*(z)}{Z} \quad \text{where} \quad Z := \int_{\text{all } z} \tilde{p}_*(z)dz \quad \text{and} \quad Mp_r(z) \geq \tilde{p}_*(z) \quad \text{for all } z$$

and recall the general procedure for rejection sampling where our goal is collect a set of M independent samples $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$

ALGORITHM 1: Rejection Sampling

Initialize: $\mathcal{S} \leftarrow \emptyset$

while $|\mathcal{S}| < M$ **do**
  Draw a candidate sample from the proposal distribution: $V \sim p_r$
  Compute acceptance ratio: $a(V) = \frac{\tilde{p}_*(V)}{Mp_r(V)} \in [0, 1]$
  Draw $U \sim \text{Uniform}([0, 1])$
  if $U \leq a(V)$ **then**
    Append V to $\mathcal{S}$
  end
  Reject V
end
return $\mathcal{S}$

It follows that the probability that we accept a candidate sample is

$$\mathbb{P}[V \text{ is accepted}] = \mathbb{P}[U \le a(V)] = \int_{\text{all } z} \mathbb{P}[U \le a(V) \mid V = z]\ \mathbb{P}[V = z]\ \mathrm{d}z = \int_{\text{all } z} \mathbb{P}[U \le a(z)]\ \mathbb{P}[V = z]\ \mathrm{d}z$$

Recall that $\mathbb{P}[U \le c] = c$ for any $c \in [0,1]$ implying that $\mathbb{P}[U \le a(z)] = a(z)$. Therefore, we have

$$\mathbb{P}[V \text{ is accepted}] = \int_{\text{all } z} \frac{\tilde{p}_*(z)}{M p_r(z)}\, p_r(z)\, \mathrm{d}z = \frac{1}{M} \int_{\text{all } z} \tilde{p}_*(z)\, \mathrm{d}z = \frac{Z}{M}.$$

To complete, we solve for $M^\star$, the constant that maximizes our acceptance probability. Let $\mathcal{M} := \left\{M > 0 \mid M \ge \frac{\tilde{p}_*(z)}{p_r(z)}\ \forall z\right\}$ represent the space of M that satisfy $\frac{\tilde{p}_*(z)}{M p_r(z)} \le 1$ for all z. We have

$$M^\star = \underset{M \in \mathcal{M}}{\operatorname{argmax}} \left\{ \mathbb{P}[V \text{ is accepted}] \right\} = \underset{M \in \mathcal{M}}{\operatorname{argmax}} \left\{ \frac{Z}{M} \right\} = Z / \max_z \left\{ \frac{\tilde{p}_*(z)}{p_r(z)} \right\} \quad \Longrightarrow \quad M^\star = \max_z \left\{ \frac{\tilde{p}_*(z)}{p_r(z)} \right\}$$

For this model, the unnormalized target distribution $\tilde{p}_*(\alpha, \beta)$ is given by

$$p(\alpha, \beta \mid x_j, y_j, n_j) \propto p(y_j \mid \alpha, \beta, x_j, n_j)\, p(\alpha, \beta, x_j, n_j) = p(y_j \mid n_j, \theta_j)\, p(\alpha)\, p(\beta)\, p(x_j)\, p(n_j) \propto p(y_j \mid n_j, \theta_j)\, p(\alpha)\, p(\beta)$$

$$p(\alpha, \beta \mid x_j, y_j, n_j) \propto \prod_{j=1}^{J} \left[\binom{n_j}{y_j} \theta_j^{y_j} (1-\theta_j)^{n_j - y_j} \right] p(\alpha)\, p(\beta) \propto \prod_{j=1}^{J} \left[\theta_j^{y_j} (1-\theta_j)^{n_j - y_j} \right] p(\alpha)\, p(\beta) := \tilde{p}_*(\alpha, \beta)$$

and the proposal distribution $p_r(\alpha, \beta)$ is given by the prior:

$$p_r(\alpha, \beta) := p(\alpha)\ p(\beta)$$

Therefore the acceptance ratio is given by

$$a(V) := \frac{\tilde{p}_*(V)}{M p_r(V)} = \frac{\prod_{j=1}^{J} \left[\theta_j^{y_j} (1-\theta_j)^{n_j - y_j} \right] p(\alpha)\, p(\beta)}{M\, p(\alpha)\, p(\beta)} = \frac{1}{M^\star} \prod_{j=1}^{J} \left[\theta_j^{y_j} (1-\theta_j)^{n_j - y_j} \right]$$

where $M^\star$ is given by

$$M^\star = \max_z \left\{ \frac{\tilde{p}_*(z)}{p_r(z)} \right\} = \exp\left(\min_z \left\{ -\log\left(\frac{\tilde{p}_*(z)}{p_r(z)} \right) \right\} \right) = \exp\left(\min_z \left\{ \sum_{j=1}^{J} \left[-y_j \log(\theta_j) - (n_j - y_j) \log(1-\theta_j) \right] \right\} \right)$$

```python
from scipy.optimize import minimize
def neg_log_likelihood(params, x, y, n):
    alpha, beta = params
    theta = logit_inv(alpha+beta*x)
    return -np.sum(y * np.log(theta) + (n - y) * np.log(1 - theta))

bounds = ((0, None), (0, None))
result = minimize(neg_log_likelihood, (0, 0), args=(x_j, y_j, n_j), method='SLSQP', bounds=bounds)
alpha_star, beta_star = result.x
M_star = np.exp(-result.fun)
print(f"(alpha_star={alpha_star:0.3f}, beta_star={beta_star:0.3f})")
print(f"M_star={M_star:0.3f}")

def acceptance_ratio(M_star, params, x, y, n):
    M_cand = np.exp(-neg_log_likelihood(params, x, y, n))
    return M_cand / M_star

#---------------------------------#
#---Rejection Sampling Algorithm---#
#---------------------------------#
N_samples = 1000
```

```python
samples = []
count = 0
while len(samples)<1000:

    # Sample candidate alpha and beta from proposal distribution
    alpha_cand, beta_cand = stats.t.rvs(df=4, loc=0, scale=2), stats.t.rvs(df=4, loc=0, scale=1)
    candidate = alpha_cand, beta_cand

    # Compute acceptance ratio
    a = acceptance_ratio(M_star,candidate, x_j, y_j, n_j)

    # Sample Uniform([0,1]) random variable
    U = stats.uniform.rvs()

    # If U <= a, accept candidate
    if U <= a:
        # print(f"Accept candidate: {alpha_cand:0.3f}, {beta_cand:0.3f}")
        samples.append(candidate)

    count+=1

print(f"Rejection sampling required {count} attempts to collect {N_samples} samples")
```

Problem 5: Chapter 10, Problem 6 [BDA3]

Importance sampling when the importance weights are well behaved: consider a univariate posterior distribution, $p(\theta|y)$, which we wish to approximate and then calculate moments of, using importance sampling from an unnormalized density, $g(\theta)$. Suppose the posterior distribution is normal, and the approximation is t_3 with mode and curvature matched to the posterior density.

The posterior follows a normal distribution,

$$p_*(\theta) = p(\theta \mid y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{\theta-\mu}{\sigma}\right)^2\right)$$

Recall that X that follows a location-scale t distribution has a density of

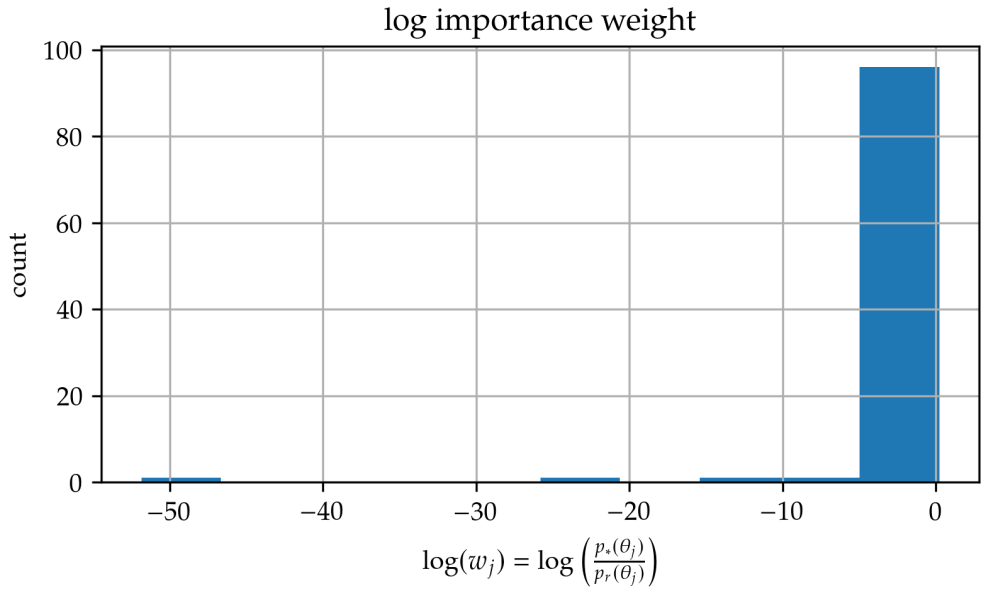
$$p(x \mid \nu, \mu, \tau) := \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}\tau}\left(1+\frac{1}{\nu}\left(\frac{x-\mu}{\tau}\right)^2\right)^{-(\nu+1)/2} \quad \Longrightarrow \quad p_r(\theta) \propto \left(1+\frac{1}{3}\left(\frac{\theta-\mu}{\tau}\right)^2\right)^{-2}$$

Recall that the curvature is given by $I(\hat{\theta}) = -\frac{\partial^2}{\partial\theta^2}\log\big(p(\theta \mid y)\big)$ where $\hat{\theta}$ is the mode of the distribution. The curvature for target and proposal distributions are given by

$$-\frac{\partial^2}{\partial\theta^2}\log\big(p_*(\theta)\big) = \frac{\partial^2}{\partial\theta^2}\left\{\frac{1}{2}\left(\frac{\theta-\mu}{\sigma}\right)^2 + f(\sigma^2)\right\}\Bigg|_{\theta=\hat{\theta}} = \frac{1}{\sigma^2}$$

$$-\frac{\partial^2}{\partial\theta^2}\log\big(p_r(\theta)\big) = \frac{\partial^2}{\partial\theta^2}\left\{2\log\left(1+\frac{1}{3}\left(\frac{\theta-\mu}{\tau}\right)^2\right)\right\}\Bigg|_{\theta=\hat{\theta}} = \left\{\frac{4}{3\tau^2\left(\frac{(\theta-\mu)^2}{3\tau^2}+1\right)} - \frac{8\,(\theta-\mu)^2}{9\tau^4\left(\frac{(\theta-\mu)^2}{3\tau^2}+1\right)^2}\right\}\Bigg|_{\theta=\hat{\theta}} = \frac{4}{3\tau^2}$$

(a) Draw a sample of size $S = 100$ from the approximate density and compute the importance ratios. Plot a histogram of the log importance ratios.




```python
import numpy as np
np.random.seed(7)
np.set_printoptions(precision=4)
import scipy.stats as stats
import matplotlib.pyplot as plt

```

```python
def check_samples(samples, pr, pt):
    "Resample to avoid dividing by zero or log(0)"
    approx_densities = pr.pdf(samples)
    post_densities   = pt.pdf(samples)
    while np.any((approx_densities == 0) | (post_densities == 0)):
        zero_indices = np.where((approx_densities == 0) | (post_densities == 0))[0]
        new_samples  = pr.sample(size=len(zero_indices))
        samples[zero_indices] = new_samples
        approx_densities[zero_indices] = pr.pdf(new_samples)
        post_densities[zero_indices]   = pt.pdf(new_samples)
    return samples

class StudentsT:

    def __init__(self, mean, tau, df=3):
        self._loc   = mean
        self._scale = tau
        self._df    = df
        self.var    = (self._scale**2) * self._df / (self._df - 2)

    def sample(self, size=100):
        return stats.t.rvs(df=self._df, loc=self._loc, scale=self._scale, size=size)

    def pdf(self, samples):
        return stats.t.pdf(samples, df=self._df, loc=self._loc, scale=self._scale)

class Normal:

    def __init__(self, mean, sigma):
        self._loc   = mean
        self._scale = sigma
        self.var    = self._scale**2

    def sample(self, size=100):
        return stats.norm.rvs(loc=self._loc, scale=self._scale, size=size)

    def pdf(self, samples):
        return stats.norm.pdf(samples, loc=self._loc, scale=self._scale)

S      = 100
mu     = 4
sigma  = 3
sigma2 = sigma**2
tau2   = (4/3)*sigma2
tau    = np.sqrt(tau2)

p_r = StudentsT(mean=mu, tau=tau)
p_t = Normal(mean=mu,sigma=sigma)

samples = p_r.sample(size=S)
samples = check_samples(samples, p_r, p_t)
importance_w = p_t.pdf(samples)/p_r.pdf(samples)

plt.figure(figsize=(6,3))
plt.hist(np.log(importance_w))
plt.xlabel(r"$\log(w_j) = \log\left(\frac{p_*(\theta_j)}{p_r(\theta_j)}\right)$")
plt.ylabel(r"count")
plt.title(r"log importance weight")
plt.grid(True)
plt.savefig('log_importance_w.png', bbox_inches='tight')
plt.show()
```

(b) Estimate $\mathbb{E}(\theta|y)$ and $\text{var}(\theta|y)$ using importance sampling. Compare to the true values.

	$\mathbb{E}(\theta\|y)$	$\text{var}(\theta\|y)$
true	4	9
approximate	3.869	9.825

```python
def calc(x, w):
    return np.mean(x*w)/np.mean(w)
mean = calc(samples**1,importance_w)
var  = calc(samples**2,importance_w) - mean**2

print(f"approx mean = {mean:0.3f}, true mean = {mu:0.3f}")
print(f"approx var  = {var:0.3f}, true var = {p_t.var:0.3f}")
```

(c) Repeat (a) and (b) for $S = 10{,}000$.

	$\mathbb{E}(\theta\|y)$	$\text{var}(\theta\|y)$
true	4	9
approximate	3.979	8.944

(d) Using the sample obtained in (c), compute an estimate of effective sample size using (10.4) on page 266.

The effective sample size is given by

$$S_{\text{eff}} = \frac{1}{\sum_{s=1}^{S}(\tilde{w}(\theta^s))^2} \quad \text{where} \quad \tilde{w}(\theta^s) := \frac{w(\theta^s)}{\sum_{s'=1}^{S} w(\theta^{s'})} \quad \text{are the normalized importance weights.}$$

I found that the effective sample size is $\boxed{S_{\text{eff}} \approx 8703}$

```python
w_tilde = importance_w / np.sum(importance_w)
S_eff   = 1/(np.sum(w_tilde**2))
print(f"S_eff = {S_eff:0.3f}")
```

Problem 6: Chapter 10, Problem 7 [BDA3]

Importance sampling when the importance weights are too variable: repeat the previous exercise, but with a t_3 posterior distribution and a normal approximation. Explain why the estimates of $\text{var}(\theta|y)$ are systematically too low.

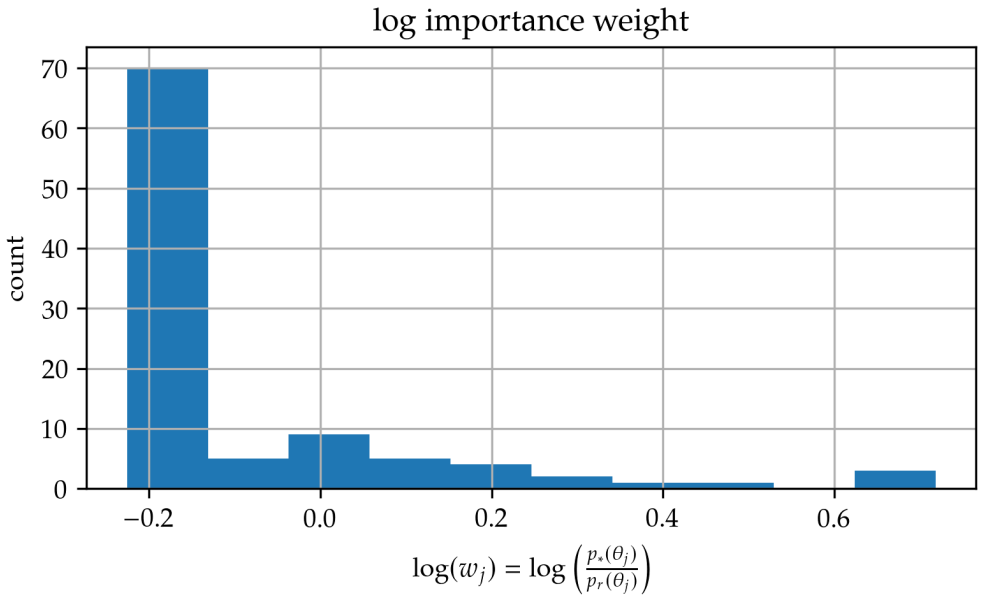
We run the same code as Chapter 10, Problem 6 except we now use

```
p_r = Normal(mean=mu,sigma=sigma)
p_t = StudentsT(mean=mu, tau=tau)
```

Additionally we change the parameters to such that they have the same true variance:

```
mu     = 4
tau2   = 3
tau    = np.sqrt(tau2)
sigma2 = (3/4)*tau2
sigma  = np.sqrt(sigma2)
```

(a) Draw a sample of size $S = 100$ from the approximate density and compute the importance ratios. Plot a histogram of the log importance ratios.



(b) Estimate $\mathbb{E}(\theta|y)$ and $\text{var}(\theta|y)$ using importance sampling. Compare to the true values.

Note that using the same parameters as before we have

	$\mathbb{E}(\theta\|y)$	$\text{var}(\theta\|y)$
true	4	9
approximate	4.044	3.087

(c) Repeat (a) and (b) for $S = 10{,}000$.

	$\mathbb{E}(\theta\|y)$	$\text{var}(\theta\|y)$
true	4	9
approximate	3.972	4.132

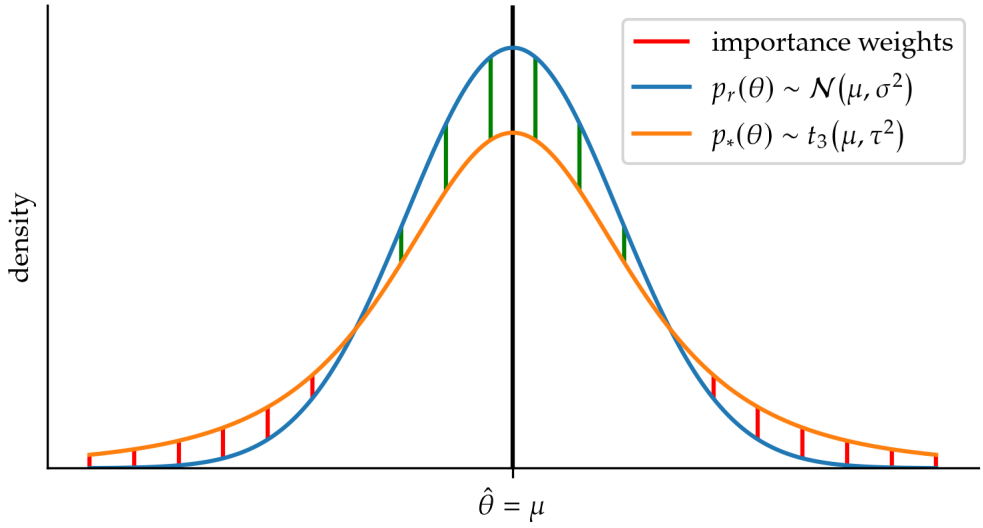
(d) Using the sample obtained in (c), compute an estimate of effective sample size using (10.4) on page 266.

The effective sample size is $\boxed{S_{\text{eff}} \approx 5407}$

Explanation as to why the estimates of var$(\theta|y)$ are systematically too low:
This mainly stems from the rapidly decaying tails of the normal distribution. In the figure, we plot the densities of both the proposal and target density. In the tails the importance weights ($w_j = \frac{p_*(\theta_j)}{p_r(\theta_j)}$) are extremely large because $p_r(\theta_j)$ is close to zero (see red bars in the figure below). Therefore, we put a large amount of weight on extreme samples. Then in our calculation of the variance, we have

$$\widehat{\text{var}}(\theta|y) = \frac{\sum_{s=1}^{S} w_s \theta_s^2}{\sum_{s'=1}^{S} w_{s'}} - \left(\frac{\sum_{s=1}^{S} w_s \theta_s}{\sum_{s'=1}^{S} w_{s'}} \right)^2 \approx \frac{\sum_{s=1}^{S} w_s \theta_s^2}{\sum_{s'=1}^{S} w_{s'}} - \mathbb{E}[\theta \mid y]^2$$

where a few the extreme samples will dominant both the numerator and denominator. As a result, our effective sample size decreases as our weights are extremely variable.



Problem 7: Chapter 11, Problem 3 [BDA3]

Gibbs sampling: Table 1 contains quality control measurements from 6 machines in a factory. Quality control measurements are expensive and time-consuming, so only 5 measurements were done for each machine. In addition to the existing machines, we are interested in the quality of another machine (the seventh machine). Implement a separate, a pooled and hierarchical Gaussian model with common variance described in Section 11.6. Run the simulations long enough for approximate convergence. Using each of three models—separate, pooled, and hierarchical—report:

Machine	Measurements
1	83, 92, 92, 46, 67
2	117, 109, 114, 104, 87
3	101, 93, 92, 86, 67
4	105, 119, 116, 102, 116
5	79, 97, 103, 79, 92
6	57, 92, 104, 77, 100

TABLE 1: Quality control measurements from 6 machines in a factory.

We begin by summarizing the three models:

1. **Seperate model**:
   - $p(\zeta_j) \propto 1$ for each machine $j \in \{1, 2, \ldots, J\}$,
   - $y_{ij} \mid \theta_j, \sigma \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\theta_j, \sigma^2)$ for $i \in \{1, 2, \ldots, n_j\}$ for a total of $n = \sum_{j=1}^{J} n_j$ total measurements.
2. **Pooled model**:
   - $y_{ij} \mid \eta, \sigma \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\eta, \sigma^2)$ for each machine $j \in \{1, 2, \ldots, J\}$ and for $i \in \{1, 2, \ldots, n_j\}$.
3. **Hierarchical model**:
   - $p(\mu, \log\sigma, \log\tau) \propto \tau$, we then draw J machine means,
   - $\theta_j \mid \mu, \tau \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mu, \tau^2)$ for $j \in \{1, 2, \ldots, J\}$, and then we draw n_j measurements from each of the J machines,
   - $y_{ij} \mid \theta_j, \sigma \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\theta_j, \sigma^2)$ for $i \in \{1, 2, \ldots, n_j\}$.

(i) the posterior distribution of the mean of the quality measurements of the sixth machine,

TODO

(ii) the predictive distribution for another quality measurement of the sixth machine,

TODO

(iii) the posterior distribution of the mean of the quality measurements of the seventh machine.

TODO

Problem 8: Chapter 11, Problem 5 [BDA3]

Monitoring convergence:

(a) Prove that $\widehat{\text{var}}^+(\psi \mid y)$ as defined in (11.3) is an unbiased estimate of the marginal posterior variance of ϕ, if the starting distribution for the Markov chain simulation algorithm is the same as the target distribution, and if the m parallel sequences are computed independently. (Hint: show that $\widehat{\text{var}}^+(\psi \mid y)$ can be expressed as the average of the halved squared differences between simulations ϕ from different sequences, and that each of these has expectation equal to the posterior variance.)

TODO

(b) Determine the conditions under which $\widehat{\text{var}}^+(\psi \mid y)$ approaches the marginal posterior variance of ϕ in the limit as the lengths n of the simulated chains approach ∞.

TODO

Problem 9: Chapter 11, Problem 6 [BDA3]

Effective sample size:

(a) Derive the asymptotic formula (11.5) for the variance of the average of correlated simulations.

Our goal is to show

$$\lim_{n\to\infty} mn \ \operatorname{Var}\left(\overline{\psi}_{\cdot\cdot}\right) = \left(1 + 2\sum_{t=1}^{\infty}\rho_t\right)\operatorname{Var}\left(\psi \mid y\right)$$

where ρ_t is the autocorrelation of the sequence ψ at lag t, i.e., $\rho_t = \frac{\operatorname{Cov}(\psi_{(i)1},\psi_{(i+t)1})}{\operatorname{Var}(\psi|y)}$ by wide-sense stationarity.

Proof. We have

$$\operatorname{Var}\left(\overline{\psi}_{\cdot\cdot}\right) = \operatorname{Var}\left(\frac{1}{m}\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\psi_{ij}\right) = \frac{1}{m^2n^2}\operatorname{Var}\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\psi_{ij}\right) = \frac{1}{m^2n^2}\sum_{j=1}^{m}\operatorname{Var}\left(\sum_{i=1}^{n}\psi_{ij}\right) = \frac{1}{mn^2}\operatorname{Var}\left(\sum_{i=1}^{n}\psi_{i1}\right)$$

as each of the $j \in \{1, 2, \ldots, m\}$ sequences are independent and identically distributed. Focusing on the term $\operatorname{Var}\left(\sum_{i=1}^{n}\psi_{i1}\right)$, we have

$$\operatorname{Var}\left(\sum_{i=1}^{n}\psi_{i1}\right) = \sum_{k=1}^{n}\sum_{\ell=1}^{n}\operatorname{Cov}\left(\psi_{k1},\psi_{\ell 1}\right) = \sum_{i=1}^{n}\operatorname{Var}\left(\psi_{i1}\right) + \sum_{\substack{k,\ell=1\\k\neq\ell}}^{n}\operatorname{Cov}\left(\psi_{k1},\psi_{\ell 1}\right) = n\operatorname{Var}\left(\psi \mid y\right) + 2\sum_{k=1}^{n}\sum_{\substack{\ell=1\\k<\ell}}^{n}\operatorname{Cov}\left(\psi_{k1},\psi_{\ell 1}\right)$$

Focusing on the last term, $\sum_{k=1}^{n}\sum_{\substack{\ell=1\\k<\ell}}^{n}\operatorname{Cov}\left(\psi_{k1},\psi_{\ell 1}\right)$ where we let $\operatorname{Cov}\left(\psi_{(i)1},\psi_{(i+t)1}\right) = \rho_t \operatorname{Var}(\psi \mid y)$, we have

$$\sum_{k=1}^{n}\sum_{\substack{\ell=1\\k<\ell}}^{n}\operatorname{Cov}\left(\psi_{k1},\psi_{\ell 1}\right) = (n-1)\operatorname{Cov}\left(\psi_{(i)1},\psi_{(i+1)1}\right) + (n-2)\operatorname{Cov}\left(\psi_{(i)1},\psi_{(i+2)1}\right) + \cdots + (n-(n-1))\operatorname{Cov}\left(\psi_{n1},\psi_{n1}\right)$$

$$\sum_{k=1}^{n}\sum_{\substack{\ell=1\\k<\ell}}^{n}\operatorname{Cov}\left(\psi_{k1},\psi_{\ell 1}\right) = \sum_{t=1}^{n}(n-t)\rho_t \operatorname{Var}(\psi \mid y) = n\sum_{t=1}^{n}\rho_t \operatorname{Var}(\psi \mid y) - \sum_{t=1}^{n}t\rho_t \operatorname{Var}(\psi \mid y)$$

Substituting our results in, we arrive at

$$\operatorname{Var}\left(\overline{\psi}_{\cdot\cdot}\right) = \frac{1}{mn^2}\left(n\operatorname{Var}\left(\psi \mid y\right) + 2n\sum_{t=1}^{n}\rho_t \operatorname{Var}(\psi \mid y) - 2\sum_{t=1}^{n}t\rho_t \operatorname{Var}(\psi \mid y)\right)$$

$$\operatorname{Var}\left(\overline{\psi}_{\cdot\cdot}\right) = \frac{1}{mn}\left(\operatorname{Var}\left(\psi \mid y\right) + 2\sum_{t=1}^{n}\rho_t \operatorname{Var}(\psi \mid y)\right) - \frac{2}{mn^2}\sum_{t=1}^{n}t\rho_t \operatorname{Var}(\psi \mid y)$$

Multiplying by mn and taking the limit, we arrive at

$$\lim_{n\to\infty} mn \ \operatorname{Var}\left(\overline{\psi}_{\cdot\cdot}\right) = \lim_{n\to\infty}\left\{\left(\operatorname{Var}\left(\psi \mid y\right) + 2\sum_{t=1}^{n}\rho_t \operatorname{Var}(\psi \mid y)\right) - \frac{2}{n}\sum_{t=1}^{n}t\rho_t \operatorname{Var}(\psi \mid y)\right\}$$

$$\lim_{n\to\infty} mn \ \operatorname{Var}\left(\overline{\psi}_{\cdot\cdot}\right) = \left(1 + 2\sum_{t=1}^{\infty}\rho_t\right)\operatorname{Var}\left(\psi \mid y\right) \quad \checkmark$$

□

STAT238: Bayesian Statistics April 29th, 2025

Homework 5: Tempering, HMC, & Model Checking

Reece D. Huff

Problems

Problem 1: MCMC Sampling in Mixture Models [BDA3]

Sometimes it helps to expand the space of unknowns we aim to sample to include additional variables (e.g. the temperature in simulated tempering, or the momentum in Hamiltonian Monte Carlo). We've also seen that it can be useful to expand models hierarchically, by expressing a desired distribution as a mixture of simpler distributions.

Let X denote the unknowns of interest, and Z the additional unknowns introduced in the model or sampler. Let $Y = y$ denote the observed data.

In both settings, we usually want to sample from a target distribution on X. In a Bayesian setting, this target is usually either the marginal posterior for X marginalizing out Z (as in HMC, or when Z is a latent parameter in a mixture model), or the conditional posterior for X conditioned on a particular value of Z (as in simulated tempering).

As a specific example, consider the mixture model setting where we first sample $X \sim p_X$, then $Z|X \sim p_{Z|X}$, and finally, $Y|Z \sim p_{Y|Z}$. Suppose, in addition, that we can derive, and sample from, the exact conditional $Z|X = x, Y = y$.

Then, consider the following joint MCMC procedure on X, Z. Given X_t, Z_t:

1. Sample a candidate X' from a symmetric proposal distribution given the current iterate X_t. Let $p_r(x \to x')$ denote the probability that we propose $X' = x'$ given $X_t = x$.
2. Sample $Z'|X', Y = y$ using its exact conditional, $p_{Z|X,Y}$.
3. Accept or reject the joint candidate X', Z' as a single Metropolis-Hasting step on the joint posterior.

(a) Derive the acceptance probability needed so that the MH procedure described above samples from the joint posterior $X, Z|Y = y$.

In general, the Metropolis-Hasting acceptance probability is given by

$$a(s \to s') = \min\left\{1, \frac{\tilde{p}_*(s')\, p_r(s' \to s)}{\tilde{p}_*(s)\, p_r(s \to s')}\right\}$$

where s is the unknown, $\tilde{p}_*$ is the unnormalized target distribution and $p_r(s \to s')$ is the proposal distribution. In our case, we have

$$a(X, Z \to X', Z') = \min\left\{1, \frac{\tilde{p}_*(X', Z' \mid Y = y)\, p_r(X', Z' \to X, Z \mid Y = y)}{\tilde{p}_*(X, Z \mid Y = y)\, p_r(X, Z \to X', Z' \mid Y = y)}\right\}$$

By our sampling stategy, we have that

$$p_r(X, Z \to X', Z' \mid Y = y) = p_r(X \to X')\, p(Z' \mid X', Y = y)$$

Therefore, our acceptance probability is given by

$$a(X, Z \to X', Z') = \min\left\{1, \frac{\tilde{p}_*(X', Z' \mid Y = y)\, p_r(X' \to X)\, p(Z \mid X, Y = y)}{\tilde{p}_*(X, Z \mid Y = y)\, p_r(X \to X')\, p(Z' \mid X', Y = y)}\right\}$$

$$a(X, Z \to X', Z') = \min\left\{1, \frac{\tilde{p}_*(X', Z' \mid Y = y)\, p(Z \mid X, Y = y)}{\tilde{p}_*(X, Z \mid Y = y)\, p(Z' \mid X', Y = y)}\right\}$$

(b) Describe how you could use this joint sampling procedure to sample from the marginal posterior $X|Y = y$. Is your procedure the same as Metropolis sampling on the target $X|Y = y$ for some choice of proposal?

We can on sample the target $X|Y = y$ using the same procedure as above. We note that we can split the target into

$$\tilde{p}_*(X, Z \mid Y = y) = p(Z \mid X, Y = y)\, \tilde{p}_*(X \mid Y = y)$$

as we have access to $p(Z \mid X, Y = y)$. Substituting our results in, the acceptance probability is then given by

$$a(X, Z \to X', Z') = \min\left\{1, \frac{p(Z' \mid X', Y = y)\, \tilde{p}_*(X' \mid Y = y)\, p(Z \mid X, Y = y)}{p(Z \mid X, Y = y)\, \tilde{p}_*(X \mid Y = y)\, p(Z' \mid X', Y = y)}\right\} = \min\left\{1, \frac{\tilde{p}_*(X' \mid Y = y)}{\tilde{p}_*(X \mid Y = y)}\right\}$$

which equal to the acceptance probability of a Metropolis algorithm sampling from the target $X|Y = y$.

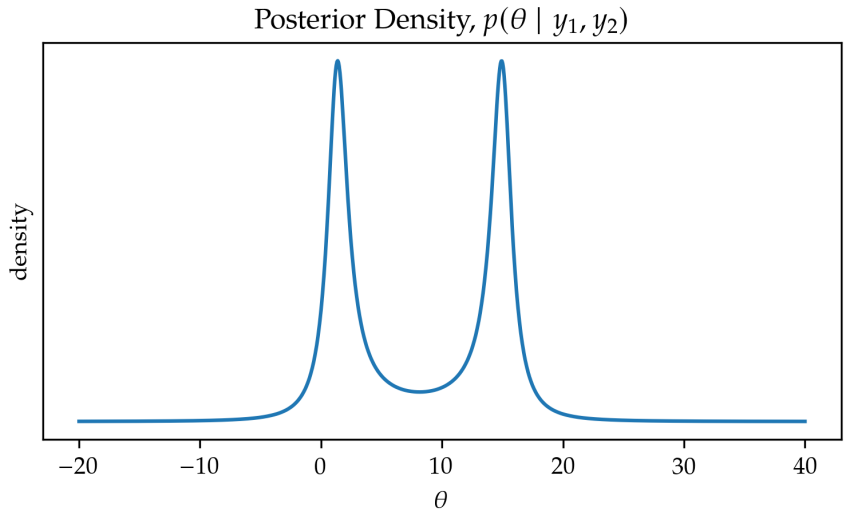
Problem 2: Chapter 12, Problem 2: Simulated Tempering [BDA3]

Consider the Cauchy model, $y_i \sim \text{Cauchy}(\theta, 1)$, $i = 1, \ldots, n$, with uniform prior on θ, and two data points, $y_1 = 1.3$, $y_2 = 15.0$.

(a) Graph the posterior density.

We have $y_i \mid \theta \overset{\text{i.i.d.}}{\sim} y_i \sim \text{Cauchy}(\theta, 1)$ with density $p(y_i \mid \theta) = \frac{1}{\pi[1+(y_i-\theta)^2]}$. The uniform prior has density $p(\theta) \propto 1$. It then follows that

$$p(\theta \mid y_1, y_2) \propto p(y_1, y_2 \mid \theta)\, p(\theta) = p(y_1 \mid \theta)\, p(y_2 \mid \theta)\, p(\theta) \propto \left(\frac{1}{\pi\left[1+(y_1-\theta)^2\right]}\right)\left(\frac{1}{\pi\left[1+(y_2-\theta)^2\right]}\right)$$




```python
import numpy as np
np.random.seed(7)
import scipy.stats as stats
import matplotlib.pyplot as plt

def posterior(t,y1=1.3,y2=15.0):
    return stats.cauchy.pdf(t,loc=y1)*stats.cauchy.pdf(t,loc=y2)

theta = np.linspace(-20,40,1000)
posterior_density = posterior(theta)
posterior_density /= np.trapezoid(posterior_density, theta)

plt.figure(figsize=(6,3))
plt.gca().set_yticks([])
plt.plot(theta, posterior_density)
plt.xlabel(r'$\theta$')
plt.ylabel('density')
plt.title('Posterior Density, $p(\\theta \\mid y_1, y_2)$')
plt.savefig('post_density.12.2.png', bbox_inches='tight')
plt.show()
```

(b) Program the Metropolis algorithm for this problem using a symmetric Cauchy jumping distribution. Tune the scale parameter of the jumping distribution appropriately.

We begin by writing out the Metropolis algorithm:

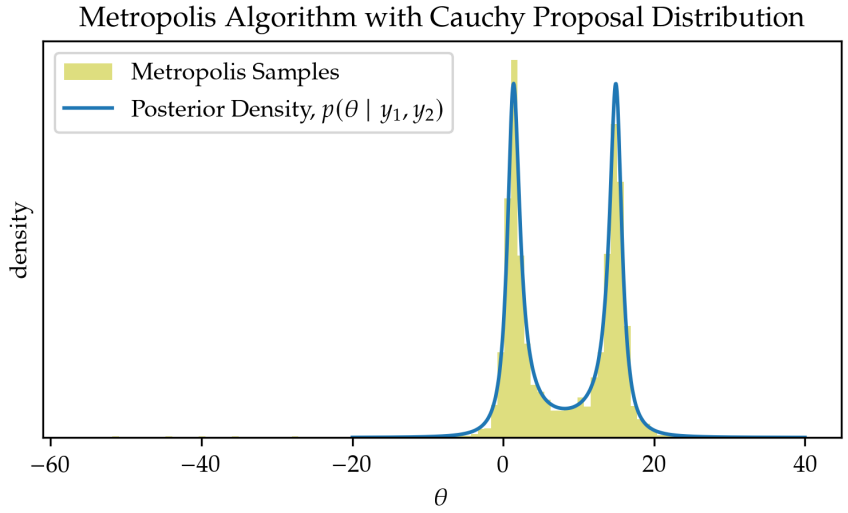
ALGORITHM 2: Metropolis Algorithm

Input: unnormalized target distribution $\tilde{p}_*$ and symmetric proposal distribution p_r

Initialize: Initialize empty set of samples $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**
  Draw a candidate sample from the proposal distribution: $Y \sim p_r(X_t \to Y)$
  Compute acceptance ratio: $r = \tilde{p}_*(Y)/\tilde{p}_*(X_t)$ and $a(Y) = \min(r, 1)$
  Draw $U \sim \text{Uniform}([0, 1])$
  if $U \leq a(Y)$ **then**
    Accept candidate Y, set $X_{t+1} = Y$, and append Y to $\mathcal{S}$
  else
    Reject candidate Y, set $X_{t+1} = X_t$, and discard Y
  end
end
return $\mathcal{S}$

I tuned the scale parameter to 2 as it accepted about $\sim 50\%$ of the candidates.




```python
# Metropolis sampler
def metropolis_cauchy(n_iter, theta_init, scale):

    theta_samples = np.zeros(n_iter)
    theta_current = theta_init
    n_accepted    = 0

    for i in range(n_iter):

        # Symmetric Cauchy jumping distribution
        theta_proposed = stats.cauchy.rvs(loc=theta_current, scale=scale)

        # Acceptance probability
        accept_prob = min(1, posterior(theta_proposed) / posterior(theta_current))

        # Accept or reject
        if np.random.rand() < accept_prob:
            theta_current = theta_proposed
            n_accepted   += 1

        theta_samples[i] = theta_current

    print(f"Percent of candidates accepted {100*(n_accepted/n_iter):0.2f}%")
    return theta_samples
```

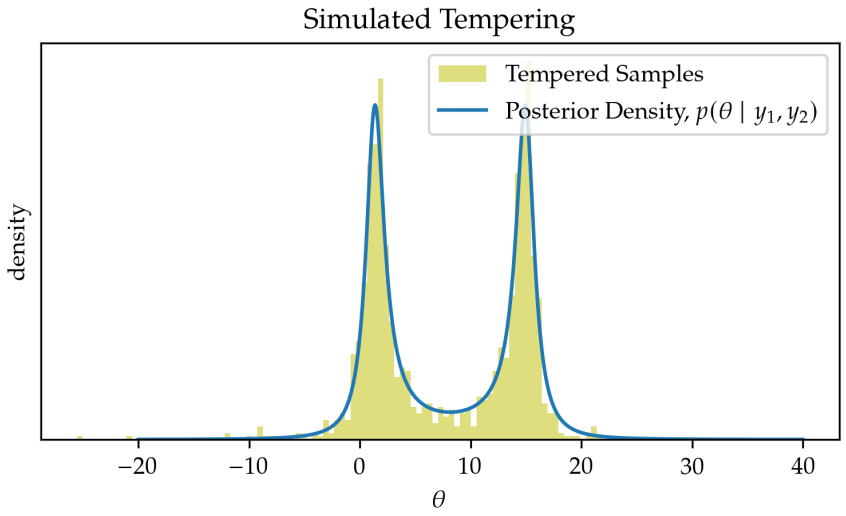
```
25
26 # Parameters
27 n_iter     = 10000
28 scale      = 2.0
29 theta_init = (1.3+15.0)/2
30
31 # Run Metropolis sampler
32 samples = metropolis_cauchy(n_iter, theta_init, scale)
33
34 # Plot the posterior samples
35 plt.figure(figsize=(6,3))
36 plt.gca().set_yticks([])
37 plt.hist(samples, bins=100, density=True, alpha=0.5, label='Metropolis Samples', color='y')
38 plt.plot(theta, posterior_density, label='Posterior Density, $p(\\theta \\mid y_1, y_2)$')
39 plt.xlabel(r'$\theta$')
40 plt.ylabel('density')
41 plt.title('Metropolis Algorithm with Cauchy Proposal Distribution')
42 plt.legend()
43 plt.savefig('metropolis.12.2.png', bbox_inches='tight')
44 plt.show()
```

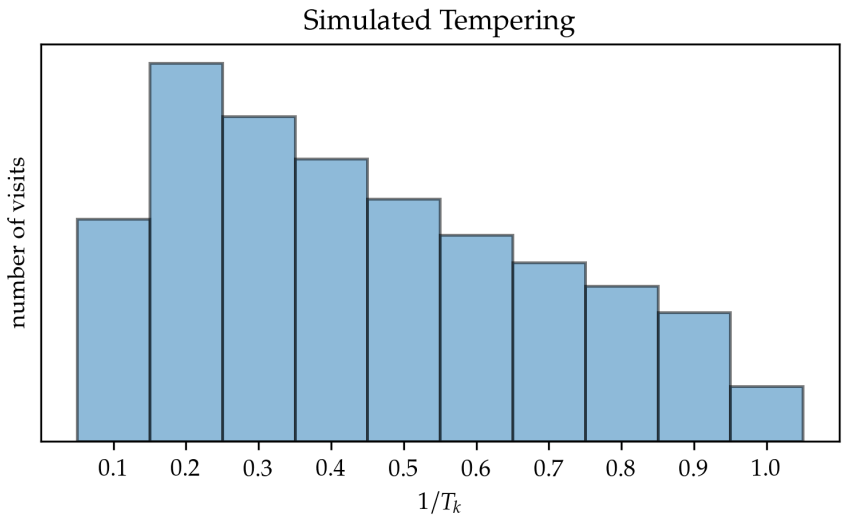
(c) Program simulated tempering with a ladder of 10 inverse-temperatures, $0.1, \ldots, 1$.

Simulated tempering works by setting a ladder of unnormalized densities q_k

$$q_k(\theta) = p(\theta \mid y)^{1/T_k} p_0(\theta)^{1-1/T_k} \quad \text{for} \quad k = 0, 1, 2, \ldots, K$$

for a set of 'temperature' parameters $T_k > 0$. Setting $T_k = 1$ reduces to the original density, and large values of T_k produce less highly peaked modes, going in the limit to some 'base measure' p_0 that is some convenient prechosen distribution with high variance [BDA3].






```python
# Tempered unnormalized density
def q_k(theta_proposed,inv_temp,p_0_scale=10):
    post_density = posterior(theta_proposed)
    p_0_density  = (stats.cauchy.pdf(theta_proposed,loc=theta_proposed, scale=p_0_scale))**2
    return (post_density**inv_temp)*(p_0_density**(1-inv_temp))

# Simulated tempering
def simulated_tempering(n_iter, theta_init, scale, taus):

    theta_samples    = np.zeros(n_iter)
    tau_samples      = np.zeros(n_iter)
    theta_t          = theta_init
    tau_idx          = len(taus) - 1
    tau_t            = taus[tau_idx]
```

```
    tau_indices      = list(np.arange(len(taus)))
    n_theta_accepted = 0
    n_tau_accepted   = 0

    for i in range(n_iter):

        # Symmetric Cauchy jumping distribution
        candidate = stats.cauchy.rvs(loc=theta_t, scale=scale)

        # Acceptance probability
        accept_prob = min(1, posterior(candidate) / posterior(theta_t))

        # Accept or reject
        if np.random.rand() < accept_prob:
            theta_t           = candidate
            n_theta_accepted += 1

        #---My version---#
        # # Sample candidate tau (inverse temperate)
        # tau_idx_candidate = np.random.choice(np.delete(np.arange(len(taus)),tau_idx))
        # tau_candidate     = taus[tau_idx_candidate]

        # # Acceptance probability
        # counts, _ = np.histogram(np.array(tau_indices), bins=np.arange(len(taus)+1))
        # c_k = counts / counts.sum()
        # c_j = c_k[tau_idx_candidate]
        # c_s = c_k[tau_idx]
        # tau_accept_prob = min(1, ((1/c_j)*q_k(theta_t,tau_candidate)) / ((1/c_s)*q_k(theta_t,tau_t)))

        #---Simplier version---#
        # Sample candidate tau (inverse temperate)
        if tau_idx == 0:
            tau_idx_candidate = tau_idx + 1
        elif tau_idx == len(taus)-1:
            tau_idx_candidate = tau_idx - 1
        else:
            tau_idx_candidate = tau_idx + np.random.choice([-1,1])

        tau_candidate     = taus[tau_idx_candidate]

        # Acceptance probability
        tau_accept_prob = min(1, q_k(theta_t,tau_candidate) / q_k(theta_t,tau_t))

        # Accept or reject
        if np.random.rand() < tau_accept_prob:
            tau_t           = tau_candidate
            tau_idx         = tau_idx_candidate
            n_tau_accepted += 1

        # Save samples
        theta_samples[i] = theta_t
        tau_samples[i]   = tau_t
        tau_indices.append(tau_idx)

    print(f"Percent of theta candidates accepted {100*(n_theta_accepted/n_iter):0.2f}%")
    print(f"Percent of  tau  candidates accepted {100*(n_tau_accepted/n_iter):0.2f}%")

    return theta_samples, tau_samples

# Parameters
n_iter     = 50000
scale      = 2.0
taus       = np.linspace(0.1, 1.0, 10) # tau_k = 1/T_k
theta_init = (1.3+15.0)/2
```

```python

# Run simulated tempering
theta_samples, tau_samples = simulated_tempering(n_iter, theta_init, scale, taus)

samples = theta_samples[tau_samples==1]

# Plot simulated tempering
plt.figure(figsize=(6,3))
plt.gca().set_yticks([])
plt.hist(samples, bins=100, density=True, alpha=0.5, label='Tempered Samples', color='y')
plt.plot(theta, posterior_density, label='Posterior Density, $p(\\theta \\mid y_1, y_2)$')
plt.xlabel(r'$\theta$')
plt.ylabel('density')
plt.title('Simulated Tempering')
plt.legend()
plt.savefig('tempering.12.2.png', bbox_inches='tight')
plt.show()

# Check that we visit each distribution equally
plt.figure(figsize=(6,3))
plt.gca().set_yticks([])
plt.hist(tau_samples, bins=[t-0.05 for t in taus] + [1.05], alpha=0.5, edgecolor='black', linewidth=1.2)
plt.gca().set_xticks([t for t in taus],[f"{t:0.1f}" for t in taus])
plt.xlabel(r'$1/T_k$')
plt.ylabel('number of visits')
plt.title('Simulated Tempering')
plt.savefig('tempering_visits.12.2.png', bbox_inches='tight')
plt.show()
```

(d) Compare your answers in (b) and (c) to the graph in (a).

The results between the two are quite similar.

Problem 3: Chapter 12, Problem 3: Hamiltonian Monte Carlo [BDA3]

Program Hamiltonian Monte Carlo (HMC) in `Python` for the bioassay logistic regression example from Chapter 3.

Dose, x_i (log g/ml)	Number of animals, n_i	Number of deaths, y_i
−0.86	5	0
−0.30	5	1
−0.05	5	3
0.73	5	5

Table 2: Bioassay data from Racine et al. (1986)

(a) Code the gradients analytically and numerically and check that the two programs give the same result.

Recall the logistic regression model from Chapter 3 [BDA3]. We have that

$$y_i \mid n_i, \theta_i \overset{\text{i.i.d.}}{\sim} \text{Binomial}(n_i, \theta_i) \quad \text{where} \quad \theta_i \mid x_i = \text{logit}^{-1}(\alpha + \beta x_i) = \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)} = \frac{1}{1 + \exp(-\alpha - \beta x_i)}$$

We assume a uniform prior over α, β such that $p(\alpha, \beta) \propto 1$. It then follows that

$$p(\alpha, \beta \mid y, n, x) \propto p(\alpha, \beta \mid n, x)\, p(y \mid \alpha, \beta, n, x) = p(\alpha, \beta) \prod_{i=1}^{k} p(y_i \mid \alpha, \beta, n_i, x_i)$$

$$p(\alpha, \beta \mid y, n, x) \propto \prod_{i=1}^{k} \left[\text{logit}^{-1}(\alpha + \beta x_i)\right]^{y_i} \left[1 - \text{logit}^{-1}(\alpha + \beta x_i)\right]^{n_i - y_i}$$

Then the log of the unnormalized target distribution is given by

$$\log(\tilde{p}_*(\alpha, \beta)) = \sum_{i=1}^{k} y_i \log\left[\text{logit}^{-1}(\alpha + \beta x_i)\right] + (n_i - y_i) \log\left[1 - \text{logit}^{-1}(\alpha + \beta x_i)\right]$$

$$\log(\tilde{p}_*(\alpha, \beta)) = \sum_{i=1}^{k} y_i \log\left[\text{logit}^{-1}(\alpha + \beta x_i)\right] + (n_i - y_i) \log\left[1 - \text{logit}^{-1}(\alpha + \beta x_i)\right]$$

We have

$$\log\left[\text{logit}^{-1}(\theta_i)\right] = \log\left[\frac{1}{1 + \exp(-\theta_i)}\right] \qquad \log\left[1 - \text{logit}^{-1}(\theta_i)\right] = \log\left[\frac{1}{1 + \exp(\theta_i)}\right]$$

$$\log\left[\text{logit}^{-1}(\theta_i)\right] = -\log\left[1 + \exp(-\theta_i)\right] \qquad \log\left[1 - \text{logit}^{-1}(\theta_i)\right] = -\log\left[1 + \exp(\theta_i)\right]$$

$$= -\log\left[\exp(\theta_i)(1 + \exp(-\theta_i))\right]$$

$$= -\theta_i - \log\left[1 + \exp(-\theta_i)\right]$$

It then follows that

$$\log(\tilde{p}_*(\alpha, \beta)) = \sum_{i=1}^{k} y_i \log\left[\text{logit}^{-1}(\theta_i)\right] + (n_i - y_i) \log\left[1 - \text{logit}^{-1}(\theta_i)\right]$$

$$\log(\tilde{p}_*(\alpha, \beta)) = \sum_{i=1}^{k} -y_i \log\left[1 + \exp(-\theta_i)\right] - (n_i - y_i)\, \theta_i - (n_i - y_i) \log\left[1 + \exp(-\theta_i)\right]$$

$$\log(\tilde{p}_*(\alpha, \beta)) = \sum_{i=1}^{k} -(n_i - y_i)\, \theta_i - n_i \log\left[1 + \exp(-\theta_i)\right]$$

Taking the derivative of $\log(\tilde{p}_*(\alpha, \beta))$ with respect to α and β, we arrive at

$$\frac{\partial \log(\tilde{p}_*(\alpha, \beta))}{\partial \alpha} = \sum_{i=1}^{k} -(n_i - y_i) + n_i \frac{\exp(-\alpha - \beta x_i)}{1 + \exp(-\alpha - \beta x_i)} = \sum_{i=1}^{k} -(n_i - y_i) + \frac{n_i}{1 + \exp(\alpha + \beta x_i)}$$

$$\frac{\partial \log(\tilde{p}_*(\alpha,\beta))}{\partial \beta} = \sum_{i=1}^{k} -(n_i - y_i)x_i + \frac{n_i x_i}{1+\exp(\alpha+\beta x_i)}$$

Next we verify that the analytical gradient is the same as the numerical gradient using autograd:

```python
import torch as pt

def log_grad(y,n,x,a,b):
    theta  = a+b*x
    a_grad = -(n-y) + n/(1+pt.exp(theta))
    b_grad = a_grad*x
    return pt.sum(a_grad), pt.sum(b_grad)

def log_target(y,n,x,a,b):
    theta = a+b*x
    term1 = y*pt.log(pt.sigmoid(theta))
    term2 = (n-y)*pt.log(1-pt.sigmoid(theta))
    return pt.sum(term1+term2)

y_i   = pt.tensor([0., 1., 3., 5.])
n_i   = pt.tensor([5., 5., 5., 5.])
x_i   = pt.tensor([-0.86, -0.30, -0.05, 0.73])
alpha = pt.tensor(1.0, requires_grad=True)
beta  = pt.tensor(2.0, requires_grad=True)

analytical = log_grad(y_i,n_i,x_i,alpha,beta)
autograd   = log_target(y_i, n_i, x_i, alpha, beta)
autograd.backward()

print("Analytical gradient: ", f"{analytical[0].item():0.2f}, {analytical[1].item():0.2f}")
print("Autograd gradient  : ", f"{alpha.grad.item():0.2f}, {beta.grad.item():0.2f}")
```

```
Analytical gradient:  −3.79, 2.32
Autograd gradient  :  −3.79, 2.32
```

(b) Pick reasonable starting values for the mass matrix, step size, and number of steps.

I started with a mass matrix of $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, step size of 0.1, and number of steps of 10. This is just to estimate $\text{Cov}(\alpha,\beta \mid y,n,x)^{-1}$ that I use in the next step.

(c) Tune the algorithm to an approximate 65% acceptance rate.

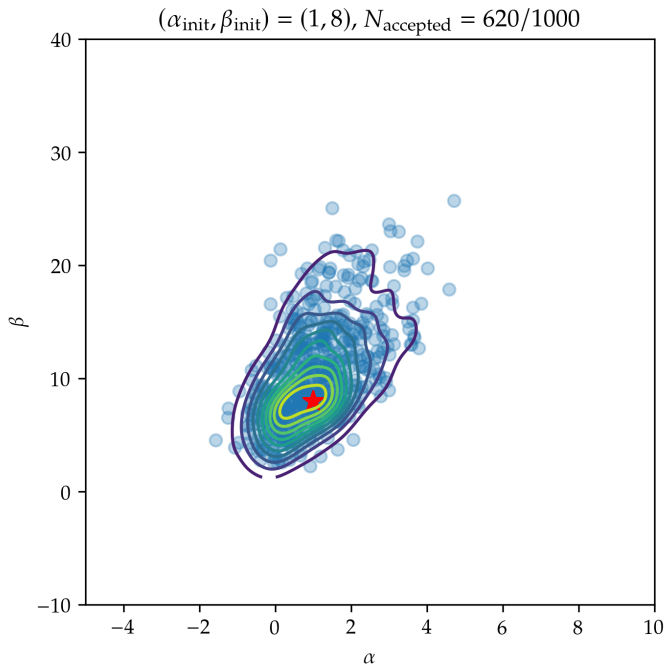
After a bit of burn in, I estimated the variance of the posterior from my samples and found that $\text{Cov}(\alpha,\beta \mid y,n,x) \approx \begin{bmatrix}1 & 3\\ 3 & 25\end{bmatrix}$ and therefore I set my mass matrix to $M \leftarrow \text{Cov}(\alpha,\beta \mid y,n,x)^{-1} \approx \begin{bmatrix}1.5625 & -0.1875\\ -0.1875 & 0.0625\end{bmatrix}$. From there, I tuned the step size to 0.6 and the number of steps to 20 to achieve an approximate 65% acceptance rate and an effective sample size of at least 100.

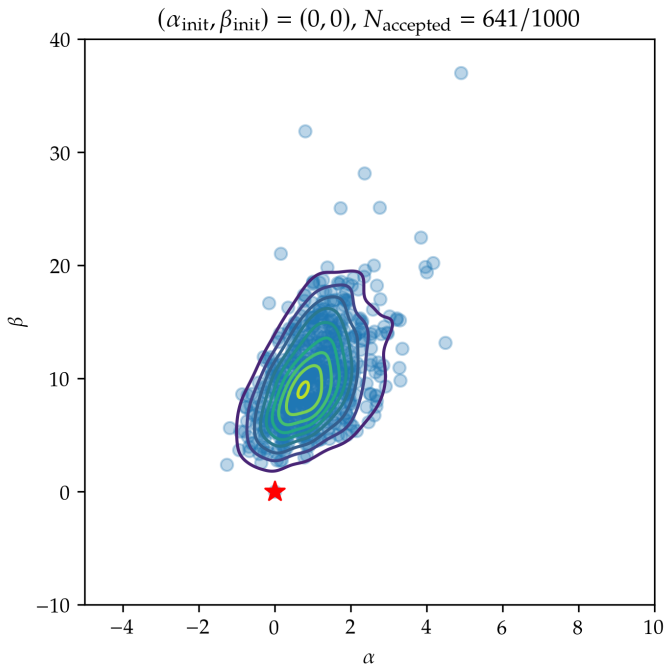
(d) Run 4 chains long enough so that each has an effective sample size of at least 100. How many iterations did you need?

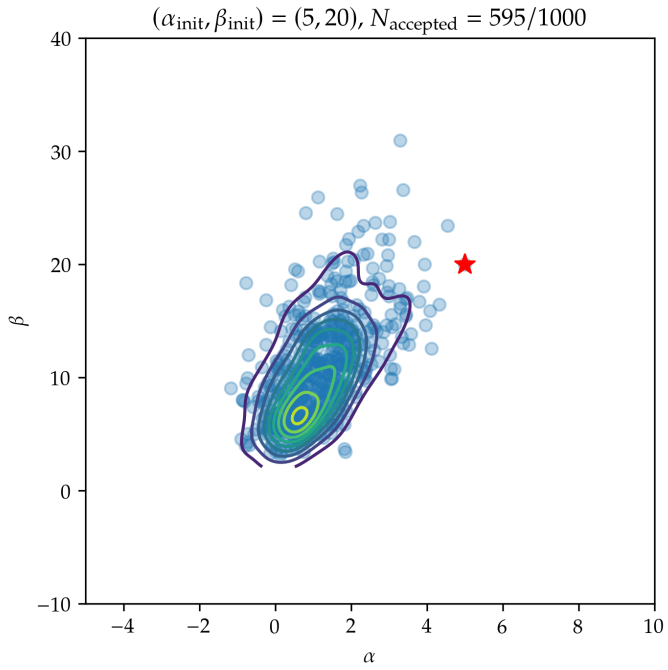
I needed about 1000 iterations to achieve an effective sample size of at least 100. Increasing the number of steps led to a large effective sample size with fewer iterations.

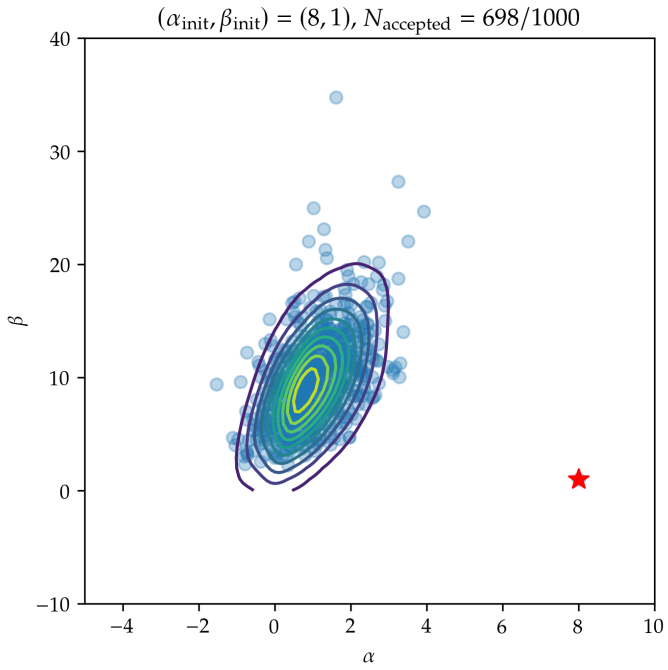
(e) Check that your inferences are consistent with those from the direct approach in Chapter 3.

Both the samples are the contours are quite similar to those in Chapter 3 which is a good sign. Even when using an initial guess in regions of extremely low probability the HMC algorithm was ultimately able to collect useful samples for making posterior inference.


$(\alpha_{\text{init}}, \beta_{\text{init}}) = (1, 8)$, $N_{\text{accepted}} = 620/1000$
α
β


$(\alpha_{\text{init}}, \beta_{\text{init}}) = (0, 0)$, $N_{\text{accepted}} = 641/1000$
α
β


$(\alpha_{\text{init}}, \beta_{\text{init}}) = (5, 20)$, $N_{\text{accepted}} = 595/1000$
α
β


$(\alpha_{\text{init}}, \beta_{\text{init}}) = (8, 1)$, $N_{\text{accepted}} = 698/1000$
α
β

```python
import numpy as np
np.random.seed(7)
import scipy
import scipy.stats as stats
import arviz as az
import matplotlib.pyplot as plt

class HMC:
    def __init__(self,y,n,x,Mass,epsilon,n_steps):
        self.y = y
        self.n = n
        self.x = x
        self.M = Mass
        self.Minv = np.linalg.inv(self.M)
        self.eps     = epsilon
        self.n_steps = n_steps

    def log_grad(self,x_t):
        a, b   = x_t
        theta  = a+b*self.x
        a_grad = self.y - self.n * scipy.special.expit(theta)
        b_grad = a_grad*self.x
        return np.array([np.sum(a_grad), np.sum(b_grad)])

    def target_log_prop(self,x_t):
        a, b  = x_t
        theta = a+b*self.x
        term1 = -self.y*np.logaddexp(0.0, -theta)
        term2 = -(self.n-self.y)*np.logaddexp(0.0, theta)
        return np.sum(term1+term2)

    def phi_sample(self,size=1):
        return stats.multivariate_normal.rvs(mean=np.array([0.,0.]), cov=self.M, size=size)

    def phi_pdf(self,samples):
        return stats.multivariate_normal.pdf(samples,mean=np.array([0.,0.]), cov=self.M)

    def phi_log_prop(self,samples):
        return np.log(self.phi_pdf(samples))

    def run(self,x_init,n_iter=1000):

        samples    = [x_init.copy()]
        x_t        = x_init.copy()
        n_accepted = 0

        for i in range(n_iter):
            x_t, accepted = self.step(x_t)
            n_accepted   += int(accepted)
            if accepted:
                samples.append(x_t.copy())

        print(f"Percent of candidates accepted {100*(n_accepted/n_iter):0.2f}%")
        return np.array(samples)

    def step(self,x_init):

        phi0    = self.phi_sample()
        phi_t   = phi0
        x_t     = x_init
        all_phi = [phi_t]

        for _ in range(self.n_steps):
            phi_t_half = phi_t + (1/2)*self.eps*self.log_grad(x_t)
```

```
            x_t_half   =  x_t  + self.eps*(self.Minv@phi_t_half)
            phi_t      = phi_t_half + (1/2)*self.eps*self.log_grad(x_t_half)
            x_t        = x_t_half
            all_phi.append(phi_t)

        phi_log_prob_0 = self.phi_log_prop(phi0)
        phi_log_prob_T = self.phi_log_prop(phi_t)
        x_log_prob_0   = self.target_log_prop(x_init)
        x_log_prob_T   = self.target_log_prop(x_t)
        delta          = x_log_prob_T - x_log_prob_0 + phi_log_prob_T - phi_log_prob_0
        accept_prop    = min(1, np.exp(delta))

        if np.random.rand() < accept_prop:
            return x_t, True
        else:
            return x_init, False

    def plot_contour(self,samples,n_iters, save=None):

        from scipy.stats import gaussian_kde
        x1 = samples[:, 0]
        x2 = samples[:, 1]
        values = np.vstack([x1, x2])
        kernel = gaussian_kde(values)
        xmin, xmax = x1.min() - 1, x1.max() + 1
        ymin, ymax = x2.min() - 1, x2.max() + 1
        X, Y = np.meshgrid(
            np.linspace(xmin, xmax, 200),
            np.linspace(ymin, ymax, 200)
        )
        positions = np.vstack([X.ravel(), Y.ravel()])
        Z = np.reshape(kernel(positions).T, X.shape)

        plt.figure(figsize=(5,5))
        plt.contour(X, Y, Z, levels=10)
        plt.scatter(x1, x2, alpha=0.3)
        plt.scatter(x1[0], x2[0], marker='*', color='red', s=100)
        plt.xlabel(r'$\alpha$')
        plt.ylabel(r'$\beta$')
        plt.title(f'$(\\alpha_{{\\text{{init}}}},\\beta_{{\\text{{init}}}})=({x1[0]:0.0f},{x2[0]:0.0f})
        ↪   ↪$, $N_{{\\text{{accepted}}}} = {len(x1)-1}/{n_iters}$')
        # Use sample limits as Figure 3.3 in BDA3
        plt.xlim([-5,10])
        plt.ylim([-10,40])
        if save:
            plt.savefig(save, bbox_inches='tight')
        plt.show()

# Data
y_i     = np.array([0., 1., 3., 5.])
n_i     = np.array([5., 5., 5., 5.])
x_i     = np.array([-0.86, -0.30, -0.05, 0.73])

# Initial sampling
# epsilon = 0.1
# n_steps = 10
# n_iter  = 1000
# M       = np.array([[1., 0.],
#                     [0., 1.]])

# 65% acceptance probability
epsilon = 0.6
n_steps = 20
n_iter  = 1000
```

```
M       = np.linalg.inv( np.array([[1., 3.],
                                   [3., 25.]]))

# Initial guesses
x_inits = [np.array([1, 8]),
           np.array([0, 0]),
           np.array([5, 20]),
           np.array([8, 1])]

# Run Hamiltonian Monte Carlo
hmc     = HMC(y_i, n_i, x_i, M, epsilon, n_steps)
for i, x_init in enumerate(x_inits):

    # Run HMC
    samples = hmc.run(x_init,n_iter)
    hmc.plot_contour(samples,n_iter,save=f"hmc.plot.{i}.12.3.png")

    # Calculate effective sample size
    idata    = az.from_dict(posterior={"a": samples[:,0], "b": samples[:,1]})
    ess_dict = az.ess(idata)
    print(ess_dict)
```

Problem 4: Chapter 6, Problem 1: Posterior Predictive Checking [BDA3]

(a) On page 120 [BDA3], the data from the SAT coaching experiments were checked against the model that assumed identical effects in all eight schools: the expected order statistics of the effect sizes were (26, 19, 14, 10, 6, 2, −3, −9), compared to observed data of (28, 18, 12, 8, 7, 1, −1, −3). Express this comparison formally as a posterior predictive check comparing this model to the data. Does the model fit the aspect of the data tested here?

School	Estimated treatment effect, y_j	Standard error of effect estimate, σ_j	Expected treatment effect
A	28	15	26
B	8	10	10
C	−3	16	−9
D	7	11	6
E	−1	9	−3
F	1	11	2
G	18	10	19
H	12	18	14

Table 5.2 from [BDA3]. *Observed effects of special preparation on SAT-V scores in eight randomized experiments. Estimates are based on separate analyses for the eight experiments.*

Recall the pooled estimate where our posterior is given by

$$\theta \mid y \sim \mathcal{N}\left(\frac{\sum_{j=1}^{8} \frac{1}{\sigma_j^2} y_j}{\sum_{j=1}^{8} \frac{1}{\sigma_j^2}} , \left(\sum_{j=1}^{8} \frac{1}{\sigma_j^2} \right)^{-1} \right)$$

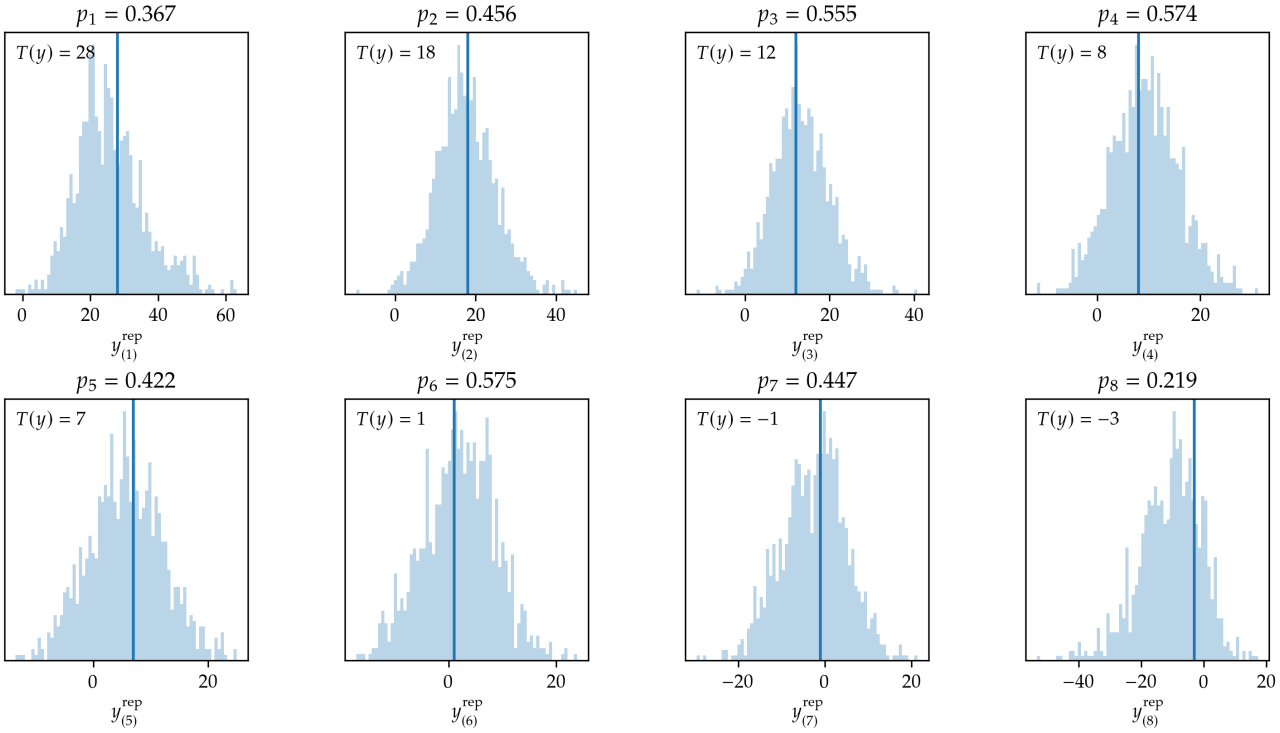
From the data, we find that $\theta \mid y \sim \mathcal{N}(7.7, 16.6)$. Our sampling procedure of the posterior predictive distribution is as follows:

(1.) Sample θ' from $\mathcal{N}(7.7, 16.6)$

(2.) Sample $y_j^{\text{rep}} \mid \theta'$ from $\mathcal{N}\left(\theta', \sigma_j^2\right)$ for $j = 1, 2, \ldots, 8$

(3.) Calculate our test statistic as $T(y^{\text{rep}}) = [y_{(1)}^{\text{rep}}, y_{(2)}^{\text{rep}}, \ldots, y_{(8)}^{\text{rep}}]$ where $y_{(j)}^{\text{rep}}$ represents the ordered statistics of the sample, i.e., $y_{(1)}^{\text{rep}} \geq y_{(2)}^{\text{rep}} \geq \cdots \geq y_{(8)}^{\text{rep}}$

We repeat this process $N = 1000$ times plot the samples below.



We find that the p-values vary between about 0.2 and 0.6 supporting that the pooled estimate fit this aspect of the data reasonably well.

```python
import numpy as np
np.random.seed(7)
import scipy.stats as stats
import pandas as pd
import matplotlib.pyplot as plt

data = pd.DataFrame({
    'school': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'],
    'y_j': [28, 8, -3, 7, -1, 1, 18, 12],
    'sigma_j': [15, 10, 16, 11, 9, 11, 10, 18],
    'mu_j': [26, 10, -9, 6, -3, 2, 19, 14]
})

def get_pooled_params(y,sigma):
    pooled_var  = 1/sum(1/(sigma**2))
    pooled_mean = sum(y/(sigma**2)) * pooled_var
    pooled_std  = np.sqrt(pooled_var)
    return pooled_mean, pooled_var, pooled_std

pooled_mean, pooled_var, pooled_std = get_pooled_params(data['y_j'], data['sigma_j'])

# Posterior predictive distribution
n_sim   = 1000
order_rep = np.empty((n_sim, 8))

for i in range(n_sim):
    theta_i = stats.norm.rvs(loc=pooled_mean, scale=pooled_std)
    yrep_i  = stats.norm.rvs(loc=theta_i, scale=data['sigma_j'])
    order_rep[i] = np.sort(yrep_i)[::-1]

fig, axes = plt.subplots(nrows=2,ncols=4,figsize=(12,6))
plt.subplots_adjust(wspace=0.4, hspace=0.4)
for i, (ax, y, yrep) in enumerate(zip(axes.flatten(), order_obs, order_rep.T), start=1):

    p_value = np.sum(yrep>=y)/n_sim
    ax.hist(yrep,70,density=True,alpha=0.3)
    ax.set_xlabel(f"$y^{{\\text{{rep}}}}_{{({i})}}$")
    ymin, ymax = ax.get_ylim()
    ax.vlines(x=y, ymin=0, ymax=ymax)
    ax.text(0.05, y=0.9, s=f'$T(y)={y}$',transform=ax.transAxes)
    ax.set_title(f'$p_{i} = {p_value:.3g}$')
    ax.set_ylim([ymin, ymax])
    ax.set_yticks([])

plt.savefig('6.1a.png', bbox_inches='tight')
plt.show()
```

Problem 5: Chapter 6, Problem 4: Model Checking and Sensivity Analysis [BDA3]

Find a published Bayesian data analysis from the statistical literature. You may use the example from Lab 6 or try to replicate the SAT Coaching example.

Dose, x_i **(g/ml)**	**Dose,** $\log(x_i)$ **(log g/ml)**	**Number of animals,** n_i	**Number of deaths,** y_i
422	−0.86	5	0
744	−0.30	5	1
948	−0.05	5	3
2069	0.73	5	5

Bioassay data from Racine et al. (1986)

(a) Compare the data to posterior predictive replications of the data.

We use the data from Lab 6 which is the same data in Chapter 12, Problem 3. It follows the logistic regression model from Chapter 3 [BDA3]. We have that

$$y_i \mid n_i, \theta_i \overset{\text{i.i.d.}}{\sim} \text{Binomial}(n_i, \theta_i) \quad \text{where} \quad \theta_i \mid x_i = \text{logit}^{-1}(\alpha + \beta x_i) = \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)} = \frac{1}{1 + \exp(-\alpha - \beta x_i)}$$

The posterior is given by

$$p(\alpha, \beta \mid y) \propto p(\alpha, \beta)\, p(y \mid \alpha, \beta) = p(\alpha, \beta) \prod_{i=1}^{k} p(y_i \mid \alpha, \beta) = \prod_{i=1}^{k} \left[\text{logit}^{-1}(\alpha + \beta x_i)\right]^{y_i} \left[1 - \text{logit}^{-1}(\alpha + \beta x_i)\right]^{n_i - y_i}$$

The log posterior is given by

$$\log\big(p(\alpha, \beta \mid y)\big) = \sum_{i=1}^{k} -y_i \log\left[1 + \exp(-\theta_i)\right] - (n_i - y_i)\,\theta_i - (n_i - y_i) \log\left[1 + \exp(-\theta_i)\right]$$

We use a Metropolis sampling strategy from Lab 6 to collect samples of the posterior as follows:

ALGORITHM 3: Metropolis Algorithm from Lab 6

Input: unnormalized target distribution $\tilde{p}_*$ and symmetric proposal distribution p_r

Initialize: Initialize empty set of samples $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**

  Draw a candidate sample from the proposal distribution: $[\alpha', \beta'] \sim p_r([\alpha', \beta'] \to [\alpha_t, \beta_t])$:

$$[\alpha', \beta'] = [\alpha_t, \beta_t] + \texttt{lr} \times \xi \quad \text{where} \quad \xi \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}7 & 0\\0 & 1\end{bmatrix}\right)$$

  Compute log acceptance ratio: $\log(r) = \log\big(\tilde{p}_*(\alpha', \beta')\big) - \log\big(\tilde{p}_*(\alpha_t, \beta_t)\big)$

  Compute acceptance probability: $a(\alpha', \beta') = \exp\big(\min(\log(r), 0)\big) = \min(r, 1)$

  Draw $U \sim \text{Uniform}([0, 1])$

  if $U \leq a(\alpha', \beta')$ **then**

    Accept candidate α', β', set $\alpha_{t+1}, \beta_{t+1} = \alpha', \beta'$

  else

    Reject candidate α', β', set $\alpha_{t+1}, \beta_{t+1} = \alpha_t, \beta_t$

  end

  if $t \geq$ `burn_in` and t mod `stride` == 0 **then**

    Append $\alpha_{t+1}, \beta_{t+1}$ to $\mathcal{S}$

  end

end

return $\mathcal{S}$

I compare the x_i to the data in the posterior inference in the next part.

(b) Perform a sensitivity analysis by computing posterior inferences under plausible alternative models.

We consider the five models used in Lab 6 and perform a sensitivity analysis on the posterior inferences of the $\log(\text{dosage})$ that results in 50% lethality. To be specific, we collect a set of n samples from the posterior using the Metropolis algorithm, $\{\alpha_j, \beta_j\}_{j=1}^{n}$ using one of the five priors. The five priors are from Lab 6:

(1) An improper uniform prior on β (see both BDA and Racine)

(2) A Gaussian prior on β formed from consultation with toxicology experts (see Racine):

$$\beta \sim \mathcal{N}(\bar{\beta}, C_\beta) \text{ where } \bar{\beta} = [-17.31, 2.57], \quad C_\beta(\sigma) = \begin{bmatrix} 1053.72 & -156.45 \\ -156.45 & 23.24 \end{bmatrix} + \sigma^2 I$$

where $\sigma > 0$ is a free parameter introduced to weaken the prior. When $\sigma = 0$, this prior is singular (places all of the prior density on a line).

(3) A Gaussian prior on β designed to introduce a stronger regularizing effect. In particular, use the covariance:

$$R_\beta(\sigma, \mu) = (C_\beta^{-1} + \mu^{-2} I)^{-1}$$

Adding a multiple of the identity inside the inverse has the effect of soft-thresholding the largest eigenvalue of the covariance from above. The smaller μ, the larger the regularizing effect.

(4) A yet weaker prior formed by replacing a Gaussian density with a Student-t density

$$\beta \sim p_\beta \text{ where } p_\beta(\beta) \propto \left(1 + \frac{1}{\nu}(\beta - \bar{\beta})^\intercal R_\beta^{-1}(\beta - \bar{\beta})\right)^{-\frac{\nu+2}{2}}$$

where ν is a shape parameter. When $\nu = 1$ the prior is a multivariate Cauchy. When $\nu \to \infty$, the prior converges to the multivariate normal prior with mean $\bar{\beta}$ and covariance R_β.

Notice that the 4th prior family encompasses the previous three. As such, it is enough to implement (4), then pick the parameters appropriately. The first is recovered by setting μ arbitrarily close to zero. The second is recovered by setting μ and ν arbitrarily large. The third is recovered by setting ν arbitrarily large.

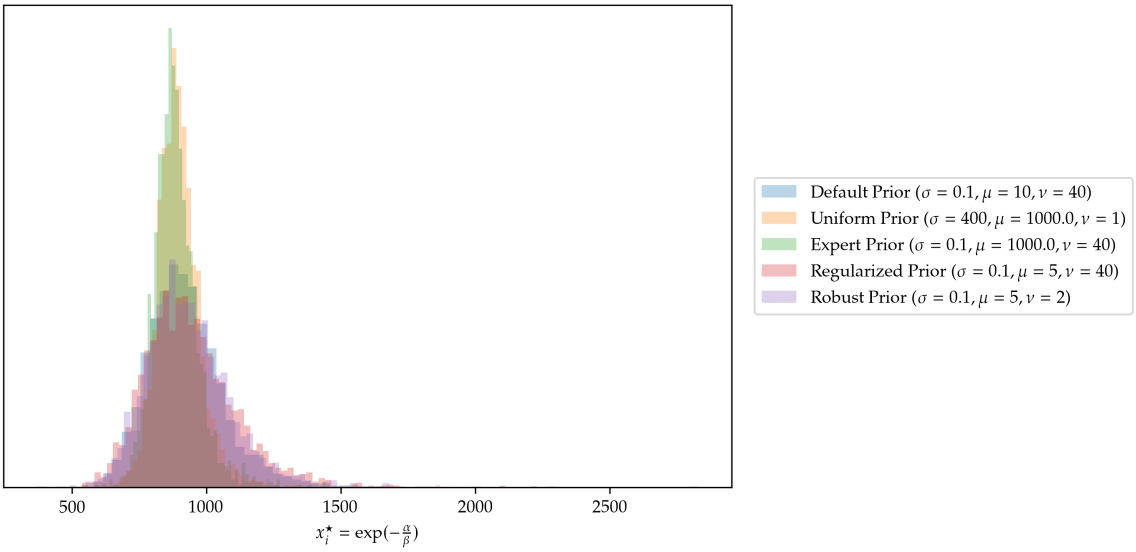
Then for each sample we calculate $\log(x_i^\star)$—that is the $\log(\text{dosage})$ that results in 50% lethality,

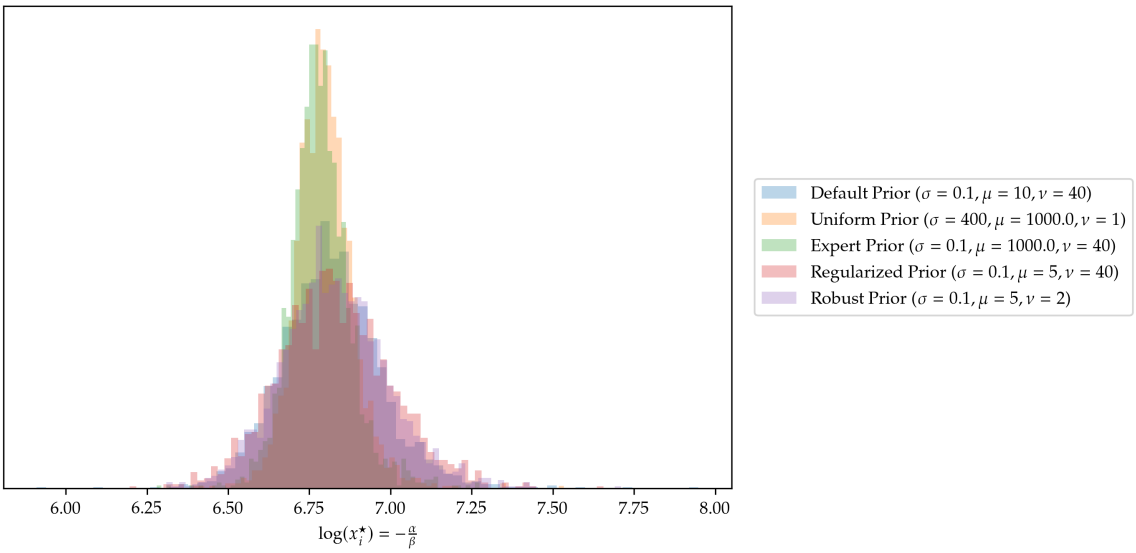
$$\frac{1}{1 + \exp(-\theta_i)} = 0.5 \implies \theta_i = 0 = \alpha + \beta \log(x_i) \implies \log(x_i^\star) = -\frac{\alpha}{\beta}$$

I summarize my posterior inferences below:

Prior	Posterior Quantiles				
	2.5%	25%	median	75%	97.5%
Default ($\sigma = 0.1$, $\mu = 10$, $\nu = 40$)	6.53	6.72	6.81	6.91	7.15
Uniform ($\sigma = 400$, $\mu = 1000$, $\nu = 1$)	6.63	6.74	6.79	6.85	6.99
Expert ($\sigma = 0.1$, $\mu = 1000$, $\nu = 40$)	6.61	6.73	6.78	6.83	7.00
Regularized ($\sigma = 0.1$, $\mu = 5$, $\nu = 40$)	6.50	6.71	6.82	6.94	7.19
Robust ($\sigma = 0.1$, $\mu = 5$, $\nu = 2$)	6.52	6.72	6.82	6.93	7.17

Prior	Posterior Quantiles				
	2.5%	25%	median	75%	97.5%
Default ($\sigma = 0.1$, $\mu = 10$, $\nu = 40$)	684.41	826.59	908.77	1002.35	1276.20
Uniform ($\sigma = 400$, $\mu = 1000$, $\nu = 1$)	757.84	845.99	891.74	940.05	1085.68
Expert ($\sigma = 0.1$, $\mu = 1000$, $\nu = 40$)	740.84	834.18	878.18	928.78	1091.17
Regularized ($\sigma = 0.1$, $\mu = 5$, $\nu = 40$)	665.27	821.61	917.32	1030.12	1330.28
Robust ($\sigma = 0.1$, $\mu = 5$, $\nu = 2$)	677.89	830.00	918.13	1022.50	1305.41





It is quite clear from these analysis that the prior does not change the posterior inference of $x_i^\star$ too much.

Problem 6: Chapter 6, Problem 7: Prior vs. Posterior Predictive Checks [BDA3]

Consider 100 observations, $y_1, \ldots, y_n$, modeled as independent samples from a $N(\theta, 1)$ distribution with a diffuse prior distribution, say, $p(\theta) = \frac{1}{2A}$ for $\theta \in [-A, A]$ with some extremely large value of A, such as 10^5. We wish to check the model using, as a test statistic, $T(y) = \max_i |y_i|$: is the maximum absolute observed value consistent with the normal model? Consider a dataset in which $\bar{y} = 5.1$ and $T(y) = 8.1$.

(a) What is the posterior predictive distribution for y^{rep}? Make a histogram for the posterior predictive distribution of $T(y^{\text{rep}})$ and give the posterior predictive p-value for the observation $T(y) = 8.1$.

We have $y_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\theta, 1)$ and $p(\theta) = \frac{1}{2A}$. Then the posterior is given by

$$p(\theta \mid y) \propto p(y \mid \theta)p(\theta) = p(\theta) \prod_{i=1}^{100} p(y_i \mid \theta) = \frac{1}{2A} \prod_{i=1}^{100} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_i - \theta}{1}\right)^2\right) \propto \exp\left(-\frac{1}{2}\sum_{i=1}^{100}(y_i - \theta)^2\right)$$

$$p(\theta \mid y) \propto \exp\left(-\frac{1}{2}\sum_{i=1}^{100} y_i^2 - 2\theta y_i + \theta^2\right) \propto \exp\left(-\frac{100}{2}\left(\bar{y}^2 - 2\theta\bar{y} + \theta^2\right)\right) = \exp\left(-\frac{1}{2}\left(\frac{\theta - \bar{y}}{1/10}\right)^2\right)$$

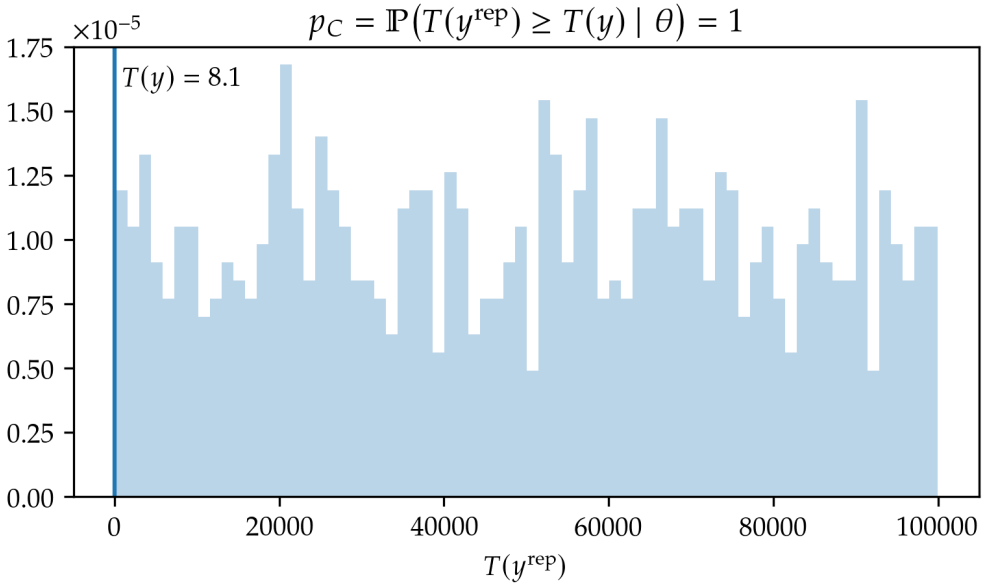
implying that the posterior is proportional to $\theta \mid y \sim \mathcal{N}(\bar{y}, 1/100)$. This implies that the posterior predictive distribution is given by

$$y^{\text{rep}} \mid y \sim \mathcal{N}\left(\mathbf{1}\bar{y}, (1/100)\mathbf{1}\mathbf{1}^\top + I\right)$$

To pull samples for the posterior predictive distribution, we...

(1.) Sample θ' from $\mathcal{N}(\bar{y}, 1/100)$,
(2.) Sample $y_1^{\text{rep}}, y_2^{\text{rep}}, \ldots, y_{100}^{\text{rep}}$ from $\mathcal{N}(\theta', 1)$
(3.) Calculate $T(y^{\text{rep}}) = \max_i \left|y_i^{\text{rep}}\right|$

We repeat this process $N = 1000$ times and plot the samples below.



The classical p-value is given by $\boxed{p_C = \mathbb{P}\left(T(y^{\text{rep}}) \geq T(y) \mid \theta\right) = 0.135.}$

(b) The prior predictive distribution is $p(y^{\text{rep}}) = \int p(y^{\text{rep}} \mid \theta)\, p(\theta)\, \mathrm{d}\theta$. (Compare to equation (6.1).) What is the prior predictive distribution for y^{rep} in this example? Roughly sketch the prior predictive distribution of $T(y^{\text{rep}})$ and give the approximate prior predictive p-value for the observation $T(y) = 8.1$.

The prior predictive distribution is given by

$$p(y^{\text{rep}}) = \int p(y^{\text{rep}} \mid \theta)\, p(\theta)\, \mathrm{d}\theta = \frac{1}{2A} \int \frac{1}{\sqrt{2\pi}^{100}} \exp\left(-\frac{1}{2}\left(y^{\text{rep}} - \mathbf{1}\theta\right)^\top \left(y^{\text{rep}} - \mathbf{1}\theta\right)\right) (\mathbb{1}\{\theta \in [-A, A]\})\, \mathrm{d}\theta$$

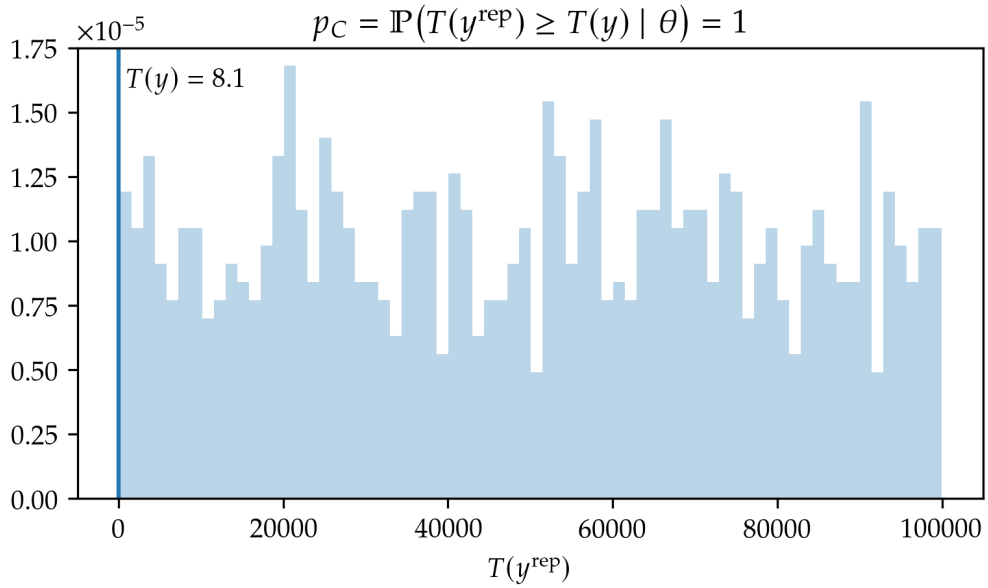
To pull samples for the prior predictive distribution, we...

(1.) Sample θ' from Uniform$(-A, A)$,

(2.) Sample $y_1^{\text{rep}}, y_2^{\text{rep}}, \ldots, y_{100}^{\text{rep}}$ from $\mathcal{N}(\theta', 1)$

(3.) Calculate $T(y^{\text{rep}}) = \max_i \left|y_i^{\text{rep}}\right|$

We repeat this process $N = 1000$ times and plot the samples below.



The classical p-value is given by $\boxed{p_C = \mathbb{P}\big(T(y^{\text{rep}}) \geq T(y) \mid \theta\big) \approx 1.}$

(c) Your answers for (a) and (b) should show that the data are consistent with the posterior predictive but not the prior predictive distribution. Does this make sense? Explain.

Since the p-value is small for the posterior predictive distribution, it means that the data is consistent with posterior predictive distribution, whereas it is not for the prior. This makes sense as the prior is essentially non-informative, so the prior should not be consistent with the data. Rather the data almost entirely describes the posterior.

```python
import numpy as np
np.random.seed(7)
import scipy.stats as stats
import matplotlib.pyplot as plt

# Posterior predictive distribution
n_sim = 1000
post_samples = np.zeros(n_sim)
for j in range(n_sim):
    theta = stats.norm.rvs(loc= 5.1 , scale=0.1, size= 1 )
    y_rep = stats.norm.rvs(loc=theta, scale= 1 , size=100)
    post_samples[j] = np.max(np.abs(y_rep))

p_value = np.sum(post_samples>=8.1)/n_sim
print(p_value)

plt.figure(figsize=(6,3))
plt.gca().set_yticks([])
plt.hist(post_samples,70,density=True,alpha=0.3)
plt.vlines(x=8.1, ymin=0, ymax=1.5)
plt.text(x=8.15, y=0.8, s=r'$T(y)=8.1$')
plt.xlabel(r"$T(y^{\text{rep}})$")
plt.title(f'$p_C = \\mathbb{{P}}\\big(T(y^{{\\text{{rep}}}}) \\geq T(y) \\mid \\theta \\big) =
↪  {p_value:.3g}$')
```

```
plt.ylim([0,1.2])
plt.savefig('y_rep.6.7.png', bbox_inches='tight')
plt.show()

# Prior predictive distribution
A = 1e5
prior_samples = np.zeros(n_sim)
for j in range(n_sim):
    theta = stats.uniform.rvs(loc=-A, scale=2*A, size= 1 )
    y_rep = stats.norm.rvs(loc=theta, scale= 1 , size=100)
    prior_samples[j] = np.max(np.abs(y_rep))

p_value = np.sum(prior_samples>=8.1)/n_sim
print(p_value)

plt.figure(figsize=(6,3))
plt.hist(prior_samples,70,density=True,alpha=0.3)
plt.vlines(x=8.1, ymin=0, ymax=1.75e-5)
plt.text(x=1000.15, y=1.6e-5, s=r'$T(y)=8.1$')
plt.xlabel(r"$T(y^{\text{rep}})$")
plt.title(f'$p_C = \\mathbb{{P}}\\big(T(y^{{\\text{{rep}}}}) \\geq T(y) \\mid \\theta \\big) =
↪  {p_value:.3g}$')
plt.ylim([0,1.75e-5])
plt.savefig('y_rep.prior.6.7.png', bbox_inches='tight')
plt.show()
```

Labs

Lab 1: The Beta, Binomial Model

Lab Date: Wednesday, January 29
Due: Wednesday, February 5

Instructions

Work with your lab group to complete the following notebook. It will be reviewed by your peers in lab next week (Wednesday, February 5th).

This notebook is only lightly scaffolded. This is intentional - as the learning goal for today's lab is as much about how to frame prior fitting and model evaluation as is it the necessary computation. As such, I have left the two major questions at the end of the lab open. If your group is stuck and unsure how to proceed, ask the instructor during lab, come into OH, or, review Chapter 6. Many of the ideas in Chapter 6 are essentially standard statistical practice, wrapped around a Bayesian pipeline, so can be discovered independently without much technical course knowledge. This lab is designed to give you enough space to discover some of these ideas if you would like.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

[1]:
```
# Basic Set Up
import numpy as np
from matplotlib import pyplot as plt
import scipy as scipy
from scipy import io, integrate, linalg, signal
import pandas as pd
# import pymc as pm
# import bambi as bmb
# import arviz as az
import statsmodels.api as sm

# Add what you like beneath here...
from scipy.stats import beta, norm
from scipy.special import betaln, binom

# My plotting settings
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['text.usetex'] = True
mpl.rcParams['text.latex.preamble'] = r'\usepackage{mathpazo}'
mpl.rcParams["font.family"] = "Palatino"
```

[2]:
```
# Load Data for the Lab
# success_counts = pd.read_csv('Beta_Binomial_Draws',header=None) # The first column is the number of trials, the second is the number of successes
# success_string = pd.read_csv('Ys',header=None) # The specific outcome string (sequence of 0, 1's) for the 100th trial
# true_probabilities = pd.read_csv('Thetas_true',header=None) # The true outcome probabilities for each row. DO NOT LOOK AT THIS UNTIL THE END. It is "unknown" and in most problems, unknowable.
success_counts = pd.read_csv('Beta_Binomial_Draws', names=['n','s']) # The first column is the number of trials, the second is the number of successes
success_string = pd.read_csv('Ys',names=['Y']) # The specific outcome string (sequence of 0, 1's) for the 100th trial
true_probabilities = pd.read_csv('Thetas_true',names=['theta']) # The true outcome probabilities for each row. DO NOT LOOK AT THIS UNTIL THE END. It is "unknown" and in most problems, unknowable.
```

The (Beta, Binomial) Model

As in class, and as in Chapter 2.4, consider the (Beta, Binomial) model. That is, we draw a success probability Θ from a Beta distribution with parameters α, β, then draw S successes from a binomial distribution on n trials, with success probability Θ:

$$\Theta \sim \text{Beta}(\alpha, \beta), \quad S \sim \text{Binomial}(n, \Theta)$$

We will assume, as usual, that n is known, S is observed, and all other variables are unknown.

It will be our goal in this lab to practice building posterior distributions, and posterior estimates to the unknown success probabilities. We learned how to do this in class for any fixed α, β. Your main goal for this lab is to learn how to reasonably estimate your prior parameters (note, I did not give them to you), and to check that your prior model leads to sensible posterior inference (model checking).

Explore the Data

The Observables

The file loaded into `success_counts` contains 100 independent draws from this model, for varying n. The first column contains the number of trials, the second contains the number of success.

The file loaded into `success_string` contains the actual sequence of outcomes for the 100th trial.

The Unknown

The file loaded into `true_probabilities` contains the true probabilities $\{\Theta_i\}_{i=1}^{1}00$ for the 100 draws from the full joint model on (Θ, S). Do not look at this until the end, or unless you are absolutely out of ideas. I have made it available here so that, after choosing your inferential methods, and evaluation scheme, you can go back and check whether we got close to the truth. You should avoid looking at this since in essentially any real Bayesian setting, this would never be known. It is common when testing pipelines to generate and save the truth, as we have done here, to validate that the pipeline is accurate had you known the truth. We'll save this evaluation for the end. For most of the lab, operate under the assumption that this file is inaccessible.

[3]:
```
# Use this cell to explore the observable data. Make whatever simple EDA plots you think you need to
get some feel for it
priors = success_counts['s']/success_counts['n']
# print(priors.sort_values().to_numpy())
```

The Inferential Pipeline

In class we discussed how to derive the posterior distribution for a model with a Beta prior and a Binomial likelihood. In particular, since the models are conjugate, the posterior is also a Beta distribution. In particular:

$$\Theta | S = s; n, \alpha, \beta \sim \text{Beta}(s + \alpha, n - s + \beta)$$

Your aim:

We want to estimate Θ_{100} from n_{100} and S_{100}. The other 99 draws from the joint model have been provided for your use either as prior data (note that α, β are the same for all 100 draws), or for model checking.

In the cell below: write a function that accepts s, n, α, β and returns the following posterior summaries: 1. The MLE estimator: $\hat{\theta}_{\text{MLE}}(s; n, \alpha, \beta)$ 1. The posterior mean: $\hat{\theta}_{\text{mean}}(s; n, \alpha, \beta)$ 1. The posterior mode: $\hat{\theta}_{\text{MAP}}(s; n, \alpha, \beta)$ 1. The posterior standard deviation: $\text{SD}[\Theta | S = s]$ 1. A basic interval estimate for $\Theta | S = s$ using posterior summaries (1. - 3.). Choose your interval estimate so that it should contain the truth with reasonably high probability. You may decide as a group what convention to use here (a Chebyshev bound, the normal approximation to the Beta, etc.).

[4]:
```
def mle(s,n,a,b):
    return s/n

def mean(s,n,a,b):
    return (s+a)/(n+a+b)

def map(s,n,a,b):
    return (s+a-1)/(n+a+b-2)

def stddev(s,n,a,b):
    return np.sqrt(
```

```
        ( mean(s,n,a,b)*(1-mean(s,n,a,b)) ) / (n+a+b+1)
    )
```

More on interval estimation

Let X be some unknown. In this section, our goal is define an interval that has a probability that the interval includes the true value, X, i.e.,

$$\mathbb{P}[X \in [\alpha_{lower}, \alpha_{upper}]] \leq \gamma$$

More formally, let γ be the coverage probability, or coverage for short, is the probability that a credible interval will include the true value (parameter) of interest.

1. Directly from PPF $F^{-1}(\cdot)$.

Let $F(x)$ be the *cummulative density function* (CDF) of a random variable X, i.e.,

$$F(x) = \mathbb{P}[X \leq x] = \int_{-\infty}^{x} f_X(x)dx$$

where $f_X(x)$ is the density of X. Additionally, let $F^{-1}(\cdot)$ be the *percent point function* (PPF) or inverse cummulative density function or quantile function. Now in this case, $F^{-1}(\cdot)$ is a function of probability, p. For example, let's say we want to know the x value that corresponds to a probability of 95%, i.e.,

$$F^{-1}(p) = F^{-1}(95\%) = x_{95\%}.$$

From PPF to credible interval: To go from Chebyshev to a credible interval, we first set our coverage (e.g., $\gamma = 0.95$), set the left probability to $\frac{1-\gamma}{2}$, and set the right probability to $1 - \frac{1-\gamma}{2}$. We have

$$\left.\begin{array}{rl} p_l = & \frac{1-\gamma}{2} \\ p_r = & 1-l \\ F^{-1}(p_l) = & x_l \\ F^{-1}(p_r) = & x_r \end{array}\right\} \implies \quad \mathbb{P}[X \in [x_l, x_r]] = \gamma$$

2. Chebyshev Inequality:

Let X (integrable) be a random variable with finite non-zero variance σ^2 (and thus finite expected value μ). Then for any real number $k > 0$,

$$\mathbb{P}(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

From Chebyshev to credible interval: To go from Chebyshev to a credible interval, we first set our coverage (e.g., $\gamma = 0.95$), upper bound our probability to $1 - \gamma$, and then solve for k. We have

$$\mathbb{P}(X \in [\mu - k\sigma, \mu + k\sigma]) \leq 1 - \gamma = \frac{1}{k^2} \quad \implies \quad \boxed{k = \frac{1}{\sqrt{1-\gamma}}}$$

In the cell below: write a function that accepts s, n, α, β and returns m i.i.d. samples from the posterior distribution:

[5]:
```
def beta_cdf(s,n,a,b, coverage=0.95):
    p_left  = (1-coverage)/2
    p_right = 1.0 - p_left
    x_left  = beta.ppf(p_left , s+a, n-s+b)
    x_right = beta.ppf(p_right, s+a, n-s+b)
    return [ x_left, x_right ]

def chebyshev(s,n,a,b, coverage=0.95):
    mean_ = mean(s,n,a,b)
    std_  = stddev(s,n,a,b)
    k     = 1 / (np.sqrt(1-coverage))
```

```
    return [ mean_-k*std_, mean_+k*std_ ]

# def normal_approx(s,n,a,b, coverage=0.95):
#     return "TODO"

interval_estimate = beta_cdf
```

In the space below: write a code that accepts

$$s, n, \alpha, \beta$$

, a chosen number of posterior samples m, and a coverage probability p. Your code should return: 1. Returns a plot of the posterior density overlaid on a histogram of the m samples 1. Print an interval estimate for $\Theta|S = s$ that contains the unknown with probability p. Your interval may be estimated from samples (as is usually done for more complex models), or, as the Beta CDF is easy to work out analytically, from exact quantiles.

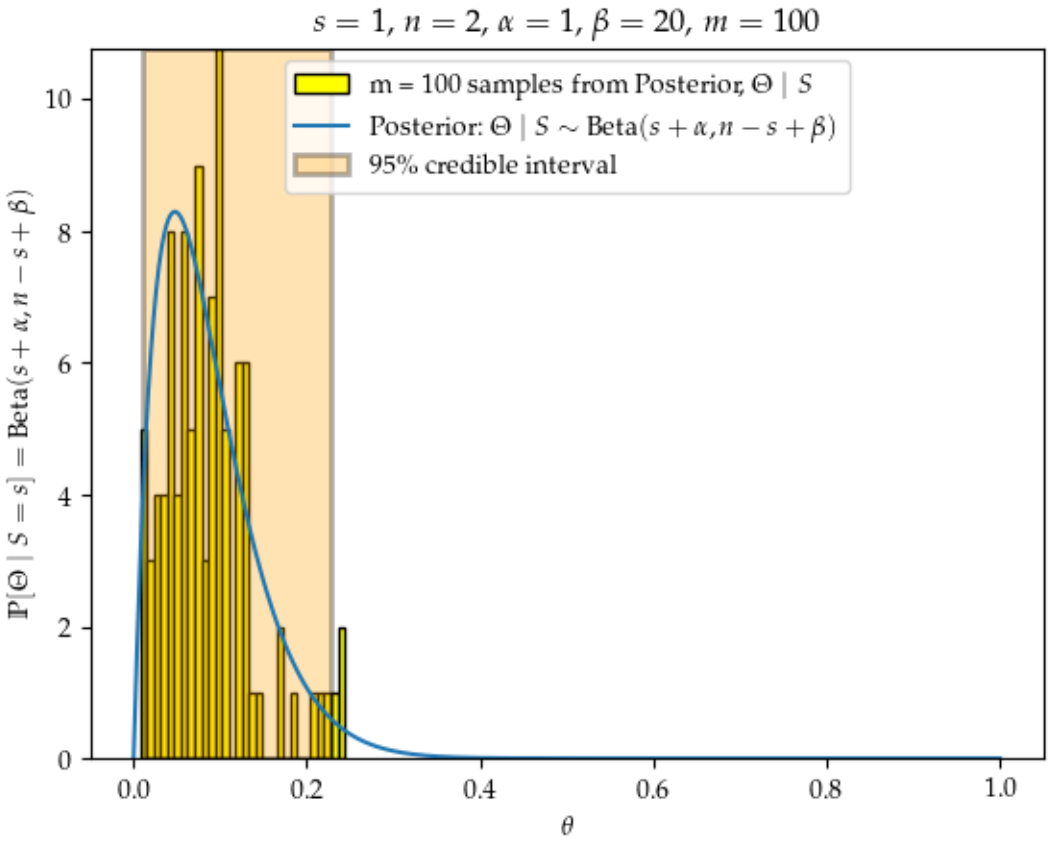
[ ]:
```
def sample_posterior(s,n,a,b,m=100):
    return beta.rvs(s+a,n-s+b,size=m)

def plot_posterior_and_samples(s,n,a,b,m=100,coverage=0.95):

    samples   = sample_posterior(s,n,a,b,m)
    posterior = beta(s+a,n-s+b)
    thetas    = np.linspace(0,1,1000)
    post_pdf  = posterior.pdf(thetas)

    plt.figure()
    plt.hist(samples, bins=30, color='yellow', edgecolor='black', label=f'm = {m} samples from
Posterior, $\\Theta\\mid S$')
    plt.plot(thetas, post_pdf, label=r'Posterior: $\Theta\mid S \sim \text{Beta}(s+\alpha,
n-s+\beta)$')
    interval_ = interval_estimate(s,n,a,b,coverage=coverage)
    plt.axvspan(interval_[0], interval_[1], alpha=0.3, facecolor='orange', edgecolor='black',
linewidth=2, label=f'${int(coverage*100)}\%$ credible interval')
    plt.xlabel(r'$\theta$')
    plt.ylabel(r'$\mathbb{P}[\Theta\mid S=s] =\text{Beta}(s+\alpha, n-s+\beta)$')
    plt.ylim([0, np.max(post_pdf) + 0.3*np.max(post_pdf)])
    plt.title(f'$s = {s}$, $n = {n}$, $\\alpha = {a}$, $\\beta = {b}$, $m = {m}$')
    plt.legend(loc='upper center')
    plt.show()

plot_posterior_and_samples(s=1,n=2,a=1,b=20)
```



In the space below: Apply the codes you wrote above to explore the range of different posterior distributions for $\Theta_{100}|S_{100} = s_{100}$ that can be produced by varying the prior parameters α and β. I am not going to give you specific prior parameter pairs to check here. Instead, be responsible for your own exploration. Stop once you think you have sufficiently demonstrated the possible posteriors, and their dependence on the possible prior assumptions.

Note: For this sort of demonstration, it helps to do a little back of the envelope thinking first about how the prior parameters bias the posterior, and the balance of information provided by the prior and by the data collected.

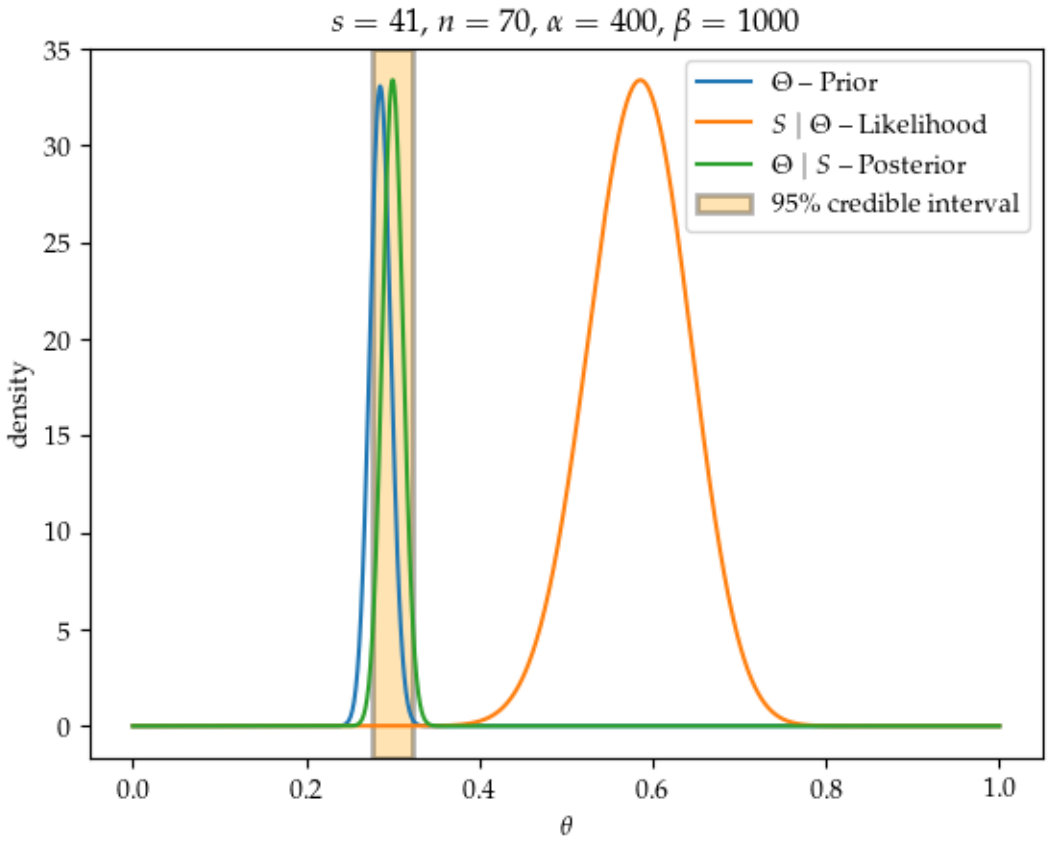
[7]:
```
def plot_dists(s,n,a,b, coverage=0.95, scale_likelihood=True):
    prior      = beta(a,b)
    posterior  = beta(s+a,n-s+b)
    def likelihood(theta,s,n):
        from scipy.special import binom
        return binom(n,s)*(theta**s)*((1-theta)**(n-s))

    thetas    = np.linspace(0,1,1000)
    prior_pdf = prior.pdf(thetas)
    post_pdf  = posterior.pdf(thetas)
    lik_pdf   = likelihood(thetas,s,n)

    if scale_likelihood:
        scale    = max(np.max(prior_pdf), np.max(post_pdf)) / np.max(lik_pdf)
        lik_pdf *= scale
    plt.figure()
    plt.plot(thetas, prior_pdf, label=r'$\Theta$ -- Prior')
    plt.plot(thetas, lik_pdf, label=r'$S\mid\Theta$ -- Likelihood')
    plt.plot(thetas, post_pdf, label=r'$\Theta\mid S$ -- Posterior')
    interval_ = interval_estimate(s,n,a,b,coverage=coverage)
    plt.axvspan(interval_[0], interval_[1], alpha=0.3, facecolor='orange', edgecolor='black',
linewidth=2, label=f'${int(coverage*100)}\%$ credible interval')
```

```
    plt.xlabel(r'$\theta$')
    plt.ylabel(r'density')
    plt.title(f'$s = {s}$, $n = {n}$, $\\alpha = {a}$, $\\beta = {b}$')
    plt.legend()
    plt.show()

s_100 = success_counts['s'].iloc[-1]
n_100 = success_counts['n'].iloc[-1]
a_est = 400
b_est = 1000
plot_dists(s_100, n_100, a_est, b_est)
```



Gathering Information

In Bayesian statistics, observations provide information via conditioning. Let's see how our posterior changes sequentially as we gain evidence.

In the cell below: Write a code that walks through the `success_string` corresponding to the 100th trial. As it goes, use the code you wrote above to print out a sequence of plots and interval estimates for the unknown. For now, initialize the process with a uniform prior ($\alpha = 1, \beta = 1$). What do you notice about the posterior (its shape, its uncertainty with respect to resampling, etc) as we gather evidence? What do you notice about its point summaries (the MLE, MAP, and posterior mode)?

Note: you don't need a unique output for all 70 trials in the success string. Instead, space your outputs to show the main trends. You should space your outputs tighter at the beginning, when our posterior is more sensitive to any individual trial outcome, and less tightly at the end. As a basic heuristic, the spacing should grow quadratically.

If you feel you already have a decent intuition for the Beta distribution, I would save this step for the end of the lab. If you'd like to try a basic widget to see how the shape of the Beta depends on its parameters, go to this article and scroll to Demo 3 (roughly 3/5ths of the way down).

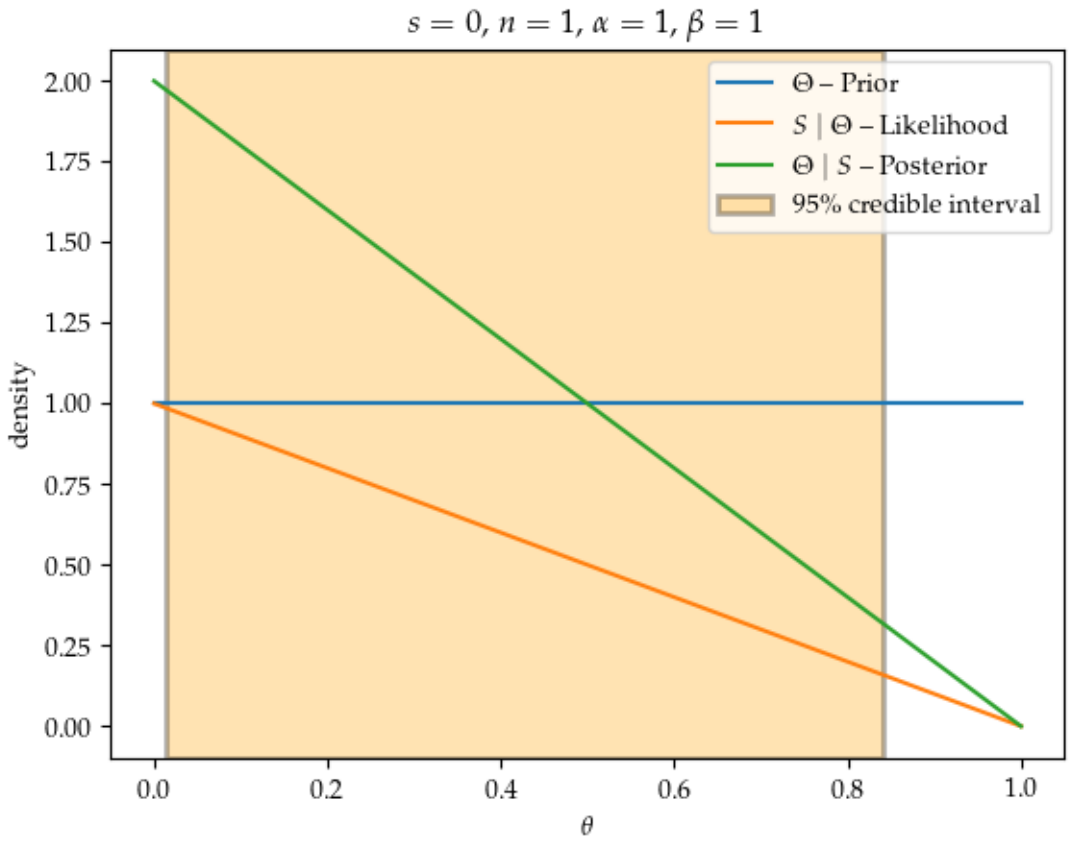
[8]:
```
n_spaces = len(success_string)
plot_when = [i**2 for i in range(1,n_spaces+1) if i**2<n_spaces] + [n_spaces]

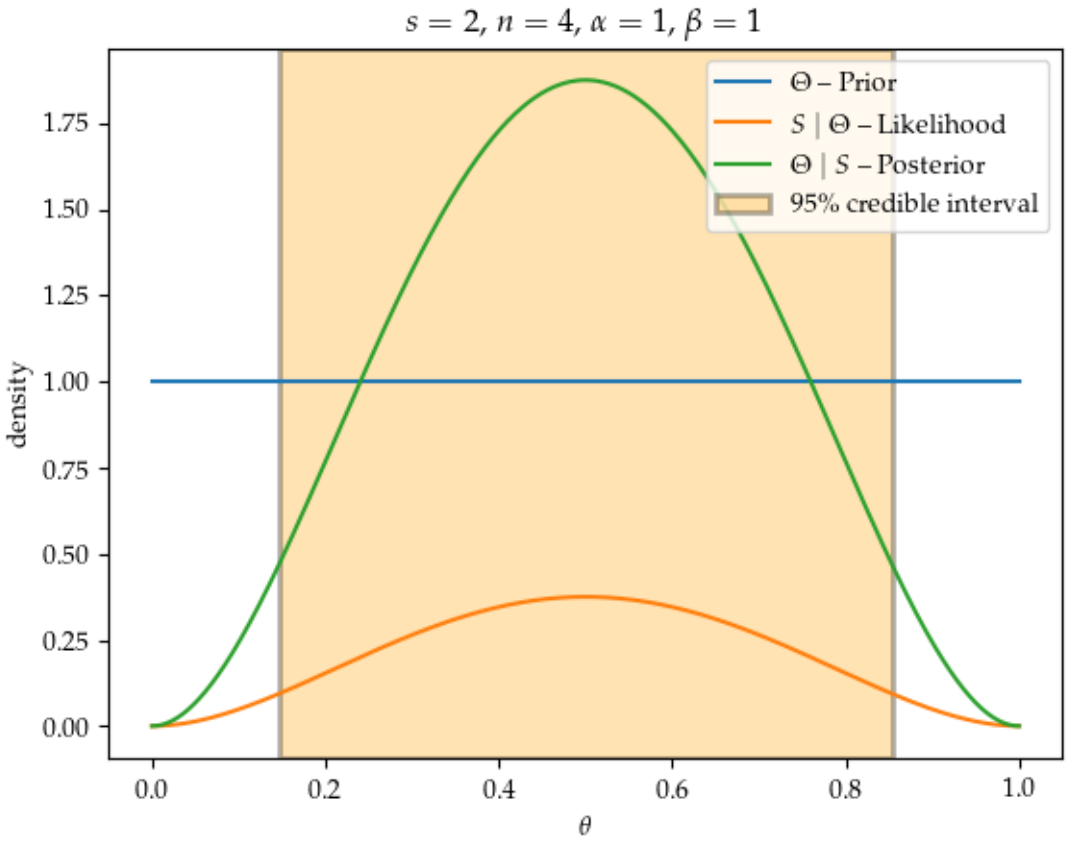
s = 0
n = 0
for i, row in success_string.iterrows():

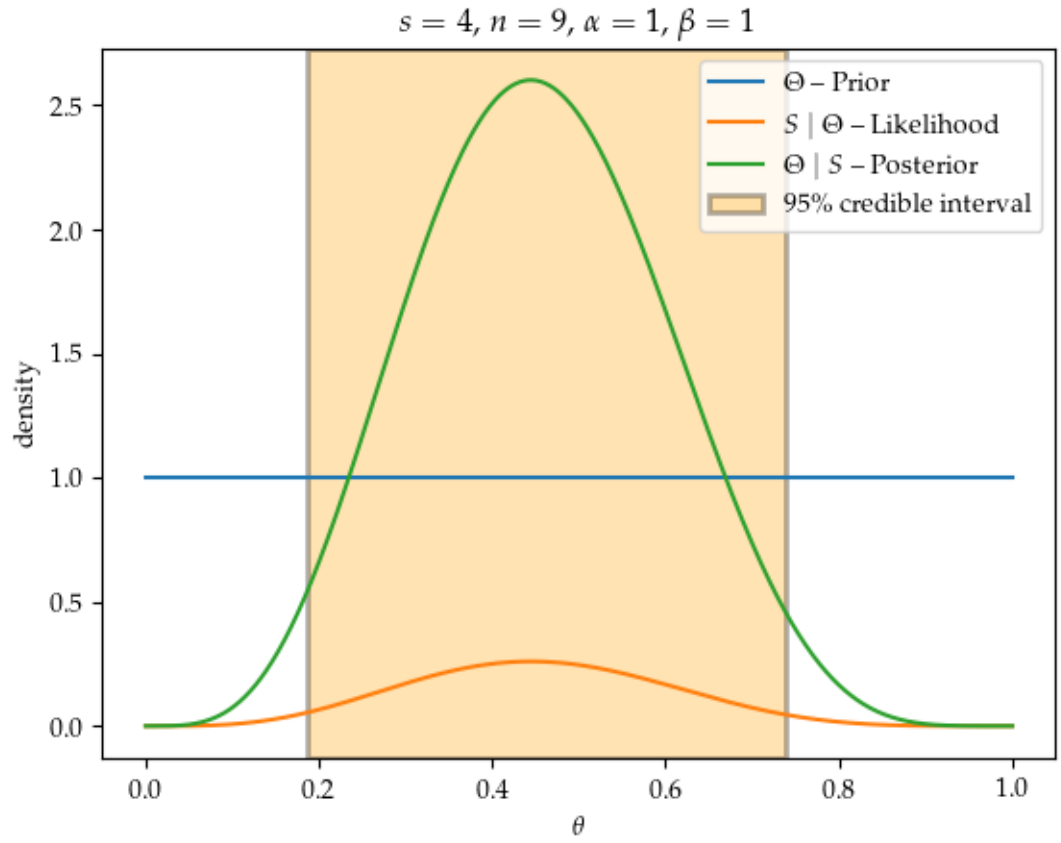
    # Add new data
    s += row['Y']
    n += 1

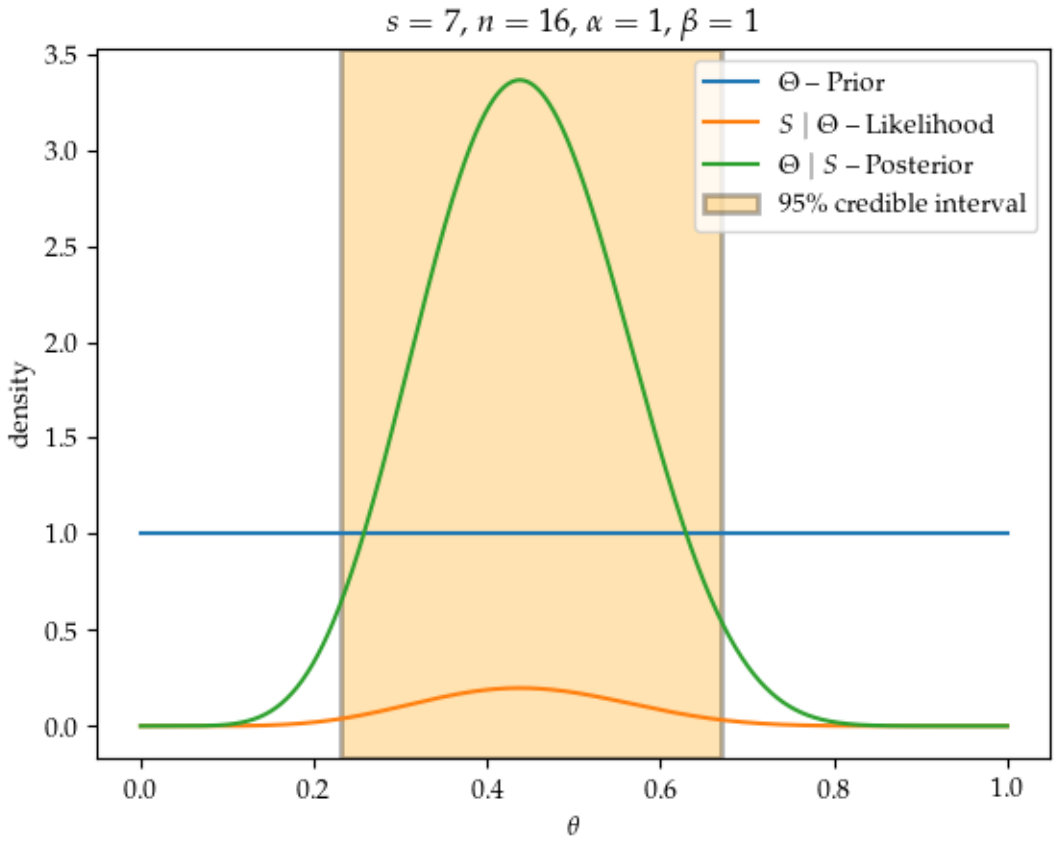
    # alpha and beta are just the uniform distribution
    alpha_ = 1
    beta_  = 1

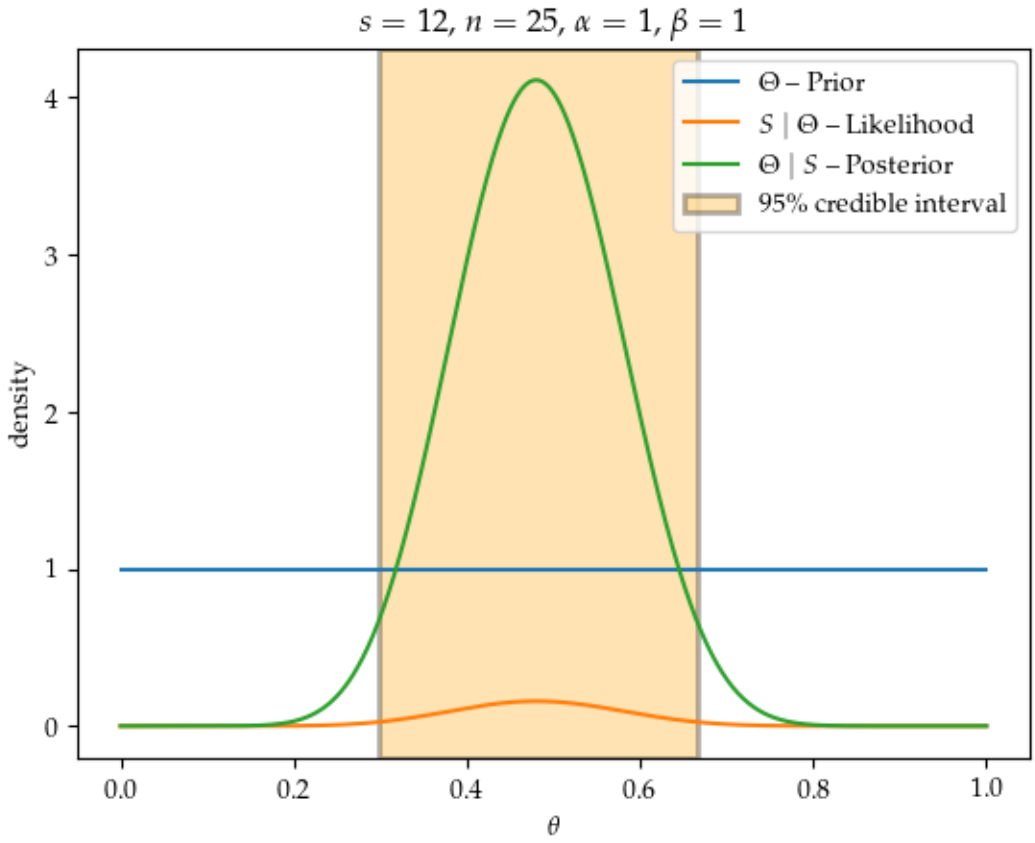
    # Plot the distributions
    if i+1 in plot_when:
        plot_dists(s,n,alpha_,beta_, scale_likelihood=False)
```

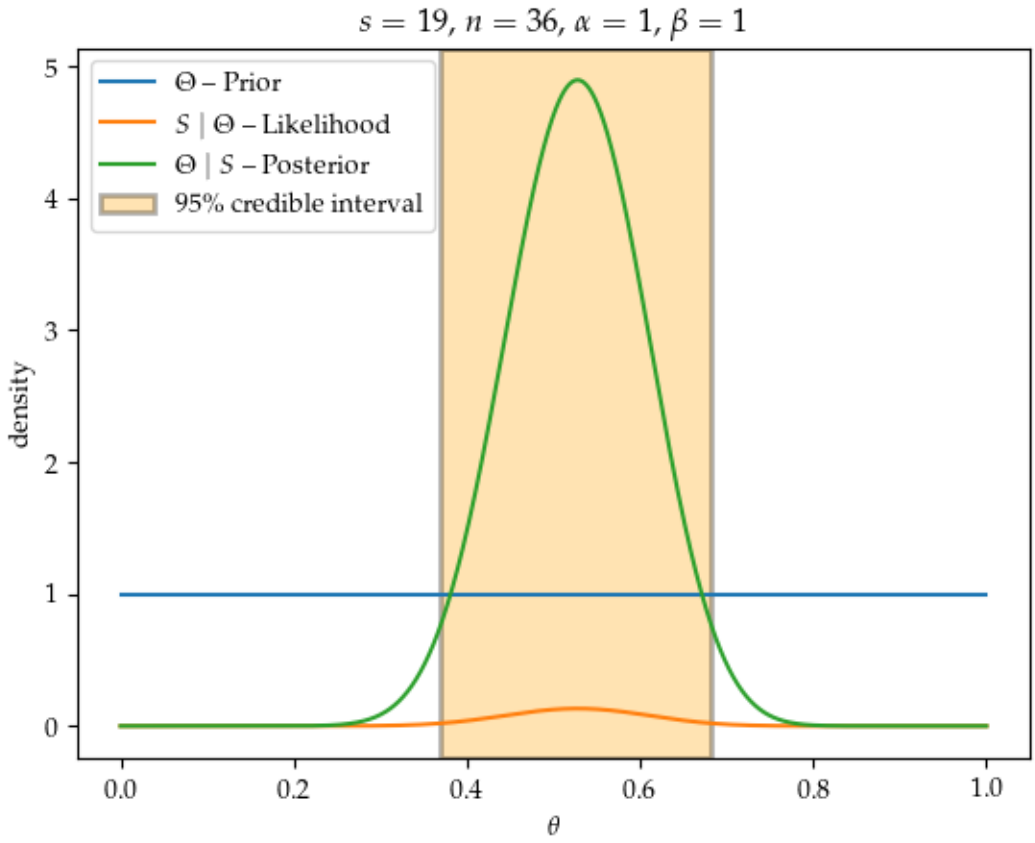


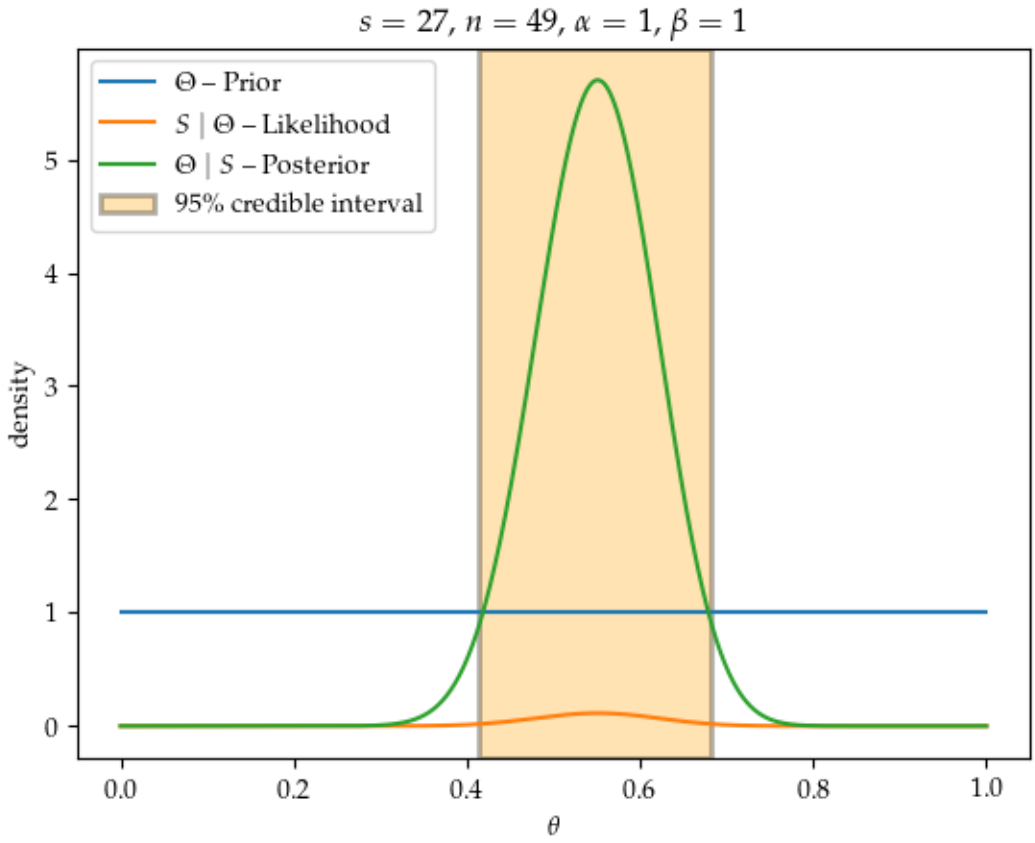

$s = 2, n = 4, \alpha = 1, \beta = 1$
Θ – Prior
S | Θ – Likelihood
Θ | S – Posterior
95% credible interval
density
θ
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.2
0.4
0.6
0.8
1.0

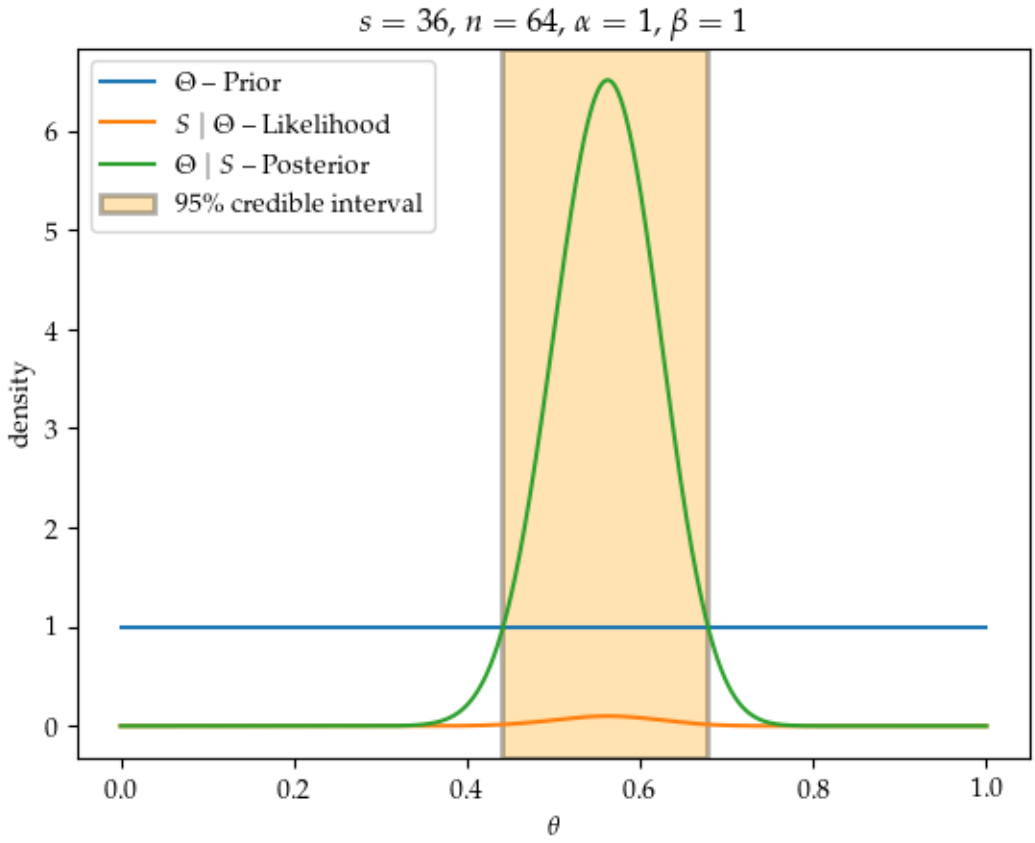

$s = 4, n = 9, \alpha = 1, \beta = 1$
Θ – Prior
S | Θ – Likelihood
Θ | S – Posterior
95% credible interval
density
θ
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

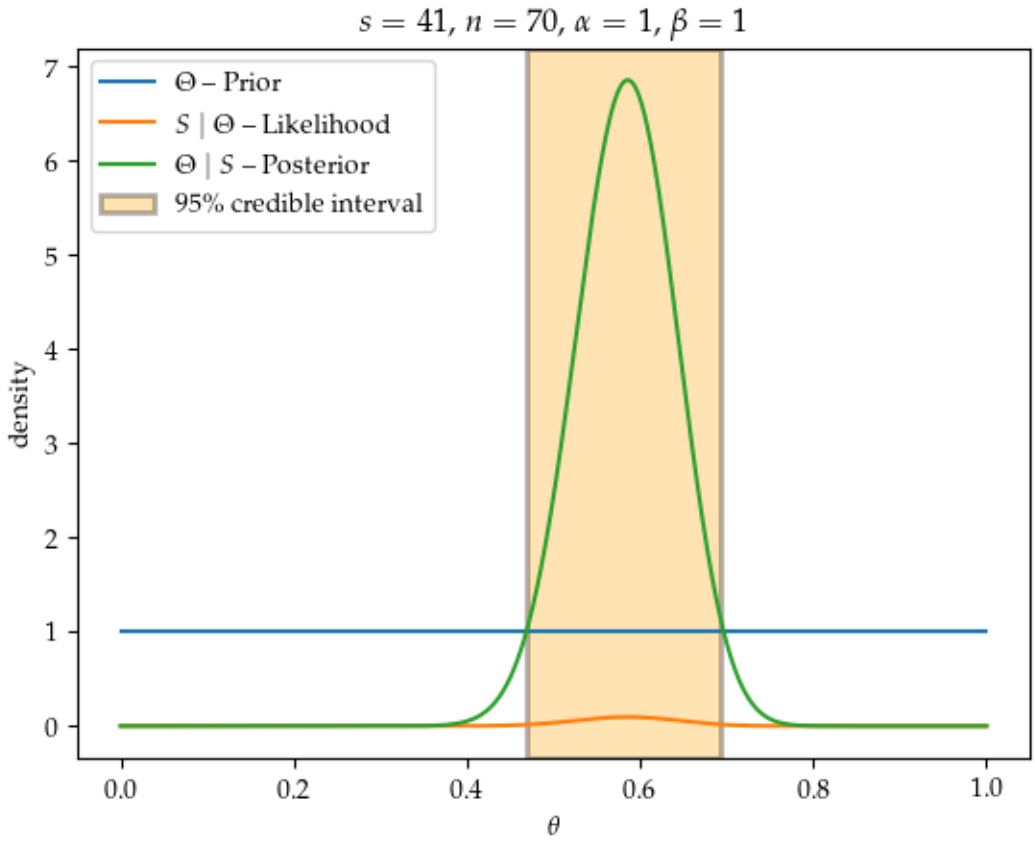

$s = 7, n = 16, \alpha = 1, \beta = 1$
Θ – Prior
$S \mid \Theta$ – Likelihood
$\Theta \mid S$ – Posterior
95% credible interval
density
θ
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0
0.2
0.4
0.6
0.8
1.0


$s = 12, n = 25, \alpha = 1, \beta = 1$
Θ – Prior
$S \mid \Theta$ – Likelihood
$\Theta \mid S$ – Posterior
95% credible interval
density
θ
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4


$s = 19, n = 36, \alpha = 1, \beta = 1$
Θ – Prior
$S \mid \Theta$ – Likelihood
$\Theta \mid S$ – Posterior
95% credible interval
density
θ
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5


$s = 27, n = 49, \alpha = 1, \beta = 1$
Θ – Prior
$S \mid \Theta$ – Likelihood
$\Theta \mid S$ – Posterior
95% credible interval
density
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
θ


$s = 36, n = 64, \alpha = 1, \beta = 1$
Θ – Prior
$S \mid \Theta$ – Likelihood
$\Theta \mid S$ – Posterior
95% credible interval
density
θ
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6


$s = 41, n = 70, \alpha = 1, \beta = 1$
Θ – Prior
$S \mid \Theta$ – Likelihood
$\Theta \mid S$ – Posterior
95% credible interval
density
θ
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7

Fitting the Prior

You should have seen that, your estimates above depend on your choice of prior parameters. This is the point of Bayesian inference. The prior *should* influence our estimation.

In order to influence our estimates informatively, the prior must encode actual information. Otherwise, the Bayesian pipeline is really just a different estimation procedure that uses the mathematics of conditioning to define intervals and to regularize the inference.

Discuss with your group how you could use the first 99 sample outcomes to estimate α and β.

For now, we will assume that the form of the prior model (i.e. the Beta) is correct. We will discuss what happens if your prior form is misspecified (does not contain the true data generating distribution for any parameter values) later in the class. I generated this data using the model specified above.

If you get stuck, consider the following pair of ideas: 1. You could, select a set of previous observations with large n_j, estimate θ_j for each with a simple point estimator, then fit the resulting ensemble of estimated θ's to a Beta distribution (either by matching moments or via an MLE). *If you choose this route, discuss the trade-off between using many past samples, and incorporating less reliable past samples* 1. Consider the marginal distribution of S drawn from the joint model (Θ, S). When evaluated at $S = s$ we called this the "evidence". Discuss the relationship between the marginal distribution of $\{S_j\}_{j=1}^{99}$ and the prior parameters. Does this relation have a familiar name? Could you use it to estimate α and β? If so, what standard framework could you adopt?

How you proceed from here is up to you. Note: if you choose 1., you will have to explain how you fit (i.e. how to solve the moment matching problem). If you choose 2, I strongly suggest you look ahead to the first part of the last section of the lab.

Estimating α and β

Below we outline the two approaches for estimating α and β from observed data. Specifically, consider a collection of 99 trials $\{S_j, \Theta_j^{(true)}\}_{j=1}^{99}$ from the joint model (Θ, S). For our analysis, we restrict ourselves to only having access to the observations of S, i.e., our dataset consist of $\{S_j\}_{j=1}^{99} \sim S \mid \Theta = \theta$.

Approach 1: Prior maximization (via estimating θ)

In one approach, we can use our observed data, S_j, to *estimate* $\hat{\theta}_j \approx \theta_j^{(true)}$ for each trial. For example, we could use an MLE estimate, $\hat{\theta}_j = s_j/n_j$. Then we aim to solve for the α and β that are most likely given these estimates of $\hat{\theta}_j$. We have

$$\mathcal{L}_I(\alpha,\beta) := \prod_{i=1}^{99} \mathbb{P}[\Theta = \hat{\theta}_j] \quad \Longrightarrow \quad \alpha^\star, \beta^\star = \underset{\alpha,\beta}{\operatorname{argmax}} \Big\{ \mathcal{L}_I(\alpha,\beta) \Big\} = \underset{\alpha,\beta}{\operatorname{argmax}} \Big\{ \log\big(\mathcal{L}_I(\alpha,\beta)\big) \Big\} = \underset{\alpha,\beta}{\operatorname{argmin}} \Big\{ -\log\big(\mathcal{L}_I(\alpha,\beta)\big) \Big\}$$

Note, that taking the $\log(\cdot)$ is done to make the maximization easier numerically.

For our model, our prior $\Theta \sim \text{Beta}(\alpha, \beta)$, so we have

$$\alpha^\star, \beta^\star = \underset{\alpha,\beta}{\operatorname{argmin}} \Big\{ -\sum_{i=1}^{99} \log\left(\frac{\hat{\theta}_i^{\alpha-1}(1-\hat{\theta}_i)^{\beta-1}}{\text{Beta}(\alpha,\beta)} \right) \Big\}$$

Approach 2: Evidence maximization (aka Empirical Bayes aka type II MLE)

In another approach, we can use our our observed data, S_j, directly. In this case, we maximize the evidence, i.e.,

$$\mathcal{L}_{II}(\alpha,\beta) := \prod_{i=1}^{99} \mathbb{P}[S = s] \quad \Longrightarrow \quad \alpha^\star, \beta^\star = \underset{\alpha,\beta}{\operatorname{argmin}} \Big\{ -\log\big(\mathcal{L}_{II}(\alpha,\beta)\big) \Big\}$$

For our model, our evidence follows $\Pr(S = s) = \binom{n}{s} \frac{\text{B}(s+\alpha, n-s+\beta)}{\text{B}(\alpha,\beta)}$, so we have

$$\alpha^\star, \beta^\star = \underset{\alpha,\beta}{\operatorname{argmin}} \Big\{ -\sum_{i=1}^{99} \log\left(\binom{n_i}{s_i} \frac{\text{B}(s_i+\alpha, n_i-s_i+\beta)}{\text{B}(\alpha,\beta)} \right) \Big\}$$

Differences in the approaches:

The biggest difference in the approach are that Approach 1 relies on an esimate of Θ which has an associated uncertainty/error whereas Approach 2 does not.

[ ]:
```
def plot_prior_and_estimates(a,b,theta_est):
    prior_est = beta(a,b)
    thetas    = np.linspace(0,1,1000)
    prior_pdf = prior_est.pdf(thetas)
    plt.figure()
    plt.hist(theta_est, bins=30, color='yellow', edgecolor='black', label=f'Top {len(theta_est)}
$\\theta_{{est}} = s_i/n_i$')
    plt.plot(thetas, prior_pdf, label=r'Prior: $\Theta \sim \text{Beta}(\alpha_{est}, \beta_{est})$')
    plt.xlabel(r'$\theta$')
    plt.ylabel(r'$\mathbb{P}[\Theta] =\text{Beta}(\alpha_{est}, \beta_{est})$')
    plt.legend(loc='upper left')
    plt.show()

# Keeping the thetas with the most n
keep_top_k_thetas  = 50
all_data_but_last  = success_counts.iloc[:-1]
top_success_counts = all_data_but_last.sort_values(by='n', ascending=False).head(keep_top_k_thetas)
theta_estimates    = (top_success_counts['s'] / top_success_counts['n']).to_numpy()

# Matching the moments (see `Method of moments' on the right: https://en.wikipedia.org/wiki/
 ↪Beta_distribution)
mean_est  = np.mean(theta_estimates)
var_est   = np.var(theta_estimates)
coeff     =  (((mean_est * (1 - mean_est)) / var_est) - 1)
alpha_mom = coeff * mean_est
beta_mom  = coeff * (1 - mean_est)
print('Method of moments: Estimated prior parameters')
print(f"alpha = {alpha_mom:0.3f}, beta = {beta_mom:0.3f}")
plot_prior_and_estimates(alpha_mom,beta_mom,theta_estimates)

# Approach 1: Prior Maximization
def prior_maximization(theta_samples, initial_alpha, initial_beta):
    def L1(params):
        a, b = params
        if a <= 0 or b <= 0:
            return np.inf
        return -np.sum(beta.logpdf(theta_samples, a, b))
    result = scipy.optimize.minimize(L1, [initial_alpha, initial_beta], method='L-BFGS-B')
    return result.x if result.success else (None, None)

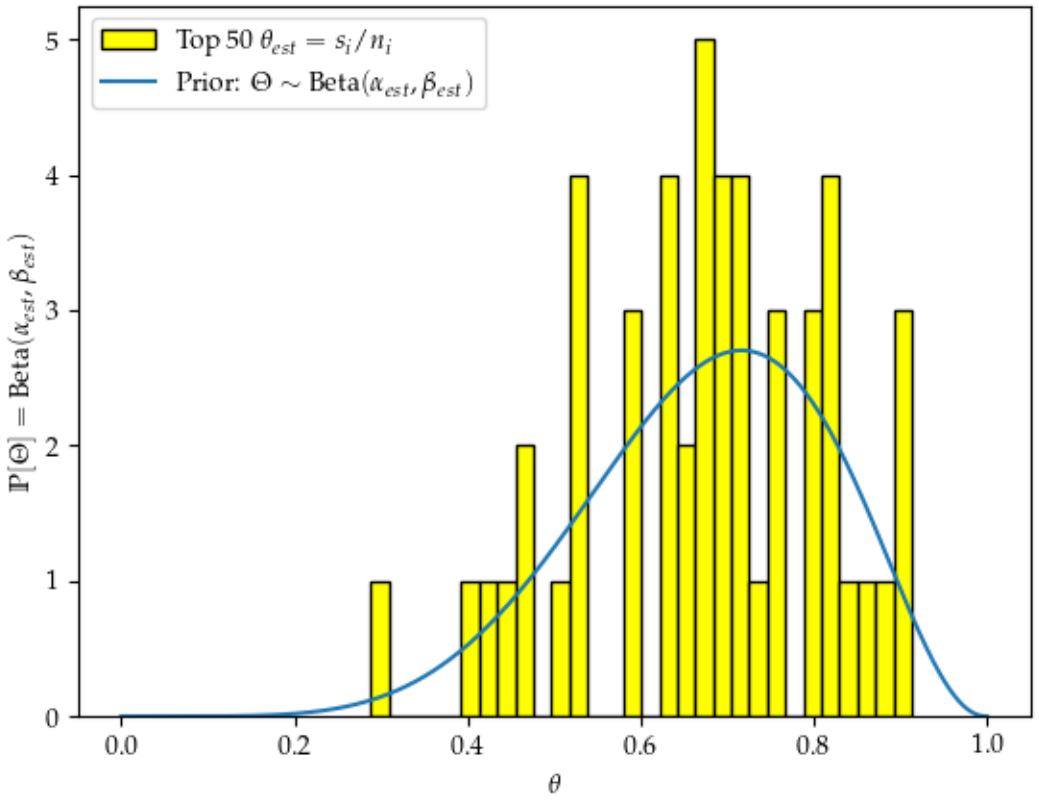
alpha_L1, beta_L1 = prior_maximization(theta_estimates, alpha_mom, beta_mom)
print('Prior maximization')
print(f"alpha = {alpha_L1:0.3f}, beta = {beta_L1:0.3f}")
plot_prior_and_estimates(alpha_L1,beta_L1,theta_estimates)

# Approach 2: Evidence Maximization
def evidence_maximization(s_samples, n_samples, initial_alpha, initial_beta):
    def L2(params):
        a, b = params
        if a <= 0 or b <= 0:
            return np.inf
        log_terms = np.log(binom(n_samples, s_samples)) \
                    + betaln(s_samples + a, n_samples - s_samples + b) \
                    - betaln(a, b)
        return -np.sum(log_terms)
```

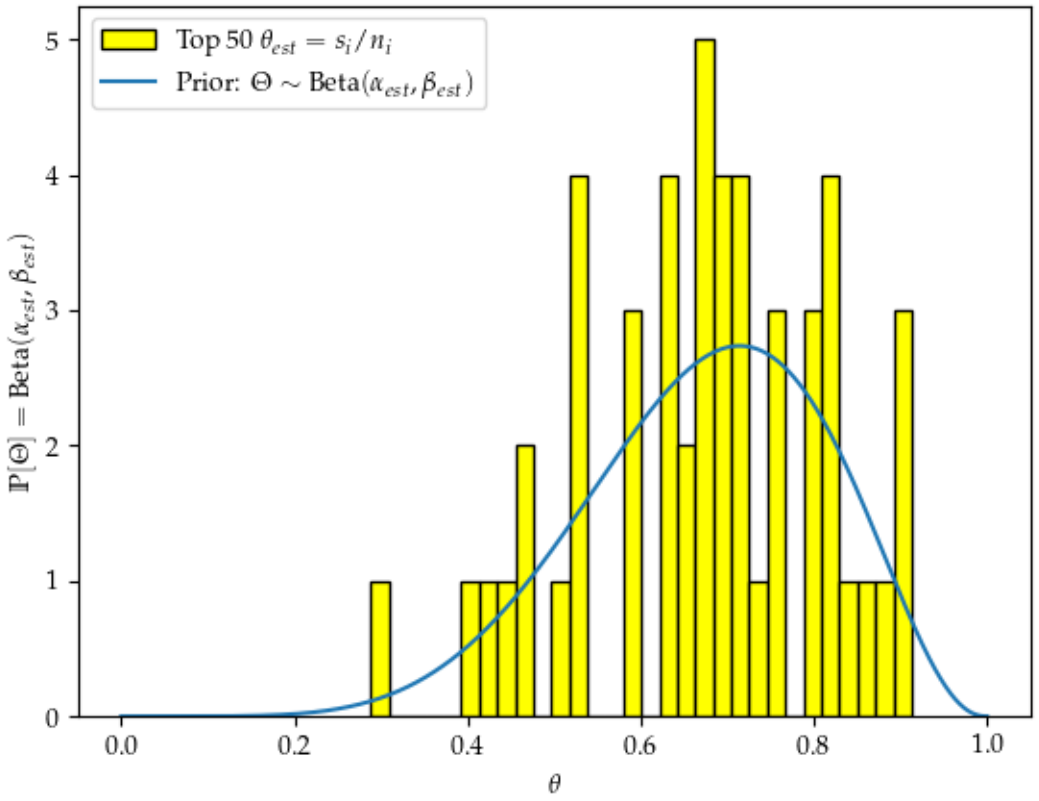
```
    result = scipy.optimize.minimize(L2, [initial_alpha, initial_beta], method='L-BFGS-B')
    return result.x if result.success else (None, None)

s_samples = all_data_but_last['s'].to_numpy()
n_samples = all_data_but_last['n'].to_numpy()
alpha_L2, beta_L2 = evidence_maximization(s_samples, n_samples, alpha_mom, beta_mom)
print('Evidence maximization')
print(f"alpha = {alpha_L2:0.3f}, beta = {beta_L2:0.3f}")
plot_prior_and_estimates(alpha_L2,beta_L2,(all_data_but_last['s'] / all_data_but_last['n']).
 ↪to_numpy())
```

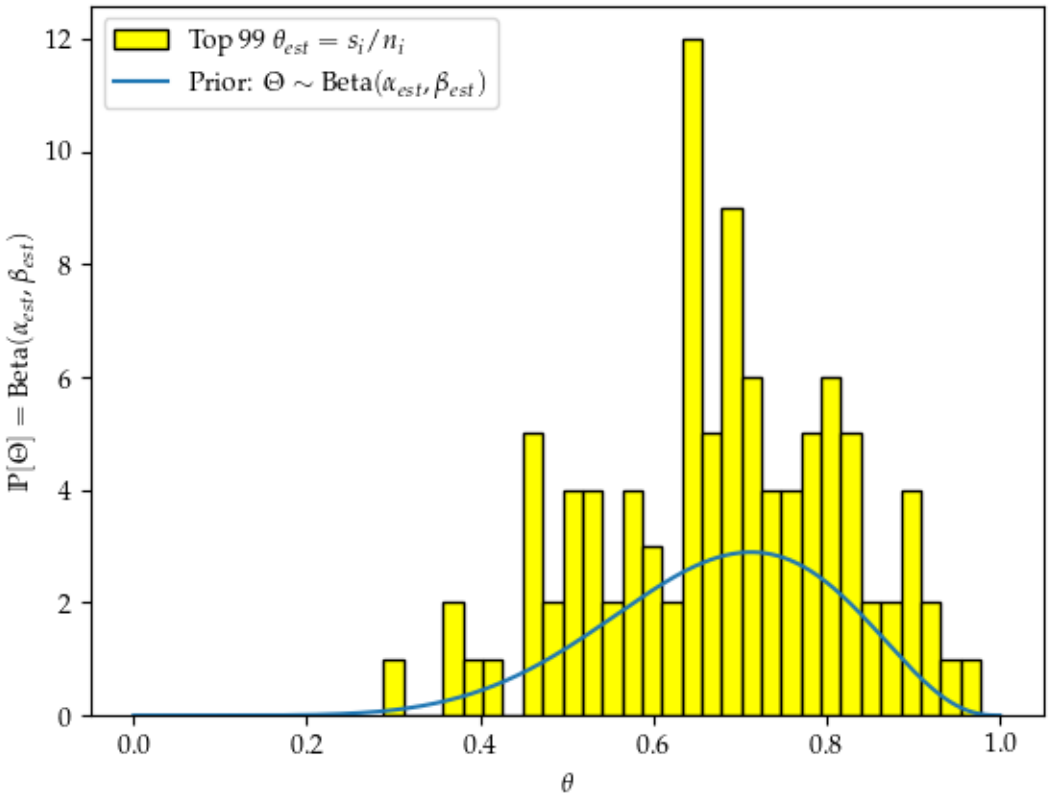
```
Method of moments: Estimated prior parameters
alpha = 6.719, beta = 3.265
```




```
Prior maximization
alpha = 6.898, beta = 3.361
```




```
Evidence maximization
alpha = 7.734, beta = 3.714
```



Comment on your degree of confidence in your prior parameter fits. Think about simple procedures for evaluating your uncertainty in the prior parameter fits with your group. Based on your experience toying with the posterior for different prior choices, do you believe you've resolved the parameters with enough confidence so that your posterior is reliable (i.e. the uncertainty captured in the posterior fully captures your uncertainty about the unknown)?

Replace this text with your discussion. You are welcome to attempt the procedure you sketch out below.

TODO

[ ]:

Model Checking

The Evidence

Derive the marginal distribution for the number of observed successes S given (Θ, S) drawn with parameters α, β, n. That is, derive an explicit formula for $\Pr(S = s)$ for all $s \in \{1, 2, ..., n\}$. You may express your answer using any standard combinatorial functions (factorial, choose, Γ, Beta, etc). Do not try to close the integrals directly. Instead, reference the normalization factor used in the Beta distribution's density function.

We have

$$\Theta \sim \text{Beta}(\alpha, \beta) \quad \text{and} \quad S \mid \Theta = \theta \sim \text{Binomial}(n, \theta)$$

where the densities are given by

$$f_\Theta(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\text{B}(\alpha, \beta)} \quad \text{for all } \theta \in [0, 1]$$

$$f_{S|\Theta=\theta}(s) = \binom{n}{s}\theta^s(1-\theta)^{n-s} \quad \text{for all } s \in \{0, 1, 2, \ldots, n\}$$

Here, $\text{B}(\alpha, \beta)$ represents the Beta function which can be written as a product of Gamma functions, i.e.,

$$\text{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \quad \text{where} \quad \Gamma(z) = \begin{cases} (z-1)! & \text{when } z \text{ is a positive integer} \\ \int_0^\infty t^{z-1}e^{-t}\,\text{d}t & \text{otherwise} \end{cases}$$

The evidence is calculated as

$$\Pr(S = s) = \int_{\theta=0}^{\theta=1} f_{S|\Theta=\theta}(s) f_\Theta(\theta) d\theta$$

$$\Pr(S = s) = \int_{\theta=0}^{\theta=1} \binom{n}{s}\theta^s(1-\theta)^{n-s} \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\text{B}(\alpha, \beta)} d\theta$$

$$\Pr(S = s) = \frac{\binom{n}{s}}{\text{B}(\alpha, \beta)} \int_{\theta=0}^{\theta=1} \theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1} d\theta$$

$$\Pr(S = s) = \binom{n}{s}\frac{\text{B}(s+\alpha, n-s+\beta)}{\text{B}(\alpha, \beta)} \underbrace{\int_{\theta=0}^{\theta=1} \frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}}{\text{B}(s+\alpha, n-s+\beta)} d\theta}_{\text{CDF of Beta}(s+\alpha, n-s+\beta) \text{ over the support}}$$

$$\implies \boxed{\Pr(S = s) = \binom{n}{s}\frac{\text{B}(s+\alpha, n-s+\beta)}{\text{B}(\alpha, \beta)}}$$

Give an argument justifying the word "evidence" to describe $\Pr(S = s)$. Relate your answer to the likelihood function of the *prior* parameters α, β.

Replace this text with your discussion

TODO

Posterior Predictive Distribution

Use your work above, and the rule that updates the prior (produces the posterior), given an observation, to derive the posterior predictive distribution for a new set of m sample outcomes, given an observed set of n outcomes containing s successes.

Let S_{new} represent the set of m sample outcomes.

$$\mathbb{P}\left(S_{new} = s_{new}\right) = \int_0^1 \mathbb{P}[S_{new} = s_{new}, \Theta = \theta] d\theta = \int_0^1 \mathbb{P}[S_{new} = s_{new} \mid \Theta = \theta]\, \mathbb{P}[\Theta = \theta] d\theta$$

$$\mathbb{P}\left(S_{new} = s_{new}\right) = \int_0^1 \binom{m}{s_{new}} \theta^{s_{new}} (1-\theta)^{m-s_{new}} \frac{\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta}}{\mathrm{B}(s+\alpha, n-s+\beta)} d\theta$$

$$\mathbb{P}\left(S_{new} = s_{new}\right) = \binom{m}{s_{new}} \frac{\mathrm{B}(s+\alpha+s_{new}, n-s+\beta+m-s_{new})}{\mathrm{B}(s+\alpha, n-s+\beta)}$$

where we note that our prior now follows that of the previous posterior ($\mathbb{P}[\Theta] = \mathbb{P}[\Theta \mid S = s] = \mathrm{B}(s+\alpha, n-s+\beta)$).

With your group, discuss how you could use this distribution to check whether your model (and, deductively, your inferences regarding the unknown) are realistic given your observations.

Hint: 1. It may help to first think about trying to evaluate the probability of observing what you observed, conditioned on what you observed. This will motivate the need for the word "prediction", and the distinction between evaluating a likelihood, evaluating the evidence, and predictive checking. 1. Since we only have access to a fixed data set, consider breaking your data set into pieces that serve separate purposes. Just as we secluded a prior data set ($j = 1, 99$), we could further split the data to include an evaluation or test set. Discuss how you might compare posterior predictions to this test set, whether it would make sense to evaluate the posterior predictive probability of observing the test set, and what does or doesn't go wrong if you tried to evaluate the posterior predictive probability of observing the data *that you did observe (i.e. conditioned on).*

If you start going in circles here, call over the instructor, or review BDA chapter 6.

Replace this text with your discussion. Then, describe your method, and implement an example below.

TODO

[12]:
```
# Split data into training and test sets
train_percent = 0.7
split_idx     = int(len(success_counts)*train_percent)
shuffle_idx = np.arange(0,len(success_counts))
np.random.shuffle(shuffle_idx)
train_idx  = shuffle_idx[:split_idx]
test_idx   = shuffle_idx[split_idx:]
train_data = success_counts.iloc[train_idx]
test_data  = success_counts.iloc[test_idx]

# Train using evidence maximization
s_samples = train_data['s'].to_numpy()
n_samples = train_data['n'].to_numpy()
alpha_train, beta_train = evidence_maximization(s_samples, n_samples, alpha_mom, beta_mom)
print('Alpha and beta via evidence maximization on training data')
print(f"alpha = {alpha_train:0.3f}, beta = {beta_train:0.3f}")

# Define the predictive posterior distribution
def predict_post_dist(s, n, alpha, beta):
    from scipy.special import betaln, comb
    log_prob = np.log(comb(n, s)) + betaln(s + alpha, n - s + beta) - betaln(alpha, beta)
    return np.exp(log_prob)

# Compute predictive probabilities for each test observation
test_s = test_data['s'].to_numpy()
test_n = test_data['n'].to_numpy()
predictive_probs     = predict_post_dist(test_s, test_n, alpha_train, beta_train)
predictive_probs_mom = predict_post_dist(test_s, test_n, alpha_mom, beta_mom)
print("")
print("Predictive probabilities for test data via evidence maximization \n:", predictive_probs)
```

```
print("")
print("Predictive probabilities for test data from matching of moments  \n:", predictive_probs_mom)
print("")
log_pred_probs     = np.log(predictive_probs)
log_pred_probs_mom = np.log(predictive_probs_mom)
log_pred_prob      = np.sum(log_pred_probs)
log_pred_prob_mom  = np.sum(log_pred_probs_mom)
print(f"Log predictive probability via evidence maximization : {log_pred_prob:0.3f}")
print(f"Log predictive probability from matching of moments  : {log_pred_prob_mom:0.3f}")
print(f"Predictive probability via evidence maximization : {np.exp(log_pred_prob)}")
print(f"Predictive probability via evidence maximization : {np.exp(log_pred_prob)}")
if log_pred_prob > log_pred_prob_mom:
    print("Maximizing evidence leads to a more predictive posterior")
else:
    print("Matching of moments leads to a more predictive posterior...")



# From Taiki
df = pd.DataFrame(np.zeros((len(test_data), 4)), columns=["n", "s", "lower", "upper"])
df["n"] = test_n
df["s"] = test_s
for i, (s, n) in enumerate(zip(test_s, test_n)):
    Pr_leq_j = 0
    # print("="*50)
    # print(f"s = {s}, n = {n}")
    for j in range(n+1):
        Pr_leq_j += predict_post_dist(j, n, alpha_train, beta_train)
        if Pr_leq_j < 0.025:
            df.iloc[i, 2] = j
        if Pr_leq_j < 0.975:
            df.iloc[i, 3] = j
        # print(f"Pr[S <= {j} | n={n}] = {Pr_leq_j:0.3f}")

fig, ax = plt.subplots(figsize=(10, 5))
mid = (df["lower"] + df["upper"]) / 2
ax.errorbar(np.arange(len(test_s)) + 1, mid, yerr=[mid - df["lower"], df["upper"] - mid], fmt='none',
            color='black', capsize=5, label = r"95\% prediction interval")
ax.scatter(np.arange(len(test_s)) + 1, df["s"], color='red', label="Actual Success Number", zorder=5)
ax.legend(prop={'size': 14})
ax.set_ylabel("Number of Success", fontsize = 14)
ax.set_xlabel("Test Data Set", fontsize = 14)
ax.tick_params(axis='both', labelsize=14)
plt.show()
```

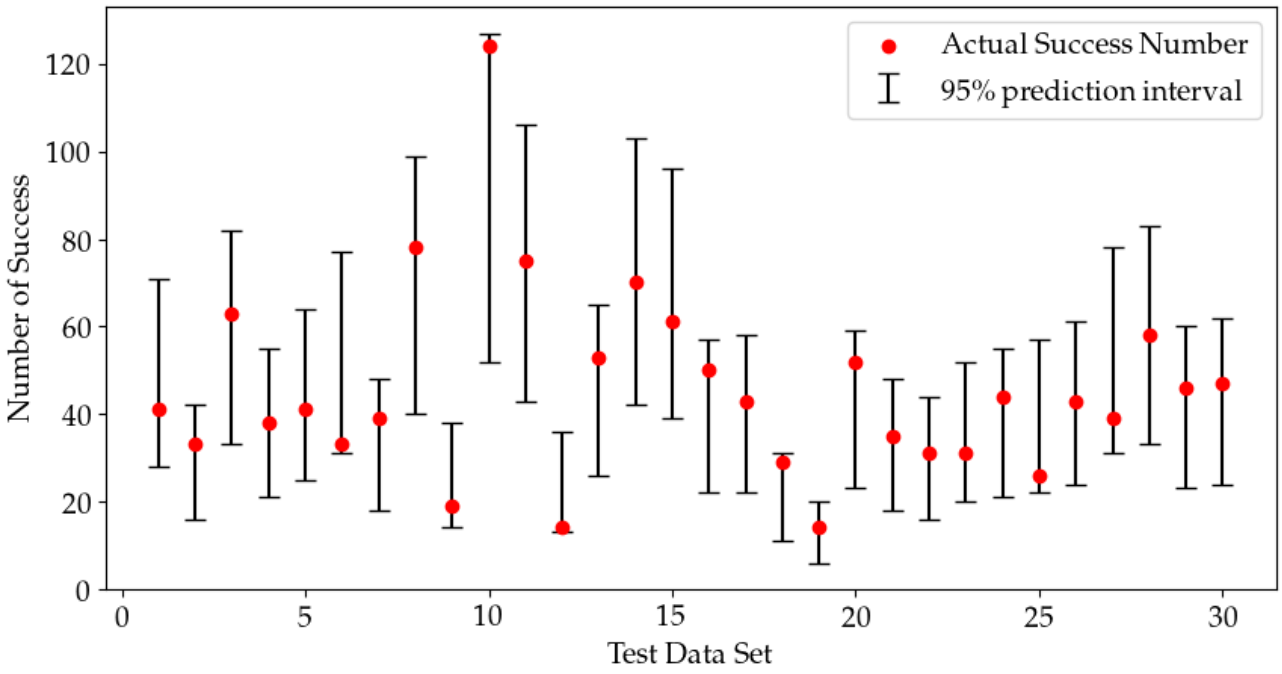
```
Alpha and beta via evidence maximization on training data
alpha = 7.240, beta = 3.278

Predictive probabilities for test data via evidence maximization
: [0.01693112 0.05495341 0.02901418 0.03645364 0.02613248 0.00513109
 0.04934936 0.02488342 0.02135094 0.00935108 0.02029415 0.00888386
 0.03702969 0.01953757 0.01722656 0.0379811  0.03945225 0.05583738
 0.09073218 0.03610738 0.04623669 0.04656511 0.02577699 0.0433887
 0.00939628 0.03374658 0.00949239 0.02489231 0.03971017 0.03728432]

Predictive probabilities for test data from matching of moments
: [0.01873743 0.05318098 0.02830217 0.03714955 0.02756127 0.00645069
 0.04685151 0.02392285 0.02454432 0.00818313 0.02051582 0.01118349
 0.03515035 0.02005847 0.0182802  0.03492559 0.03894128 0.05064897
 0.09196794 0.03308797 0.04592088 0.04704389 0.02800004 0.04154266
```

```
 0.01138522 0.03420408 0.01116304 0.0253202  0.03858836 0.03659695]

Log predictive probability via evidence maximization : -108.852
Log predictive probability from matching of moments  : -108.206
Predictive probability via evidence maximization : 5.322505799397041e-48
Predictive probability via evidence maximization : 5.322505799397041e-48
Matching of moments leads to a more predictive posterior...
```



Post-Script (Peaking at the Answer)

This part is optional.

Now that you have a pipeline for: a. fitting the prior, b. using it to produce inferences, and c. rejecting models that produce implausible inferences, it's worth seeing whether that pipeline typically returns accurate inferences.

Select a subset of the provided data as a prior set for fitting the prior parameters, then apply steps b and c to the remaining test cases. Think carefully about whether you should use step c to eliminate specific test cases, or to eliminate all inferences if enough test cases are sufficiently implausible. Then, compare your remaining answers to the true success probabilities.

Note: A nice way to summarize whether your pipeline is producing accurate answers when you have access to the truth is via a coverage test. That is, return an interval estimate for each unknown Θ_j in the test set that should hold the unknown with some large success probability (I usually repeat this for success probabilities 0.5, 0.75, 0.875, 0.95, 0.99 or something to the effect). Then, compute the fraction of the intervals that *actually* contained the truth. These should match. Notice that this is essentially a procedural guarantee. Consider the analogy to confidence intervals. This process of checking whether the pipeline produces intervals that actually cover the unknown as often as they purport to cover the unknown is especially important in more complicated examples when we can't perform step b exactly, and instead use approximations to perform posterior inference.

[ ]:

Lab 2: Bayesian Pipeline for Kidney Cancer

Lab Date: Wednesday, February 5
Due: Wednesday, February 13

Instructions

Work with your lab group to complete the following notebook. It will be reviewed by your peers in lab next week (Wednesday, February 13th).

This lab builds directly on your first lab. The idea is to use the pipeline you developed last week to solve two Bayesian decision theory problems. Both are optimal resource allocations problems.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials. You may use the same procedure you used last week to work with that folder.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

[1]:
```
# load whatever packages you prefer here. I've added a reference list
import numpy as np
import pandas as pd
import scipy
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
# sns.set()
# import pymc as pm
# import bambi as bmb
# import arviz as az
# import statsmodels.api as sm

# Add what you like beneath here...
from typing import Literal
# from scipy.stats import beta, norm
# from scipy.special import betaln, binom

# My plotting settings
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['text.usetex'] = True
mpl.rcParams['text.latex.preamble'] = r'\usepackage{mathpazo}'
mpl.rcParams["font.family"] = "Palatino"
```

The Setting

This week's lab will use the "Kidney Cancer" data set from BDA (Gelman, Third Edition). The data set contains death counts due to kidney cancer in the United States. The counts are subdivided at a state and county level. The data set includes the county populations, and death counts over two 5 year periods (1980 - 1984, 1985 - 1989).

Focus on the following columns in the kidney cancer dataset: * `state`: the US state * `Location`: the county and state as a string * `fips`, which provides the `FIPS code` for each county: this is a standard identifier that can often be used to join datasets with county-level information. * `dc` and `dc.2`: the number of kidney cancer deaths between 1980-1984 and 1985-1989, respectively * `pop` and `pop.2`: the population between 1980-1984 and 1985-1989, respectively

[2]:
```
kc_full = pd.read_csv('kidney_cancer_1980.csv', skiprows=4)
# There are many other interesting columns, but we'll focus on these:
kc = kc_full.loc[:, ['state', 'Location', 'dc', 'dc.2', 'pop', 'pop.2']]
```

```
kc.head()
```

[2]:

	state	Location	dc	dc.2	pop	pop.2
0	ALABAMA	Autauga County, Alabama	2	1	61921	64915
1	ALABAMA	Baldwin County, Alabama	7	15	170945	195253
2	ALABAMA	Barbour County, Alabama	0	1	33316	33987
3	ALABAMA	Bibb County, Alabama	0	1	30152	31175
4	ALABAMA	Blount County, Alabama	3	5	88342	91547

Take a moment with your group to explore this data set. You should notice that, outside of the specific context, it takes the same form as the synthetic data set you studied last lab. The synthetic data set consisted of a collection of the form $\{n_j, S_j\}_{j=1}^{100}$ where each n_j was a number of draws of a Bernoulli random variable and S_j was the number of succesful draws.

Now, we have two datasets of this form. For each 5 year span you have a collection of the form $\{(n_j(t), D_j(t))\}_{j=1}^{2997}$. Here t indexes a timespan (either 1980 - 1984 or 1985 - 1989).

[3]:
```
# use this cell to perform whatever data exploration you want.
# I suggest querying for the quantiles of the population and death count columns.
# In particular, the min and max are important.
# Notice also that, while the population sizes are large, the observed rate of deaths per individual
are small.
kc['death_rate']   = kc['dc'] / kc['pop']
kc['death_rate.2'] = kc['dc.2'] / kc['pop.2']
kc.describe()
```

[3]:

	dc	dc.2	pop	pop.2	death_rate	death_rate.2
count	3110.000000	3110.000000	3.110000e+03	3.110000e+03	3110.000000	3110.000000
mean	7.527974	8.359164	1.550087e+05	1.606611e+05	0.000050	0.000058
std	23.095402	23.765728	4.722345e+05	4.972750e+05	0.000044	0.000050
min	0.000000	0.000000	2.580000e+02	2.670000e+02	0.000000	0.000000
25%	1.000000	1.000000	2.238250e+04	2.213000e+04	0.000023	0.000028
50%	2.000000	3.000000	4.762800e+04	4.806950e+04	0.000045	0.000052
75%	6.000000	7.000000	1.146262e+05	1.179355e+05	0.000070	0.000077
max	647.000000	598.000000	1.526640e+07	1.660789e+07	0.000452	0.000718

The Model

We will work with the same model we used last week. This is, formally, our first example of a hierarchical model. A hierarchical model is simply a sampling model where the samples are performed sequentially, and where new samples may be sampled conditioned on the old samples.

In particular, we will assume that, for each county and time frame (indexed by j and t):

1. **Background Parameters:** $\alpha(t), \beta(t)$. These are assumed to be fixed but unknown.
2. **Death Rates:** $\Theta_j(t) \sim \text{Beta}(\alpha(t), \beta(t))$.
3. **Death Counts:** $D_j(t)|\Theta_j(t) = \theta \sim \text{Binomial}(n_j(t), \theta)$.

At each level of the hierarchy the set of samples are drawn i.i.d.

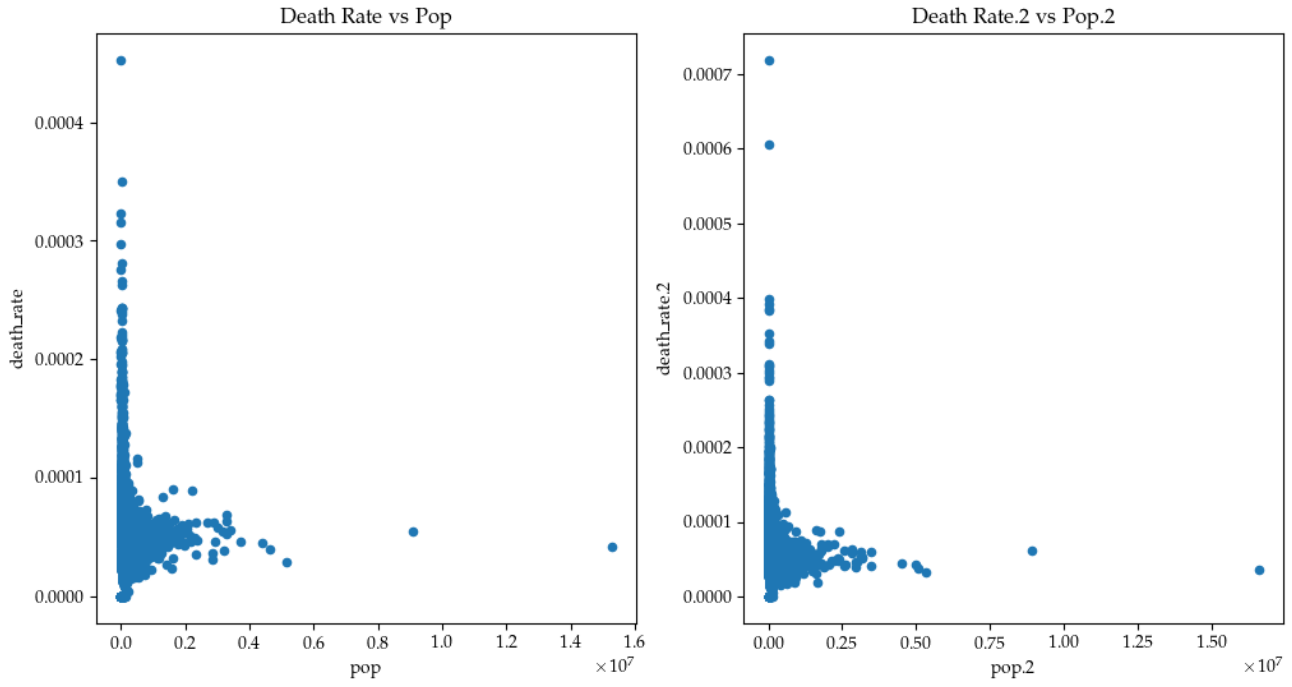
Hierarchical models are helpful since they can be used to pool and share information. You practiced this last week when you used the first 99 draws to estimate the background parameters α, β that parameterized the prior needed for posterior inference on the 100th draw. In this lab, we will work with the full model. Here, sharing information is important since

death rates due to kidney cancer are relatively low on a per capita basis. As a result, some small counties have very few observed deaths, even over a five-year period. Direct estimation of the death rates in those counties would be unreliable due to the small expected number of observed events in small counties. We will use the death rates observed in large counties to regularize our inference on small counties.

Model Assumptions (A Warning)

The model stated above makes a number of assumptions about the counties. Notice, for example, that it assumes that the prior distribution on the death rates is the same in all counties. This might not be the case, as counties with large populations tend to be urban (search for Cook County Illinois or Orange County California) while small counties tend to be rural. We might expect systematic differences in the death rates associated with county size that might break the exchangeability assumption used to "borrow" information from large counties when estimating death rates in small counties. We don't have the tools to treat this yet, but, it is worth remembering. The appropriate choice of Bayesian model should be informed by context knowledge. (Note: this is not too hard to check with a scatterplot. Try comparing county size to the MLE estimator for the death rate in each county... how concerned are you about our assumptions?).

[4]:
```
_, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 6))
kc.plot(x='pop', y='death_rate', kind='scatter', ax=ax1, title='Death Rate vs Pop')
kc.plot(x='pop.2', y='death_rate.2', kind='scatter', ax=ax2, title='Death Rate.2 vs Pop.2')
plt.show()
```



The Inference Pipeline

In order to work with this model, we will need code to fit for the prior parameters, compute a posterior and perform posterior inference using the beta, binomial model, and to validate our models. Thankfully, you did all this already in Lab 1.

Go back to Lab 1, and copy the functions you wrote there into the cells below. Change any variable names needed so that they can reference the data used in this lab.

In this lab, we will focus on the first two stages of the BDA cycle, namely, fitting for the fixed prior parameters $\alpha(t)$ and $\beta(t)$ and deriving a posterior over the death rates conditioned on the observed death counts.

Evidence maximization (aka Empirical Bayes aka type II MLE)

In another approach, we can use our our observed data, S_j, directly. In this case, we maximize the evidence, i.e.,

$$\mathcal{L}_{II}(\alpha,\beta) := \prod_{i=1}^{99} \mathbb{P}[S=s] \quad \Longrightarrow \quad \alpha^\star,\beta^\star = \underset{\alpha,\beta}{\operatorname{argmin}} \left\{ -\log\left(\mathcal{L}_{II}(\alpha,\beta)\right)\right\}$$

For our model, our evidence follows $\Pr(S=s) = \binom{n}{s}\frac{\mathrm{B}(s+\alpha,n-s+\beta)}{\mathrm{B}(\alpha,\beta)}$, so we have

$$\alpha^\star,\beta^\star = \underset{\alpha,\beta}{\operatorname{argmin}} \left\{ -\sum_{i=1}^{99} \log\left(\binom{n_i}{s_i}\frac{\mathrm{B}(s_i+\alpha,n_i-s_i+\beta)}{\mathrm{B}(\alpha,\beta)}\right)\right\}$$

[5]:
```
#%% functions for fitting the prior parameters
def plot_prior_and_estimates(dataframe, s_col, n_col, a, b):

    # Isolate s and n samples from dataframe
    s_samples = dataframe[s_col].to_numpy()
    n_samples = dataframe[n_col].to_numpy()
    theta_est = s_samples/n_samples
    prior_est = scipy.stats.beta(a,b)
    thetas    = np.linspace(0,0.0005,10000)
    prior_pdf = prior_est.pdf(thetas)

    # Plot histogram and pdf
    _, ax1 = plt.subplots()
    # Plot a histogram of theta hats left y-axis
    custom_bins = np.linspace(0.0, 0.0005, 31)
    ax1.hist(theta_est, bins=custom_bins, color='green', edgecolor='black', alpha=0.7)
    ax1.set_ylabel(r'Counts of $\theta_{{est}} = s_i/n_i$', color='green')
    ax1.tick_params(axis='y', labelcolor='green')
    ax1.grid(False)

    # Plot the prior pdf on the right y-axis
    ax2 = ax1.twinx()
    ax2.plot(thetas, prior_pdf, color='blue',
            label=r'Prior: $\Theta \sim \textnormal{Beta}(\alpha_{est}, \beta_{est})$')
    ax2.set_xlabel(r'$\theta$')
    ax2.set_ylabel(r'$\mathbb{P}[\Theta] =\textnormal{Beta}(\alpha_{est}, \beta_{est})$',
color='blue')
    ax2.tick_params(axis='y', labelcolor='blue')
    ax2.set_xlim([0, 0.0005])
    ax2.set_ylim([0.0, np.max(prior_pdf)*1.1])
    ax2.grid(False)

    plt.tight_layout()
    plt.show()

# Matching the moments (see `Method of moments' on the right: https://en.wikipedia.org/wiki/
 ↪Beta_distribution)
def match_beta_moments(theta_estimates):
    mean_est  = np.mean(theta_estimates)
    var_est   = np.var(theta_estimates)
    coeff     =  (((mean_est * (1 - mean_est)) / var_est) - 1)
    alpha_mom = coeff * mean_est
    beta_mom  = coeff * (1 - mean_est)
    return alpha_mom, beta_mom

# Evidence Maximization (aka Empirical Bayes aka type II MLE)
```

```
def fit_prior_beta(dataframe, s_col, n_col, initial_alpha=None, initial_beta=None):

    # Isolate s and n samples from dataframe
    s_samples = dataframe[s_col].to_numpy()
    n_samples = dataframe[n_col].to_numpy()

    # If initial guess is not provided, match moments
    if initial_alpha is None or initial_alpha is None:
        initial_alpha, initial_beta = match_beta_moments(s_samples/n_samples)

    # Optimize
    from scipy.special import betaln, gammaln
    def L2(params):
        a, b = params
        if a <= 0 or b <= 0:
            return np.inf
        log_terms = gammaln(n_samples + 1) - gammaln(s_samples + 1) - gammaln(n_samples - s_samples +
1) \
                  + betaln(s_samples + a, n_samples - s_samples + b) \
                  - betaln(a, b)
        return -np.sum(log_terms)

    #---Option 1
    bounds = [(1e-8, None), (1e-8, None)]
    result = scipy.optimize.minimize(L2, [initial_alpha, initial_beta],
                                     method='L-BFGS-B', bounds=bounds)
    #---Option 2
    # result = scipy.optimize.minimize(L2, [initial_alpha, initial_beta])
    if not result.success:
       print("WARNING! Evidence maximization failed! Returning matching of moments alpha and beta...
 ↪")
       print(result)

    return result.x if result.success else (initial_alpha, initial_beta)

alpha1, beta1 = fit_prior_beta(kc, s_col='dc'  , n_col='pop'  )
alpha2, beta2 = fit_prior_beta(kc, s_col='dc.2', n_col='pop.2')
print(f"alpha_1 = {alpha1:0.3f}, beta_1 = {beta1:0.3f}")
print(f"alpha_2 = {alpha2:0.3f}, beta_2 = {beta2:0.3f}")
```

```
alpha_1 = 1.604, beta_1 = 26641.129
alpha_2 = 1.577, beta_2 = 22886.708
```

[6]:
```
# functions for performing posterior inference
def posterior_inference(dataframe, index, population, alpha_prior, beta_prior, visualize=True):

        # Isolate data from DataFrame
        if population == 2:
                dc_key, pop_key = 'dc.2', 'pop.2'
        else:
                dc_key, pop_key = 'dc', 'pop'
        s, n = dataframe[dc_key][index], dataframe[pop_key][index]
        county = dataframe["Location"][index]

        # Perform inference (make estimates of \theta and define credible interval)
        posterior = scipy.stats.beta(s+alpha_prior,n-s+beta_prior)
        thetas    = np.linspace(0,0.0005,10000)
        post_pdf  = posterior.pdf(thetas)
        estimates = {
```

```
                'MLE'  : s / n,
                'mean' : (s + alpha_prior) / (n + alpha_prior + beta_prior),
                'MAP'  : (s + alpha_prior - 1) / (n + alpha_prior + beta_prior - 2),
                'lower': posterior.ppf(0.025),
                'upper': posterior.ppf(0.975),
                'alpha_post': s+alpha_prior,
                'beta_post': n-s+beta_prior,
                'Location':county
        }

        # If you wish to visualize your inference
        if visualize:
                # Create figure
                plt.figure()

                # Plot the estimators
                plt.axvline(estimates['MLE'] , color='red',
label=f'$\\hat{{\\theta}}_{{MLE}}=\\frac{{s}}{{n}}$')# + f"={estimates['MLE']:.3e}")
                plt.axvline(estimates['mean'], color='green',
label=f'$\hat{{\\theta}}_{{mean}}=\\frac{{s+\\alpha}}{{n-s+\\beta}}$')# + f"={estimates['mean']:.
 ↪3e}")
                plt.axvline(estimates['MAP'] , color='orange',
label=f'$\hat{{\\theta}}_{{MAP}}=\\frac{{s+\\alpha-1}}{{n-s+\\beta-2}}$')# + f"={estimates['MAP']:.
 ↪3e}")

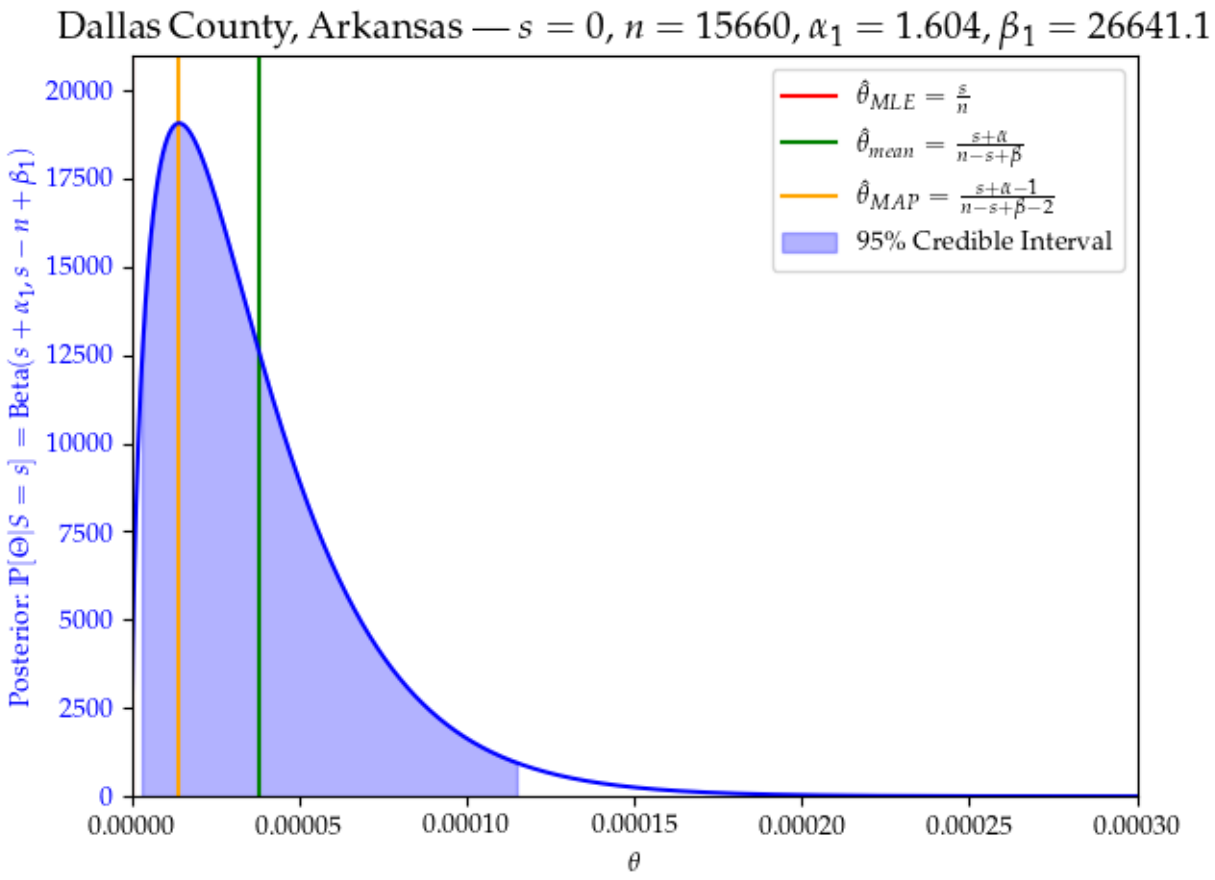
                # Plot the pdf of the posterior
                plt.plot(thetas, post_pdf, color='blue')

                # Compute 95% credible interval
                lower_bound = estimates['lower']
                upper_bound = estimates['upper']
                mask = (thetas >= lower_bound) & (thetas <= upper_bound)
                plt.fill_between(thetas[mask], post_pdf[mask], color='blue', alpha=0.3, label='95\%
Credible Interval')
                # plt.plot([lower_bound, lower_bound], [0, posterior.pdf(lower_bound)],
color='black')
                # plt.plot([upper_bound, upper_bound], [0, posterior.pdf(upper_bound)],
color='black')

                # Axes, labels, etc
                plt.xlabel(r'$\theta$')
                plt.ylabel(f'Posterior: $\mathbb{{P}}[\Theta | S=s]=
\\textnormal{{Beta}}(s+\\alpha_{population}, s-n+\\beta_{population})$', color='blue')
                plt.title(f'{county} --- $s = {s}$, $n = {n}$, $\\alpha_{population} = {alpha_prior:
 ↪0.3f}$, $\\beta_{population} =  {beta_prior:0.1f}$', fontsize=14)
                plt.tick_params(axis='y', labelcolor='blue')
                plt.xlim([0, 0.0003])
                plt.ylim([0.0, np.max(post_pdf)*1.1])
                plt.legend()
                plt.grid(False)
                plt.tight_layout()
                plt.show()

        return estimates

alpha_prior, beta_prior = alpha1, beta1
est = posterior_inference(kc, 100, 1, alpha_prior, beta_prior)
print(f"{est}")
```




```
{'MLE': 0.0, 'mean': 3.792657840537214e-05, 'MAP': 1.428811928801674e-05,
'lower': 3.1233515578234807e-06, 'upper': 0.00011505388683919638, 'alpha_post':
1.604397918584054, 'beta_post': 42301.12856775275, 'Location': 'Dallas County,
Arkansas'}
```

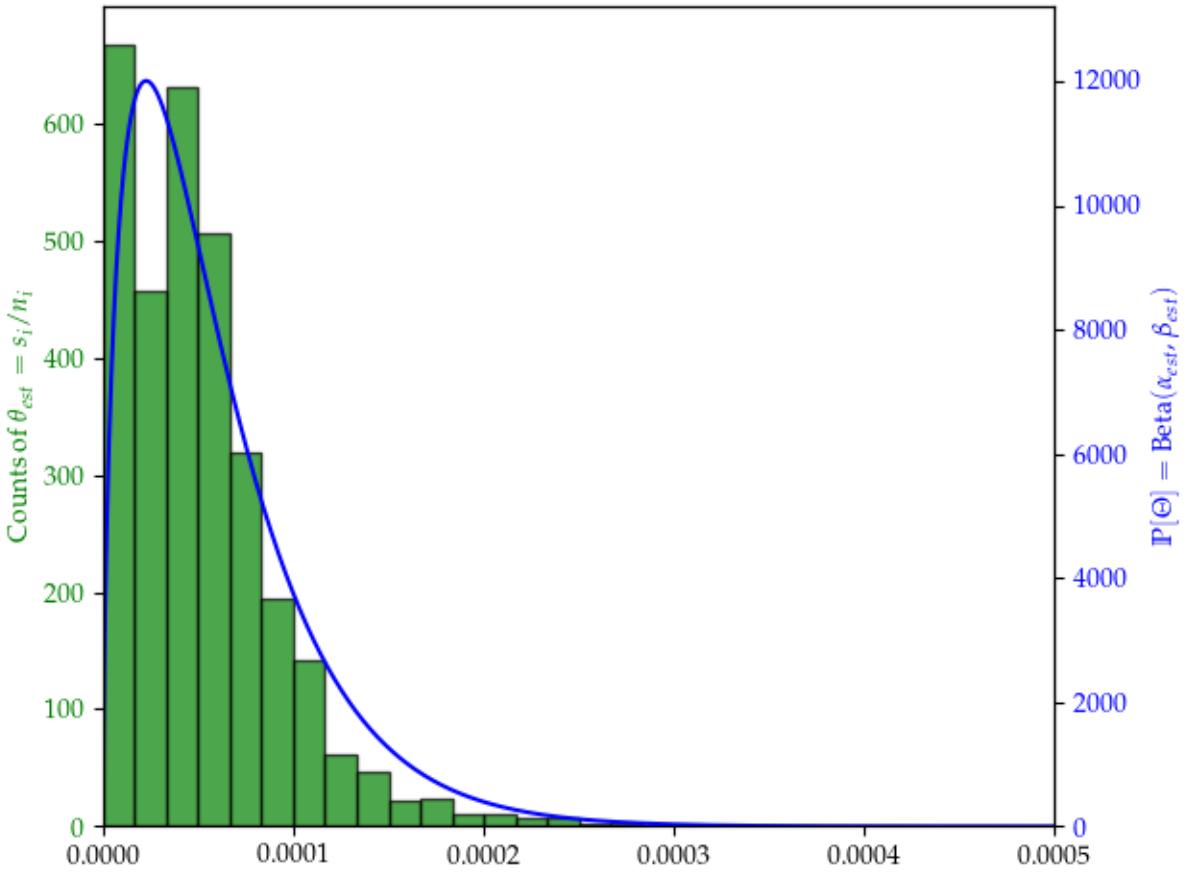
[7]:
```
# functions for validating the model (this is optional)
```

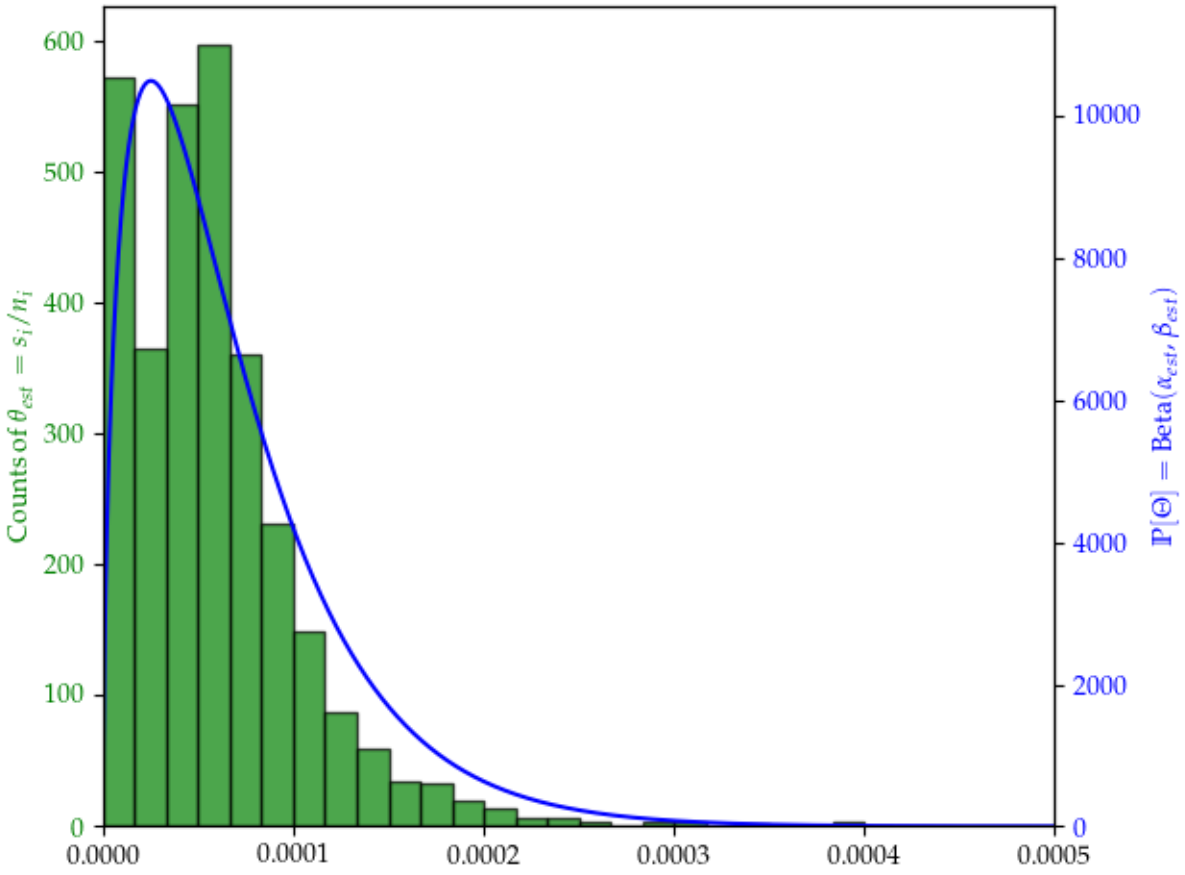
Fit for the Prior Parameters

In the cell below, use the fitting procedure you developed in the last lab to get a guess at $\alpha(t), \beta(t)$ for the two time-spans. If your group did not build functioning code for this task last lab, either raise a hand to ask the TA for advice, or, talk to the other lab groups. You should have discussed strategies with those groups in your evaluations. You are welcome to try and implement any good ideas they described.

[8]:
```
#%% Fit prior parameters
alpha1, beta1 = fit_prior_beta(kc, s_col='dc'  , n_col='pop'  )
alpha2, beta2 = fit_prior_beta(kc, s_col='dc.2', n_col='pop.2')
print(f"alpha_1 = {alpha1:0.3f}, beta_1 = {beta1:0.3f}")
print(f"alpha_2 = {alpha2:0.3f}, beta_2 = {beta2:0.3f}")
plot_prior_and_estimates(kc, s_col='dc'  , n_col='pop'  , a=alpha1, b=beta1)
plot_prior_and_estimates(kc, s_col='dc.2', n_col='pop.2', a=alpha2, b=beta2)
```

```
alpha_1 = 1.604, beta_1 = 26641.129
alpha_2 = 1.577, beta_2 = 22886.708
```


Counts of $\theta_{est} = s_i/n_i$
$\mathbb{P}[\Theta] = \text{Beta}(\alpha_{est}, \beta_{est})$
0
100
200
300
400
500
600
0
2000
4000
6000
8000
10000
12000
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005



Clearly report your estimates for $\alpha(t), \beta(t)$. - What do you notice about these parameters? - What do they imply about the shape of your prior? - How does the prior mean compare to the MLE for the national death rate due to kidney cancer? How does it compare to the MLE among just the largest counties (say, the top twenty)?

[9]:
```
# Comparison
kc_large_1980 = kc.nlargest(20, 'pop')
kc_large_1985 = kc.nlargest(20, 'pop.2')
print('prior mean (1980-1984): ', alpha1 / (alpha1 + beta1))
print('MLE (1980-1984):        ', sum(kc['dc']) / sum(kc['pop']))
print('MLE (1980-1984, large counties):        ', sum(kc_large_1980['dc']) /
sum(kc_large_1980['pop']))
print('prior mean (1985-1989): ', alpha2 / (alpha2 + beta2))
print('MLE (1985-1989):        ', sum(kc['dc.2']) / sum(kc['pop.2']))
print('MLE (1985-1989, large counties):       ', sum(kc_large_1985['dc.2']) / sum(kc_large_1985['pop.
 ↪2']))
```

```
prior mean (1980-1984):  6.021896930210918e-05
MLE (1980-1984):         4.856485743364176e-05
MLE (1980-1984, large counties):         4.8538193977842634e-05
prior mean (1985-1989):  6.891784429434295e-05
MLE (1985-1989):         5.2029789627331375e-05
MLE (1985-1989, large counties):        4.673749581116483e-05
```

- Looking at the parameters, we can see that β is much larger than α, and the probability of kidney cancer is extremely low. Θ has a shape where the density gradually decreases after peaking at a value very close to zero, so the average value of the data is higher than the mode.
- In addition, the value of the prior mean is larger than the MLE, and this trend is robust even when the MLE is calculated by aggregating the 20 counties with the largest population.

Infer Death Rates

Pick three different counties from the data set and form posterior estimates for their death rates in each of the two time periods. Select a small county, a large county, and a county of intermediate size (say, size close to the median size over all counties.

[10]:
```
# Perform posterior inference for each county
# NOTE: We use the first time point as we will use it to infer into future
for c in range(len(kc)):
    estimates = posterior_inference(kc, c, population=1, alpha_prior=alpha1, beta_prior=beta1,
visualize=False)
    for key, value in estimates.items():
        kc.loc[c, key] = value

# Small, medium, and large counties
small  = 10000
middle = 50000
large  = 2000000
small_county = (kc['pop'] - small).abs().idxmin()
med_county   = (kc['pop'] - middle).abs().idxmin()
large_county = (kc['pop'] - large).abs().idxmin()

# Perform posterior inference on the small, medium, and large counties in each of the two time
periods
small_est_1 = posterior_inference(kc, small_county, population=1, alpha_prior=alpha1,
beta_prior=beta1)
small_est_2 = posterior_inference(kc, small_county, population=2, alpha_prior=alpha2,
beta_prior=beta2)
med_est_1   = posterior_inference(kc, med_county  , population=1, alpha_prior=alpha1,
beta_prior=beta1)
med_est_2   = posterior_inference(kc, med_county  , population=2, alpha_prior=alpha2,
beta_prior=beta2)
large_est_1 = posterior_inference(kc, large_county, population=1, alpha_prior=alpha1,
beta_prior=beta1)
large_est_2 = posterior_inference(kc, large_county, population=2, alpha_prior=alpha2,
beta_prior=beta2)

# Save MLE estimates
mle = [small_est_1['MLE'],
       small_est_2['MLE'],
       med_est_1['MLE'],
       med_est_2['MLE'],
       large_est_1['MLE'],
       large_est_2['MLE']]

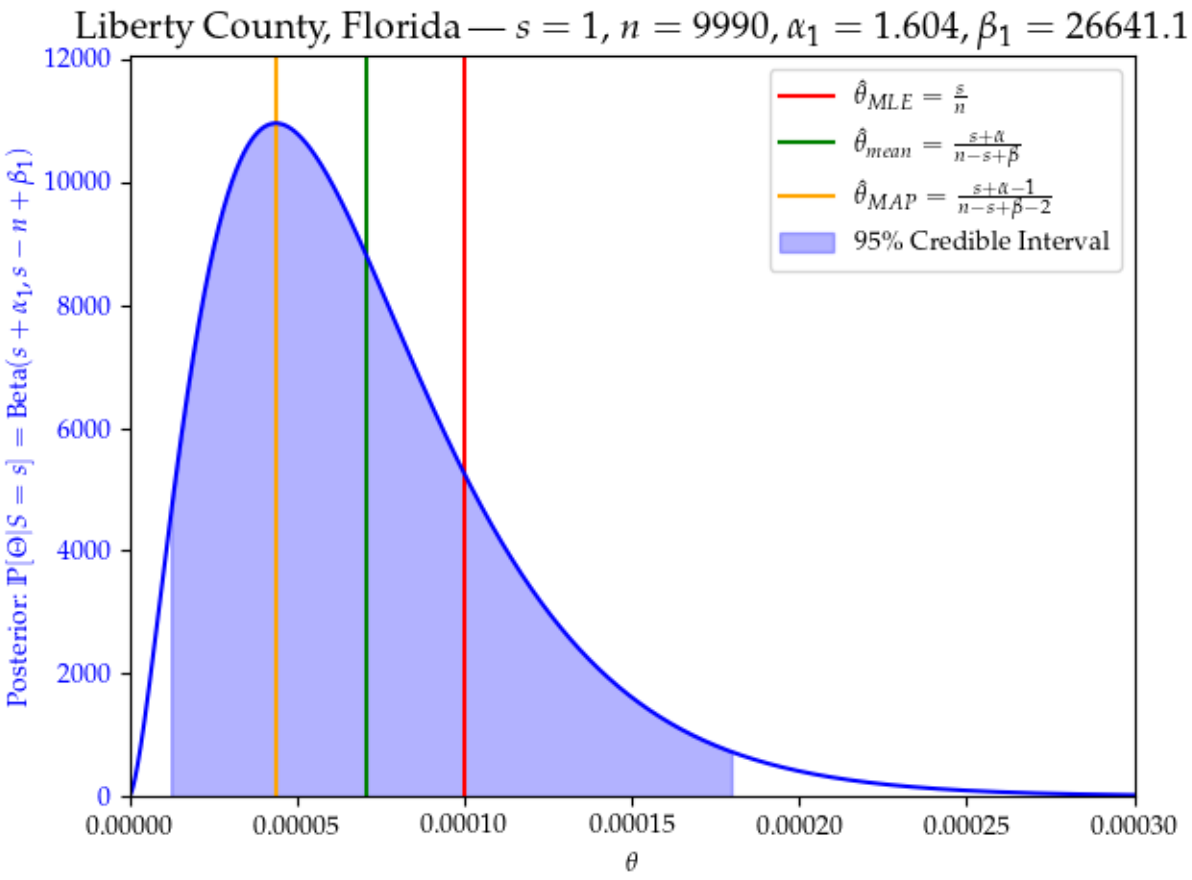
params = {
    "Small 1" : (small_est_1['alpha_post'], small_est_1['beta_post']),
    "Small 2" : (small_est_2['alpha_post'], small_est_2['beta_post']),
    "Middle 1": (med_est_1['alpha_post']  , med_est_1['beta_post']),
    "Middle 2": (med_est_2['alpha_post']  , med_est_2['beta_post']),
    "Large 1" : (large_est_1['alpha_post'], large_est_1['beta_post']),
    "Large 2" : (large_est_2['alpha_post'], large_est_2['beta_post']),
}

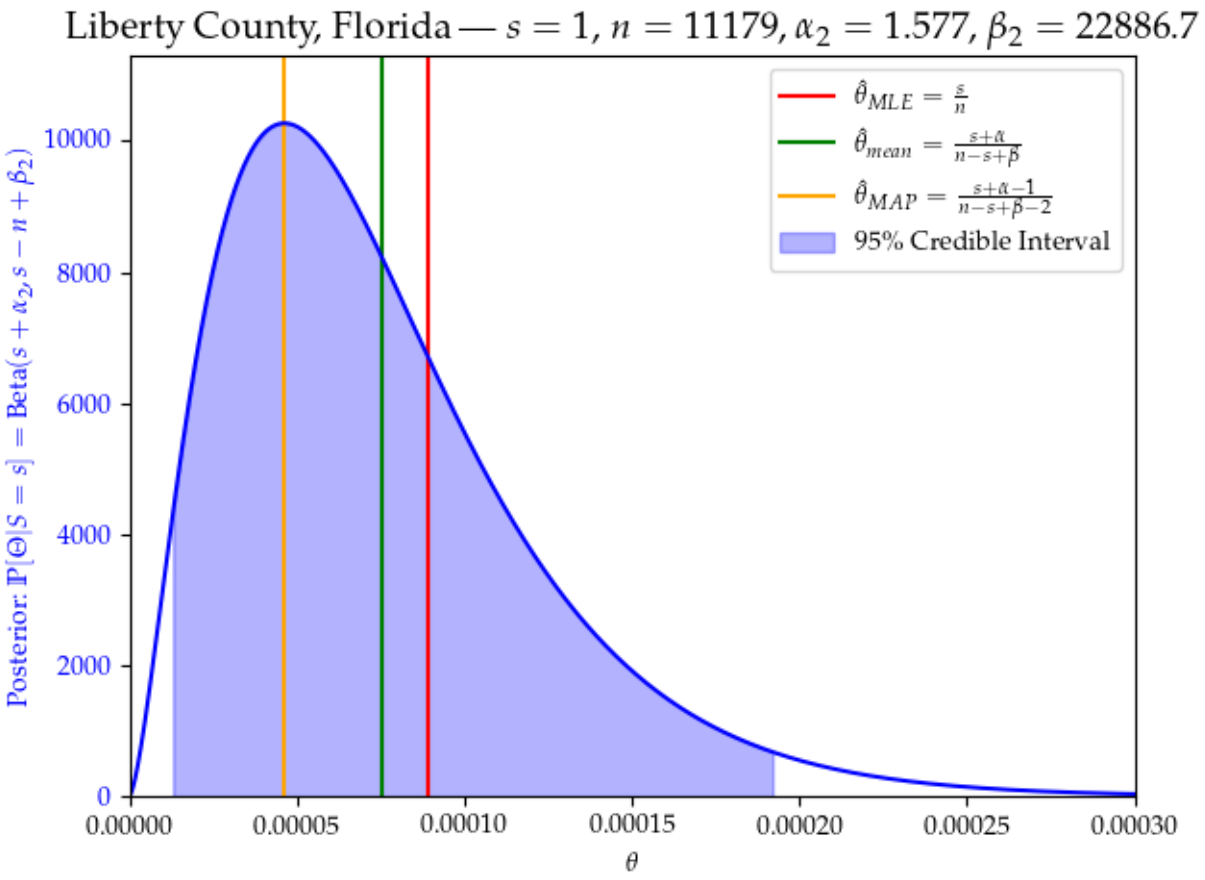
cred_intervals = {}
for label, (alpha, beta) in params.items():
    lower, upper = stats.beta.ppf([0.025, 0.975], alpha, beta)
    cred_intervals[label] = (lower, upper)
```

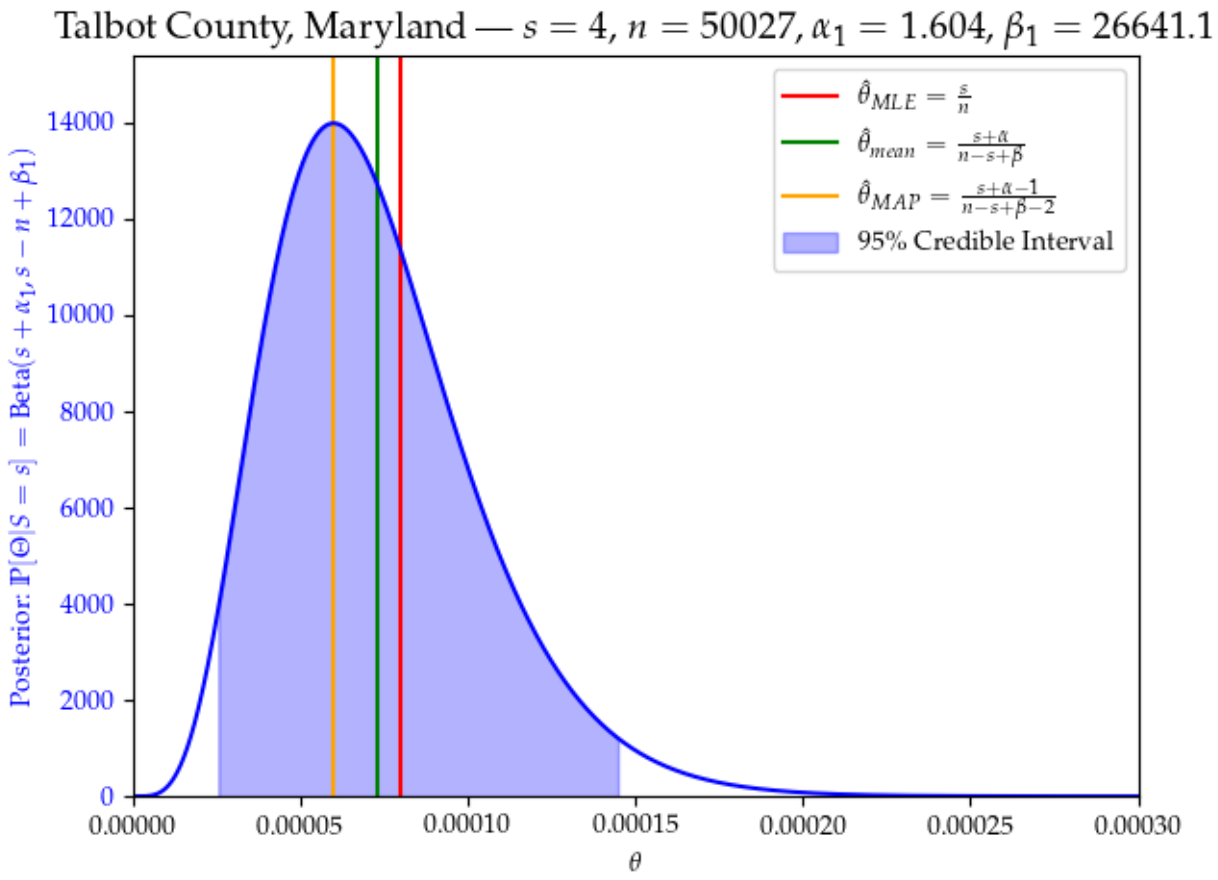
```
fig, ax = plt.subplots(figsize=(6, 8))
x_labels = list(cred_intervals.keys())
x_pos = np.arange(len(x_labels))
lower_bounds = [cred_intervals[label][0] for label in x_labels]
upper_bounds = [cred_intervals[label][1] for label in x_labels]

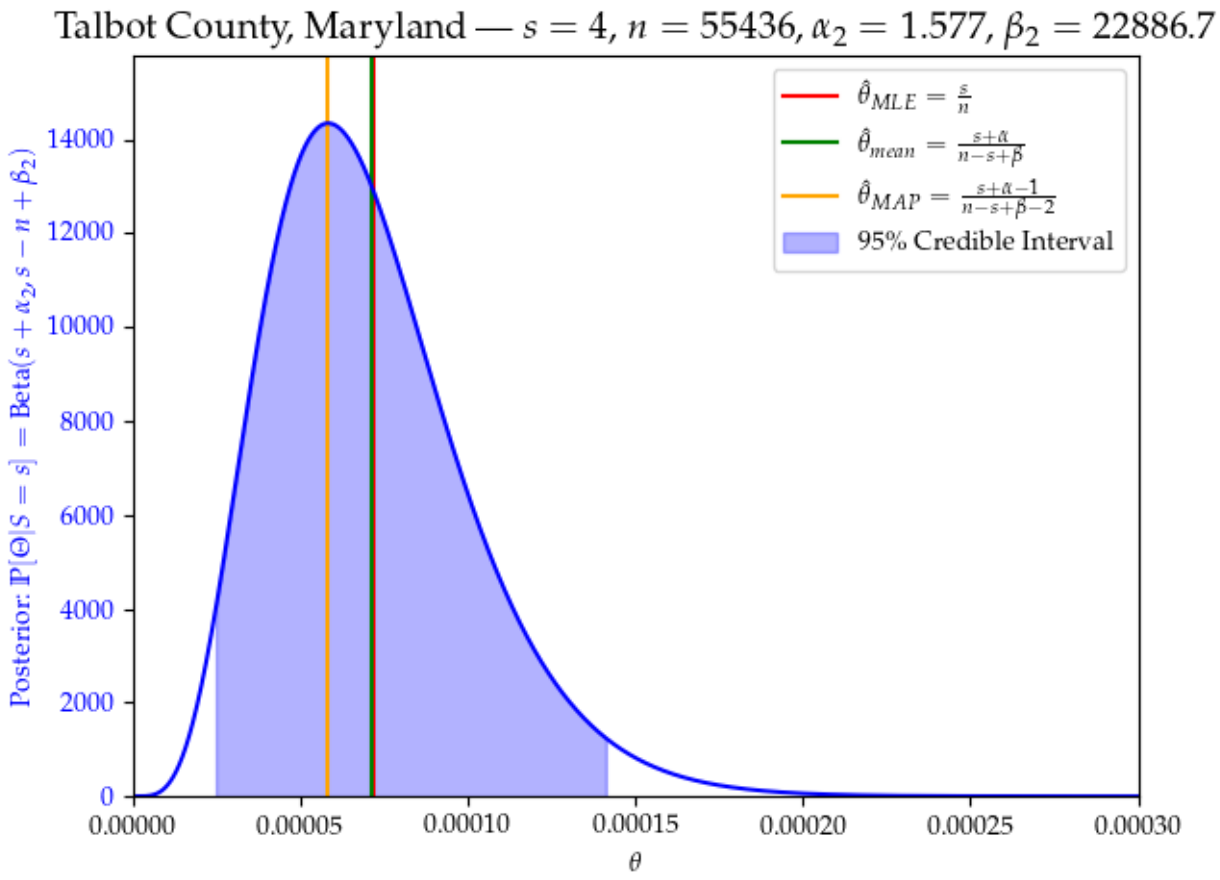
xtick_labels = [small_est_1['Location'] + "(1980-1984)",
                small_est_1['Location'] + "(1985-1989)",
                med_est_1['Location'] + "(1980-1984)",
                med_est_1['Location'] + "(1985-1989)",
                large_est_1['Location'] + "(1980-1984)",
                large_est_1['Location'] + "(1985-1989)"]

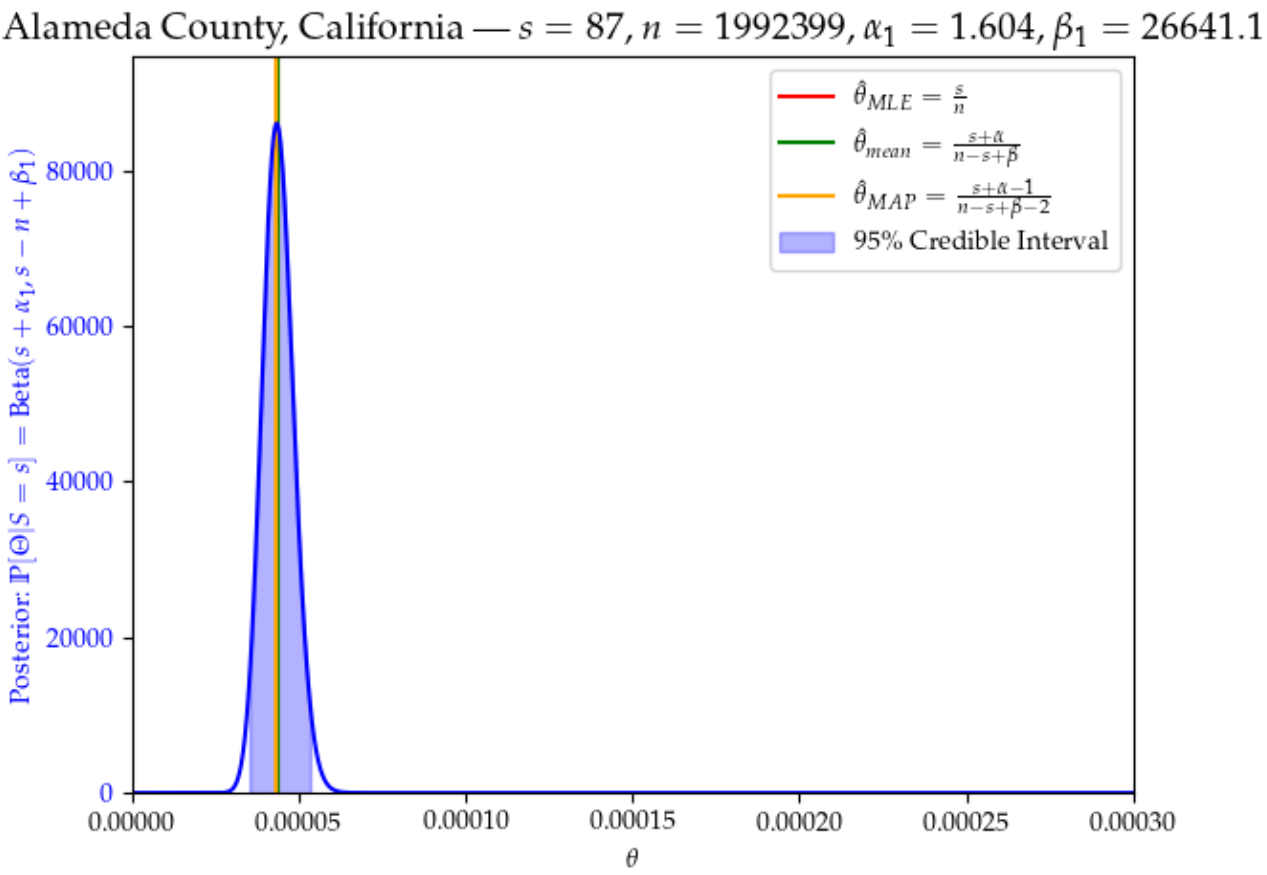
ax.vlines(x_pos, lower_bounds, upper_bounds, colors='blue', lw=2, label = "95% Credible interval")
ax.scatter(x_pos, mle, color='red', label = "MLE")
ax.set_xticks(x_pos)
ax.set_xticklabels(xtick_labels, rotation=45, ha="right")
ax.set_ylabel("Probability")
ax.set_title("95\% Credible Intervals")
ax.set_ylim(0, 0.0002)
ax.legend()
plt.show()
```

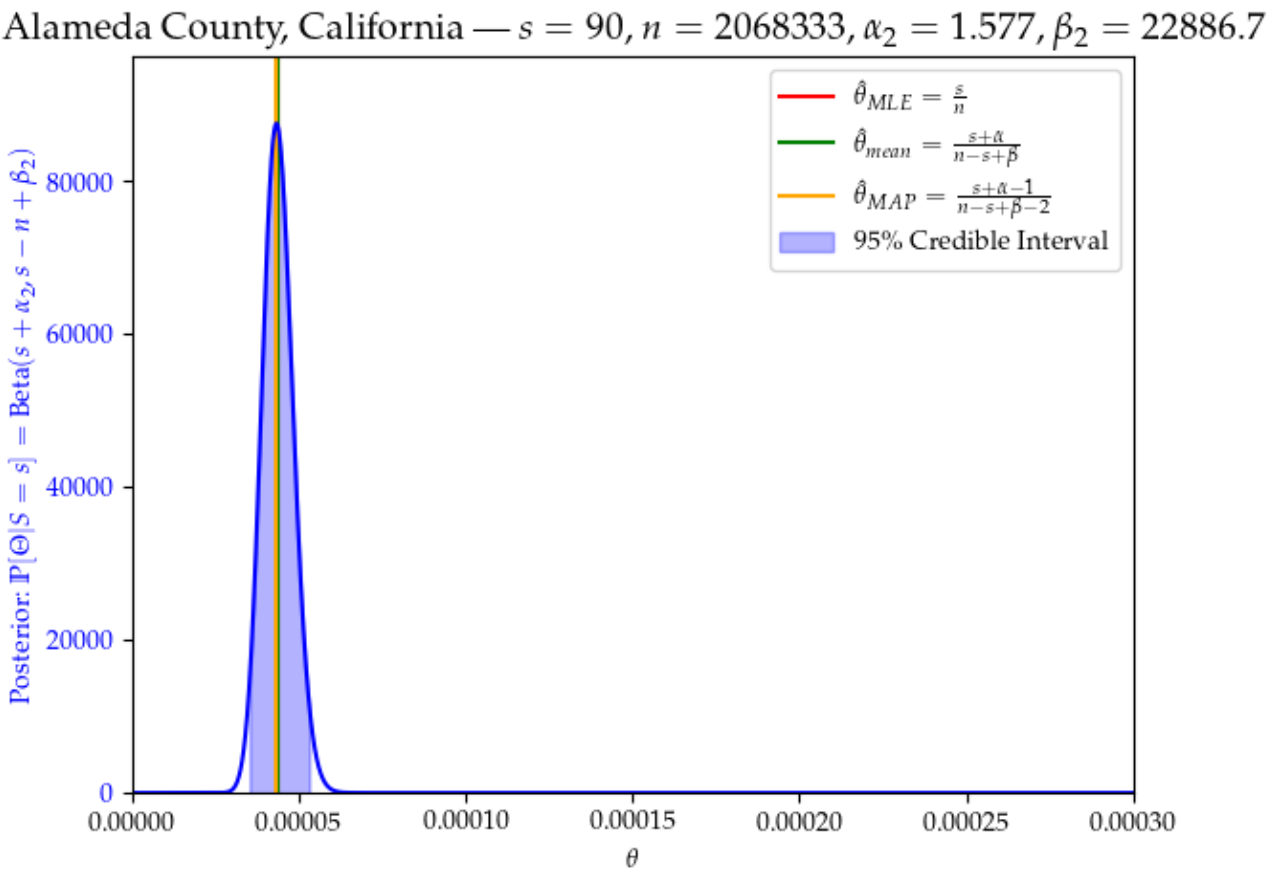


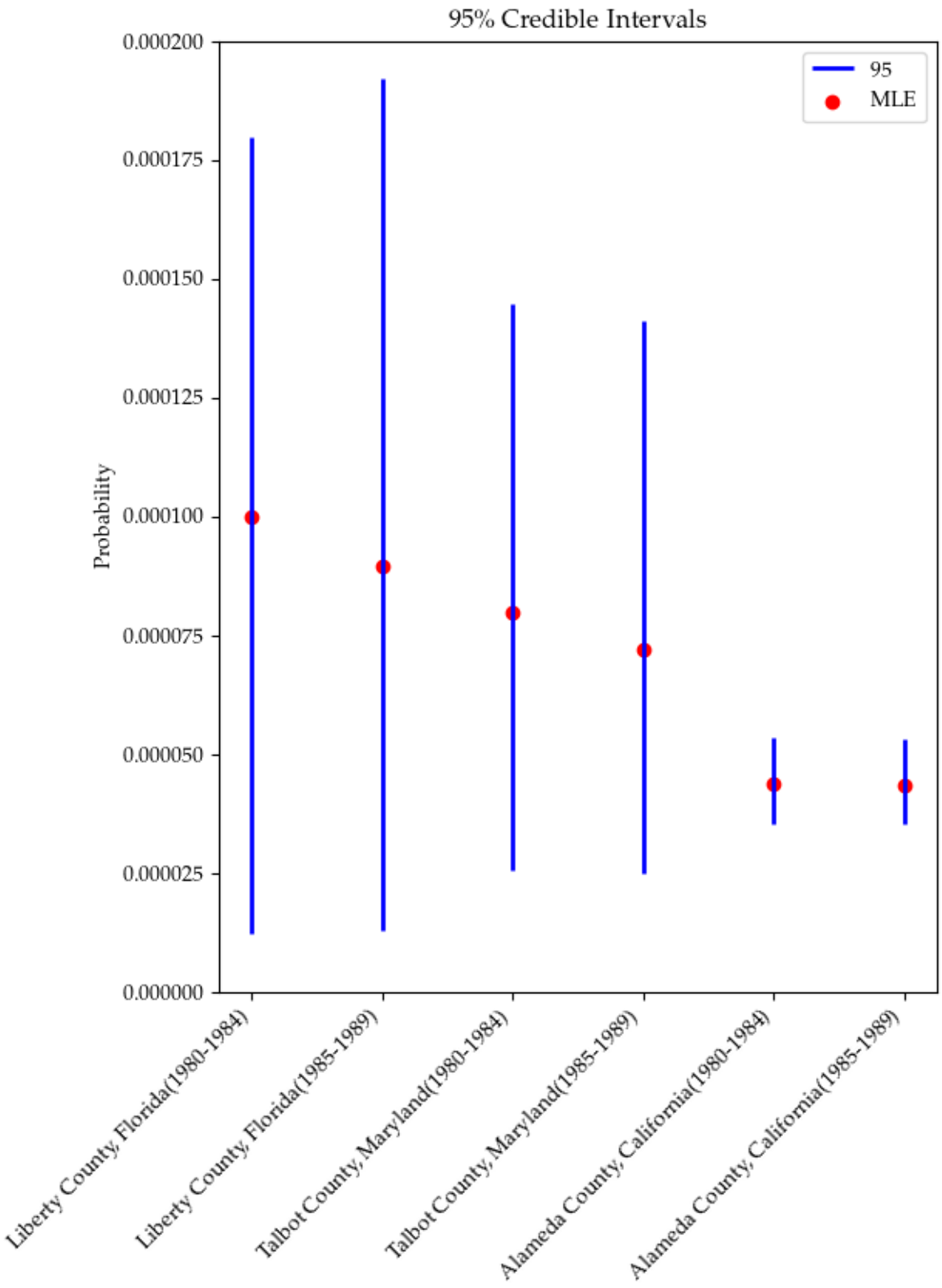

Liberty County, Florida — $s = 1, n = 11179, \alpha_2 = 1.577, \beta_2 = 22886.7$
$\hat{\theta}_{MLE} = \frac{s}{n}$
$\hat{\theta}_{mean} = \frac{s+\alpha}{n-s+\beta}$
$\hat{\theta}_{MAP} = \frac{s+\alpha-1}{n-s+\beta-2}$
95% Credible Interval
Posterior: $\mathbb{P}[\Theta|S=s] = \text{Beta}(s+\alpha_2, s-n+\beta_2)$
0
2000
4000
6000
8000
10000
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
θ


Talbot County, Maryland — $s = 4$, $n = 50027$, $\alpha_1 = 1.604$, $\beta_1 = 26641.1$
$\hat{\theta}_{MLE} = \frac{s}{n}$
$\hat{\theta}_{mean} = \frac{s+\alpha}{n-s+\beta}$
$\hat{\theta}_{MAP} = \frac{s+\alpha-1}{n-s+\beta-2}$
95% Credible Interval
Posterior: $\mathbb{P}[\Theta|S=s] = \text{Beta}(s+\alpha_1, s-n+\beta_1)$
0
2000
4000
6000
8000
10000
12000
14000
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
θ


Talbot County, Maryland — $s = 4, n = 55436, \alpha_2 = 1.577, \beta_2 = 22886.7$
$\hat{\theta}_{MLE} = \frac{s}{n}$
$\hat{\theta}_{mean} = \frac{s+\alpha}{n-s+\beta}$
$\hat{\theta}_{MAP} = \frac{s+\alpha-1}{n-s+\beta-2}$
95% Credible Interval
Posterior: $\mathbb{P}[\Theta|S=s] = \text{Beta}(s+\alpha_2, s-n+\beta_2)$
0
2000
4000
6000
8000
10000
12000
14000
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
θ


Alameda County, California — $s = 87, n = 1992399, \alpha_1 = 1.604, \beta_1 = 26641.1$
$\hat{\theta}_{MLE} = \frac{s}{n}$
$\hat{\theta}_{mean} = \frac{s+\alpha}{n-s+\beta}$
$\hat{\theta}_{MAP} = \frac{s+\alpha-1}{n-s+\beta-2}$
95% Credible Interval
Posterior: $\mathbb{P}[\Theta|S=s] = \text{Beta}(s+\alpha_1, s-n+\beta_1)$
0
20000
40000
60000
80000
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
θ


Alameda County, California — $s = 90, n = 2068333, \alpha_2 = 1.577, \beta_2 = 22886.7$
$\hat{\theta}_{MLE} = \frac{s}{n}$
$\hat{\theta}_{mean} = \frac{s+\alpha}{n-s+\beta}$
$\hat{\theta}_{MAP} = \frac{s+\alpha-1}{n-s+\beta-2}$
95% Credible Interval
Posterior: $\mathbb{P}[\Theta|S=s] = \text{Beta}(s+\alpha_2, s-n+\beta_2)$
0
20000
40000
60000
80000
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
θ



In the space below: 1. Report an interval estimate for the true death rates in each of your selected counties given the observed death counts over each time period. Select the intervals so that they should contain the true value with at least a 95% chance. 1. Comment on the difference in the width of the intervals between the large and the small counties. 1. Compare the interval estimate to the MLE in each county. Which counties have it

1. The interval estimation of the true mortality rate is shown in the graph above.
2. This estimated interval tends to become narrower for larger countries. This is because the parameter of the beta distribution in the posterior distribution increases in proportion to the size of the population, resulting in a smaller variance of the posterior distribution.
3. In addition, the credibility intervals for each county all cover the MLE.

Decision Theory

Imagine that you work for the NIH, and have been asked to advise the federal government in an effort to distribute research and treatment funds to US counties. For the sake of this exercise, imagine that you are performing this task at the end of the survey that ran from 1980 - 1984. Your resource allocation problems are:

1. Given a fixed total budget to be spent on treatment programs, distributed at a county level, how should treatment spending be distributed amongst the counties to best protect the American public?
2. Given a fixed total research budget, where should the NIH send targeted survey teams to maximize the amount of information they expect to gain from a follow-up survey?

These two scenarios are explained in detail below. (*Note: the following scenarios are stylized to make the subsequent exercise feasible.*)

Scenario 1: Treatment Allocation Suppose that, as part of the war on cancer, the federal government is considering allocating 140 million dollars per year to kidney cancer treatment programs. These funds will be allocated at the county level. Adjusted for inflation (adjusted to 1985), the average cost of kidney cancer treatment in fatal cases would have been roughly 35,000 dollars. The proposed total allocation is close to, but not quite, enough to cover the cost of each fatal kidney cancer case recorded in the 1980 - 1985 period (the total necessary cost would have been 163.9 million).

You are asked to give an initial suggestion for how to distribute these funds across the counties based on your analysis of the available data. To start, you pose the following optimization problem:

1. **Allocation:** You represent your allocation as a distribution vector with elements $\{a_j\}_{j=1}^{2997}$, where a_j represents the fraction of the total allocation you would allocate to county j.
2. **Loss (per county):** You aim to allocate enough funds to each county so that, the money sent to each county could cover the treatment cost for each fatal case in the county in the coming year. Let $D_j(t')$ represent the number of potentially fatal cases you hope to treat in the next year, t'. If $D_j(t') = d_j$ and you allocated $a_j \times 140 \times 10^6$ dollars to county j, then you compute your loss piecewise as:

$$l_j(d_j, a_j) = \max\{d_j \times 35 \times 10^3 - a_j \times 140 \times 10^6, 0\}$$

   Then, your loss is equal to the total shortfall in treatment funding you assigned to the j^{th} county. Since the federal funding is not enough to cover all counties, you do not add a penalty for over-allocating to a county (*this will be accounted for by trying to distribute limited funds to minimize the shortfall across all counties. If you'd like to try adding a cost for overbudgeting, add a separate loss term that is proportional to $a_j \times 140 \times 10^6 - d_j \times 35 \times 10^3$ when this difference is positive. To prioritize treatment over efficiency, you should weight this term by some factor less than one.*)
3. **Total Loss:** You define your total loss, given $\{D_j(t') = d_j, a_j\}_{j=1}^{2997}$, as the total loss over all counties:

$$l(d, a) = \sum_{j=1}^{2997} l_j(d_j, a_j)$$

4. **Bayes Risk:** You define your Bayesian risk, conditioned on the observed data, to be equal to your expected loss, given an allocation a, and given your past data:

$$\begin{aligned} r(a|D(t) = d) &= \mathbb{E}_{D(t')|D(t)=d}[l(D(t'), a)] \\ &= \sum_{j=1}^{2997} \mathbb{E}_{D_j(t')|D_j(t)=d_j}[l_j(D_j(t'), a_j)] \end{aligned}$$

   Note that each expectation runs over the *posterior predictive* distribution for the number of fatal kidney cancer cases in 1 new year, given the data collected over the past 5.
5. You aim to select the allocation that minimizes your Bayesian risk:

$$a_*(d) = \text{argmin}_a\{r(a|D(t) = d)\}$$

Our approach

Recall the *posterior predictive distribution* for a new set of m samples, given an observed set of n outcomes containing d deaths. Here we let D' the distribution over the set of m new samples. Then $\mathbb{P}[D' = d', D = d]$ follows the Beta–Binomial distribution, i.e.,

$$\mathbb{P}\left(D' = d' \mid D = d; n, \alpha, \beta\right) = \binom{m}{d'} \frac{\mathrm{B}\left(d' + [d + \alpha], m - d' + [(n - d) + \beta]\right)}{\mathrm{B}(d + \alpha, n - d + \beta)} \sim \text{Beta–Binomial}(m, d + \alpha, n - d + \beta).$$

[11]:
```
# write a code here to compute the Bayes risk as a function of:
# the prior parameters and observation record
# the allocation strategy
# you can either estimate the Bayes risk by sampling from the posterior predictive distribution
# you can define a quadrature rule that integrates over the beta-binomial CDF
def beta_binomial_interval(n, a, b, coverage=0.999):
    mean  = n*a/(a+b)
    var   = (n*a*b*(a+b+n)) / (((a+b)**2)*(a+b+1))
    std   = np.sqrt(var)
    pm    = 1/(np.sqrt(1-coverage))
    lower = np.floor(mean-pm*std)
    upper = np.ceil( mean+pm*std)
    return max(lower,0), min(upper,n)

def loss_j(d_j, a_j, fund=140e6):
    return np.maximum( d_j * 35e3 - a_j * fund , 0.0)

def expected_loss_j(a_j, m, d, n, alpha_, beta_, fund=140e6):

    # Our expectation is over the Posterior Predictive Distribution
    from scipy.stats import betabinom
    #---Option 1: Use just 0.999 of the PMF (via Chebyshev's inequality)
    lower, upper = beta_binomial_interval(m, d+alpha_, n-d+beta_)
    d_new = np.arange(lower, upper+1)
    post_pred_dist_ = betabinom.pmf(d_new, m, d+alpha_, n-d+beta_)
    #---Option 2: Use entire support of PMF (computationally expensive)
    # d_new = np.arange(0, m+1)
    # post_pred_dist_ = betabinom.pmf(d_new, m, d+alpha_, n-d+beta_)

    # Calculate losses --- loss_j( D_j(t'),a_j ) --- a m-dim vector over possible values of D_j(t')
    losses = loss_j(d_new, a_j, fund)

    # Calculate expected loss over Posterior Predictive Distribution $E_{D_j(t')|D_j(t)=d_j}
[loss_j(D_j(t'),a_j)]$
    return sum(losses*post_pred_dist_)

def bayes_risk(allocations, dataframe, alpha_prior, beta_prior, fund=140e6):
    ms   = dataframe['pop']
    ds   = dataframe['dc']
    ns   = dataframe['pop']
    risk = 0.0
    for a_j, m_j, d_j, n_j in zip(allocations, ms, ds, ns):
        risk += expected_loss_j(a_j, m_j, d_j, n_j, alpha_prior, beta_prior, fund*5)
    return risk / 5
```

This problem is too big to optimize easily (2997 variables! Noisy evaluation!). So, instead of trying to optimize it exactly, let's try some heuristic allocation rules, and select the one with the least risk.

Implement the heuristic allocation procedures suggested below then select the one with the smallest risk: 1. Set the allocation across the counties proportional to (the MAP estimators for the death rates) × (the respective county populations) 1. Set the allocation across the counties proportional to (the p% upper credible bound on the death rates) × (the respective

county populations). The p% upper credible bound on the rate in county j is the value $\theta_{uj}(p)$ such that $\Pr(\Theta_j(t) \leq \theta_{uj}(p)) = p$. If you wrote a code last week to produce credible intervals based on the CDF of the posterior, you've already written the code to evaluate this. Try varying p.

Hint: for an alternative heuristic, think about whether the loss function above is more directly related to the posterior over the unknown rates, or the posterior predictive distribution. If you'd rather try a heuristic based on the posterior predictive distribution, go ahead!

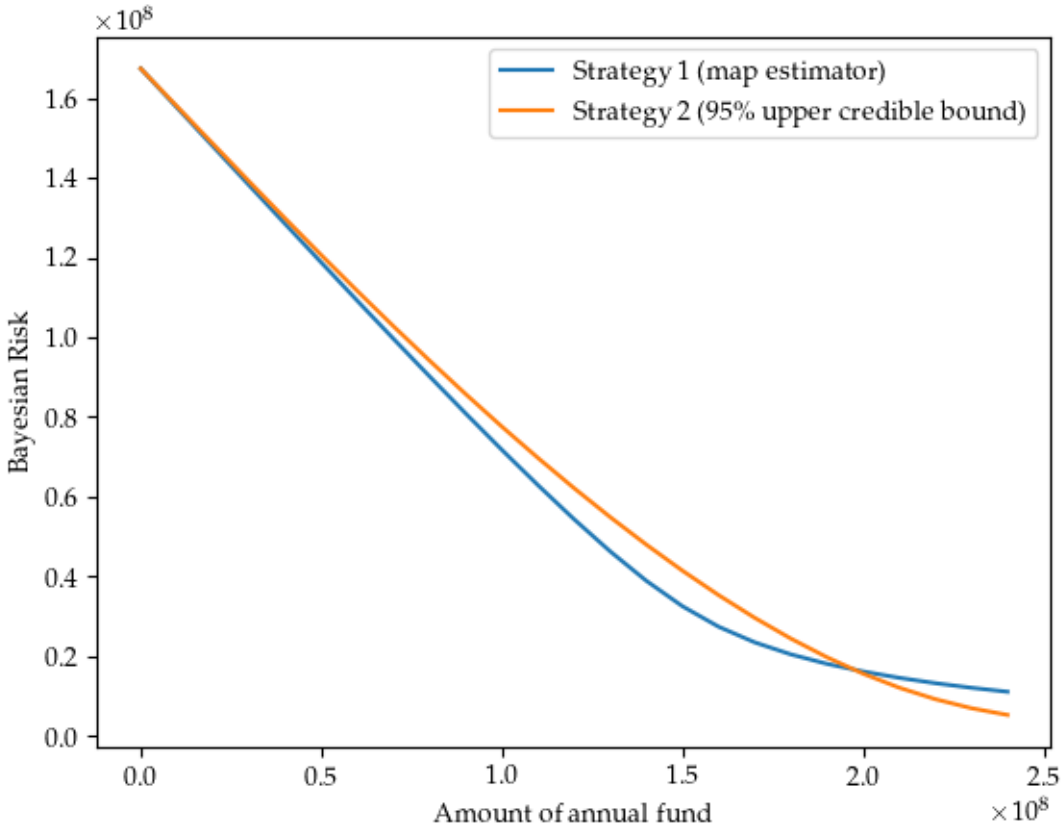
[12]:
```
# experiment with heuristics here, then select the best.

# ==================== #
# === HEURISTIC 1 === #
# ==================== #
MAPs          = kc['MAP'].to_numpy()
pops          = kc['pop'].to_numpy()
allocations_1 = (MAPs*pops) / sum(MAPs*pops)
risk_1        = bayes_risk(allocations_1, kc, alpha1, beta1)
print(f"Risk 1: {risk_1:0.3f}")

# ==================== #
# === HEURISTIC 2 === #
# ==================== #
uppers        = kc['upper'].to_numpy()
pops          = kc['pop'].to_numpy()
allocations_2 = (uppers*pops) / sum(uppers*pops)
risk_2        = bayes_risk(allocations_2, kc, alpha1, beta1)
print(f"Risk 2: {risk_2:0.3f}")
```

```
Risk 1: 38854860.485
Risk 2: 47944659.978
```

[13]:
```
# (Optional) try changing the total allocation. Make it much smaller than
# the value needed to cover all cases on average (say, 60 million)
# try making it much larger
# does your best allocation strategy change in a sensible way?
fund = np.arange(0, 25 * 10**7, 10**7)
loss_strategy1 = []
loss_strategy2 = []
for i in range(len(fund)):
    loss_strategy1.append(bayes_risk(allocations_1, kc, alpha1, beta1, fund[i]))
    loss_strategy2.append(bayes_risk(allocations_2, kc, alpha1, beta1, fund[i]))
plt.plot(fund, loss_strategy1, label = "Strategy 1 (map estimator)")
plt.plot(fund, loss_strategy2, label = 'Strategy 2 (95\% upper credible bound)')
plt.ylabel('Bayesian Risk')
plt.xlabel('Amount of annual fund')
plt.legend(loc='best')
plt.show()
```



[14]:
```
# (Optional) if you like thorny optimization problems, try passing this to
# an optimization scheme and see if you can solve for an allocation that beats
# your best heuristic. It may help to warm-up on a single state before trying the
# federal problem.
```

Scenario 2: Research Allocation

Suppose, instead, that the NIH intends to run a follow-up survey that can screen a sample of up to 25,000 individuals drawn from a given county. The NIH wants to pick a few candidate counties to survey. They hope to pick counties such that, the effort and money spent on the survey are expected to lead to the greatest gain in knowledge.

This is an example of a Bayesian optimal search or optimal experimental design question. Given a set of available measurements, which should I pick to maximize the amount I expect to learn from the measurement?

One way to frame these problems is to define a utility function, or information acquisition function, which measures how much we would learn after an observation. Then, we might try to maximize the expected utility over all possible observations given the choice of measurement. (*Near-to-optimal search trees are often constructed by greedy iteration of this procedure. The tree terminates when the unknown is determined within a chosen uncertainty. In cases when the measurements return answers drawn from a finite set, the sequence of answers to the sequence of questions form a string that points to a conclusion, thus constructs an encoding. These encodings are typically close to information theoretic optimal in the sense that they are maximally efficient (require as few questions as possible, in expectation, to reach a conclusion). They are a classical tool of statistical data compression.*)

The decision problem can be posed as follows:

1. You get to choose which county to survey. Your space of available actions is the set of counties.
2. Consider two different criteria for selecting a county:
    1. Pick the county where you have the most opportunity to learn from observations. That is, choose the county, j_* for which, you would expect the greatest reduction in posterior standard deviation in the death rate, if you had access to another 5 year record from that county among, **at most**, a sample population of 25,000 individuals drawn county j_*. (*This assumes that the survey data produced by screening can provide the equivalent information content as an additional five year sample, and that after converting to a count, is identically distributed to actual count data. In other*

words, the survey data can be used to produce predictions which are an accurate proxy for real events. This is to establish a simple heuristic. We'll see better motivated examples later in the class.) That is, select the county:

$$j_* = \text{argmax}_{j|n_j} \left\{ \mathbb{E}_{D_j(t')|d_j(t)} \left[\text{SD}[\Theta_j|d_j] - \text{SD}[\Theta_j|d_j(t), D_j(t')] \right] \right\}$$

As usual, other reasonable definitions are possible here. (*For example, we could pick the county where we expect the greatest reduction in variance, greatest reduction of coefficient of variation, or, to approach information-theoretic optimality, the greatest reduction in entropy.*)

2. Pick the county that could plausibly have the largest death rate. This choice could be motivated if our aim is to identify causes of kidney cancer mortality. In particular, let's say that a county could plausibly admit a death rate of, at most θ_{uj} if θ_{uj} acts as an upper credible bound on the unknown. That is, if $\Theta_j \leq \theta_{uj}$ with high probability. For a 90% upper credible interval, we need to find a bound θ_{uj} such that $\Pr(\Theta_j \leq \theta_{uj}|d_j(t)) \geq 0.90$. To find such a bound, use the analytic CDF of the posterior on each county, or estimate it by sampling from the posterior. You should have written the code to perform one of these two procedures last week.

[15]:
```
# write a code here to compute the expected reduction in posterior standard deviation in each county
# if we drew 5 new years worth of data for min{n_j,25000} individuals
# You can look up the variance in the beta-binomial, or check your work in HW 1
# Then, simply evaluate the expected reduction on every county, and pick the best
kc['theta_sd'] = np.sqrt(kc['alpha_post'] * kc['beta_post'] /
                    ((kc['alpha_post'] + kc['beta_post'])**2 * (kc['alpha_post'] + kc['beta_post'] +
1)))
kc['theta_sd_update'] = 0.0
for i in range(len(kc)):
    n = kc.loc[i, 'pop']
    alpha = kc.loc[i, 'alpha_post']
    beta = kc.loc[i, 'beta_post']
    n_survey = min(n, 25000)
    survey_positive = np.arange(0, n_survey)
    post_predictive = scipy.stats.betabinom.pmf(survey_positive, n_survey, alpha, beta)
    survey_post_alpha = alpha + survey_positive
    survey_post_beta = beta + n_survey - survey_positive
    theta_sd_update = np.sqrt((survey_post_alpha * survey_post_beta) /\
                        ((survey_post_alpha + survey_post_beta)**2 * (survey_post_alpha +
survey_post_beta + 1)))
    theta_sd_update = np.sum(post_predictive * theta_sd_update)
    kc.loc[i, 'theta_sd_update'] = theta_sd_update
kc['theta_sd_decrease'] = kc['theta_sd_update'] - kc['theta_sd']
list_survey1 = kc.nsmallest(10, 'theta_sd_decrease')
list_survey1
```

[15]:

	state	Location	dc	dc.2	pop	pop.2	death_rate	\
2128	OKLAHOMA	Haskell County, Oklahoma	5	1	24288	23418	0.000206	
2119	OKLAHOMA	Dewey County, Oklahoma	5	1	14299	13397	0.000350	
2135	OKLAHOMA	Kiowa County, Oklahoma	6	3	27698	24807	0.000217	
2657	TEXAS	Mitchell County, Texas	4	0	21276	20023	0.000188	
1512	MISSOURI	Maries County, Missouri	4	1	19261	19702	0.000208	
894	KANSAS	Harper County, Kansas	4	0	18141	17239	0.000220	
677	INDIANA	Crawford County, Indiana	4	2	24108	24659	0.000166	
2142	OKLAHOMA	Marshall County, Oklahoma	4	0	24149	23893	0.000166	
935	KANSAS	Rice County, Kansas	5	5	27283	25668	0.000183	
509	GEORGIA	White County, Georgia	4	0	24847	28098	0.000161	

	death_rate.2	MLE	mean	MAP	lower	upper	\
2128	0.000043	0.000206	0.000130	0.000110	0.000050	0.000246	
2119	0.000075	0.000350	0.000161	0.000137	0.000063	0.000306	
2135	0.000121	0.000217	0.000140	0.000122	0.000059	0.000256	
2657	0.000000	0.000188	0.000117	0.000096	0.000041	0.000232	
1512	0.000051	0.000208	0.000122	0.000100	0.000043	0.000242	

```
894       0.000000  0.000220  0.000125  0.000103  0.000044  0.000248
677       0.000081  0.000166  0.000110  0.000091  0.000039  0.000219
2142      0.000000  0.000166  0.000110  0.000091  0.000039  0.000219
935       0.000195  0.000183  0.000122  0.000104  0.000048  0.000232
509       0.000000  0.000161  0.000109  0.000089  0.000038  0.000216

      alpha_post     beta_post  theta_sd  theta_sd_update  theta_sd_decrease
2128    6.604398  50924.128568  0.000050         0.000041          -0.000009
2119    6.604398  40935.128568  0.000063         0.000054          -0.000009
2135    7.604398  54333.128568  0.000051         0.000042          -0.000009
2657    5.604398  47913.128568  0.000049         0.000041          -0.000009
1512    5.604398  45898.128568  0.000052         0.000043          -0.000009
894     5.604398  44778.128568  0.000053         0.000044          -0.000009
677     5.604398  50745.128568  0.000047         0.000038          -0.000008
2142    5.604398  50786.128568  0.000047         0.000038          -0.000008
935     6.604398  53919.128568  0.000048         0.000039          -0.000008
509     5.604398  51484.128568  0.000046         0.000038          -0.000008
```

[16]:
```
# write a code here to compute the 90% upper credible bound on the death rate in each county
# return the county with the largest 90% upper credible bound
kc['upper_90'] = stats.beta.ppf(0.9, kc['alpha_post'], kc['beta_post'])
list_survey2 = kc.nlargest(10, 'upper_90')
list_survey2
```

[16]:
```
         state                 Location  dc  dc.2    pop  pop.2  death_rate  \
2119  OKLAHOMA    Dewey County, Oklahoma   5     1  14299  13397    0.000350
1457  MISSOURI    Bates County, Missouri   9     2  37094  36524    0.000243
575   ILLINOIS     Clay County, Illinois   8     5  37117  35052    0.000216
2695     TEXAS    San Saba County, Texas   4     0  14253  13744    0.000281
2507     TEXAS      Bosque County, Texas   7     1  31982  34646    0.000219
844       IOWA        Union County, Iowa   7     2  32300  30043    0.000217
2135  OKLAHOMA    Kiowa County, Oklahoma   6     3  27698  24807    0.000217
2497     TEXAS      Austin County, Texas   8     4  40725  42896    0.000196
2128  OKLAHOMA  Haskell County, Oklahoma   5     1  24288  23418    0.000206
894     KANSAS     Harper County, Kansas   4     0  18141  17239    0.000220

      death_rate.2       MLE      mean       MAP     lower     upper  \
2119      0.000075  0.000350  0.000161  0.000137  0.000063  0.000306
1457      0.000055  0.000243  0.000166  0.000151  0.000082  0.000280
575       0.000143  0.000216  0.000151  0.000135  0.000071  0.000260
2695      0.000000  0.000281  0.000137  0.000113  0.000048  0.000272
2507      0.000029  0.000219  0.000147  0.000130  0.000066  0.000260
844       0.000067  0.000217  0.000146  0.000129  0.000065  0.000258
2135      0.000121  0.000217  0.000140  0.000122  0.000059  0.000256
2497      0.000093  0.000196  0.000143  0.000128  0.000067  0.000246
2128      0.000043  0.000206  0.000130  0.000110  0.000050  0.000246
894       0.000000  0.000220  0.000125  0.000103  0.000044  0.000248

      alpha_post     beta_post  theta_sd  theta_sd_update  theta_sd_decrease  \
2119    6.604398  40935.128568  0.000063         0.000054          -0.000009
1457   10.604398  63726.128568  0.000051         0.000043          -0.000008
575     9.604398  63750.128568  0.000049         0.000041          -0.000008
2695    5.604398  40890.128568  0.000058         0.000050          -0.000008
2507    8.604398  58616.128568  0.000050         0.000042          -0.000008
844     8.604398  58934.128568  0.000050         0.000042          -0.000008
2135    7.604398  54333.128568  0.000051         0.000042          -0.000009
2497    9.604398  67358.128568  0.000046         0.000039          -0.000007
2128    6.604398  50924.128568  0.000050         0.000041          -0.000009
894     5.604398  44778.128568  0.000053         0.000044          -0.000009
```

```
      upper_90
2119  0.000245
1457  0.000234
575   0.000215
2695  0.000214
2507  0.000213
844   0.000212
2135  0.000208
2497  0.000204
2128  0.000197
894   0.000196
```

Lab 3: Kalman Filtering and Online Bayesian Inference

Lab Date: Wednesday, February 12
Part I Due: Wednesday, February 19

Instructions

Work with your lab group to complete the following notebook. Your work on Part I will be reviewed by your peers in lab next week (Wednesday, February 19th). This is a multi-part lab. If you finish lab 3, you may go to Lab 4 to complete parts II and III. Lab 4 (Parts II and III) will be due in two weeks (Wednesday, February 26).

In the next two labs, you will:

1. Practice with the normal-normal model, [Part I]
2. *See how easily (some) conjugate models extend to thorny multi-dimensional problems, and, accordingly, why they are so popular as building blocks for large problems, [Part I and II]*
3. *See how the self-consistency of Bayesian inference allows for straightforward on-line inference (inference as the data is collected). [Parts II and III]*

Together, these demonstrate two of the common selling points for Bayesian methods: (1) they extend easily and consistently to multiple dimensions without requiring the user to make arbitrary choices to reduce dimension (e.g. picking a test statistic), (2) they are very natural to work with when tracking sequential data. It also demonstrates a subtler computational message. The larger our problems, the more useful analytic results are. As you read ahead, try to imagine what the procedure would look like if we replaced the simple algebra that follows from conjugacy with sampling, or, god forbid, numerical integration.

These learning goals will be presented to you through the Kalman filter, a fundamental tool for data assimilation in dynamical systems. The Kalman filter is an iterated application of a Gaussian likelihood, a Gaussian prior on the mean, and a linear update model to represent a dynamical system with a measurement protocol. Until this year, this approach was state-of-the-art for very large online prediction problems such as weather forecasting (see GraphCast, 2023 and GenCast, Nature, 2025).

The Kalman filter involves a lot of algebra. We've scaffolded that algebra, and the associated inferential code, for you. Focus on the statistical story represented in the algebra.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials. You may use the same procedure you used last week to work with that folder.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

[1]:
```
# load whatever packages you prefer here. We've added a reference list here
import numpy as np
import matplotlib.pyplot as plt

# My packages
import plotly.graph_objs as go
```

[PART I] Conjugate Gaussian Models

So far we've looked at one family of conjugate models, the (Beta, Binomial). In this lab we'll be working with the next natural conjugate family, the (Normal, Normal) family. For reference, see BDA Chapter 2.5, 3.2, and 3.3.

The Model:

Consider a joint model of the form: - **Prior:** $M \sim \mathcal{N}(\hat{m}, C_m)$ where $\hat{m} = \mathbb{E}[M]$ and $C_m = \text{Cov}[M]$ - **Likelihood:** $Y|M = m \sim \mathcal{N}(m, C_y)$ where C_y is the covariance in X given m.

This is a conditionally Gaussian model for Y, conditioned on an unknown mean, M. In this context, C_y is often interpreted as the covariance of some noise. Combining a Gaussian prior with a conditionally Gaussian likelihood produces

an elegant conjugate family. This family is very convenient for large problems, since it allows analytic posterior inference in arbitrary dimension. The inference proceeds via a sequence of linear algebraic updates, so is easy to apply as a step in larger inferential problems.

Solving for the Posterior (in 1 Dimension)

Suppose that $M \in \mathbb{R}^1$ and $X \in \mathbb{R}^1$: - **Prior:** $M \sim \mathcal{N}(\hat{m}, \sigma_m^2)$ - **Likelihood:** $Y|M = m \sim \mathcal{N}(m, \sigma_y^2)$

Q1.1: In the space below, write out the explicit form for the joint distribution at (m, y) by evaluating the product of the prior and likelihood. As usual, drop any normalizing constants (the evidence is implied). Based on your answer, what class of distributions must the posterior be a member of?

We have

$$p_M(m) := \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left\{-\frac{(m-\hat{m})^2}{2\sigma_m^2}\right\} \quad \text{and} \quad p_{Y|M=m}(y) := \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left\{-\frac{(y-m)^2}{2\sigma_y^2}\right\}$$

Therefore the joint distribution at (m, y) is given by

$$p_{M,Y}(m,y) = p_{Y|M=m}(y)p_M(m) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left\{-\frac{(y-m)^2}{2\sigma_y^2}\right\} \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left\{-\frac{(m-\hat{m})^2}{2\sigma_m^2}\right\}$$

$$p_{M,Y}(m,y) = \frac{1}{2\pi\sigma_m\sigma_y} \exp\left\{-\frac{(y-m)^2}{2\sigma_y^2} - \frac{(m-\hat{m})^2}{2\sigma_m^2}\right\}$$

which is proportional to the posterior distribution

$$p_{M|Y=y}(m) \propto \frac{1}{2\pi\sigma_m\sigma_y} \exp\left\{-\frac{(y-m)^2}{2\sigma_y^2} - \frac{(m-\hat{m})^2}{2\sigma_m^2}\right\}$$

Q1.2: Complete the square inside the exponential to solve for the mean and variance in $M|Y = y$. Simplify your answers in terms of the signal-to-noise ratio (SNR), σ_m^2/σ_y^2, and the precisions, σ_m^{-2}, σ_y^{-2} (inverse variances). As usual, it helps to remember that you can ignore any constant multiplicative factors. These appear as constant terms inside the exponential. Since we are looking for a distribution over M, think carefully about which terms are constant after observing $Y = y$. See equation (2.10) in BDA.

Note: it helps to start by assuming $\hat{m} = 0$, then noticing that, introducing $\hat{m} \neq 0$ is equivalent to translating the origin of your space to be located at $\hat{m}$.

We continue with our simplication of the posterior distribution

$$p_{M|Y=y}(m) \propto \exp\left\{-\frac{(y-m)^2}{2\sigma_y^2} - \frac{(m-\hat{m})^2}{2\sigma_m^2}\right\}$$

$$p_{M|Y=y}(m) \propto \exp\left\{-\frac{1}{2}\left[\left(\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}\right)m^2 - 2\left(\frac{y}{\sigma_y^2} + \frac{\hat{m}}{\sigma_m^2}\right)m\right]\right\} \quad \text{(only keep terms that depend on } m\text{)}$$

$$p_{M|Y=y}(m) \propto \exp\left\{-\frac{1}{2}\left(\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}\right)\left[m^2 - 2\left(\frac{\frac{y}{\sigma_y^2} + \frac{\hat{m}}{\sigma_m^2}}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}\right)m\right]\right\}$$

$$p_{M|Y=y}(m) \propto \exp\left\{-\frac{1}{2}\left(\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}\right)\left[\left[m - \left(\frac{\frac{y}{\sigma_y^2} + \frac{\hat{m}}{\sigma_m^2}}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}\right)\right]^2 - \underbrace{\left(\frac{\frac{y}{\sigma_y^2} + \frac{\hat{m}}{\sigma_m^2}}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}\right)^2}_{\text{independent of } m}\right]\right\}$$

$$p_{M|Y=y}(m) \propto \exp\left\{-\frac{1}{2}\left(\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}\right)\left[m - \left(\frac{\frac{y}{\sigma_y^2} + \frac{\hat{m}}{\sigma_m^2}}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}\right)\right]^2\right\}$$

to arrive at

$$p_{M|Y=y}(m) \propto \exp\left\{-\frac{(m-\hat{m}_{|y})^2}{2\sigma^2_{m|y}}\right\}$$

where

$$\hat{m}_{|y} = \mathbb{E}[M \mid Y = y] = \frac{\frac{\hat{m}}{\sigma_m^2} + \frac{y}{\sigma_y^2}}{\frac{1}{\sigma_m^2} + \frac{1}{\sigma_y^2}}$$
$$\sigma^2_{m|y} = \mathrm{Var}[M \mid Y = y] = \frac{1}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}$$

Q1.3: Express the posterior mean, $\hat{m}_{|y}$ as a convex combination of the form $(1 - s(\mathrm{SNR}))\hat{m} + s(\mathrm{SNR})y$ where $s(\cdot)$ is some monotonic, saturating function of the SNR. Is your answer sensible?

We have that $\mathrm{SNR} = \frac{\sigma_m^2}{\sigma_y^2}$. Therefore, we can rewrite the mean as

$$\hat{m}_{|y} = \frac{\frac{\hat{m}}{\sigma_m^2} + \frac{y}{\sigma_y^2}}{\frac{1}{\sigma_m^2} + \frac{1}{\sigma_y^2}} = \frac{\hat{m} + \frac{\sigma_m^2}{\sigma_y^2}y}{1 + \frac{\sigma_m^2}{\sigma_y^2}} = \frac{\hat{m} + (\mathrm{SNR})y}{1 + \mathrm{SNR}} = \left(1 - \frac{\mathrm{SNR}}{1+\mathrm{SNR}}\right)\hat{m} + \left(\frac{\mathrm{SNR}}{1+\mathrm{SNR}}\right)y$$

which implies that

$$\hat{m}_{|y} = \big(1 - s(\mathrm{SNR})\big)\hat{m} + s(\mathrm{SNR})y \quad \text{where} \quad s(\mathrm{SNR}) = \frac{\mathrm{SNR}}{1+\mathrm{SNR}}$$

This makes sense because if our sample y is... - **super** noisy (i.e., $\mathrm{SNR} \ll 1$), our posterior mean is approximately equal to our prior mean (i.e., $\hat{m}_{|y} \to \hat{m}$ as $\mathrm{SNR} \to 0$) because we aren't gaining information from the data so we default to the prior. - **not** noisy (i.e., $\mathrm{SNR} \gg 1$), our posterior mean is approximately equal to the sample y (i.e., $\hat{m}_{|y} \to y$ as $\mathrm{SNR} \to \infty$) because we are gaining a lot information from the data so we ignore our prior belief.

Q1.4: Express the posterior precision, $\sigma^{-2}_{m|y}$, in terms of the precisions σ_m^{-2} and σ_y^{-2}. Is your answer sensible?

$$\sigma^2_{m|y} = \frac{1}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}} \quad \Longrightarrow \quad \frac{1}{\sigma^2_{m|y}} = \frac{1}{\sigma_m^2} + \frac{1}{\sigma_y^2}$$

This makes sense because our posterior precision is the sum of our prior precision and likelihood precision.

Q1.5: Write out, a simple rule that explains how to update the prior (produce the posterior from the prior) after an observation.

$$\hat{m}_{|y} = \frac{\frac{\hat{m}}{\sigma_m^2} + \frac{y}{\sigma_y^2}}{\frac{1}{\sigma_m^2} + \frac{1}{\sigma_y^2}} = \frac{\hat{m} + \frac{\sigma_m^2}{\sigma_y^2}y}{1 + \frac{\sigma_m^2}{\sigma_y^2}} = \hat{m} + \left(\frac{\frac{\sigma_m^2}{\sigma_y^2}}{1 + \frac{\sigma_m^2}{\sigma_y^2}}\right)(y - \hat{m})$$
$$\hat{m}_{|y} = \hat{m} + \left(\frac{\sigma_m^2}{\sigma_m^2 + \sigma_y^2}\right)(y - \hat{m})$$
$$\sigma^2_{m|y} = \frac{1}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}$$

Solving for the Posterior (in d Dimensions)

Save deriving the update in multiple dimensions for the end of your lab work. It is worth doing once to justify the heavier formulas that come later. All the main ideas were captured in the 1D example. We will do parts of this in class on Thursday.

Q1.6: In the cell below, repeat the analysis you performed in the 1D case for the generic case. That is, for M and Y in $\mathbb{R}^d$. It helps to be strategic here. Follow the same sequence of steps you performed in the 1D case, dropping constants

everywhere you can. If you need help completing the square in multiple dimensions, come talk to us in office hours or raise a hand.

$$\hat{m}_{|y} = \frac{\frac{\hat{m}}{\sigma_m^2} + \frac{y}{\sigma_y^2}}{\frac{1}{\sigma_m^2} + \frac{1}{\sigma_y^2}} = \frac{\hat{m} + \frac{\sigma_m^2}{\sigma_y^2} y}{1 + \frac{\sigma_m^2}{\sigma_y^2}} = \hat{m} + \left(\frac{\frac{\sigma_m^2}{\sigma_y^2}}{1 + \frac{\sigma_m^2}{\sigma_y^2}} \right) (y - \hat{m})$$

$$\hat{m}_{|y} = \hat{m} + \left(\frac{\sigma_m^2}{\sigma_m^2 + \sigma_y^2} \right) (y - \hat{m})$$

$$\sigma_{m|y}^2 = \frac{1}{\frac{1}{\sigma_y^2} + \frac{1}{\sigma_m^2}}$$

Inspecting the Update Rule

In the previous cell you should have derived the following update:

$$M|Y = y \sim \mathcal{N}(\hat{m}_{|y}, C_{m|y}) \text{ where: } \begin{cases} \hat{m}_{|y} = \hat{m} + (I + (C_m C_y^{-1})^{-1})^{-1}(y - \hat{m}) \\ C_{m|y} = (C_m^{-1} + C_y^{-1})^{-1} \end{cases}$$

There are a couple things to note here. 1. The update is entirely expressed by updating the parameters (here, the mean and the covariance). Here we see the power of conjugacy on full display. Moreover, if we observe multiple samples observations sequentially, then we can just update the mean and covariance recursively. 1. The mean update is sensible. As before, it updates the mean by shifting it towards the observation. The product $C_m C_y^{-1}$ is playing the role of the SNR, and the prefactor is the matrix equivalent to $s(\text{SNR}) = [1 + \text{SNR}^{-1}]^{-1}$. If the covariance in the measurement is much larger than the covariance in the noise, then this term is small, so $(I + C_y C_m^{-1})^{-1} \approx I$, so $\hat{m}_{|y} \approx \hat{m}$. 1. The covariance is update is sensible. As before, the precisions add after the observation.

These sorts of updates can be easily extended to similar scenarios. For example, since all linear transformations of normal distributions are normal, we can derive a similar rule for the conditional distribution of some unknown mean M when the observable Y is expressed as a linear transformation of M, plus Gaussian noise. That is $y = Hm + \epsilon$ where $\epsilon \sim \mathcal{N}(0, C_y)$ represents the noise, H represents the mapping responsible for converting state to measurement, and where we have used lower case letters for y and m to distinguish vectors from matrices. This is a standard convention at this stage. In general, we will continue to use hats for fixed quantities (expected values), and capital letters for matrices.

Introducing a linear mapping between the unknown m and the observation y is useful since, in most high-dimensional settings, we have (far) fewer measurements than unknowns. We will assume that H is $n \times d$ where, for, most practical purposes, $n \ll d$.

Introducing the linear mapping only decorates the original expression. Read through the expression carefully to track the change in dimensions. Each time we switch from n to d we are moving from the measurement domain to the state space. Each time we map from d to n we are moving from the state space to the measurement space.

$$m|y \sim \mathcal{N}(\hat{m}_{|y}, C_{m|y}) \text{ where: } \begin{cases} \hat{m}_{|y} = \hat{m} + (C_m^{-1} + H^\intercal C_y^{-1} H)^{-1} H^\intercal C_y^{-1}(y - H\hat{m}) \\ C_{m|y} = (C_m^{-1} + H^\intercal C_y^{-1} H)^{-1} \end{cases} \tag{56}$$

For the next part, it will also be convenient to understand how the posterior distribution updates if we replace m with $Am + \zeta$ where $\zeta \sim \mathcal{N}(0, C_\zeta)$, and $A \in \mathbb{R}^{d \times d}$ is invertible. This represents a noisy linear transformation of m.

Q 1.7: In the space below, find the distribution for $Am + \zeta$. Remember, any linear transformation of a Gaussian vector produces a Gaussian vector, and any sum of Gaussian vectors is a Gaussian vector. Also remember that expectation is linear, so commutes with linear transformations, and the covariance of a sum of independent vectors is the sum of their covariances.

Given that $m \sim \mathcal{N}(\hat{m}, C_m)$, we have

$$\mathbb{E}[Am + \zeta] = A\mathbb{E}[m] + \mathbb{E}[\zeta] = A\hat{m} \quad \text{and}$$
$$\text{Cov}[Am + \zeta] = \text{Cov}[Am] + \text{Cov}[\zeta] = A\text{Cov}[m]A^\intercal + C_\zeta = AC_m A^\intercal + C_\zeta$$

Therefore, we have

$$\boxed{Am + \zeta \sim \mathcal{N}(A\hat{m}, AC_m A^\intercal + C_\zeta)}$$

Visualization

Q 1.8: In the cell below, write a code that can visualize any 2-D Gaussian density function as a surface. Use color to represent the height of the surface (*and, optionally, add contour lines*). Note, your function should work for singular Gaussian densities. Since we don't care about normalization, your function does not need to plot a properly normalized Gaussian density.

[2]:
```
# function to display 2D Gaussian
def plot_2d_gaussian(mu, Sigma, title=None, grid_points=500, xlim=(-4, 4), ylim=(-4, 4),
add_contours=True):

    # Mesh grid
    x = np.linspace(xlim[0], xlim[1], grid_points)
    y = np.linspace(ylim[0], ylim[1], grid_points)
    X, Y = np.meshgrid(x, y)

    # Compute the pseudo-inverse of Sigma (works even if Sigma is singular)
    Sigma_inv = np.linalg.pinv(Sigma)

    # Build a position array with shape (grid_points, grid_points, 2)
    pos = np.empty(X.shape + (2,))
    pos[:, :, 0] = X
    pos[:, :, 1] = Y
    diff = pos - mu  # difference from the mean

    # Compute the quadratic form for the exponent
    exponent = -0.5 * np.einsum('...i,ij,...j', diff, Sigma_inv, diff)

    # Compute the unnormalized density
    Z = np.exp(exponent)

    # Configure contour options
    if add_contours:
        contours = dict(
            z = dict(show=True, usecolormap=True, highlightcolor="limegreen", project_z=True)
        )
    else:
        contours = dict(z = dict(show=False))

    # Create the interactive 3D surface plot
    fig = go.Figure(data=[go.Surface(x=X, y=Y, z=Z,
                                     colorscale='Viridis',
                                     contours=contours)],
                    )
    custom_title = [f"$",
                    f"p(X_1, X_2 ; \mu, \Sigma), \quad" ,
                    f"\mu = \\begin{{bmatrix}}{mu[0]:0.3f} \\\ {mu[1]:0.3f} \end{{bmatrix}}, \quad" ,
                    f"\Sigma = \\begin{{bmatrix}} {Sigma[0,0]:0.3f} & {Sigma[0,1]:0.3f} \\\
{Sigma[1,0]:0.3f} & {Sigma[1,1]:0.3f} \end{{bmatrix}}",
                f"$"]
    if title:
        custom_title.insert(1, r"\text{%s}: \quad " % title)
    fig.update_layout(
        title=''.join(custom_title),
        scene=dict(
            xaxis=dict(title="X_1"),
            yaxis=dict(title="X_2"),
            zaxis=dict(title="Density")
        )
    )
```

```
    fig.show()

# Example usage:
mu    = np.array([1, 0])
Sigma = np.array([[4, 2],
                  [1, 4]])
plot_2d_gaussian(mu, Sigma, title='2D Gaussian Distribution', grid_points=200, xlim=(-10, 10),
ylim=(-10, 10))
```

Q 1.9: Then create a visual showing: 1. A standard normal prior density with $\hat{m} = 0$ and $C_m = I$. 1. The likelihood function when $H = [1, -2]$ and $y = 0.5$. You may assume the noise variance, $C_y = \sigma_y^2 = 0.25$. 1. The resulting posterior distribution.

[3]:
```
def gaussian_posterior(H, m_hat, C_m, y, C_y):
    # Compute the posterior covariance
    C_m_inv = np.linalg.inv(C_m)
    C_y_inv = np.linalg.inv(C_y)
    C_m_given_y = np.linalg.inv(C_m_inv + H.T @ C_y_inv @ H)
    # Compute the posterior mean
    m_hat_given_y = m_hat + C_m_given_y @ H.T @ C_y_inv @ (y - H @ m_hat)
    return m_hat_given_y, C_m_given_y
```

[4]:
```
# visualizate prior, likelihood, posterior
prior_mean = np.array([0, 0])
prior_cov = np.array([[1, 0], [0, 1]])
plot_2d_gaussian(prior_mean, prior_cov)
H = np.array([[1, -2]])
y = 0.5
likelihood_cov = np.array([[0.25]])
post_mean, post_cov = gaussian_posterior(H, prior_mean[:, np.newaxis], prior_cov, y, likelihood_cov)
print(post_mean)
print(post_cov)
plot_2d_gaussian(post_mean.squeeze(), post_cov)
```

```
[[ 0.0952381 ]
 [-0.19047619]]
[[0.80952381 0.38095238]
 [0.38095238 0.23809524]]
```

Q 1.10: Set the posterior produced from (Q 1.9), and use it as your prior. Make plots showing: 1. The likelihood function when $H = [1, -1.6]$ and $y = 0.4$. You may assume the noise variance, $C_y = \sigma_y^2 = 0.25$. 1. The resulting posterior.

[5]:
```
# visualize likelihood and posterior
H = np.array([[1, -1.6]])
y = 0.4
post_mean_2, post_cov_2 = gaussian_posterior(H, post_mean, post_cov, y, likelihood_cov)
print(post_mean_2)
print(post_cov_2)
plot_2d_gaussian(post_mean_2.squeeze(), post_cov_2)
```

```
[[ 0.0952381 ]
 [-0.19047619]]
[[0.72063492 0.38095238]
 [0.38095238 0.23809524]]
```

Q 1.11: In the space below, comment on how the posterior updated in each step. Are the resulting updates sensible?
The first step updates the covariance, the second step did not change all that much.

Q 1.12: Did the second measurement effectively reduce the uncertainty in the posterior? Why or why not? What do you notice about the direction in the state space where uncertainty is constrained when using a one dimensional measurement (*how does it relate to the rows of H?*). Propose a new 1×2 measurement matrix H that you think will lead to the greatest

reduction in posterior uncertainty. Explain how you selected your measurement matrix H and in what sense you expect it to maximally reduce uncertainty.

Not particularly reduced uncertainty. Higher values of H should reduce uncertainty, as they'll be multiplicative factors in the inverse of the posterior covariance matrix.

Q 1.13: Use the posterior from (Q 1.12) as your prior and make plots showing: 1. The likelihood function using your new H and a hypothesize measurement that is consistent with past observations (*you may calculate the prior mean $\hat{m}$, and use $H\hat{m}$ so that our posterior mean stays still.*). Keep the noise variance, $C_y = \sigma_y^2 = 0.25$. 1. The new posterior.

Comment on whether your choice of measurement effectively reduced uncertainty.

[6]:
```
# visualize likelihood and posterior
prior_mean = np.array([0, 0])
prior_cov = np.array([[1, 0], [0, 1]])
plot_2d_gaussian(prior_mean, prior_cov)
H = np.array([[-10, 10]])
y = 0.5
likelihood_cov = np.array([[0.25]])
post_mean, post_cov = gaussian_posterior(H, prior_mean[:, np.newaxis], prior_cov, y, likelihood_cov)
print(post_mean)
print(post_cov)
plot_2d_gaussian(post_mean.squeeze(), post_cov)
```

```
[[-0.02496879]
 [ 0.02496879]]
[[0.50062422 0.49937578]
 [0.49937578 0.50062422]]
```

Congratulations! You've reached the end of part I. You should have finished up to here by Wednesday, February

[PART II - Preview] Kalman Filters

Consider the following problem:

You are a meteorologist insterested in predicting the weather. You have access to a large, varied, set of real-time data sources. These provide a constant stream of measurements. Even so, the real weather system is unimaginably high-dimensional. So, even with input from satellites, and weather stations, and weather bouys, and balloon launches, and everything else NOA attempts, omniscience is still far off. Given your data, the true state of the weather at any time is unknown.

Worse still, the weather is constantly changing.

These two ingredients pose a data assimilation problem. In data assimilation, we make measurements in real time of an unknown, *potentially changing*, quantity, usually called the state vector. Unlike the examples we've considered before, asymptotic consistency guarantees don't apply since the unknown is constantly moving. Our aim is to keep "assimilating" new data into a model for the motion of the unknown in order to keep a good guess at the range of possible states given the observed sequence of measurements. In other words, we want to sequentially update a posterior after each measurement.

Data assimilation has two key components: 1. A dynamical systems model that specifies how we would expect the state vector to change over time if we knew it's exact initial conditions at some point in time. This system may be deterministic, or stochastic. In the weather analogy, this is a massive PDE simulator, built on a combination of fundamental physics (i.e. Navier-Stokes) and geochemistry (water cycles, etc.). In all that follows the true, but unknown, state of the system will be represented as $x(t)$ (if in continuous time), or x_t in discrete time. We will treat this as random quantity, but use lower case letters to maintain a clear distinction between matrices and vectors. 1. A statistical model that specifies a prior distribution over the unknowns before any measurement, and an error model relating the true state of the system to the observed measurements. In all that follows we will let y_t represent the measurements made at time t. Typically, y_t is a low dimensional object relative to x_t. Hence, we are in the standard "small sample" statistics regime of Bayesian statistics where we aim to repeatedly solve an ill-posed problem.

Then, we update a posterior distribution after each new measurement, using the posterior after the previous measurement as our prior, and using the dynamical systems model and measurement model to specify the likelihood.

A Kalman filter linearizes the dynamical system and measurement model (assumes small time steps and concentrated posteriors), and uses a (Normal, Normal) conjugate model. In this setting, the posterior over the unknown state remains Gaussian at all times. Since a Gaussian is fully specified by its mean and covariance, applying data assimilation reduces to running an algebraic recursion driven by a constant stream of input measurements.

An overview.

The idea in a Kalman filter is to continuously update a Gaussian approximation to the posterior distribution of the unknown state given a series of sequential measurements. In what follows, let x_t denote the (unknown) state at time t, and y_t denote the corresponding observation. Between each observations, x_t is updated by a noisy dynamical system. As long as the time step is small, the mapping imposed by the system is approximately linear. Similarly, as long as we can collect enough measurements to keep the posterior concentrated, then the action of any, potentially nonlinear, set of measurement functions, are approximately linear. Then, every update is either, a linear transformation plus Gaussian noise, or a posterior inference step based on a linear measurement. Each of these steps preserve the family of Gaussian distributions, so the entire pipeline can be represented by deriving a recursion on the mean and covariance of the state vector $\{\hat{x}_t, P_t\}_{t=1}^{\cdots}$. Since we will condition on all past observations to constrain our inference, we will write, $\hat{x}_{t|t-1}, P_{t|t-1}$ to denote the posterior mean and covariance over the state vector after measurement y_t.

Then, after initializing with a Gaussian prior mean $\hat{x}_0$ and covariance P_0, we will run a recursive process of the form:

1. Push the posterior distribution left after the observation at time t forward to a prior for time $t+1$ by running the dynamics (apply the linear update equation you derived in Q 1.7).
2. Make an observation, y_t
3. Condition on the observation, using the new prior, to derive a new posterior via equation (1).
4. Iterate.

Link to Lab 4:

To go to lab four click this link: Lab 4

Note: Lab 4 is a work in progress. Most of the exposition and code is available for you. Activity prompts will post by next Monday.

Lab 4: Kalman Filtering and Online Bayesian Inference

Lab Date: Wednesday, February 19
Parts II and III Due: Wednesday, February 26

Instructions

Work with your lab group to complete the following notebook. Your work will be reviewed by your peers in lab next week (Wednesday, February 26th). This is the second half of the lab you started last week. To access the first half, click this link: Lab 3

In this lab, you will:

1. *Practice with our next family of conjugate models: the normal-normal model, [Part I]*
2. See how easily (some) conjugate models extend to thorny multi-dimensional problems, and thus, why they are so popular as building blocks for large problems, [Part I and II]
3. See how the self-consistency of Bayesian inference allows for straightforward on-line inference (inference as the data is collected). [Parts II and III]

Together, these demonstrate two of the common selling points for Bayesian methods: (1) they extend easily and consistently to multiple dimensions without requiring the user to make arbitrary choices to reduce dimension (e.g. picking a test statistic), (2) they are very natural to work with when tracking sequential data. It also demonstrates a subtler computational message. The larger our problems, the more useful analytic results are. As you read ahead, try to imagine what the procedure would look like if we replaced the simple algebra that follows from conjugacy with sampling, or, god forbid, numerical integration.

These learning goals will be presented to you through the Kalman filter, a fundamental tool for data assimilation in dynamical systems. The Kalman filter is an iterated application of a Gaussian likelihood, a Gaussian prior on the mean, and a linear update model to represent a dynamical system with a measurement protocol. Until this year, this approach was state-of-the-art for very large online prediction problems such as weather forecasting (see GraphCast, 2023 and GenCast, Nature, 2025).

The Kalman filter involves *a lot* of algebra. We've scaffolded that algebra, and the associated inferential code, for you. Focus on the statistical story represented in the algebra.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

[1]:
```
# load whatever packages you prefer here. We've added a reference list here
import numpy as np
import scipy as scipy
from scipy import linalg
# import matplotlib.pyplot as plt

# My plotting settings
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'''
\usepackage{newpxtext}
\usepackage[vvarbb]{newpxmath}
\usepackage{amsmath}
\DeclareMathOperator*{\E}{\mathbb{E}}
\DeclareMathOperator*{\Var}{Var}
'''
rcParams["font.family"] = "Palatino"
rcParams['figure.dpi'] = 300
```

[PART I - Reference] Conjugate Gaussian Models

This lab builds directly on the machinery you built in Lab 3. To access Lab 3 click this link: Lab 3.

You are welcome to copy other components of your group's work on lab 3 here as reference.

[PART II] Kalman Filters

Consider the following problem:

You are a meteorologist insterested in predicting the weather. You have access to a large, varied, set of real-time data sources. These provide a constant stream of measurements. Even so, the real weather system is unimaginably high-dimensional. So, even with input from satellites, and weather stations, and weather bouys, and balloon launches, and everything else NOA attempts, we are still far from omniscient. Given your data, the true state of the weather at any time is unknown.

Worse still, the weather is constantly changing.

These two ingredients pose a data assimilation problem. In data assimilation, we make measurements in real time of an unknown, *potentially changing*, quantity, usually called the state vector. Unlike the examples we've considered before, asymptotic consistency guarantees don't apply since the unknown is constantly moving. Our aim is to keep "assimilating" new data into a model for the motion of the unknown in order to keep a good guess at the range of possible states given the observed sequence of measurements. In other words, we want to sequentially update a posterior after each measurement.

Data assimilation has two key components: 1. A dynamical systems model that specifies how we would expect the state vector to change over time if we knew it's exact initial conditions at some point in time. This system may be deterministic, or stochastic. In the weather analogy, this is a massive PDE simulator, built on a combination of fundamental physics (i.e. Navier-Stokes) and geochemistry (water cycles, etc.). In all that follows the true, but unknown, state of the system will be represented as $x(t)$ (if in continuous time), or x_t in discrete time. We will treat this as random quantity, but use lower case letters to maintain a clear distinction between matrices and vectors. 1. A statistical model that specifies a prior distribution over the unknowns before any measurement, and an error model relating the true state of the system to the observed measurements. In all that follows we will let y_t represent the measurements made at time t. Typically, y_t is a low dimensional object relative to x_t. Hence, we are in the standard "small sample" regime of Bayesian statistics where we aim to repeatedly solve an ill-posed problem.

Given a model, we update a posterior distribution after each new measurement, using the posterior after the previous measurement as our prior, and using the dynamical systems model and measurement model to specify the likelihood.

A Kalman filter linearizes the dynamical system and measurement model (assumes small time steps and concentrated posteriors), and uses a (Normal, Normal) conjugate model. In this setting, the posterior over the unknown state remains Gaussian at all times. Since a Gaussian is fully specified by its mean and covariance, applying data assimilation reduces to running an algebraic recursion driven by a constant stream of input measurements.

An overview.

The idea in a Kalman filter is to continuously update a Gaussian approximation to the posterior distribution of the unknown state given a series of sequential measurements. In what follows, let x_t denote the (unknown) state at time t, and y_t denote the corresponding observation. Between each observations, x_t is updated by a noisy dynamical system. As long as the time step is small, the mapping imposed by the system is approximately linear. Similarly, as long as we can collect enough measurements to keep the posterior concentrated, then the action of any, potentially nonlinear, set of measurement functions, are approximately linear. Then, every update is either, a linear transformation plus Gaussian noise, or a posterior inference step based on a linear measurement. Each of these steps preserve the family of Gaussian distributions, so the entire pipeline can be represented by deriving a recursion on the mean and covariance of the state vector $\{\hat{x}_t, P_t\}_{t=1}^{\dots}$. Since we will condition on all past observations to constrain our inference, we will write, $\hat{x}_{t|t-1}, P_{t|t-1}$ to denote the posterior mean and covariance over the state vector after measurement y_t.

Then, after initializing with a Gaussian prior mean $\hat{x}_0$ and covariance P_0, we will run a recursive process of the form:

1. Push the posterior distribution left after the observation at time t forward to a prior for time $t+1$ by running the dynamics (apply the linear update equation you derived in Lab 3 [PART I] Q 1.7).
2. Make an observation, y_t
3. Condition on the observation, using the new prior, to derive a new posterior via Lab 3 [PART I] equation (1).
4. Iterate.

All of the algebra below is just a clever implementation of this idea.

Example: Simple Harmonic Oscillators

A simple harmonic oscillator obeys the dynamical system:

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \tag{57}$$

This is an interesting first test case since it is, in a sense, as about as easy a dynamic to track as possible without resorting to convergent models that stop changing after some time.

Harmonic oscillators are an example of a special class of dynamical systems called Hamiltonian systems. These have the remarkable property that they conserve volume. That is, if you initialize the dynamical system with an infinite collection of initial conditions on the boundary of some closed region, then the volume enclosed as the boundary evolves forward under the dynamic will remain the same. Systems with this property preserve the entropy of any continuous distribution of states evolving under the dynamic. We'll see why this is useful in a moment.

Step 1: Define the State-Space Model

A state-space model consists of a system equation and a measurement equation. The system equation describes how the true state evolves over time, while the measurement equation describes how observations are obtained.

The system equation is given by:

$$x_{t+1} = A_t x_t + w_t$$

where x_t is the state vector at time t, A_t is the transition matrix, and w_t is process noise, assumed to be Gaussian with covariance Q. Note that A_t could depend on time. This happens for nonlinear dynamics. Then, it is standard to set A_t to the Jacobian of the dynamical system evaluated at $\hat{x}_t$.

If the dynamical system is Hamiltonian, then the mapping A_t must also be volume preserving. It follows that A_t is a *unitary* transformation. This means that, in a sense, multiplying by A_t will preserve the covariance of the posterior distribution. In particular, the determinant of the covariance of a random vector is preserved, so the transformation preserves the geometric mean of the principal variances (eigenvalues of the covariance). This means that, in a sense, under the dynamic alone, the distribution of possible state vectors will not spread out too quickly. However, it does not guarantee that the dynamic will not stretch or compress the distribution along a particular direction. It only guarantees that the geometric average of the stretching and compression factors balance out to 1.

The measurement equation is given by:

$$y_t = H_t x_t + v_t$$

where y_t is the observed measurement, H_t is the observation matrix, and v_t is measurement noise, assumed to be Gaussian with covariance R. As before, the measurement matrix could, in principle, change in time if we are linearizing a nonlinear measurement function. For now, we will assume H_t is fixed.

[2]:
```
# # Define the state-space model
# dt = 0.1  # Time step
# omega = 1.0  # Natural frequency
# A_euler = np.array([[0, dt], [-omega**2 * dt, 0]])  # State transition matrix using forward Euler
# A = scipy.linalg.expm(A_euler)
# H = np.array([[1, 0]])  # Measurement matrix
# Q = np.eye(2) * 0.005 * dt  # Process noise covariance
# R = np.eye(1) * 0.5  # Measurement noise covariance

# # Initialize state estimate and covariance
# x_est = np.array([[1], [0]])  # Initial position and velocity
# P = np.eye(2) * 1.0  # Initial uncertainty

# # Simulate noisy measurements
# periods = 6
# time_steps = round(periods*2*np.pi/dt)
# true_states = []
# measurements = []
# for _ in range(time_steps):
#     x_est = A @ x_est + np.random.multivariate_normal(mean=[0, 0], cov=Q).reshape(-1, 1) #
Propagate state forward
#     measurement = H @ x_est + np.random.normal(0, np.sqrt(R[0, 0]))  # Add measurement noise
#     true_states.append(x_est.flatten())
```

```
#     measurements.append(measurement.flatten())
```

Step 2: Compute the Prediction Step

At each time step, we predict the next state using the system dynamics.

1. **State Prediction:** Set:
$$\hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1}$$
This equation propagates the previous estimate through the state model to predict the next state.
2. **Covariance Prediction:**
$$P_{t|t-1} = AP_{t-1|t-1}A^T + Q$$
Here, we compute the predicted covariance, which accounts for the uncertainty in the process model.

Step 3: Compute the Kalman Gain

The Kalman gain determines how much weight is given to the new measurements:

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}$$

This equation balances trust in the prediction versus the new observation.

Step 4: Compute the Update Step

When a new measurement y_t is available, we update our state estimate:

1. **State Update:**
$$x_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - H\hat{x}_{t|t-1})$$
This correction ensures that the estimate accounts for the measurement innovation.
2. **Covariance Update:**
$$P_{t|t} = (I - K_tH)P_{t|t-1}$$
Updating the covariance reduces uncertainty after incorporating the new measurement.

Step 5: Implement the Kalman Filter

Now that we have derived the equations, we implement the Kalman Filter iteratively over multiple time steps. At each step, we: 1. Use the state transition model to predict the next state. 2. Compute the Kalman Gain to determine the weight of the new measurement. 3. Update the state estimate based on the new measurement. 4. Update the covariance to reflect new uncertainty levels. 5. Repeat for all incoming measurements.

[3]:
```
# Implement the Kalman Filter
# filtered_states = []
# for z in measurements:
#     # Prediction Step
#     x_pred = A @ x_est
#     P_pred = A @ P @ A.T + Q

#     # Compute Kalman Gain
#     K = P_pred @ H.T @ np.linalg.inv(H @ P_pred @ H.T + R)

#     # Update Step
#     x_est = x_pred + K @ (z - H @ x_pred)
#     P = (np.eye(2) - K @ H) @ P_pred

#     filtered_states.append(x_est.flatten())
```

Step 6: Visualizing the Results

To understand how well the Kalman Filter performs, we plot the following elements: 1. **True State**: The actual underlying system state. 2. **Filtered State**: The estimated state obtained from the Kalman Filter. 3. **Measurements**: The noisy observations received over time.

The plot helps visualize how closely the Kalman Filter is able to track the true state despite noisy measurements. We expect the filtered state to be smoother than the measurements and follow the trend of the true state closely.

[4]:
```
# # Plot the results
# plt.plot([s[0] for s in true_states], label='True State')
# plt.plot([s[0] for s in filtered_states], label='Filtered State')
# plt.scatter(range(time_steps), [m[0] for m in measurements], label='Measurements', color='red',
s=1)
# plt.legend()
# plt.show()
```

Experiments To Run

Using the code provided above, run the following experiments:

Q 2.1: Set the process noise to `0.005 * dt`, the measurement noise to `0.5`, and the time step `dt` to `0.05`. Run the Kalman filter a couple times to get a sense for its general behavior. Save a representative output plot below.

Comment on how many cycles it takes before the filtered state approximation appears to converge, how smooth the filtered sequence of states are (how reactive are they to new measurements), and the accuracy of the predicted state. Do the errors between the true and filtered state decrease over time? Does it vanish?

It takes about 1 period to stabilize fully, but it stabilizes quite a bit in the first half-period. The filtered sequence appears to converge to being as smooth as the true state. The errors decrease overtime but never fully vanish.

[5]:
```
# Insert demo for Q 2.1 here...
def kalman_filter_demo(
        dt = 0.05,  # Time step
        process_noise = 0.005,
        measurement_noise = 0.5,
        periods = 6
    ):

    # Define the state-space model
    omega = 1.0  # Natural frequency
    A_euler = np.array([[0, dt], [-omega**2 * dt, 0]])  # State transition matrix using forward Euler
    A = scipy.linalg.expm(A_euler)
    H = np.array([[1, 0]])  # Measurement matrix
    Q = np.eye(2) * process_noise * dt # Process noise covariance
    R = np.eye(1) * measurement_noise  # Measurement noise covariance

    # Initialize state estimate and covariance
    x_est = np.array([[1], [0]])  # Initial position and velocity
    P = np.eye(2) * 1.0  # Initial uncertainty

    # Simulate noisy measurements
    time_steps = round(periods*2*np.pi/dt)
    true_states = []
    measurements = []
    for _ in range(time_steps):
        x_est = A @ x_est + np.random.multivariate_normal(mean=[0, 0], cov=Q).reshape(-1, 1) #
Propagate state forward
        measurement = H @ x_est + np.random.normal(0, np.sqrt(R[0, 0]))  # Add measurement noise
        true_states.append(x_est.flatten())
        measurements.append(measurement.flatten())

    # Implement the Kalman Filter
```

```
    filtered_states = []
    for z in measurements:

        # Prediction Step
        x_pred = A @ x_est
        P_pred = A @ P @ A.T + Q

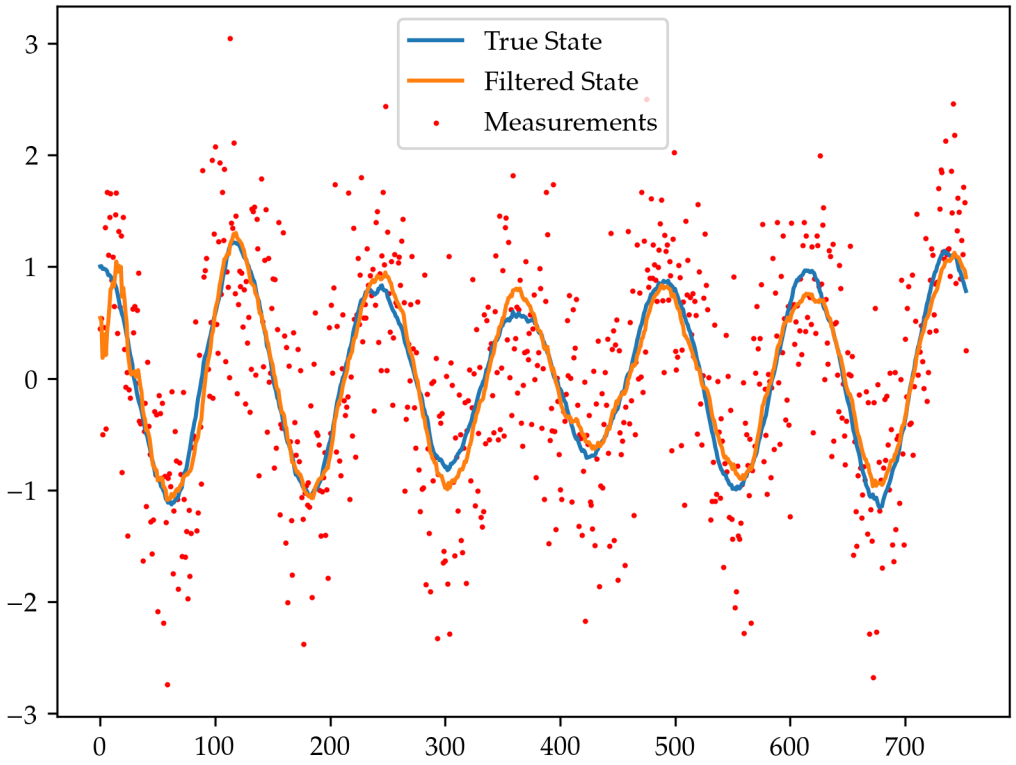
        # Compute Kalman Gain
        K = P_pred @ H.T @ np.linalg.inv(H @ P_pred @ H.T + R)

        # Update Step
        x_est = x_pred + K @ (z - H @ x_pred)
        P = (np.eye(2) - K @ H) @ P_pred

        filtered_states.append(x_est.flatten())

    # Plot the results
    plt.plot([s[0] for s in true_states], label='True State')
    plt.plot([s[0] for s in filtered_states], label='Filtered State')
    plt.scatter(range(time_steps), [m[0] for m in measurements], label='Measurements', color='red',
s=1)
    plt.legend()
    param_string = (
        f"dt = {dt},  "
        f"process_noise = {process_noise},  "
        f"measurement_noise = {measurement_noise},  "
        f"periods = {periods}"
    )
    plt.figtext(0.5, -0.001, param_string, ha='center', fontsize=10)
    plt.show()

kalman_filter_demo(dt=0.05,process_noise=0.005,measurement_noise=0.5)
```



dt = 0.05, process_noise = 0.005, measurement_noise = 0.5, periods = 6

Q 2.2: Let's try to understand the role of the process noise. Try: 1. Setting the process noise to zero 1. Setting the process noise large (large enough to see distinctly new behavior)

In each case, comment on whether the filtered values converge to the true state. Comment on whether this is even possible, and if so, how the filter manages to collect enough information to converge. Think carefully about: - The structure of this dynamical system (e.g. is it linear?). - Is convergence equivalent to resolving the exact state of the process at any specific point in time (i.e. use the full data stream to resolve the initial condition)?

Comment on how the smoothness of the filtered process changes as the process noise changes. Why should the process noise influence the smoothness of the estimated state?

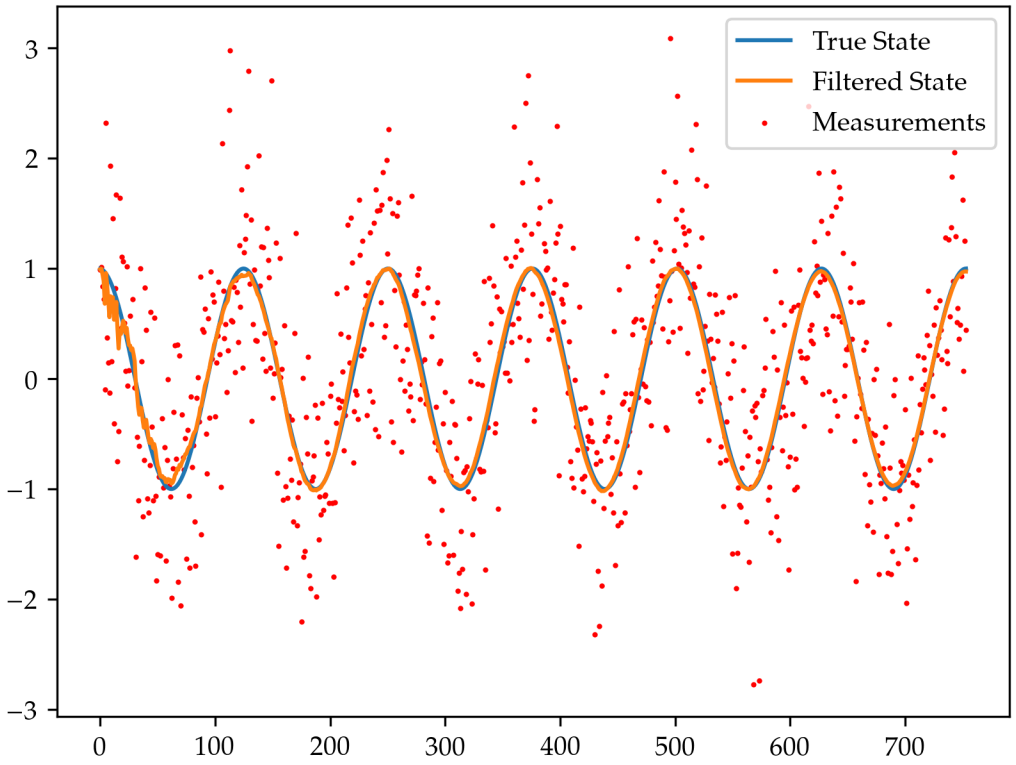
Answer:

Note that the solution to our system $\frac{d^2x}{dt^2} + \omega^2 x = 0$ is exactly given by

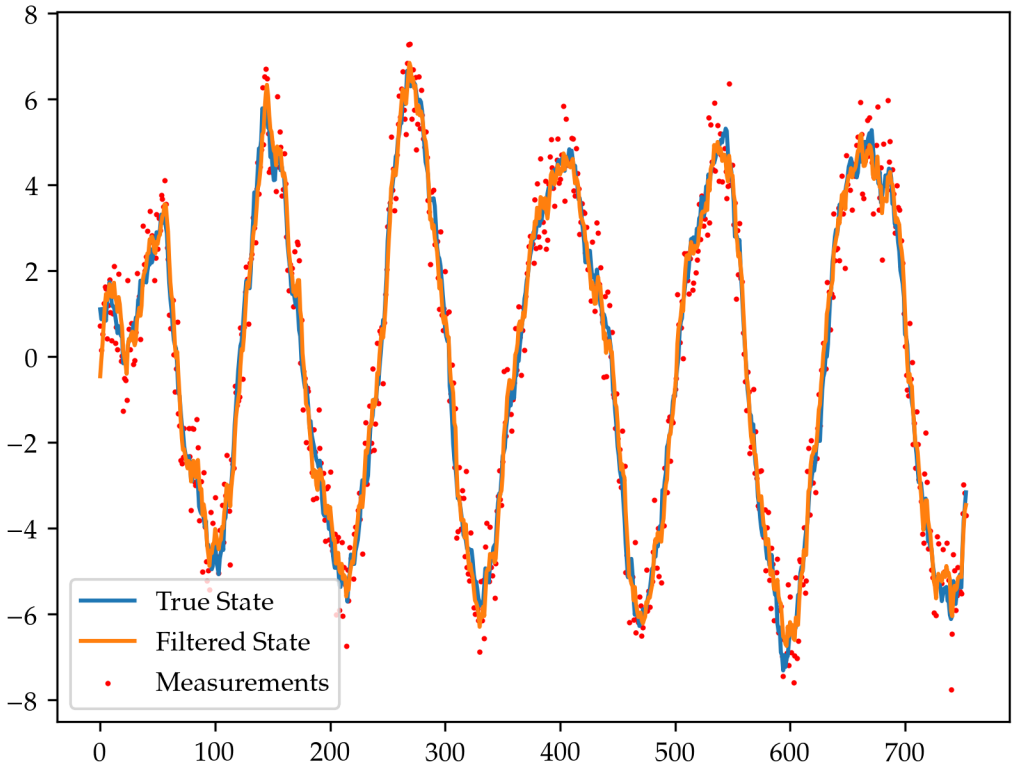
$$x(t) = A\cos(\omega t) + B\sin(\omega t),$$

where A and B are constants determined by the initial conditions (initial position and velocity). Therefore, the Kalman filter only needs to resolve the initial condition in order to infer the solution at any time point. Hence, it is able to converge to the true solution regardless of the measurement noise.

[6]:
```
# Insert demo for Q 2.2 here...
kalman_filter_demo(dt=0.05,process_noise=0.0,measurement_noise=0.5)
kalman_filter_demo(dt=0.05,process_noise=1.0,measurement_noise=0.5)
```



dt = 0.05, process_noise = 0.0, measurement_noise = 0.5, periods = 6



dt = 0.05, process_noise = 1.0, measurement_noise = 0.5, periods = 6

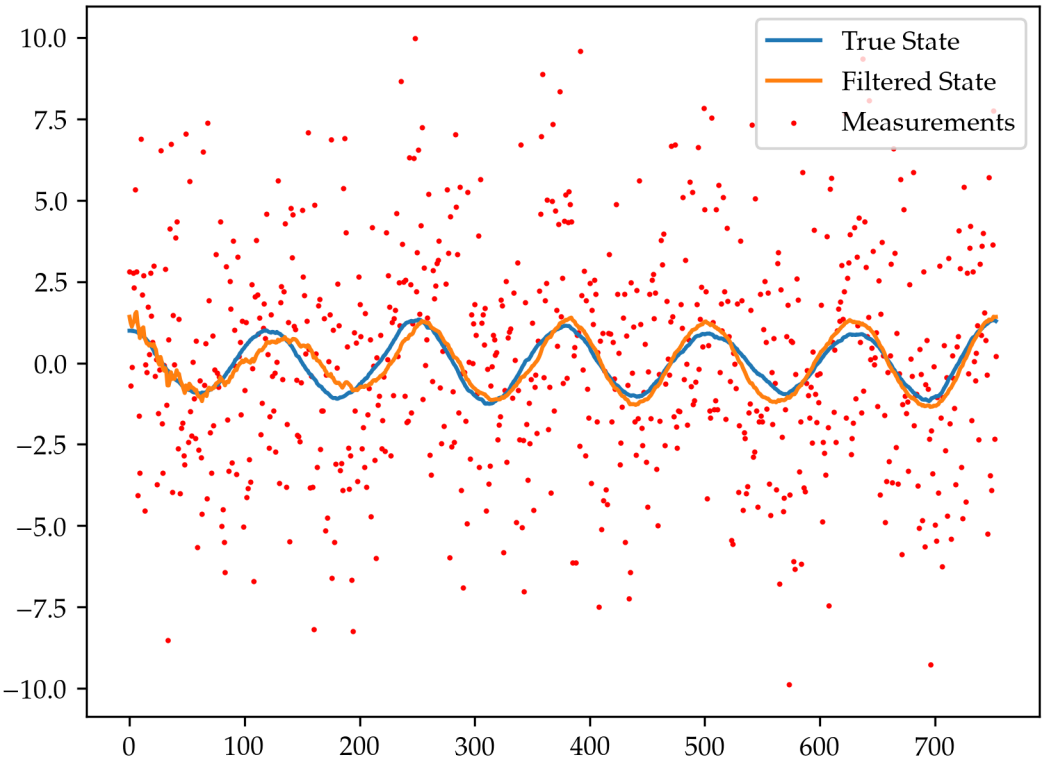
Q 2.3: Return the process noise to `0.005 * dt`. Now, let's try to understand the role of the measurement noise. As before, try:

1. Setting the measurement noise to zero
2. Setting the measurement noise large (try setting it to `10`, or `25`, or `100`).

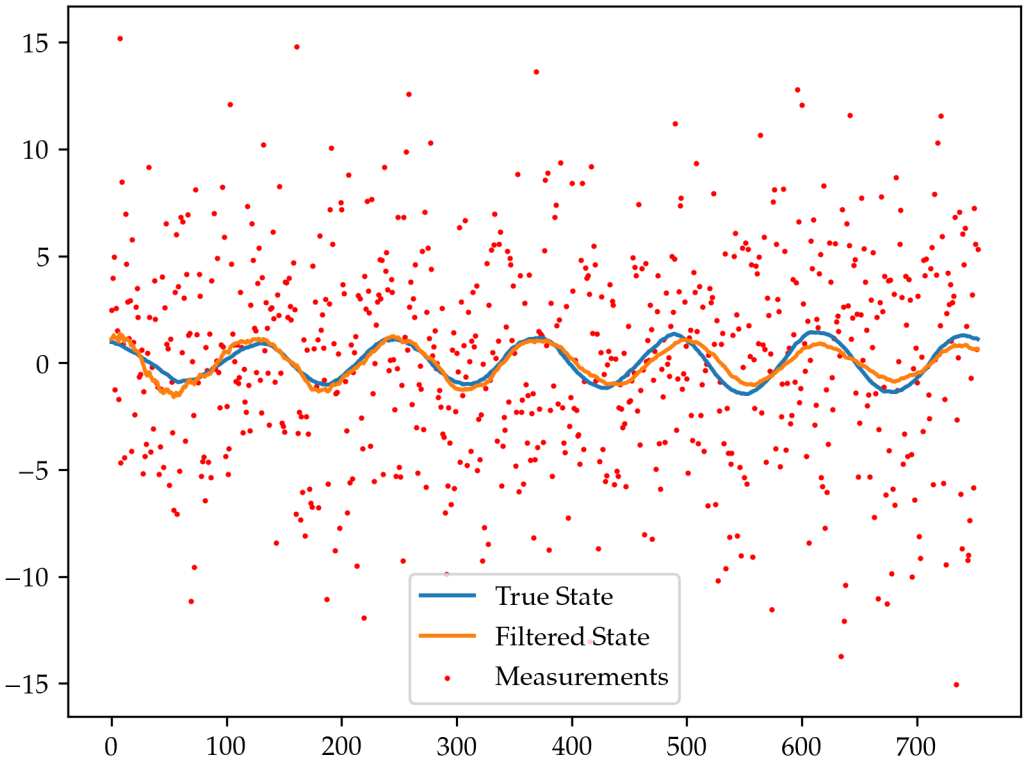
Comment on the accuracy of the filtered state as an estimate for the true state. Note whether it converges to a smooth process, whether it converges near the truth, and on the accuracy of the recovery relative to the signal-to-noise ratio in each individual measurement. Are you surprised by the amount of measurement noise we can tolerate? How is it possible to retain high accuracy estimates with such high measurement noise?

TODO

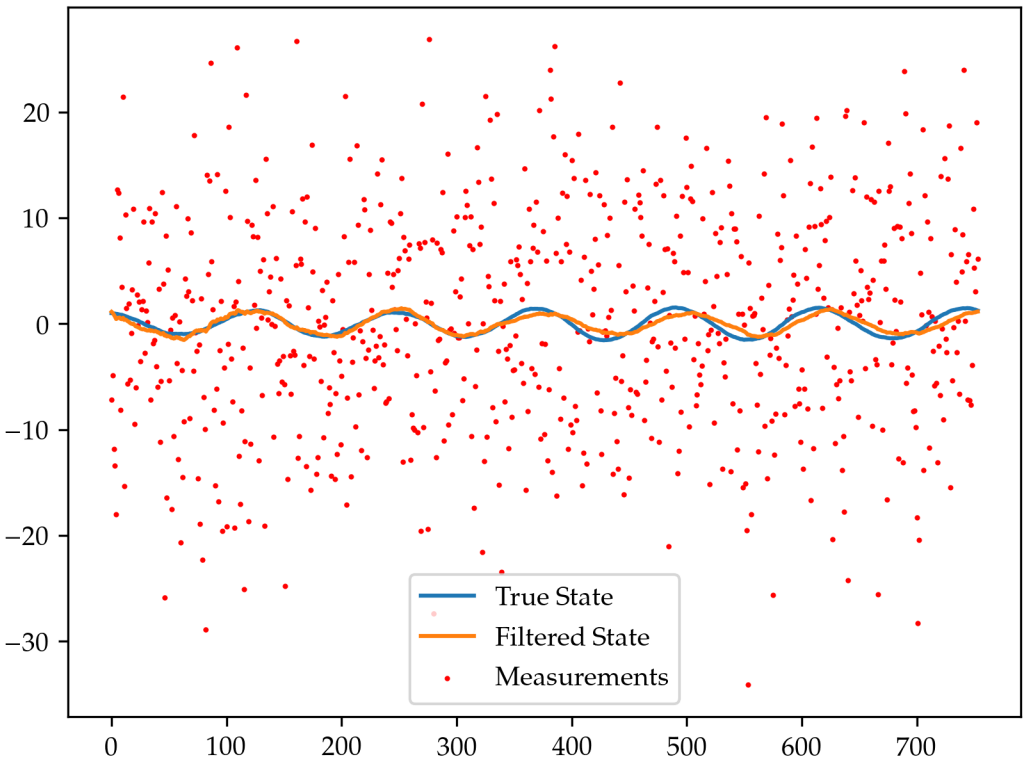
[7]:
```
# Insert demo for Q 2.3 here...
kalman_filter_demo(dt=0.05,process_noise=0.005,measurement_noise=10)
kalman_filter_demo(dt=0.05,process_noise=0.005,measurement_noise=25)
kalman_filter_demo(dt=0.05,process_noise=0.005,measurement_noise=100)
```



dt = 0.05, process_noise = 0.005, measurement_noise = 10, periods = 6



dt = 0.05, process_noise = 0.005, measurement_noise = 25, periods = 6



dt = 0.05, process_noise = 0.005, measurement_noise = 100, periods = 6

Q 2.4: We've seen the filter work. Now, let's break it.

1. Try setting the process noise to `0.1 * dt` and the measurement noise to `10`. Compare the accuracy and smoothness of the reconstruction to your results with measurement noise `10` in Q 2.4. Decrease the measurement noise until the process converges (holding the process noise fixed). Why is the threshold on acceptable measurement noise controlled by the process noise?

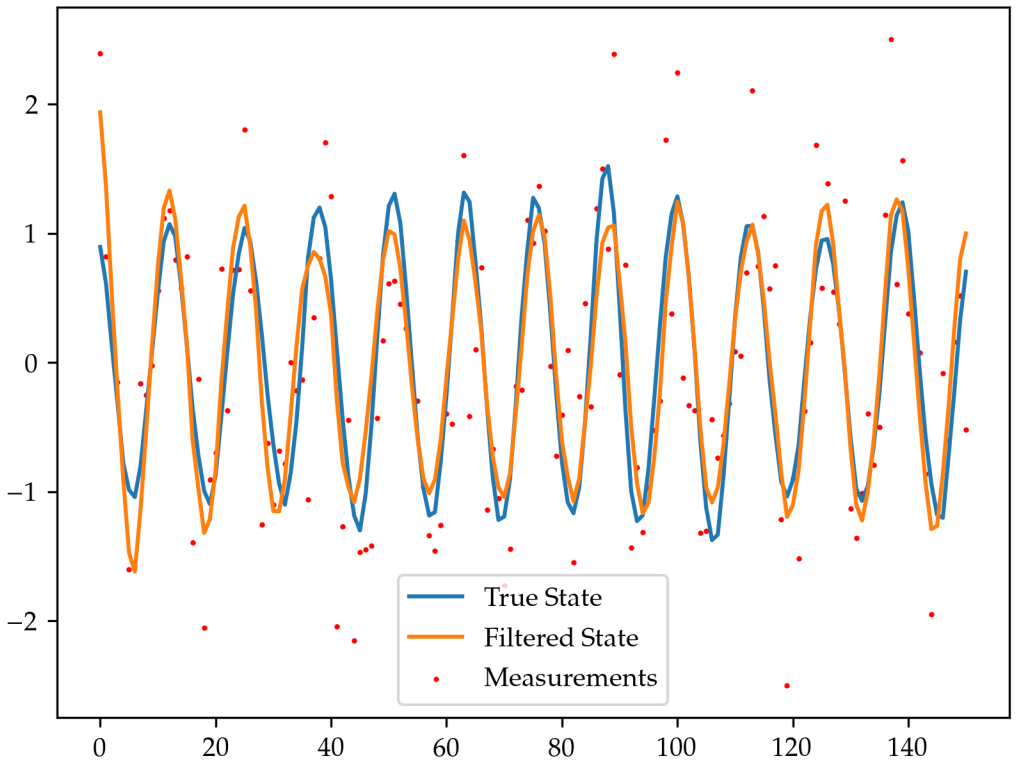
   TODO

[ ]:
```
# Insert demo for Q 2.4 here...
kalman_filter_demo(dt=0.05,process_noise=0.1,measurement_noise=10)
kalman_filter_demo(dt=0.05,process_noise=0.1,measurement_noise=5)
kalman_filter_demo(dt=0.05,process_noise=0.1,measurement_noise=1)
kalman_filter_demo(dt=0.05,process_noise=0.1,measurement_noise=0.01)
```

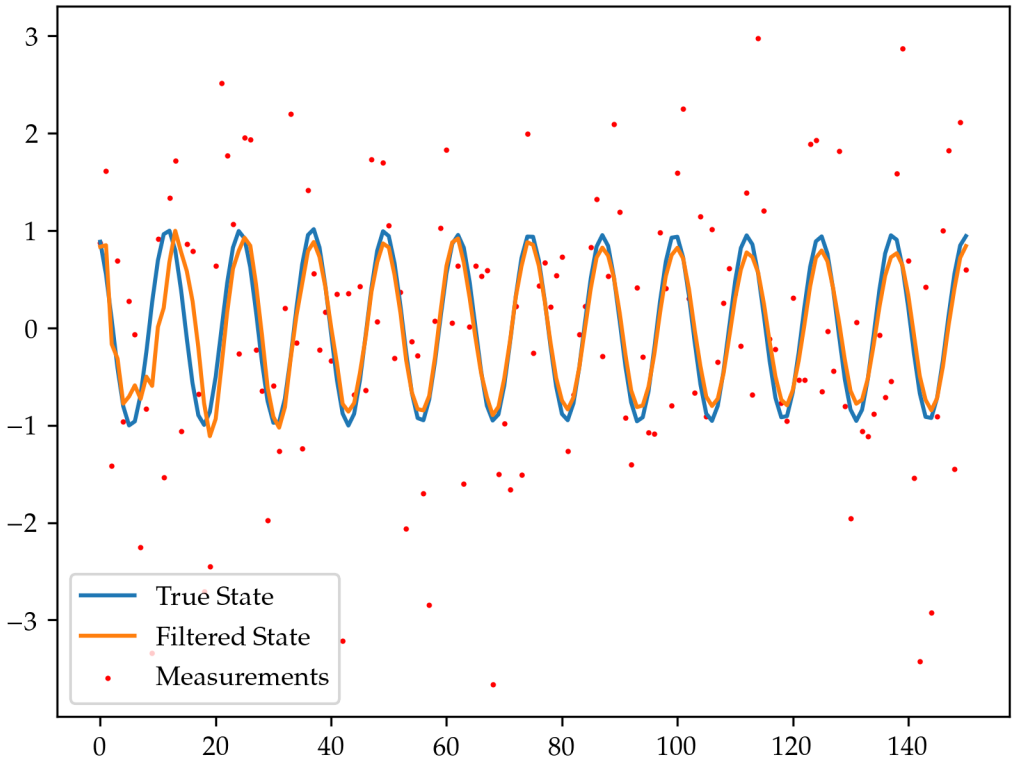
2. Set the process noise and measurement noise back to their defaults (`0.005 * dt` and `0.5`). Try increasing the sample interval to `0.5` and increase the total simulation time from `6` periods to `12`. Is the process capable of converging for sparse time sampling? If not, try decreasing either the process noise or the measurement noise until the process converges. Which controls the possibility of convergence? Why?

   TODO

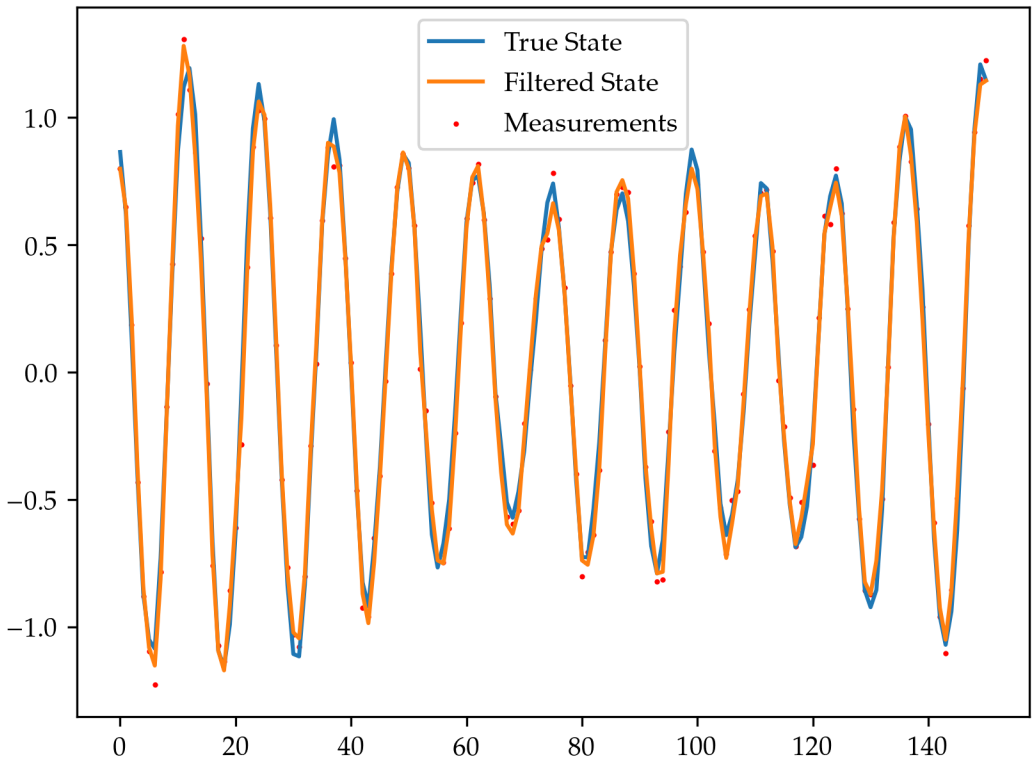
[8]:
```
# Insert demo for Q 2.4 here...
kalman_filter_demo(dt=0.5,process_noise=0.005,measurement_noise=0.5,periods=12)
kalman_filter_demo(dt=0.5,process_noise=0.00005,measurement_noise=1.5,periods=12)
kalman_filter_demo(dt=0.5,process_noise=0.005,measurement_noise=0.005,periods=12)
```



dt = 0.5, process_noise = 0.005, measurement_noise = 0.5, periods = 12



dt = 0.5, process_noise = 5e-05, measurement_noise = 1.5, periods = 12



dt = 0.5, process_noise = 0.005, measurement_noise = 0.005, periods = 12

Q 2.5: Comment on the overall success of the Kalman filter as an inferential tool. Consider the dimensionality of the object we are estimating, the dimensionality of our measurements, and the filter's capacity to integrate both past information while adapting to new observations.

TODO

[PART III] Application: Lorenz System Equations

The Lorenz system is a set of three differential equations used to model chaotic behavior:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z$$

where σ, ρ, and β are system parameters that control the chaotic behavior of the system. These equations describe the evolution of three state variables x, y, z over time. This system was first derived as an extremely simplified weather model based. It is famous as the dynamical systems model which led to the discovery of chaotic dynamical systems.

The Lorenz system is an interesting contrast to the harmonic oscillator example. First, it's dynamic is nonlinear, so the entire collection of process and measurement values are not jointly Gaussian distributed. This means that, the Kalman filter only performs *approximate* posterior inference for the Lorenz model.

Second, unlike the harmonic oscillator example, the Lorenz system is chaotic. This means that, given long enough, any pair of state vectors initialized from the same location will spread arbitrarily far apart, and mix over some attractor. For the Lorenz system, this attractor is the eponymous "butterfly" of the butterfly effect. Accurate weather prediction beyond ten days remains an unsolved scientific problem in large part because weather systems are chaotic.

Chaotic systems present the hardest possible case for data assimilation. In a chaotic system, we are constantly using measurement to tamp down the ever expanding uncertainty produced by the dynamic. Chaos is also typical for real systems where data assimilation is used.

Discretize the System

Since the Kalman filter operates in discrete time, we need to approximate the continuous Lorenz equations using a numerical method. We use the Runge-Kutta method to integrate the system over small time steps, ensuring accuracy in tracking the chaotic dynamics.

Step 1: Generate Noisy Observations

In real-world applications, measurements are subject to noise. We assume that only the (x) and (y) components are observable and introduce Gaussian noise to simulate sensor inaccuracies:

$$y_t = Hx_t + v_t$$

where (v_t) is Gaussian measurement noise with covariance (R).

[9]:
```
from scipy.integrate import solve_ivp

# # Lorenz system parameters
# sigma, rho, beta = 10, 28, 8/3
# dt = 0.02  # Time step - default 0.02

# # Define the Lorenz system
# def lorenz_system(t, state):
#     x, y, z = state
#     dxdt = sigma * (y - x)
#     dydt = x * (rho - z) - y
#     dzdt = x * y - beta * z
#     return [dxdt, dydt, dzdt]

# # Solve Lorenz system numerically
# t_end = 20
# time_span = np.linspace(0, t_end, round(t_end/dt))
# sol = solve_ivp(lorenz_system, [time_span[0], time_span[-1]], [1, 1, 1], t_eval=time_span)
```

```
# # Observation matrix (assume we only observe x and y)
# H = np.array([[1, 0, 0], [0, 1, 0]])
# Q = np.eye(3) * 0.0 * dt  # Process noise covariance - Set very small since we solved the dynamic
exactly above. Any noise here models integration error in the ODE solver
# R = np.eye(2) * 0.2  # Measurement noise covariance - default 0.2

# # Initial state and covariance
# x_est = np.array([[1], [1], [1]])
# P = np.eye(3) * 1.0

# # Generate noisy observations
# measurements = sol.y[:2].T + np.random.normal(0, 1, (len(time_span), 2)) @ scipy.linalg.sqrtm(R).T
```

Step 2: Compute the Prediction Step

At each time step, we predict the next state using the system dynamics.

Since the Lorenz system is nonlinear, we will have to derive the state transition matrix $A(\hat{x}_t)$ to approximately linearize the dynamic over the range of possible states. (Note: if the time steps are too large, it is common to replace this linear, approximate, update, with explicit simulation of a collection of states sampled from the current posterior. These may be simulated forward, then fit to a Gaussian, to give a new Gaussian prior, which is used to infer a new posterior after measurement).

For now, we will proceed by linearizing explicitly. To linearize, first compute the Jacobian of the dynamic:

$$J([x,y,z]) = [\partial_x, \partial_y, \partial_z] \begin{bmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ xy - \beta z \end{bmatrix} = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho & -1 & -x \\ y & x & -\beta \end{bmatrix}.$$

Then, set:

$$A(\hat{x}_t) = \exp(J(\hat{x}_t)\Delta t)$$

where Δt is the length of the time step, and $\exp(\cdot)$ is the matrix exponential.

Now that we've linearized the action of the dynamic over a time step, we can proceed as before. Think carefully about what assumptions on the state covariance, and the length of the time step, are needed to ensure this approximation is reliable.

1. **State Prediction:** Set $A_t = A(\hat{x}_{t-1|t-1})$. Then, set:

$$\hat{x}_{t|t-1} = A_t \hat{x}_{t-1|t-1}$$

   This equation propagates the previous estimate through the state model to predict the next state.

2. **Covariance Prediction:**

$$P_{t|t-1} = A_t P_{t-1|t-1} A_t^T + Q$$

   Here, we compute the predicted covariance, which accounts for the uncertainty in the process model.

Step 3: Compute the Kalman Gain

Proceed as before.

Step 4: Compute the Update Step

Proceed as before.

[10]:
```
# Kalman filter implementation
# filtered_states = []
# for z in measurements:
#     J = np.array([[-sigma, sigma, 0],
#                   [rho, -1, -x_est[0]],
#                   [x_est[1], x_est[0], -beta]],
#                 dtype=object)  # Jacobian of the Lorenz dynamic
#     A = scipy.linalg.expm(J * dt) # state transition matrix for linear approximation to the dynamic
```

```
#     x_pred = A @ x_est
#     P_pred = A @ P @ A.T + Q

#     K = P_pred @ H.T @ np.linalg.inv(H @ P_pred @ H.T + R)
#     x_est = x_pred + K @ (z.reshape(-1, 1) - H @ x_pred)
#     P = (np.eye(3) - K @ H) @ P_pred

#     filtered_states.append(x_est.flatten())

# filtered_states = np.array(filtered_states)
```

Step 5: Implement the Kalman Filter and Visualize Results

To evaluate the filter's performance, we plot: 1. **True State**: The numerically integrated Lorenz system. 2. **Filtered State**: The Kalman filter's best estimate. 3. **Measurements**: The noisy observations.

This visualization helps assess how well the Kalman filter mitigates chaos-induced divergence.

[11]:
```
# # Visualization
# plt.figure(figsize=(10, 5))
# plt.plot(time_span, sol.y[0], label='True x', linestyle='dashed', linewidth=2)
# plt.plot(time_span, filtered_states[:, 0], label='Filtered x', linewidth=2)
# plt.scatter(time_span, measurements[:, 0], label='Measurements x', color='red', s=0.5, alpha=0.6)
# plt.xlabel('Time Step')
# plt.ylabel('State Value')
# plt.title('Kalman Filter Tracking on Lorenz System (x component)')
# plt.legend()
# plt.grid(True)
# plt.show()
```

Q 3.1: Try increasing the process noise. In particular, try setting the process noise to `0.01 * dt` and the measurement noise to `0.2`. Compare to the results you get when the process noise is zero.

Note that, in this case, we simulated the dynamic deterministically. If you increase the process noise, you are, in effect, performing posterior inference using a softened likelihood model, that assumes more noise in the process, and thus looser temporal coupling, than there is true noise in the process. This is, in effect, using a less informative prior model at each time step. Comment on how this changes the behavior of the filtered estimate when the measurement noise is small. Do you think that adopting a weaker prior is helpful in this case?

TODO

[12]:
```
# Insert experiments for Q 3.1 here...
def kalman_Lorenz(
        dt = 0.02,  # Time step
        process_noise = 0.0,
        measurement_noise = 0.2,
        # periods = 6
    ):
  # Lorenz system parameters
  sigma, rho, beta = 10, 28, 8/3
  dt = 0.02  # Time step - default 0.02

  # Define the Lorenz system
  def lorenz_system(t, state):
      x, y, z = state
      dxdt = sigma * (y - x)
      dydt = x * (rho - z) - y
      dzdt = x * y - beta * z
      return [dxdt, dydt, dzdt]

  # Solve Lorenz system numerically
  t_end = 20
```

```
  time_span = np.linspace(0, t_end, round(t_end/dt))
  sol = solve_ivp(lorenz_system, [time_span[0], time_span[-1]], [1, 1, 1], t_eval=time_span)

  # Observation matrix (assume we only observe x and y)
  H = np.array([[1, 0, 0], [0, 1, 0]])
  Q = np.eye(3) * process_noise * dt  # Process noise covariance - Set very small since we solved the
dynamic exactly above. Any noise here models integration error in the ODE solver
  R = np.eye(2) * measurement_noise  # Measurement noise covariance - default 0.2

  # Initial state and covariance
  x_est = np.array([[1], [1], [1]])
  P = np.eye(3) * 1.0

  # Generate noisy observations
  measurements = sol.y[:2].T + np.random.normal(0, 1, (len(time_span), 2)) @ scipy.linalg.sqrtm(R).T

  # Kalman filter implementation
  filtered_states = []
  for z in measurements:
      J = np.array([[-sigma, sigma, 0],
                    [rho, -1, -x_est[0]],
                    [x_est[1], x_est[0], -beta]],
                   dtype=object)  # Jacobian of the Lorenz dynamic
      A = scipy.linalg.expm(J * dt) # state transition matrix for linear approximation to the dynamic
      x_pred = A @ x_est
      P_pred = A @ P @ A.T + Q

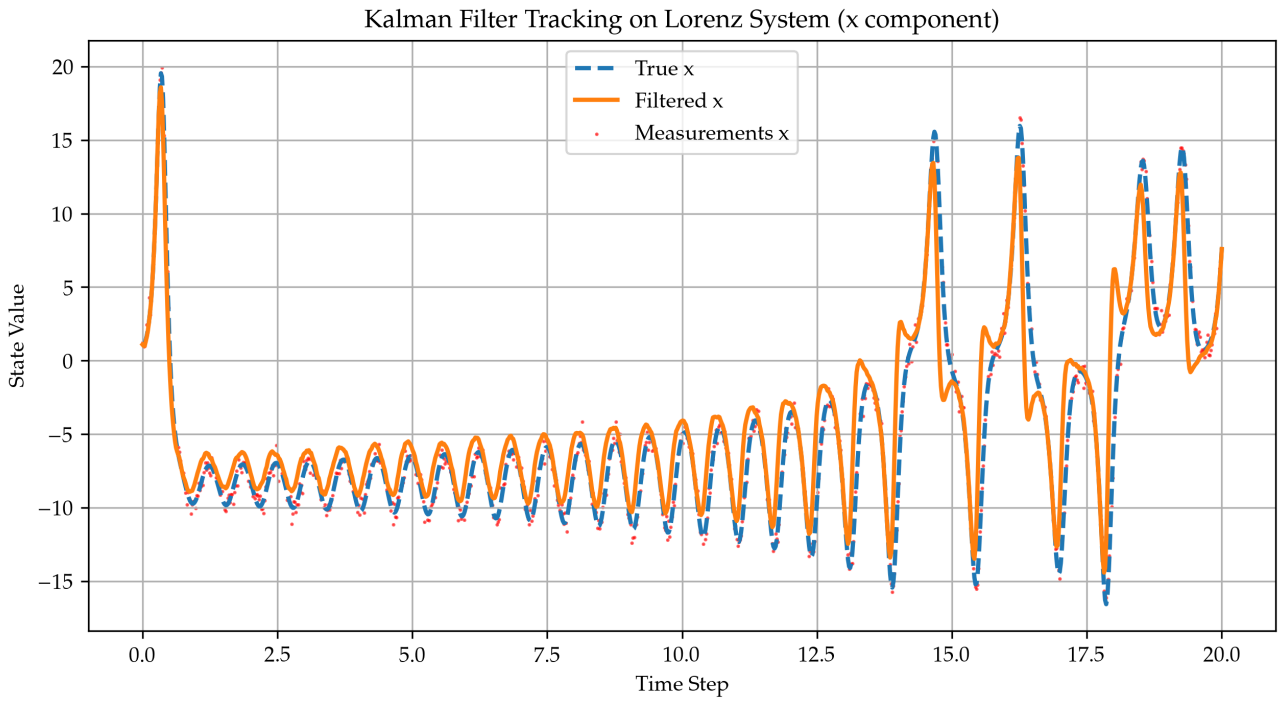
      K = P_pred @ H.T @ np.linalg.inv(H @ P_pred @ H.T + R)
      x_est = x_pred + K @ (z.reshape(-1, 1) - H @ x_pred)
      P = (np.eye(3) - K @ H) @ P_pred

      filtered_states.append(x_est.flatten())

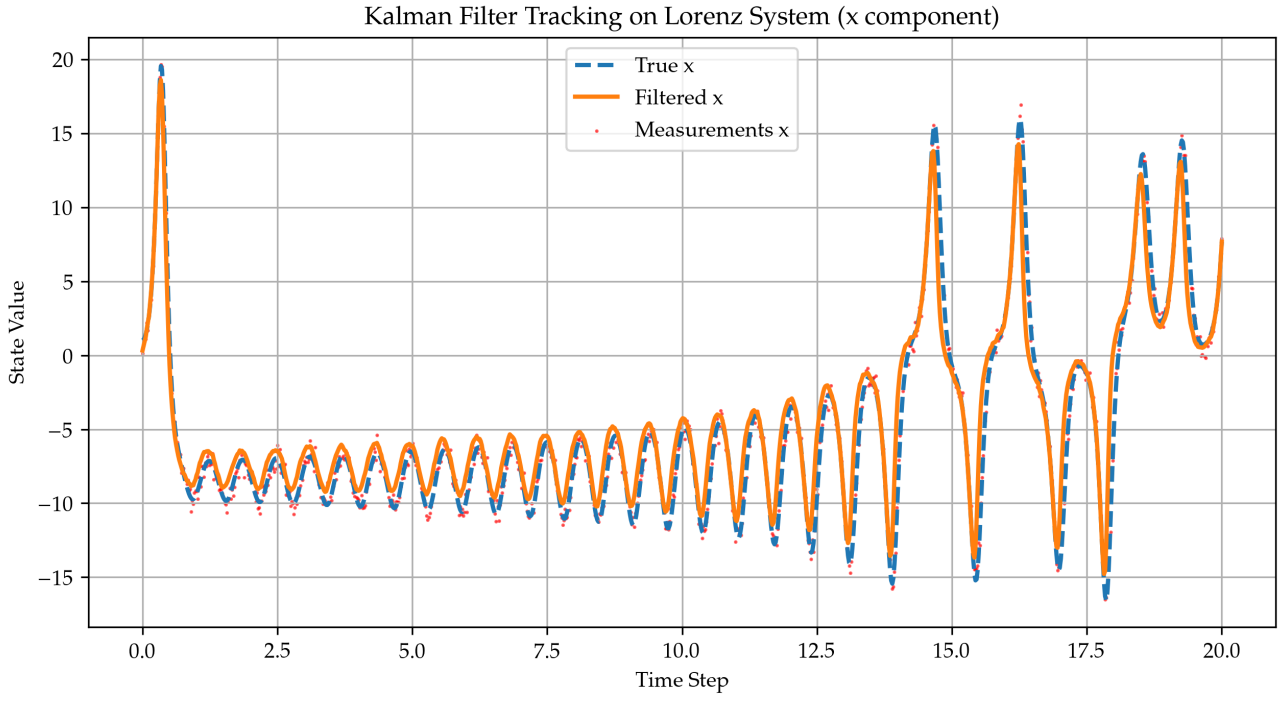
  filtered_states = np.array(filtered_states)

  # Visualization
  plt.figure(figsize=(10, 5))
  plt.plot(time_span, sol.y[0], label='True x', linestyle='dashed', linewidth=2)
  plt.plot(time_span, filtered_states[:, 0], label='Filtered x', linewidth=2)
  plt.scatter(time_span, measurements[:, 0], label='Measurements x', color='red', s=0.5, alpha=0.6)
  plt.xlabel('Time Step')
  plt.ylabel('State Value')
  plt.title('Kalman Filter Tracking on Lorenz System (x component)')
  plt.legend()
  param_string = (
        f"dt = {dt},  "
        f"process_noise = {process_noise},  "
        f"measurement_noise = {measurement_noise}"
    )
  plt.figtext(0.5, -0.05, param_string, ha='center', fontsize=10)
  plt.grid(True)
  plt.show()

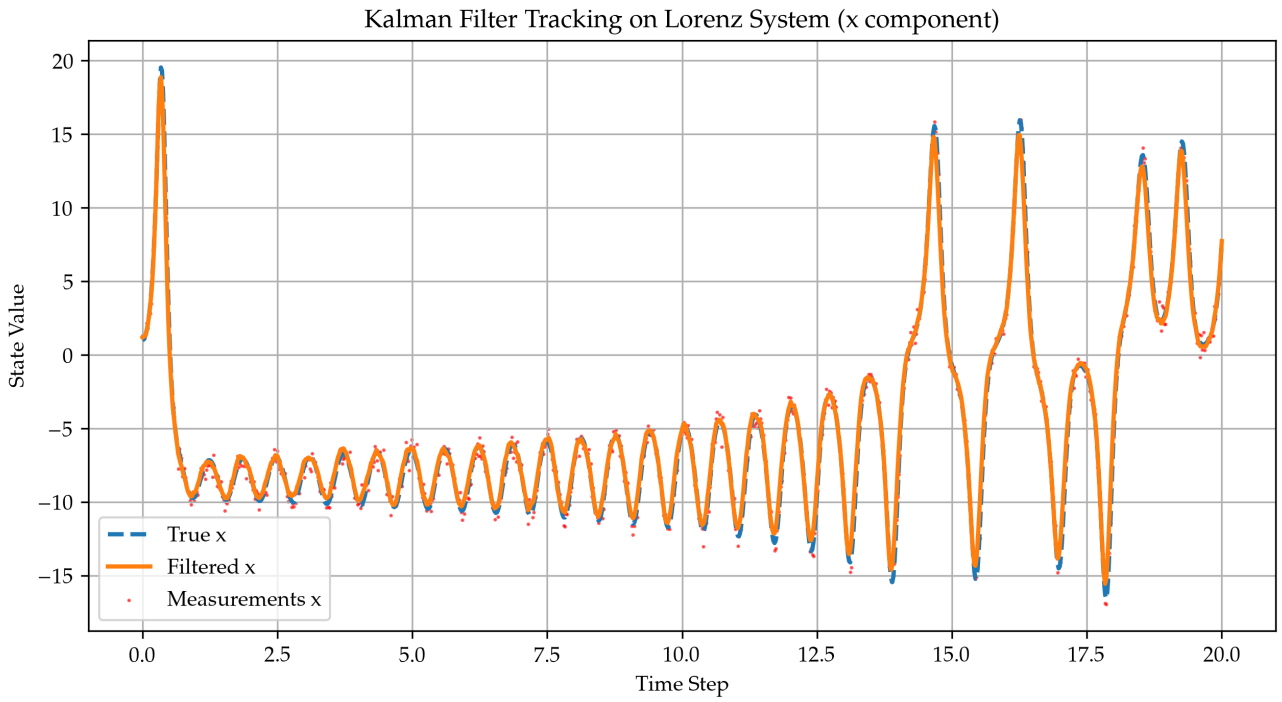
kalman_Lorenz(dt=0.02,process_noise=0.0,measurement_noise=0.2)
kalman_Lorenz(dt=0.02,process_noise=0.01,measurement_noise=0.2)
kalman_Lorenz(dt=0.02,process_noise=0.2,measurement_noise=0.2)
```



dt = 0.02, process_noise = 0.0, measurement_noise = 0.2



dt = 0.02, process_noise = 0.01, measurement_noise = 0.2

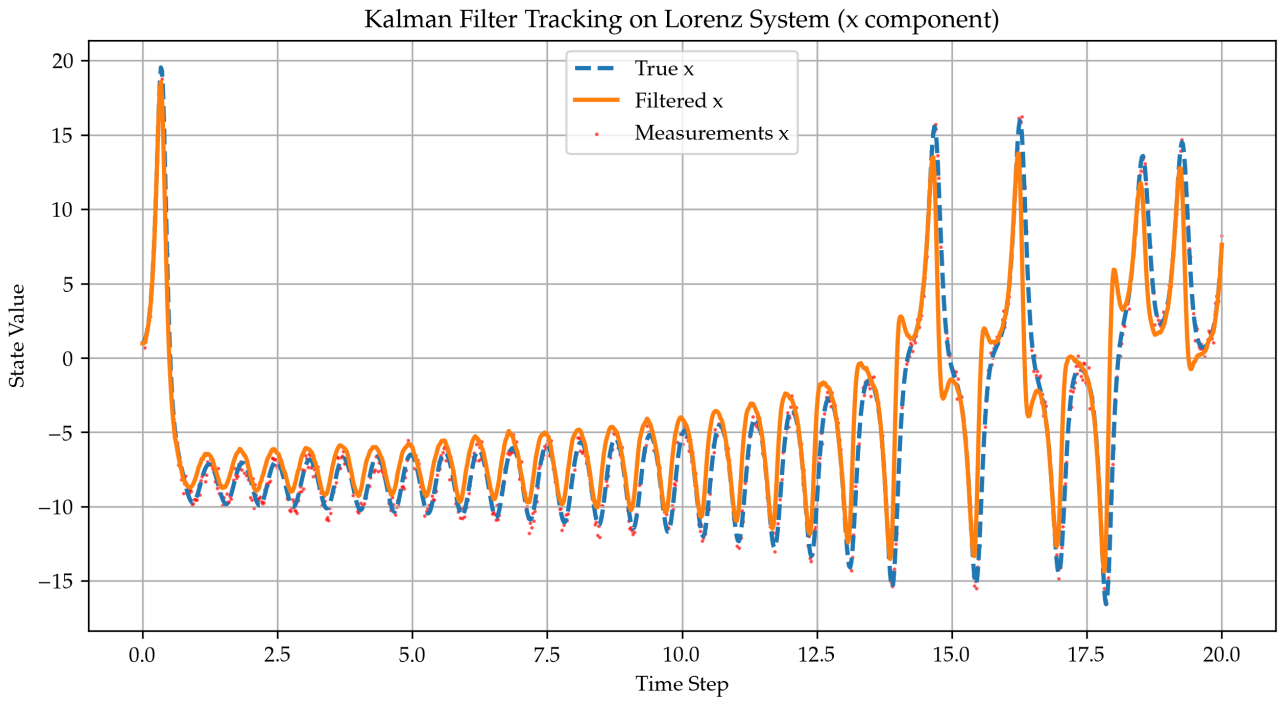


dt = 0.02, process_noise = 0.2, measurement_noise = 0.2

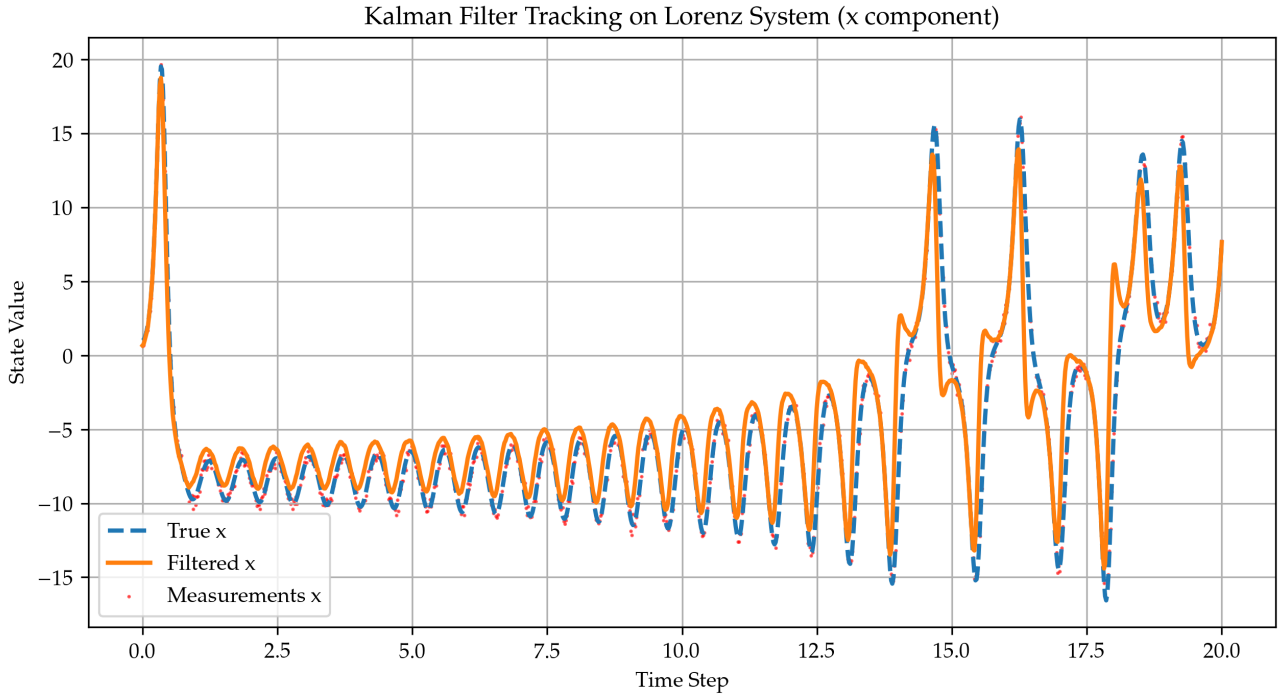
Q 3.2: Set the process noise back to zero. Try gradually decreasing the measurement noise level. Can you find a nonzero measurement noise level small enough such that the filtered process converges over the full range of simulated time steps? Compare to the results you observed for the simple harmonic oscillator with no process noise. Try to explain the phenomena you observe.

TODO

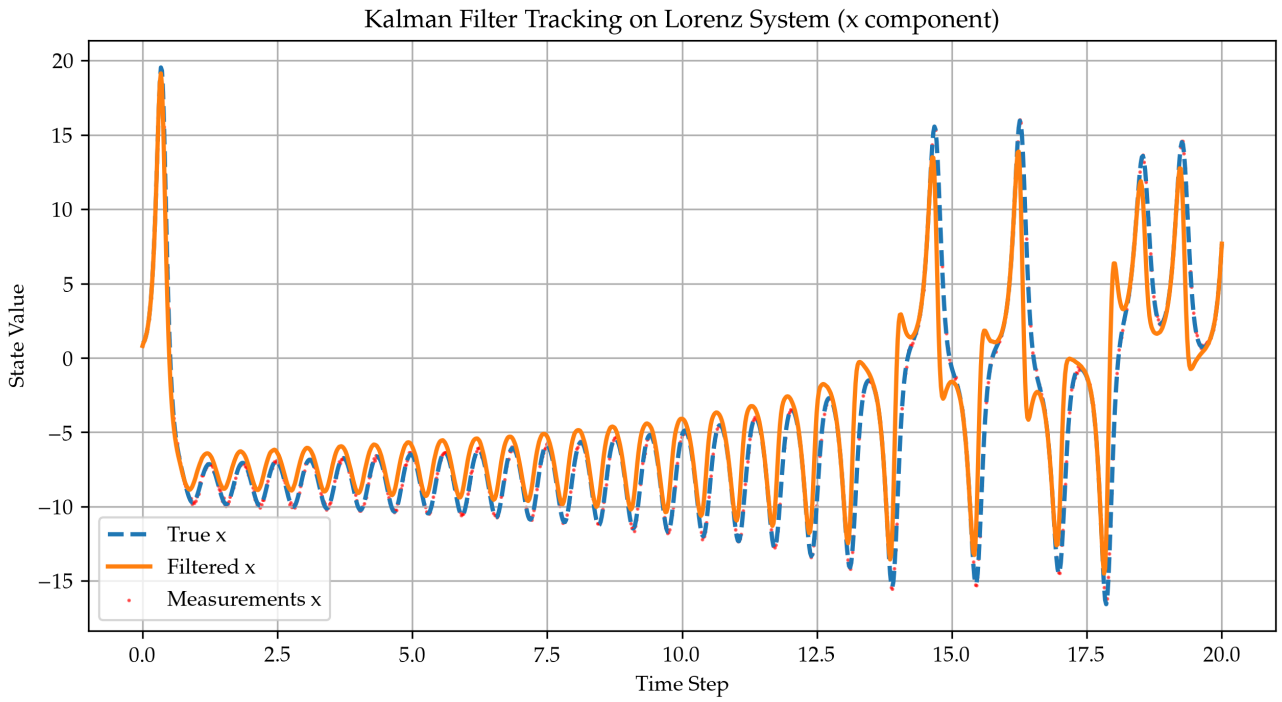
[13]:
```
# Insert experiments for Q 3.2 here...
kalman_Lorenz(dt=0.02,process_noise=0.0,measurement_noise=0.2)
kalman_Lorenz(dt=0.02,process_noise=0.0,measurement_noise=0.1)
kalman_Lorenz(dt=0.02,process_noise=0.0,measurement_noise=0.01)
kalman_Lorenz(dt=0.02,process_noise=0.0,measurement_noise=0.001)
kalman_Lorenz(dt=0.02,process_noise=0.0,measurement_noise=0.0000000000001)
```



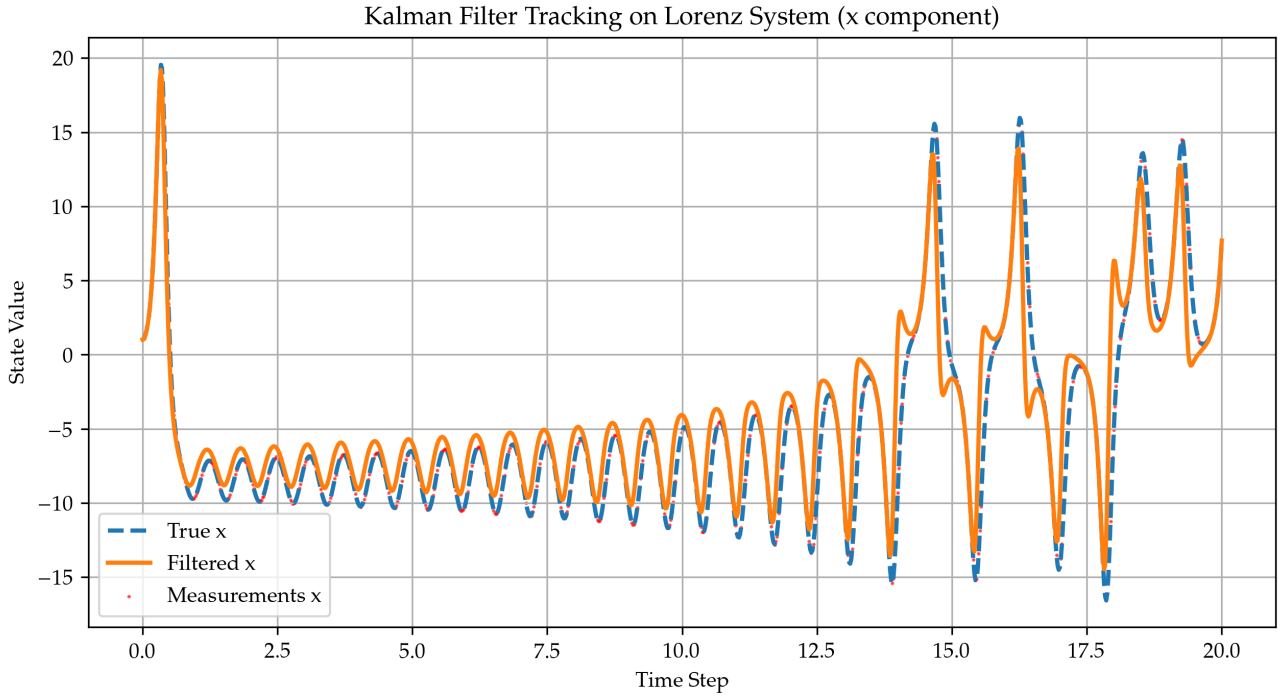
dt = 0.02, process_noise = 0.0, measurement_noise = 0.2



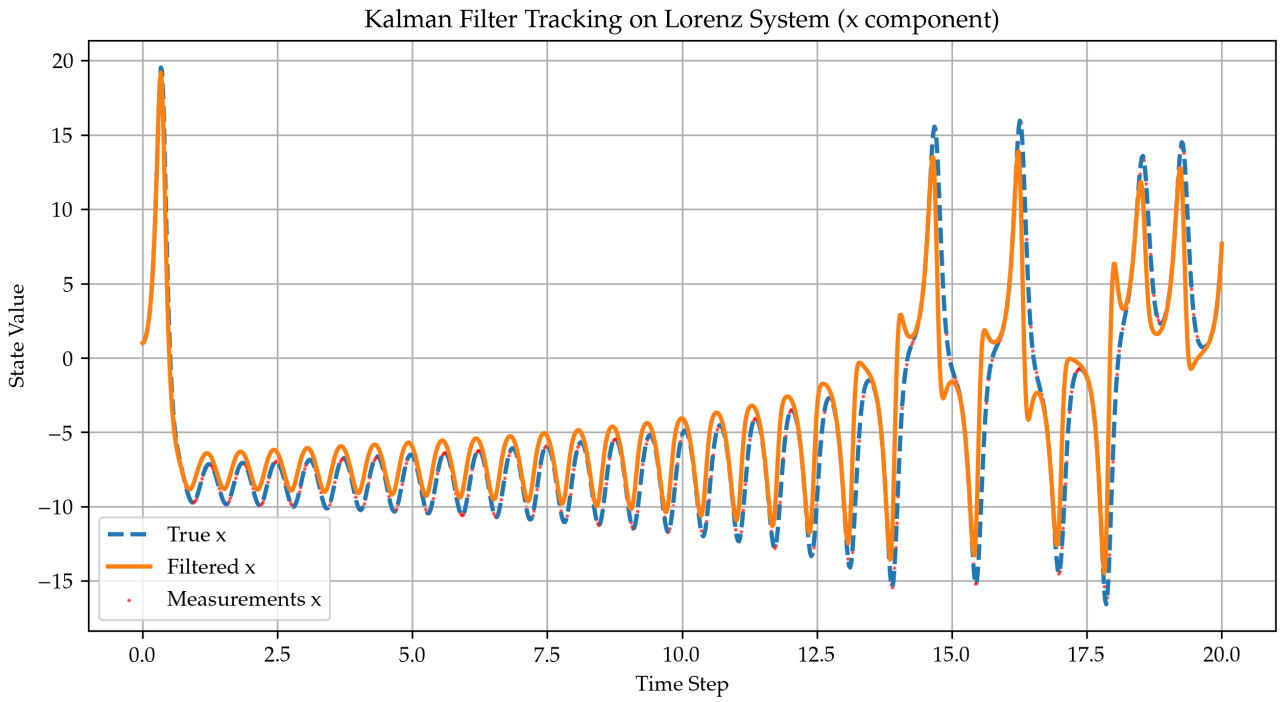
dt = 0.02, process_noise = 0.0, measurement_noise = 0.1



dt = 0.02, process_noise = 0.0, measurement_noise = 0.01



dt = 0.02, process_noise = 0.0, measurement_noise = 0.001

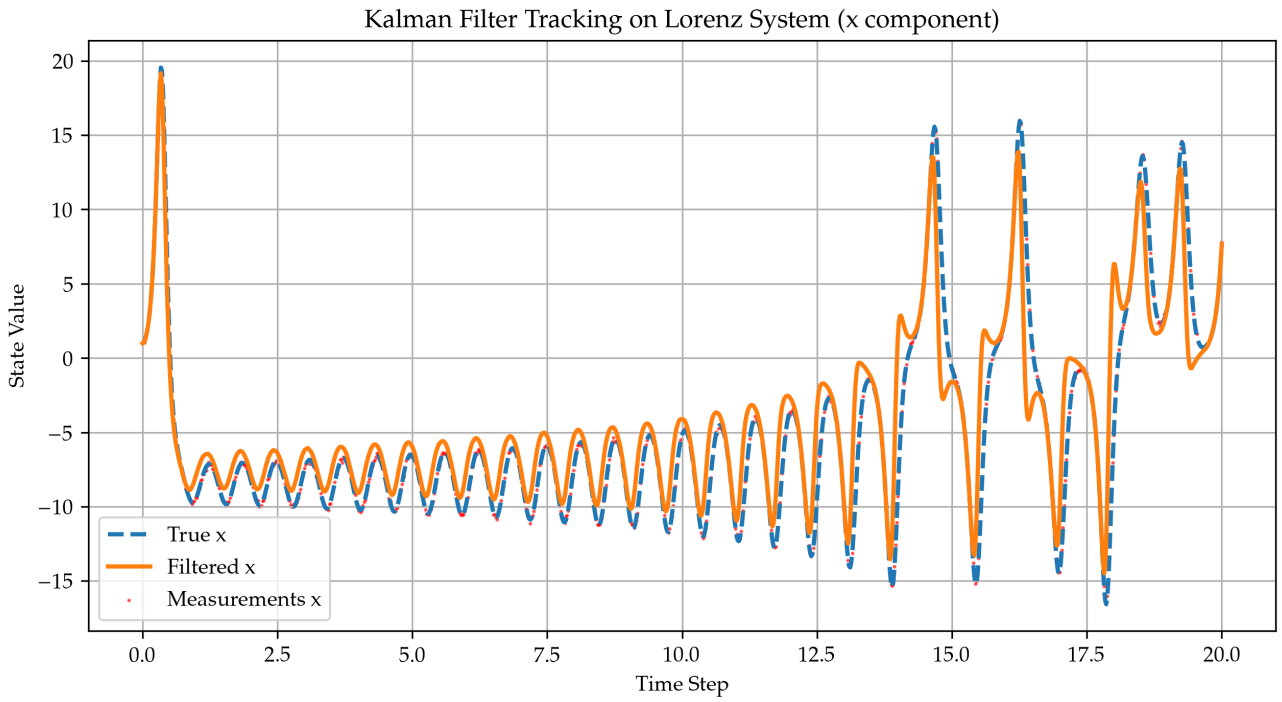


dt = 0.02, process_noise = 0.0, measurement_noise = 1e-12

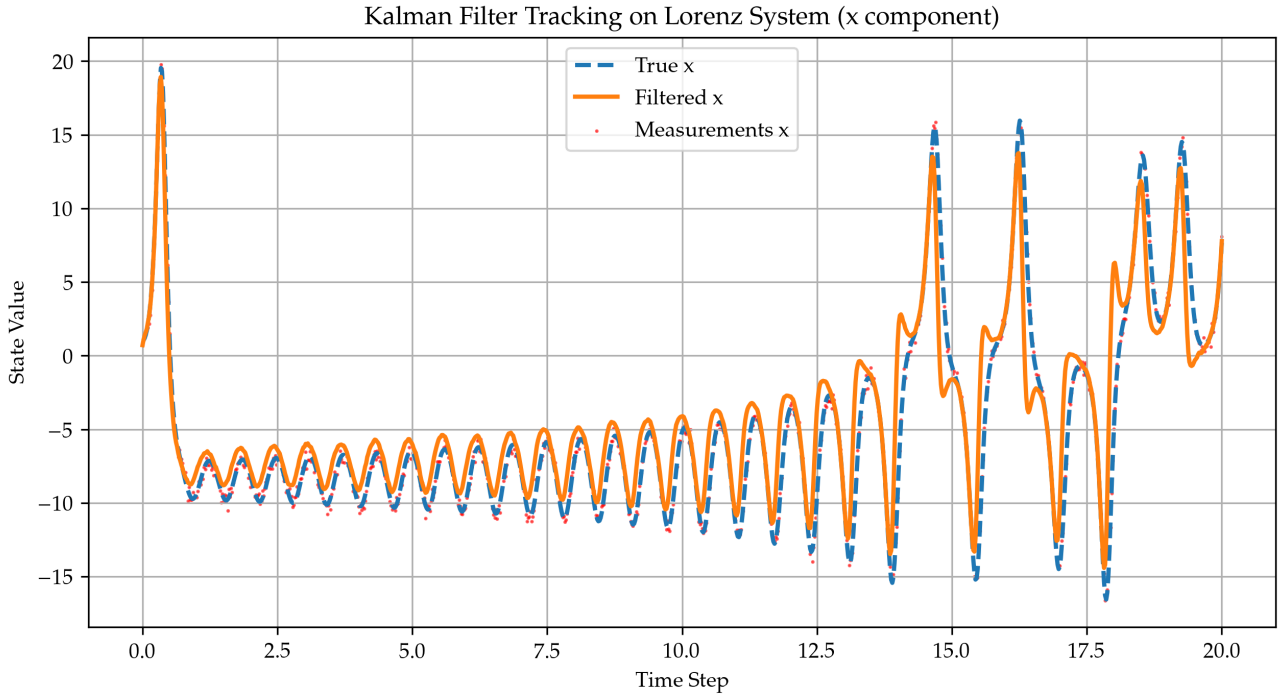
Q 3.3: Set the process noise small, then experiment with the relationship between the step size `dt` and the measurement noise needed to relaible inference. For example, try setting `dt = 0.1` and the measurement noise to `0.01`. Can you get away with larger measurement noise? Given coarse intervals, how fine must our measurements be? Comment on the feasibility of long term prediction in chaotic systems, and the balance between the uncertainty produced by the dynamic, and the information (reduction of uncertainty) produced by measurement.

TODO

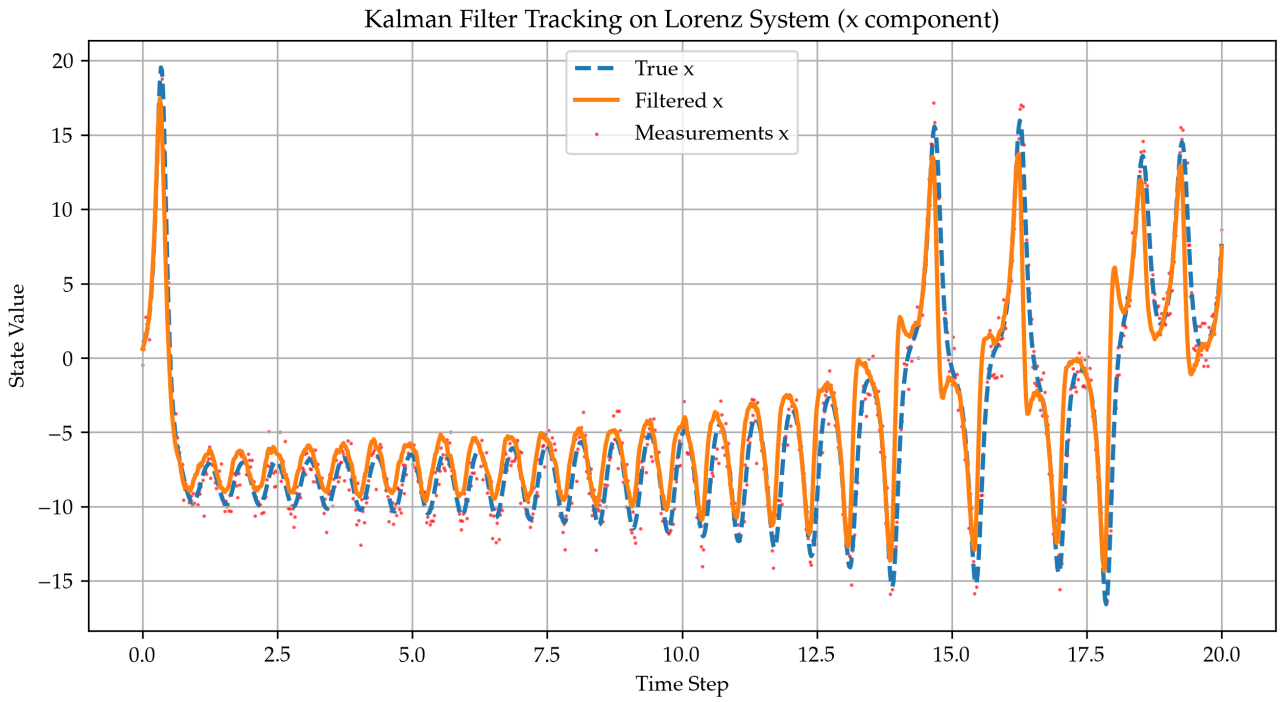
[14]:
```
# Insert experiments for Q 3.3 here...
kalman_Lorenz(dt=0.1,process_noise=0.0,measurement_noise=0.01)
kalman_Lorenz(dt=0.1,process_noise=0.0,measurement_noise=0.1)
kalman_Lorenz(dt=0.1,process_noise=0.0,measurement_noise=1.0)
kalman_Lorenz(dt=0.1,process_noise=0.0,measurement_noise=10.0)
```



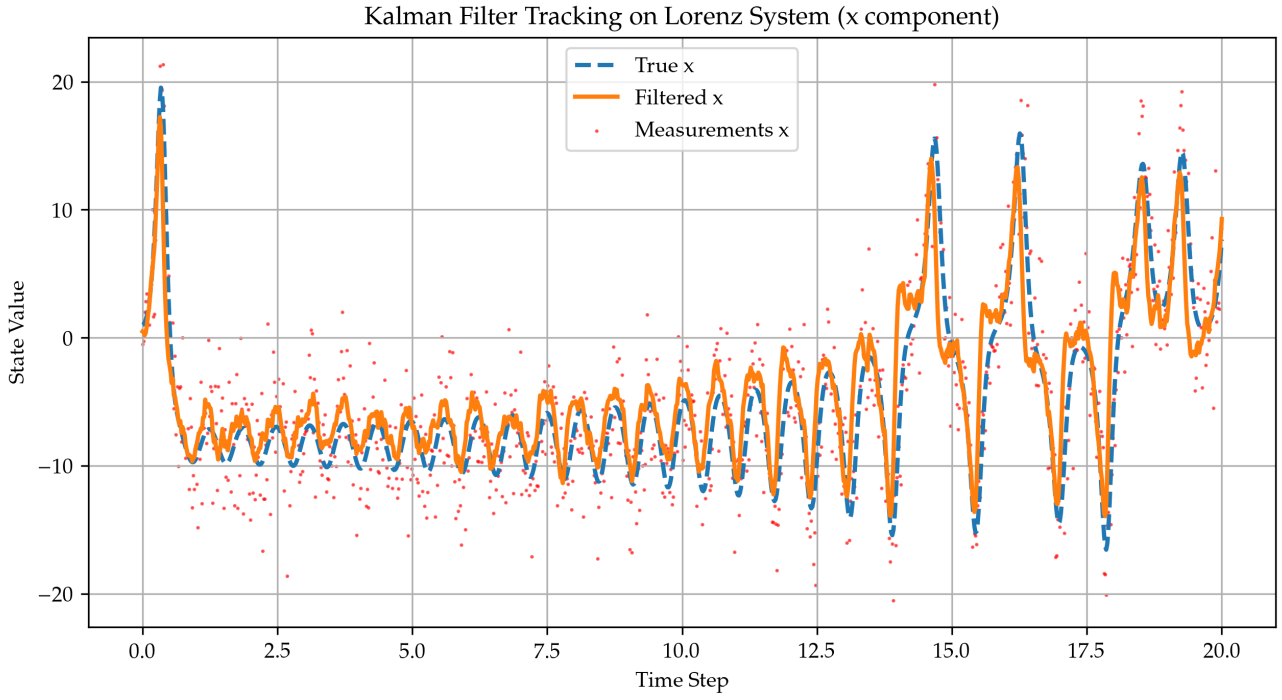
dt = 0.02, process_noise = 0.0, measurement_noise = 0.01



dt = 0.02, process_noise = 0.0, measurement_noise = 0.1



dt = 0.02, process_noise = 0.0, measurement_noise = 1.0



dt = 0.02, process_noise = 0.0, measurement_noise = 10.0

Outro: A note on revolutions

As foreshadowed at the start of this lab, the Kalman filtering technique implemented above was the gold standard in many large data assimilation problems, particularly weather prediction, until this year. AI models trained on simulated weather data may now provide more skillful predictions (see GraphCast, 2023 and GenCast, Nature, 2025).

How is this possible?

A simple answer is that trying to predict the behavior of the weather far in advance requires running many simulations of the dynamical system, initialized from different possible current states. That is, initialized from the current posterior. Because the dyanmic is chaotic, these rapidly spread out. Think about dropping a ball into a Plinko board. If you could exactly specify its position and momentum at the start, and the material properties of the ball and pins, and how the board is stabilized on the table, and how the ball interacts with the air, and its rolling friction against the board, and the exact surface topology of the board, then, in principle, you could predict where it would go, after many bounces, with high precision. Of course, this is essentially impossible. The ball's final position behaves as if it were random since we don't have complete information on the dynamic, or on the starting condition. If you have a highly concentrated prior over the initial conditions, then you may be able to precisely predict the ball's location after one or two bounces. However, the farther you look into the future, the faster your posterior will spread out.

Weather simulations have exactly the same problem. Like Plinko, they also follow regular statistical patterns over different time frames. For example, after many bounces, a ball descending a Plinko board will have a horizontal position that is roughly Gaussian distributed. If you haven't tried this, there are amazing demos (ask the instructor). So, while we may not be able to predict the exact motion of a particular ball, we could learn to approximate its distribution at any stage down the board, then use those distributions to model our uncertainty about the position of the ball. Moreover, since Plinko can be subdivided into many copies of the same problem, if we conditioned on an initial position for the ball, we could update our distributional estimate (e.g. our posterior) to reflect our observation.

In a Plinko board, you might imagine that the distribution produced by simulating the physics forward should mimic the physics closely enough that the simulated distribution would match the observed distribution over a long time frame. This is much harder for weather prediction. As a result, the simulated posterior looking far in advance may not capture long time statistical behavior of actual weather data. Alternately, since simulation is expensive, it may not sample finely enough.

In contrast, an AI model trained to imitate observed statistical patterns may sacrifice the step-by-step consistency of a physical simulation for the capacity to model observed long-time statistics. For long-term prediction in highly chaotic problems, this strategy can beat out direct simulation.

Lab 5: Bayesian Optimization and Surrogate Modeling via Gaussian Processes

Lab Date: Wednesday, February 26
Lab Due: Wednesday, March 5th

Instructions

Work with your lab group to complete the following notebook. Your work will be reviewed by your peers in lab next week (Wednesday, March 5th).

In this lab, you will:

1. Practice extending the normal, normal model to infinite dimensional regression problems
2. Learn about Gaussian Process (GP) regression and surrogate modeling
3. Learn how to use GP regression to perform Bayesian optimization

The first learning goal illustrates the power of scaleable conjugate models. The second and third demonstrate modern applications enabled by high efficiency Bayesian inference. Together, these act as a capstone demonstration of many of the ideas we've seen so far in the class and many of the advantages of a Bayesian approach: - regularization through prior assumptions, - probabilistic uncertainty quantification on the unknown, - consistent updating in sequential inference, - sensible decision theory, and - scaleable computation via conjugate modeling!

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

[21]:
```
# load whatever packages you prefer here. We've added a reference list here
import numpy as np
import scipy as scipy
from scipy import linalg
# import matplotlib.pyplot as plt

# My plotting settings
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'''
\usepackage{newpxtext}
\usepackage[vvarbb]{newpxmath}
\usepackage{amsmath}
\DeclareMathOperator*{\E}{\mathbb{E}}
\DeclareMathOperator*{\Var}{Var}
'''
rcParams["font.family"] = "Palatino"
rcParams['figure.dpi'] = 300
```

Part I: Gaussian Process Regression

Background: In your last Lab, you saw that the normal, normal model extends easily to high-dimensional problems. In this lab we will study the limiting generalization of the normal, normal model... *infinite dimensional* unknowns. In particular, we will focus on the problem of recovering an unknown function $f : \mathcal{X} \to \mathbb{R}$ given a finite set of sample nodes $\vec{x} = \{x_j\}_{j=1}^n$ and noisy samples at the nodes, $\vec{y} = [f(x_1), f(x_2), f(x_3), ..., f(x_n)] + \zeta$ where $\zeta \sim \mathcal{N}(0, C_y)$.

You may recognize this as an interpolation or fitting problem. In the standard narrative, we restrict f to a finite-dimensional set of functions with $m < n$ degrees of freedom, often by expressing f in a parametric form (say, a linear combination of feature/basis functions), then by solving for the MLE parameters.

Stochastic processes offer a non-parametric alternative. Instead of restricting f to a low-dimensional function space, we can pose a prior distribution over the set of all possible functions f by assuming that f is sampled from a stochastic process. Given a prior, we can condition on the noisy samples $\vec{y}$ to produce a posterior distribution over the possible functions f. This posterior can be used for Bayesian inference. For example identify the posterior distribution over the value of f at a finite set of query points, interval estimates for the values of f at a finite set of query points, or specify a posterior point estimator for f (either the posterior mode or mean). Importantly, by varying the stochastic process, we can vary what characteristics of f we promote in our inferences, and which we discourage.

Gaussian processes (GP's) are a popular, expressive, class of stochastic processes. A GP is a stochastic process with the property that *any set of finitely many samples of the process are multivariate normally distributed.* White noise, colored noise, auto-regressive processes with Gaussian updates of any order, Brownian motion, integrated Brownian motion, draws from stochastic differential equations, and all linear combinations of a fixed set of basis functions with Gaussian coefficients are GPs. The following summary is based on Rasmussen and Williams, *Gaussian Processes for Machine Learning*, Chapter 2. You can access select chapters through the course references listed on Ed.

A GP is parameterized by a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and a covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where, given:

$$f \sim \text{GP}(\mu, k)$$

the mean and covariance functions satisfy:

$$\mathbb{E}[f(x)] = \mu(x) \text{ and } \text{Cov}[f(x), f(x')] = k(x, x').$$

Note, the covariance function must be chosen so that it produces symmetric positive semi-definite matrices when evaluated at any finite set of sample nodes. Often, we assume that the process is stationary (its distribution is translationally symmetric). Then, the covariance function may be expressed with a kernel, κ, where $k(x, x') = \kappa(x - x')$. If, in addition, we assume that the process is isotropic, then its distribution is symmetric under all unitary coordinate transformations, so $K(x, x') = \kappa(\|x - x'\|)$.

Popular choices for κ include the squared exponential kernel, $\kappa(r) = \sigma_f \exp\left(-\frac{1}{2}\frac{r}{l}^2\right)$, the exponential kernel, $\kappa(r) = \sigma_f \exp\left(-\frac{|r|}{l}\right)$, or the Matern kernel, which interpolates between the exponential and squared exponential kernels. The parameter σ_f sets the variance in $f(x)$ at each input x. The parameter l is interpreted as a smoothness length scale. The larger l, the slower the kernels decay, so the stronger the correlation between the values of f at x and x'.

The choice of kernel controls the behavior of the functions sampled from the posterior, and the posterior estimators. In general, the squared exponential kernel leads to highly regular functions (samples are almost surely continuously differentiable to all orders, analytic, and posterior estimators are drawn from the reproducing kernel Hilbert space built from linear combinations of the kernel function) while the exponential kernel leads to highly rough functions (samples are Markovian, posterior trajectories are nowhere differentiable, and posterior estimators are piecewise linear). By adopting a Matern kernel, it is possible to vary the properties of the posterior ensemble between these two extremes (e.g. to generate posteriors whose samples are almost surely differentiable up to a finite degree). For more details on kernel modeling, see Rasmussen and Williams Chapter 4.

Now, consider the joint model:

Prior: $f \sim \text{GP}(\mu, k)$

Measurement: $\vec{y} = f(\vec{x}) + \zeta$, where $f(\vec{x}) = [f(x_1), f(x_2), ..., f(x_n)]$

Noise: $\zeta \sim \mathcal{N}(0, \sigma_y^2 I_{n\times n})$

for any finite set of sample nodes $\vec{x}$.

Since f is drawn from a GP, $f(\vec{x})$ is multivariate normally distributed with mean $\mu(\vec{x})$ and covariance $K_{\vec{x},\vec{x}}$, the matrix with i, j entries $K(x_i, x_j)$. Then, since the noise is Gaussian with known covariance, this model is, for all finite sets of sample nodes, a normal, normal model!

This model can be easily extended to include a set of query nodes where we would like to infer the value of f. Let $\vec{q} = [q_1, q_2, ..., q_m]$ denote a collection of m query nodes, $q_j \in \mathcal{X}$. Let $\vec{c} = [\vec{x}, \vec{q}]$ denote the vector formed by composing the sample nodes and query nodes. Then, since f is drawn from a GP, the vector $f(\vec{c})$ is also mutlivariate normally distributed with:

$$f(\vec{c}) \sim \mathcal{N}(\mu(\vec{c}), K_{\vec{c},\vec{c}}) \text{ where } \mu(\vec{c}) = [\mu(\vec{x}), \mu(\vec{q})] \text{ and } K_{\vec{c},\vec{c}} = \begin{bmatrix} K_{\vec{x},\vec{x}} & K_{\vec{x},\vec{q}} \\ K_{\vec{q},\vec{x}} & K_{\vec{q},\vec{q}} \end{bmatrix}.$$

The covariance matrix for the composite vector can be computed by evaluating its blocks:

$$[K_{\vec{x},\vec{x}}]_{i,j} = k(x_i, x_j), \quad [K_{\vec{x},\vec{q}}]_{i,j} = k(x_i, q_j), \quad [K_{\vec{q},\vec{q}}]_{i,j} = k(q_i, q_j)$$

Then, we can represent our measurement $\vec{y} = Mf(\vec{c})$ where M is a block matrix of the form $M = \begin{bmatrix} I_{n\times n} & 0 \end{bmatrix}$

Now, we have translated our original problem into the standard normal, normal form. Estimating f at the query points $\vec{q}$ is the same as performing posterior inference on the unknown quantities $f(\vec{q})$ given:

$$\textbf{Prior: } f(\vec{c}) \sim \mathcal{N}(\mu(\vec{c}), K_{\vec{c},\vec{c}})$$
$$\textbf{Measurement: } \vec{y} = Mf(\vec{c}) + \zeta$$
$$\textbf{Noise: } \zeta \sim \mathcal{N}(0, \sigma_y^2 I_{n\times n}).$$

Then, $f(\vec{c}), \vec{y}$ are jointly normally distributed, so the posterior $f(\vec{c})|\vec{y}$ is also normally distributed, with mean and covariance determined by the usual update formulas we derived for the normal, normal model.

Simplifying yields the posterior:

$$f(\vec{q})|\vec{y} \sim \mathcal{N}(\hat{f}_{|\vec{y}}(\vec{q}), C_{f(\vec{q})|\vec{y}})$$

where:

$$\hat{f}_{|\vec{y}}(\vec{q}) = \mu(\vec{q}) + K_{\vec{q},\vec{x}}(K_{\vec{x},\vec{x}} + \sigma_y^2 I_{n\times n})^{-1}(\vec{y} - \mu(\vec{x}))$$
$$C_{f(\vec{q})|\vec{y}} = K_{\vec{q},\vec{q}} - K_{\vec{q},\vec{x}}(K_{\vec{x},\vec{x}} + \sigma_y^2 I_{n\times n})^{-1}K_{\vec{x},\vec{q}}$$

Notice that we now have a machine that can return a posterior estimate for the unknown function at any finite set of sample points, and, the posterior distribution at that set of samples if multivariate normal. It follows that, given a GP prior, and linear measurements with Gaussian noise, the posterior process is *also a GP!* In particular, the posterior ensemble corresponds to the GP:

$$f_{|\vec{y}} \sim \mathrm{GP}(\mu_{|\vec{y}}, k_{|\vec{y}})$$

with mean and covariance functions:

$$\mu_{|y}(x) = \mu(x) + k(x,\vec{x})(K_{\vec{x},\vec{x}} + \sigma^2 I_{n\times n})^{-1}(\vec{y} - \mu(\vec{x}))$$
$$k_{|y}(x,x') = k(x,x') - k(x,\vec{x})(K_{\vec{x},\vec{x}} + \sigma^2 I_{n\times n})^{-1}k(\vec{x},x').$$

Here we see a conjugate model over an infinite dimensional function space. Given a GP prior, and linear measurements with Gaussian noise, the posterior is also a GP.

The figure below (source) illustrates a 1-dimensional example. The left most panel illustrates sample functions drawn from the prior. The middle panel shows samples drawn from the posterior given a set of (noise-less) observations. Notice that there are many different possible functions that could pass through the observations. The rightmost panel summarizes the posterior distribution by plotting the expected value of the unknown function at each possible query location (solid blue line), and by drawing a shaded region representing a credible interval at each possible query location (grey shaded region). The dashed blue line in the left most panel illustrates a sample from the posterior. Notice that the sample is a reasonable interpolant.

Usually, the posterior is summarized by: 1. Returning the posterior mean estimator $\hat{f}_{|\vec{y}}(x) = \mathbb{E}[f(x)|\vec{y}] = \mu_{|\vec{y}}(x)$. Notice that, since the posterior distribution is Gaussian at each possible sample point, the posterior mean estimator is also the MAP estimator. 1. Returning a credible interval for the value of the unknown function at each query point, $I_{f|\vec{y}}(x) = \mu_{|\vec{y}}(x) \pm s\sqrt{k_{|\vec{y}}(x,x)}$ where the number of standard deviations, s, is adopted so that the credible interval covers a chosen fraction of the sample function draws. These are related by the standard percentiles of the Gaussian distribution. Usually, $s = 2$ or 3 to achieve 95% or 99.7% coverage.

Implementation: Q 1.1: In the space below write a code that can compute the posterior mean and covariance functions for any set of query points given $\mathcal{X} = \mathbb{R}$, an arbitrary prior mean function, and a stationary covariance with an arbitrary kernel $k(x,x') = \kappa(|x - x'|)$.

[22]:
```
# Insert your code here
def get_posterior_mean_and_covariance(query_pts, data_pts, data_vals, kernel, mean, sigma_f, l,
sigma_y):

    # Calculate the means
    mu_q = mean(query_pts)
    mu_x = mean(data_pts)
```

```
    # Calculate the covariance matrices
    K_xx = kernel(data_pts , data_pts , l, sigma_f)
    K_qq = kernel(query_pts, query_pts, l, sigma_f)
    K_qx = kernel(query_pts, data_pts , l, sigma_f)
    K_xq = K_qx.T
    K    = K_xx + sigma_y**2 * np.eye(K_xx.shape[0])
    Kinv = np.linalg.inv(K)

    # Calculate the posterior mean and covariance
    mu_post  = mu_q + K_qx @ Kinv @ (data_vals - mu_x)
    cov_post = K_qq - K_qx @ Kinv @ K_xq

    return mu_post, cov_post
```

Q 1.2: In the space below, write a pair of functions, one that implements the squared exponential kernel, and one that implements the exponential kernel.

[23]:
```
def squared_exponential_kernel(x1, x2, sigma_f, l):
    # Ensure inputs are numpy arrays.
    x1 = np.atleast_1d(x1)
    x2 = np.atleast_1d(x2)
    # Create a grid of differences: shape (len(x1), len(x2))
    diff = x1[:, None] - x2[None, :]
    # Compute the squared exponential kernel matrix.
    return sigma_f**2 * np.exp(-0.5 * (diff**2) / (l**2))

def exponential_kernel(x1, x2, sigma_f, l):
    # Ensure inputs are numpy arrays.
    x1 = np.atleast_1d(x1)
    x2 = np.atleast_1d(x2)
    # Create a grid of absolute differences.
    diff = np.abs(x1[:, None] - x2[None, :])
    # Compute the exponential kernel matrix.
    return sigma_f**2 * np.exp(-diff / l)
```

Q 1.3: Write a code that, given the mean and covariance function for a GP can: 1. Plot the posterior mean estimator at a set of query points 1. Create a shaded region showing the credible interval with 95% coverage at a set of query points 1. Plot the 99% lower and upper credible bounds on the unknown function value at a set of query points

[24]:
```
# Insert your code here
def plot_with_uncertainty(query_pts, data_pts, data_vals, mu_post, cov_post, title_text,
cover_prob=0.95, ax=None):

    # If no axis is passed, create one.
    if ax is None:
        _, ax = plt.subplots()

    # Calculate the standard deviation
    if cover_prob == 0.95:
        z = 1.96
        label_text = r'95\% confidence interval'
    elif cover_prob == 0.99:
        z = 2.575
        label_text = r'99\% confidence interval'
    else:
        raise ValueError('Cover probability must be either 0.95 or 0.99')

    # Plot the data
    ax.plot(data_pts, data_vals, 'ro', label='Data points')
    ax.plot(query_pts, mu_post, 'b-', label='Posterior mean')
```

```
    ax.fill_between(query_pts, mu_post - z * np.sqrt(np.diag(cov_post)), mu_post + z * np.sqrt(np.
↪diag(cov_post)), alpha=0.2, label=label_text)
    ax.set_title(title_text)
    ax.legend()

    return ax
```

Q 1.4: Write a code that can generate samples from a GP with a known mean and covariance function at an input set of sample points, and can plot the associated sample trajectories.

[25]:
```
def zero_mean(x):
    return np.zeros(x.shape)
```

Q 1.5: Suppose that $\vec{x} = [-5, 0, 1, 3, 10]$ and $\vec{y} = [1, 10, 9, 11, 6]$. Generate plots showing the posterior mean estimator, 95% credible interval, 99% upper and lower credible bounds, and 3 sample functions for 1,000 equally spaced query points between -8 and 12. For example visualizations see Figure 2.2 in Rasmussen and William Chapter 2.

Start with $\mu = 0$, $\sigma_f = 15$, $l = 1$, and $\sigma_y = 0.2$. Compare the results using a squared exponential and exponential kernel by creating two side-by side plots illustrating the difference in the posterior distribution over possible f. Comment on the differences you observe.

[37]:
```
# Define the parameters
mu      = zero_mean
sigma_f = 15
l       = 1
sigma_y = 0.2

def plot_left_and_right(sigma_f, l, sigma_y, mu=zero_mean):

    # Define the query points, data points, and data values
    query_pts = np.linspace(-8,12,1000)
    data_pts  = np.array([-5, 0, 1, 3, 10])
    data_vals = np.array([1, 10, 9, 11, 6])

    # Compute the posterior for the first kernel and plot
    post_mean1, post_cov1 = get_posterior_mean_and_covariance(
        query_pts, data_pts, data_vals,
        squared_exponential_kernel, mu,
        sigma_f, l, sigma_y
    )

    # Compute the posterior for the second kernel and plot
    post_mean2, post_cov2 = get_posterior_mean_and_covariance(
        query_pts, data_pts, data_vals,
        exponential_kernel, mu,
        sigma_f, l, sigma_y
    )

    # Create a figure with two subplots
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))

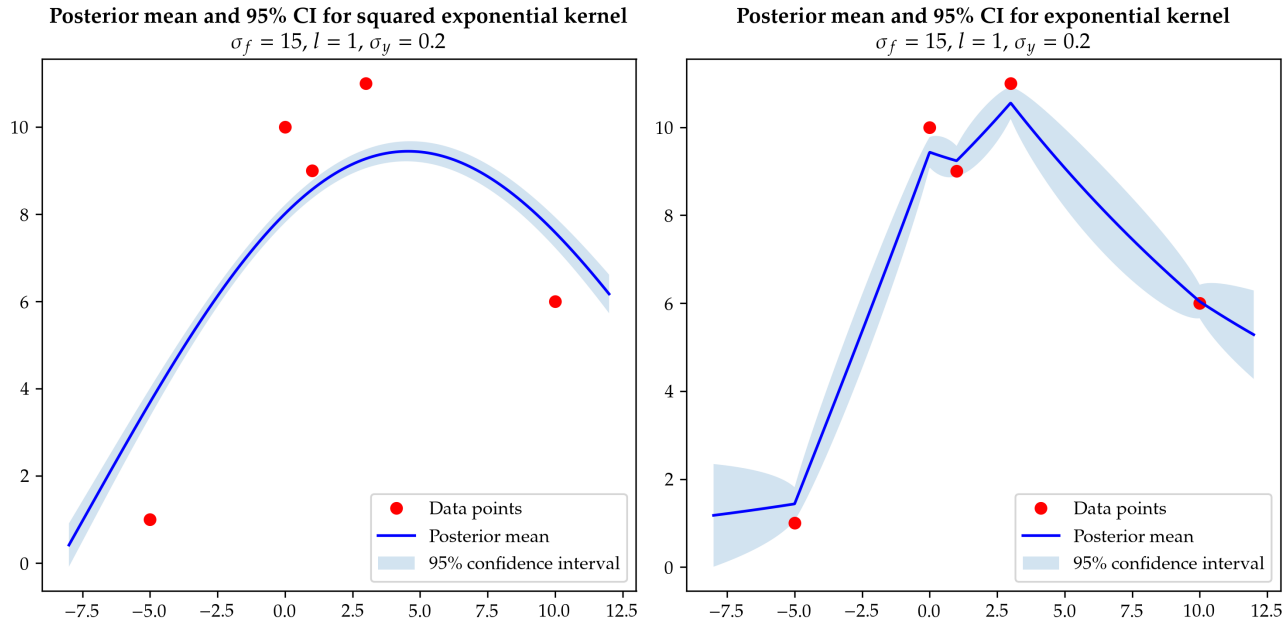
    # Pass the specific axes to the plotting function
    plot_with_uncertainty(query_pts, data_pts, data_vals,
                          post_mean1, post_cov1,
                          f'\\textbf{{Posterior mean and 95\% CI for squared exponential kernel}} \n
$\sigma_f = {sigma_f}$, $l = {l}$, $\sigma_y = {sigma_y}$',
                          ax=axes[0])

    plot_with_uncertainty(query_pts, data_pts, data_vals,
                          post_mean2, post_cov2,
```

```
                        f'\\textbf{{Posterior mean and 95\% CI for exponential kernel}} \n $\sigma_f
= {sigma_f}$, $l = {l}$, $\sigma_y = {sigma_y}$',
                        ax=axes[1])

    plt.tight_layout()
    plt.show()

plot_left_and_right(sigma_f, l, sigma_y, mu=zero_mean)
```

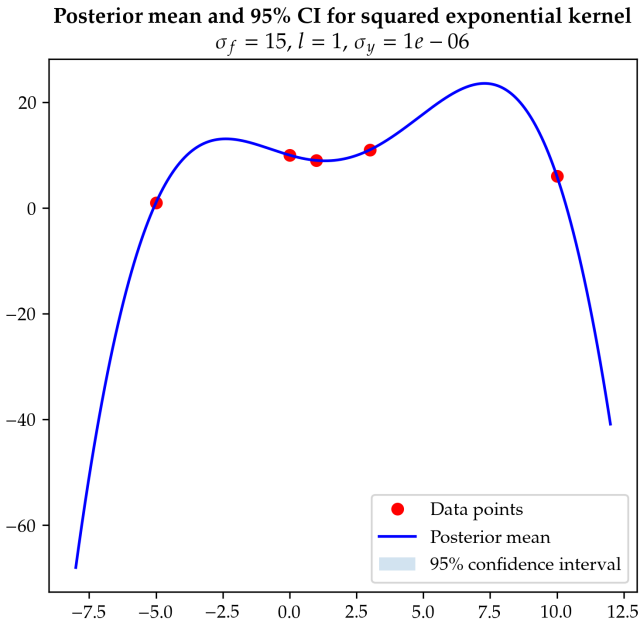


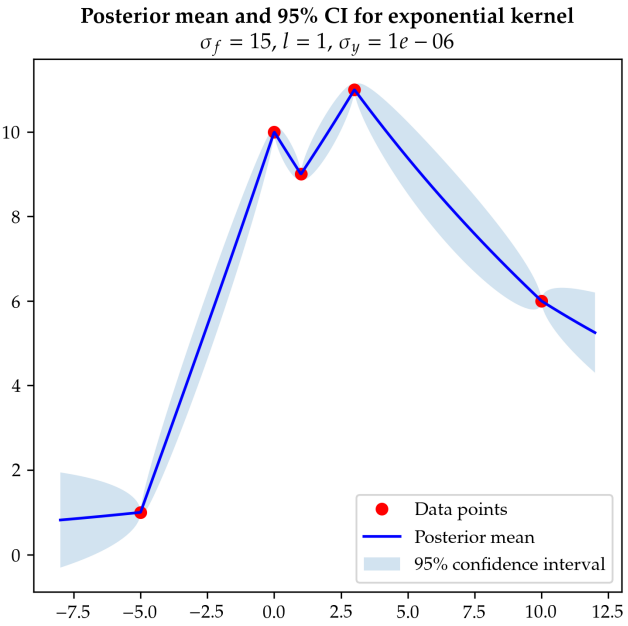
Q 1.6: Recreate the plots you generated above, but try setting: 1. $\sigma_y = 0$ and $\sigma_y = 8$. 1. $l = 0.25$ and $l = 5$.

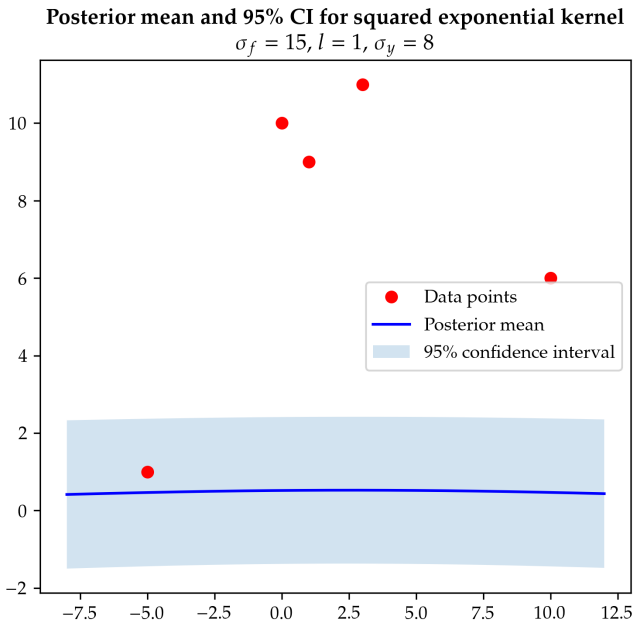
Comment on how the posterior over f changes as the likelihood changes (varying σ_y), and as the prior changes (varying the smoothness length scale l). Try to justify any phenomena you observe.

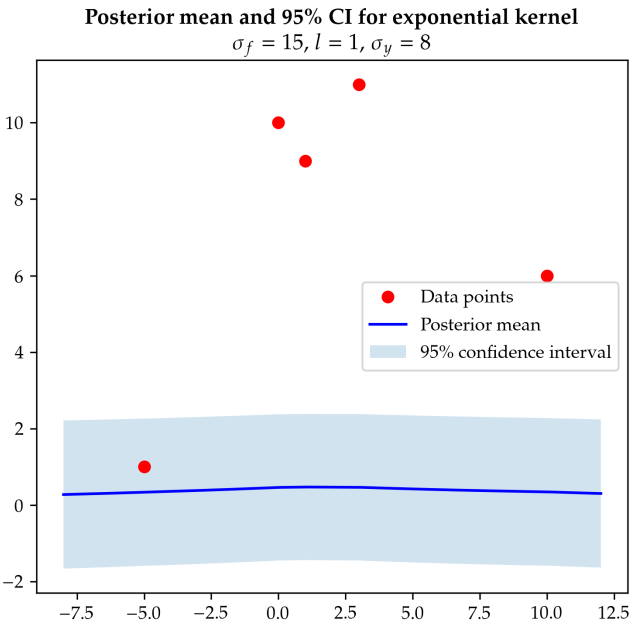
[42]:
```
# Insert your code and discussion here
plot_left_and_right(sigma_f, l  , sigma_y=0.000001)
plot_left_and_right(sigma_f, l  , sigma_y=8)
plot_left_and_right(sigma_f, l=0.25, sigma_y=0.2)
plot_left_and_right(sigma_f, l=5, sigma_y=0.2)
```

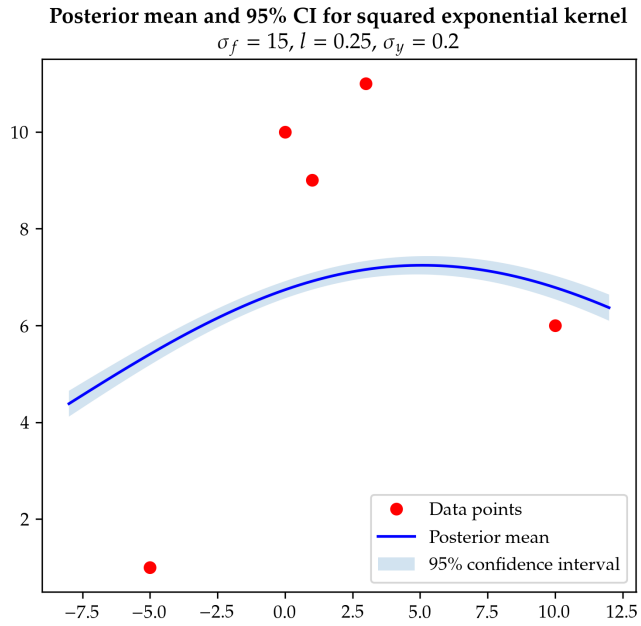
```
/var/folders/1b/4z_ghzx10cz5bmjpy5365sg40000gn/T/ipykernel_99220/808436057.py:21
: RuntimeWarning: invalid value encountered in sqrt
  ax.fill_between(query_pts, mu_post - z * np.sqrt(np.diag(cov_post)), mu_post +
z * np.sqrt(np.diag(cov_post)), alpha=0.2, label=label_text)
```

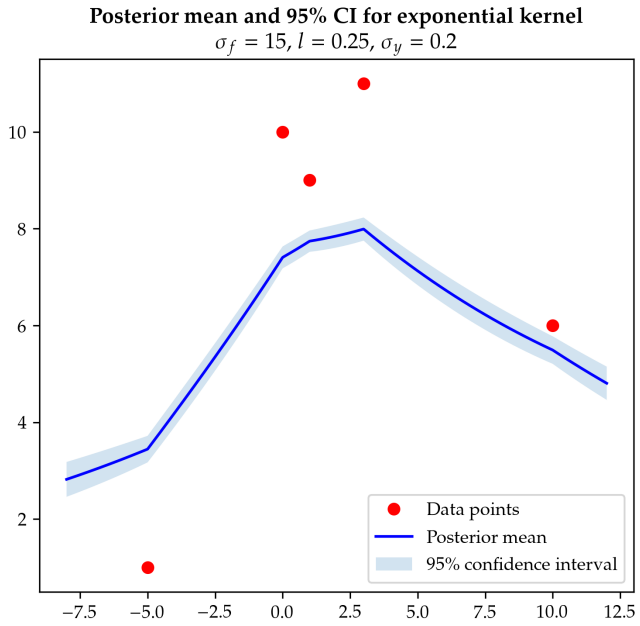

Posterior mean and 95% CI for squared exponential kernel
$\sigma_f = 15, l = 1, \sigma_y = 1e-06$
Data points
Posterior mean
95% confidence interval

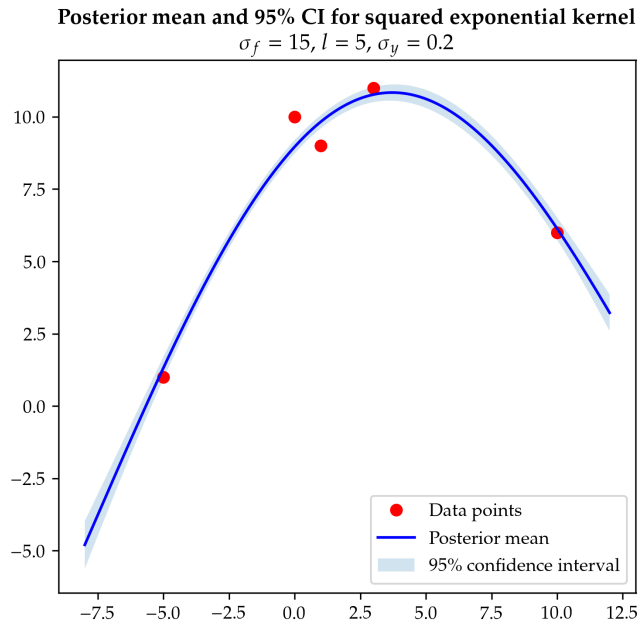

Posterior mean and 95% CI for exponential kernel
$\sigma_f = 15, l = 1, \sigma_y = 1e-06$
Data points
Posterior mean
95% confidence interval

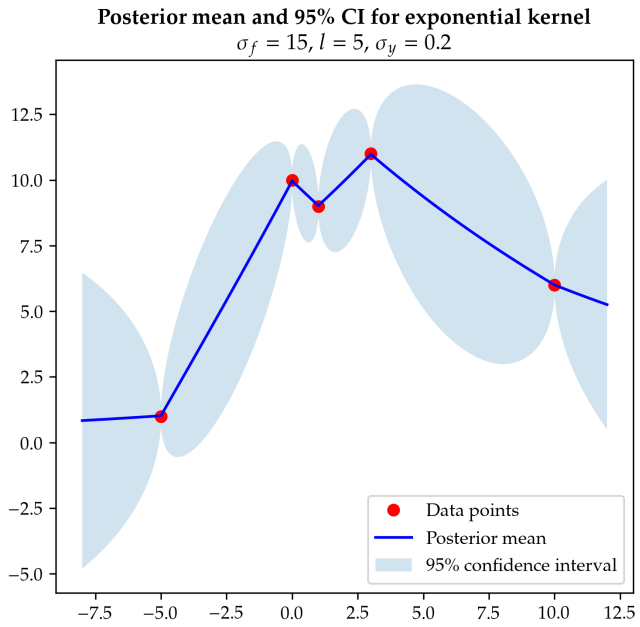

Posterior mean and 95% CI for squared exponential kernel
$\sigma_f = 15, l = 1, \sigma_y = 8$
Data points
Posterior mean
95% confidence interval


Posterior mean and 95% CI for exponential kernel
$\sigma_f = 15, l = 1, \sigma_y = 8$
Data points
Posterior mean
95% confidence interval









Part II: Surrogate Modeling and Bayesian Optimization

Background: Gaussian processes are particularly popular for *surrogate modeling*.

A surrogate model is a statistical model introduced to roughly approximate a more complicated underlying model, often, an unknown objective function. Typically the surrogate model should be fast to evaluate.

Here's a common setting:

You are asked to optimize an unknown function f over some space of possible inputs $\mathcal{X}$. You have access to an experimental procedure that returns noisy estimates of f at any sample input x, however, *each measurement is very expensive*.

In simulated situations, f can be evaluated by running a complex simulation, but the simulation is very slow, and if stochastic, returns noisy outcomes. Examples include complicated climate models, engineering models that depend on complex PDE solvers (e.g. tokamak design, aero or hydrodynamic simulation), biophysical models that involve extensive

Monte Carlo simulation involving many interacting particles (e.g. protein folding simulations), physical models that involve complicated geometries and particle cascades (e.g. particle accelerator design), or physical chemistry models for materials design (e.g. designing materials for carbon capture). In a physical setting, f may represent the idealized outcome of an experiment that may take weeks to run, and will cost real money to implement.

In all of these cases, it would be useful to have a fast, non-parameteric statistical model that can stand in as a surrogate for running real experiments, that can easily adapt to account for new measurements, and that can be used to guide new experiments.

Gaussian processes check all these boxes. They are fast to evaluate and update using the conjugate formalism developed above, nonparameteric, and, in many cases, sufficiently expressive. Their specification is also reasonably intuitive, so allows interpretable modeling. This is particularly important in the surrogate setting where we only aim for a weakly informative prior. Typically the surrogate is chosen so that sample draws from the surrogate model have the same regularity properties as the unknown objective (i.e. are continuous and sufficiently differentiable). This usually fixes the choice of kernel. They are also typically chosen so that, in absence of information, they will systematically under-estimate (for maximization) or over-estimate (for minimization) the maxima or minima of the objective. The standard σ_f is usually chosen to mildly over-estimate the range of possible objective function values, while the length scale l is usually chosen based on a rough guess at the roughness of the function (e.g. some guess at a Lipschitz constant/bound on its derivatives).

Bayesian optimization aims to select informative measurements by leveraging a surrogate model. The surrogate model is updated after each new measurement, and is used to propose new measurements. In general, we assume that the measurement procedure is vastly more expensive than any computational procedure applied to the surrogate model. So, you may assume that the cost of the method is proportional to the number of true function evaluations it demands (sample efficiency).

These procedures are designed to balance an exploration-exploitation trade-off. We want to propose new measurements that simultaneously provide as much information as possible about the optimal input, while validating the quality of proposed inputs by showing that $f(x)$ is large at any x proposed as a candidate optimizer.

Two standard procedures are outlined below:

1. **Thompson Sampling:** This approach adopts a greedy randomized heuristic; *"Select the input that optimizes a draw from the posterior."* After each observation, sample a trajectory from the GP posterior, then select the input point that maximizes the sampled trajectory. This can be accomplished by fixing a set of query points, and evaluating the posterior sample at those query points, or by optimizing the sampled trajectory with a standard optimization routine.

2. **Upper-Credible Bound (UCB):** This approach adopts a greedy optimist heuristic; *"Select the input x for which $f(x)$ could plausibly be the largest."* UCB is the reference/standard method for Bayesian optimization.

   To implement UCB, define the upper confidence bound function $b(x, p)$ where:

$$b(x, p) = \inf\{b | \Pr(f(x) < b) \geq p\}$$

   for some success probability p. Notice that, if the surrogate model uses a GP prior, and assumes Gaussian measurement noise, then $b(x, p) = \mu_{|\vec{y}}(x) + s(p)\sqrt{k_{|\vec{y}}(x, x)}$ where $s(p)$ is the p percentile of the standard normal distribution. The first term rewards inputs x which are expected to return large f. The second rewards x where f is poorly resolved.

   To select an input, search for the input $x_*(\vec{y}, p) = \text{argmax}_{x \in \mathcal{X}}(b(x, p))$. This optimization problem is ugly since $b(x, p)$ is usually a poorly behaved function of x. However, evaluating the surrogate model is cheap, so evaluating b is cheap. Usually, we adopt $p > 0.5$ and let p increase towards 1 with each subsequent observation. This ensures that, as the number of measurements increase, we demand increasing certainty in our upper confidence bounds.

Q 2.1: With your group, propose a loss function on the inferred function value and the true function value, whose posterior risk is minimized by setting the estimator equal to $b(x, p)$ (see HW 2). Explain why this loss function is sensible for an iterative optimization procedure that selects the next sample point by optimizing the estimated function value. Think carefully about what happens if you underestimate the true function value vs overestimate the function value.

Insert your answer here.

Demonstration: Bayesian optimization is particularly useful for highly multimodal objective functions in low-dimensional, or constrained $\mathcal{X}$, since it is based on global optimization of the surrogate. When $\mathcal{X}$ is low-dimensional, it is possible to exhaustively optimize the surrogate, and thus, to suggest sequences of inputs that converge to the global minimizer of the true objective f without getting stuck in local minima. In this part, you'll test this idea.

To illustrate the efficacy (or inefficacy) of Bayesian optimization using GP surrogates, consider the following three standard test functions. In all cases, we seek to find the global minimizer of these functions.

1. The 1D Levy function: $\mathcal{X} = [-10, 10]$ where:
$$f(x) = \sin^2(\pi w) + (w-1)^2(1 + 10\sin^2(\pi w + 1) + (w-1)^2(1 + \sin^2(2\pi w)$$
for $w = 0.75 + 0.25x$. The Levy function has a minimum of 0 at $x_* = 1$.
2. The 2D Ackley function: $\mathcal{X} = [-32.768, 32.768] \times [-32.768, 32.768]$ where:
$$f(x) = 20\exp\left(-0.2\sqrt{\frac{1}{2}\sum_{j=1}^{d} x_j^2}\right) - \exp\left(\frac{1}{2}\sum_{j=1}^{2}\cos(2\pi x_j)\right) + 20 + e$$
whose global minimum is 0 at $x_* = [0, 0]$.
3. The 2D Michalwicz function: $\mathcal{X} = [0, \pi] \times [0, \pi]$ where:
$$f(x) = -\sum_{j=1}^{2}\sin(x_j)\sin\left(\frac{j}{\pi}x_j^2\right)^{10}$$
whose global minimum is -1.8 at $x_* = [2.20, 1.57]$.

Q 2.2: In the space below implement the Levy test function. *Note: to complete the lab, you will only need the Levy function. The other pair of test functions are optional, to be used to complete a mini project.*

[ ]:
```
# Insert code here
def levy(x):
    w = 0.75 + 0.25*x
    f_x = np.sin(np.pi*w)**2 + (w-1)**2 * (1 + 10*np.sin(np.pi*w+1)**2) + (w-1)**2 * (1 + np.
↪sin(2*np.pi*w)**2)
    return f_x
```

Q 2.3: Based on your experiments in part 1, and inspection of the test functions (follow the links above) propose a GP surrogate. Choose between an exponential or square exponential kernel, suggest a choice for the length scale l, and standard deviation $\kappa(0) = \sigma_f$.

Then, suggest an alternate surrogate choice that you expect to perform poorly. Think about the regularizing effect of the prior model when selecting your surrogates, and how the biases incurred by the prior interact with the optimization routine.

Propose your surrogates here. Explain why you think the first will perform well, and why the second will perform poorly.

Proposed *Well*-Performing Surrogate: - Squared Exponential Kernel — - Length scale $\ell = 0.3$ — - Standard deviation $\sigma_f = 0.3$ —

Proposed *Poor*-Performing Surrogate: - Squared Exponential Kernel — - Length scale $\ell = 0.3$ — - Standard deviation $\sigma_f = 0.3$ —

Q 2.4: In the space below prepare two separate codes, one that implements UCB, and one that implements Thompson Sampling for a true objective f, domain $\mathcal{X}$, noise level σ_y^2, kernel function κ, and smoothness length scale l. - *You need only prepare code for the Levy case. As above, the Ackley and Michalwicz examples are for a project mini.* - You may assume that $\mu = 20$ for the Levy and Ackley functions, and $\mu = 0$ for the Michalwicz function. These are intentionally pessimistic choices that must be disproved by near-to-optimal samples. - You may assume that $\mathcal{X}$ is an interval in one-dimension, or a rectangular region in two-dimensions - To optimize over query points simply evaluate the acquisition function (either the sampled surrogate function, or the credible bound) on a finely spaced grid. If multiple sample points return the same acquisition value, pick between them uniformly at random. More sophisticated methods use an adaptive grid that adapt to a partitioning of $\mathcal{X}$ induced by the current set of sampled nodes. - To initialize, select a sample point uniformly at random.

[ ]:
```
# Insert code here
```

Q 2.5: Apply UCB (*Lower*-Credible Bound for minimization), and Thompson sampling to minimize the Levy function using your chosen kernel and best guess at the smoothness parameter l. Terminate after 160 iterations. Use your code from Part 1 to display the posterior distribution used for selecting the next sample node after 10, 20, 40 and 160 iterations. Set $\sigma_y = 0$. Add scatter points for each observed sample node, function value pair, and add a vertical line for the next proposed sample location.

[30]:
```
# Insert code here
```

Q 2.6: Repeat 2.5 but try increasing σ_y to 0.1, then to 1. Only produce plots for iterations 10, 40, and 160. Comment on how the behavior of the optimizer changes.

[31]:
```
# Insert code here
```

Completing the subsequent parts for all three test functions is optional, and will contribute to the completion of a project mini. For the lab, you need only complete the subsequent tasks for the Levy test function.

Q 2.7: Apply UCB (LCB for minimization), and Thompson sampling to minimize each of the three test functions using your preferred surrogate model. Return two sets of three (*if you run all three tests*) plots: - Make three adjacent plots, one per objective, showing the locations of the sample nodes selected for evaluation. Use a scatter plot, and color the nodes according to their iteration number (e.g. blue for early iterations, red for late iterations), and change the marker to distinguish UCB from Thompson. Jitter the points slightly so that it is possible to distinguish multiple evaluations of the same sample node. For the Levy function, overlay the scatter cloud on a plot of the true objective. For the 2D test functions, overlay the scatter on a heatmap showing the true objectives. Set the transparency of the heat map to 0.5 or less so that the scatter points are clearly visible.

- Make three comparison plots, one per objective, showing the sequence of sampled objective function values (scatter), and the running minimum of the sampled objective function values (line) under each optimization rule. Your plots should include three panels, one per test function. Overlay the sampled values, and running minimums, for each approach. Add a legend to distinguish Thompson sampling from UCB. Set the minimum value of the vertical axis to the true minimal value of each test function. For this experiment you may use $\sigma_y = 0$.
- Stop the optimization process after, at most, 160 iterations. You may terminate earlier if the value of the sampled objective functions converge near the true minimum values provided above.

[32]:
```
# Insert code here
```

Q 2.8: Repeat the same process but select means μ that underestimate the desired minima (say $\mu = -0.2$ for the Levy and Ackley functions, and $\mu = -2$ for the Michalwicz function). How does the behavior of the Bayesian optimizer change? (Hint: consider how it prioritizes exploration vs exploitation when μ underestimates the minimizer.)

[33]:
```
# Insert code here
```

Q 2.9: Repeat the same process, but using the surrogate choice you expected to perform poorly. Comment on the result.

[34]:
```
# Insert code here
```

Q 2.10: Comment on the efficacy of your Bayesian optimizers in the test cases proposed here. Focus on sample efficiency (the number of samples of f that are evaluated) to evaluate quality and discuss the sensitivity of the method to the choice of surrogate model. Do you think it would be easy to select a suitable surrogate in practical problems? What principles for prior modeling would guide its selection?

Insert commentary here.

Lab 6: Bayesian Logistic Regression

Introduction

This lab has been modified into a tutorial activity.

In this tutorial, you can explore the regularizing impact of different features of a prior in a Bayesian logistic regression problem. You will learn how to: 1. Understand a logistic likelihood model 2. Select non-conjugate priors for their regularizing effects 3. Compare the biasing effects of different priors based on their tail-behavior

The data in this lab is drawn from: - Racine et al. *Bayesian Methods in Practice: Experiences in the Pharmaceutical Industry.* Journal of the Royal Statistical Society. 1986. JSTOR link (this is the source for the toxicity data set used in BDA Chapter 3.7 and in Chapter 5)

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

[3]:
```
# load whatever packages you prefer here. We've added a reference list here
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm, beta
```

Data

[26]:
```
# Load toxicity data
file_path = "toxicology_data.csv"
toxicity_data = pd.read_csv(file_path)
print(toxicity_data)
print(np.log(toxicity_data['Dose_(mg/ml)']))
```

```
   Dose_(mg/ml)  Number_Exposed  Number_Harmed
0           422               5              0
1           744               5              1
2           948               5              3
3          2069               5              5
0    6.045005
1    6.612041
2    6.854355
3    7.634821
Name: Dose_(mg/ml), dtype: float64
```

Bayesian Logistic Regression Models

In this lab, you will explore the regularizing effects of different prior models on two generalized linear models (likelihood models that compose a linear combination of features with a nonlinear link). Here we focus on a logistic regression model for the toxicity of a compound.

Likelihoods:

Toxicity Study The following example is based on the first logistic regression example in BDA (see Chapter 3.7, page 74). The data is excerpted from Racine et al.'s review article on applications of Bayesian statistics in the pharmaceutical industry.

The article introduces a Bayesian regression model for the toxicity of a substance based on an animal trial. While establishing the toxicity of chemical substances is important in pharmaceutical research, animal trials that harm their subjects should only be performed when necessary and no reasonable alternatives exist. Ethical and legal requirements demand that any such studies should be performed on as few animals as possible while retaining enough statistical power to establish the conclusions needed for safe drug development. Since these studies were, as of Racine's writing, performed

routinely, past information, or information from studies on similar substances, could be used to inform priors. The need for robust analyses from small sample sizes motivates the use Bayesian analysis in these examples. For further reasons to adopt a Bayesian approach, see Racine et al.

The data consists of four trials for the acute toxicity of a substance, each involving five subject animals. Note the small number of subjects used in the reported trial (20 in total, 5 per treatment group). The dose $\{x_i\}_{i=1}^4$ varied between the four groups, increasing from 422 mg/ml to 2069 mg/ml. In each, the number of animals killed by the substance, $\{y_i\}_{i=1}^4$, was recorded.

Following BDA, consider the logistic regression model:

$$\begin{aligned}
&\textbf{Linear Comb. of Features:} && \eta(x,\beta) = \beta_0 + \beta_1 \log(x) \\
&\textbf{Logistic Link:} && q(x,\beta) = \text{logistic}(\eta(x,\beta)) = [1+\exp(-\eta(x,\beta))] \\
&\textbf{Binomial Sampling Dist:} && Y|x,\beta \sim \text{Bin}(n, q(x,\beta)).
\end{aligned}$$

Then, given the data $\{(x_i, y_i)\}_{i=1}^4$, the likelihood of the regression parameters β is:

$$\text{Lik}(\beta) \propto \prod_{i=1}^4 q(x_i,\beta)^{y_i}(1-q(x_i,\beta))^{n-y_i} = \prod_{i=1}^4 (1+e^{-\beta_0-\beta_1\log(x_i)})^{-y_i}(1+e^{\beta_0+\beta_1\log(x_i)})^{-(n-y_i)}$$

We want to infer the regression coefficients, $\beta = [\beta_0, \beta_1]$ from the data $\{(x_i, y_i)\}_{i=1}^4$. In particular, toxicity regulation is based on the dosage level x for which the substance has a 50% chance of causing harm (here defined as causing a fatal reaction). This is the level at which $\eta(x,\beta) = 0$, or, is $x_*(\beta) = \exp(-\beta_0/\beta_1)$. We will study how the posterior on x_* changes under different prior specifications.

Toxicity: Define likelihood function for logistic regression

[5]:
```
def log_likelihood_toxicity(beta, data):
    log_prob = 0
    for _, row in data.iterrows():
        x, y = row['Dose_(mg/ml)'], row['Number_Harmed']
        eta = beta[0] + beta[1]*np.log(x)

        log_prob += -y*np.log(1 + np.exp(-eta)) - (5 - y)*np.log(1 + np.exp(eta))
    return log_prob
```

Priors:

Toxicity Study: We will consider four possible choices of prior. They are listed below in order of generality. Each includes its predecessors as a special case.

1. An improper uniform prior on β (see both BDA and Racine)
2. A Gaussian prior on β formed from consultation with toxicology experts (see Racine):

$$\beta \sim \mathcal{N}(\bar{\beta}, C_\beta) \text{ where } \bar{\beta} = [-17.31, 2.57], \quad C_\beta(\sigma) = \begin{bmatrix} 1053.72 & -156.45 \\ -156.45 & 23.24 \end{bmatrix} + \sigma^2 I$$

   where $\sigma > 0$ is a free parameter introduced to weaken the prior. When $\sigma = 0$, this prior is singular (places all of the prior density on a line).
3. A Gaussian prior on β designed to introduce a stronger regularizing effect. In particular, use the covariance:

$$R_\beta(\sigma,\mu) = (C_\beta^{-1} + \mu^{-2} I)^{-1}$$

   Adding a multiple of the identity inside the inverse has the effect of soft-thresholding the largest eigenvalue of the covariance from above. The smaller μ, the larger the regularizing effect.
4. A yet weaker prior formed by replacing a Gaussian density with a Student-t density

$$\beta \sim p_\beta \text{ where } p_\beta(\beta) \propto \left(1 + \frac{1}{\nu}(\beta-\bar{\beta})^\intercal R_\beta^{-1}(\beta-\bar{\beta})\right)^{-\frac{\nu+2}{2}}$$

   where ν is a shape parameter. When $\nu = 1$ the prior is a multivariate Cauchy. When $\nu \to \infty$, the prior converges to the multivariate normal prior with mean $\bar{\beta}$ and covariance R_β.

Notice that the 4th prior family encompasses the previous three. As such, it is enough to implement (4), then pick the parameters appropriately. The first is recovered by setting μ arbitrarily close to zero. The second is recovered by setting μ and ν arbitrarily large. The third is recovered by setting ν arbitrarily large. In general, the prior parameters σ, μ, ν may be interpreted as:

1. σ weakens the expert provided prior by adding variance. When $\sigma = 0$ the expert provided prior assumes that β_0 and β_1 satisfy a strict linear relationship. Using (small) nonzero σ allows some variation about this relationship.
2. μ tightens the prior by introducing a soft threshold on the prior variance. The smaller μ, the smaller the allowed variance. This is an important regularization parameter since the expert provided prior allows a large degree of variation in the regression parameters along a direction in the parameter space that is not well constrained by the likelihood.
3. ν relaxes the tails of the prior so that the tails decay according to a power-law. This fat-tailed behavior allows more outliers when sampling from the prior, so does not as strongly suppress the tails of the posterior. This allows more robust inference since, the slower the prior tails decay, the less bias is introduced by the prior given data that suggests an outlier. The slower the tailes decay, the fewer samples are needed to move the posterior towards the likelihood when the likelihood suggests that the true parameters would have been surprising under the prior. This makes the posterior inference more reactive to the likelihood (it is more amenable to accepting outliers), so increases the variance in the posterior with respect to sampling while decreasing the biases introduced by regularization.

[6]:
```
# Implement the uniform prior. Your code should return the log prior density up to an additive
constant
def log_prior_uni_toxicity(beta):
    return 0
```

[7]:
```
# Implement the normal prior. Your code should return the log prior density up to an additive
constant
def log_prior_normal_toxicity(beta, sigma):
    beta_centered = beta - np.array([-17.31,2.57])
    C = np.array([[1053.72, -156.45], [-156.45, 23.24]]) + (sigma**2)*np.eye(2)
    log_prob = -0.5*beta_centered.T @ np.linalg.inv(C) @ beta_centered
    return log_prob
```

[8]:
```
# Implement the regularized normal prior. Your code should return the log prior density up to an
additive constant
def log_prior_reg_normal_toxicity(beta, sigma, mu):
    beta_centered = beta - np.array([-17.31,2.57])
    C = np.array([[1053.72, -156.45], [-156.45, 23.24]]) + (sigma**2)*np.eye(2)
    R = np.linalg.inv(np.linalg.inv(C) + mu**(-2)*np.eye(2))
    log_prob = -0.5*beta_centered.T @ np.linalg.inv(R) @ beta_centered
    return log_prob
```

[9]:
```
# Implement the student-t prior. Your code should return the log prior density up to an additive
constant
def log_prior_student_toxicity(beta, sigma, mu, nu):
    beta_centered = beta - np.array([-17.31,2.57])
    C = np.array([[1053.72, -156.45], [-156.45, 23.24]]) + (sigma**2)*np.eye(2)
    R = np.linalg.inv(np.linalg.inv(C) + mu**(-2)*np.eye(2))
    quadratic_term = beta_centered.T @ np.linalg.inv(R) @ beta_centered
    log_prob = -((nu + 2)/2)*np.log(1 + quadratic_term/nu) # the larger nu, the closer this is to
Gaussian
    return log_prob
```

Posterior Inference

Sampling:

The code below implements a Markov chain Monte-Carlo sampler (MCMC) that can approximately sample from a posterior distribution by iteratively proposing new values for the unknowns, and accepting or rejecting the proposals based on the

posterior odds. The specific algorithm is an instance of the Metropolis-Hastings method. We will discuss this in the next unit. For now, you may accept this as a black box tool that produces samples from the posterior.

[ ]:
```
def metropolis_hastings(beta, data, prior_param, iterations=200000, step_size=0.1, burn_in = 1000,
stride = 100):
    # initialize chain
    samples = []

    current_ll = log_likelihood_toxicity(beta, data)

    sigma = prior_param[0]
    mu = prior_param[1]
    nu = prior_param[2]
    current_lpr = log_prior_student_toxicity(beta, sigma, mu, nu)

    current_lpo = current_ll + current_lpr

    k = 0 # counter for step of sampler

    # iterate the Markov chain
    for _ in range(iterations):
        # propose new sample location
        new_beta = beta + step_size*np.random.normal(0, [7,1], size=2) # default scales [7, 1]

        # evaluate proposal
        new_ll = log_likelihood_toxicity(new_beta, data)

        new_lpr = log_prior_student_toxicity(new_beta, sigma, mu, nu)

        new_lpo = new_ll + new_lpr

        # decide whether to accept or reject the proposal
        acceptance_ratio = np.exp(min(new_lpo - current_lpo, 0)) # set the acceptance ratio equal to
the posterior odds, capped at 1
        if np.random.rand() < acceptance_ratio:
            beta = new_beta
            current_lpo = new_lpo

        # add to the running list of samples
        k += 1
        if np.mod(k,stride) == 1 and k >= burn_in:
            samples.append(beta.copy())
    return np.array(samples)
```

[11]:
```
# Set the prior parameters
sigma = 0.1 # default 0.1 (reasonably close to the expert provided prior)
mu = 10 # default 10 (weak regularization)
nu = 40 # defualt 40 (close to Gaussian tails)

# Intialize to the expert prior expectation
beta = np.array([-17.31,2.57])

# Run MCMC sampling ... this may take a while. Let it simmer
prior_param = np.array([sigma, mu, nu])
post_samples = metropolis_hastings(beta, toxicity_data, prior_param)
```

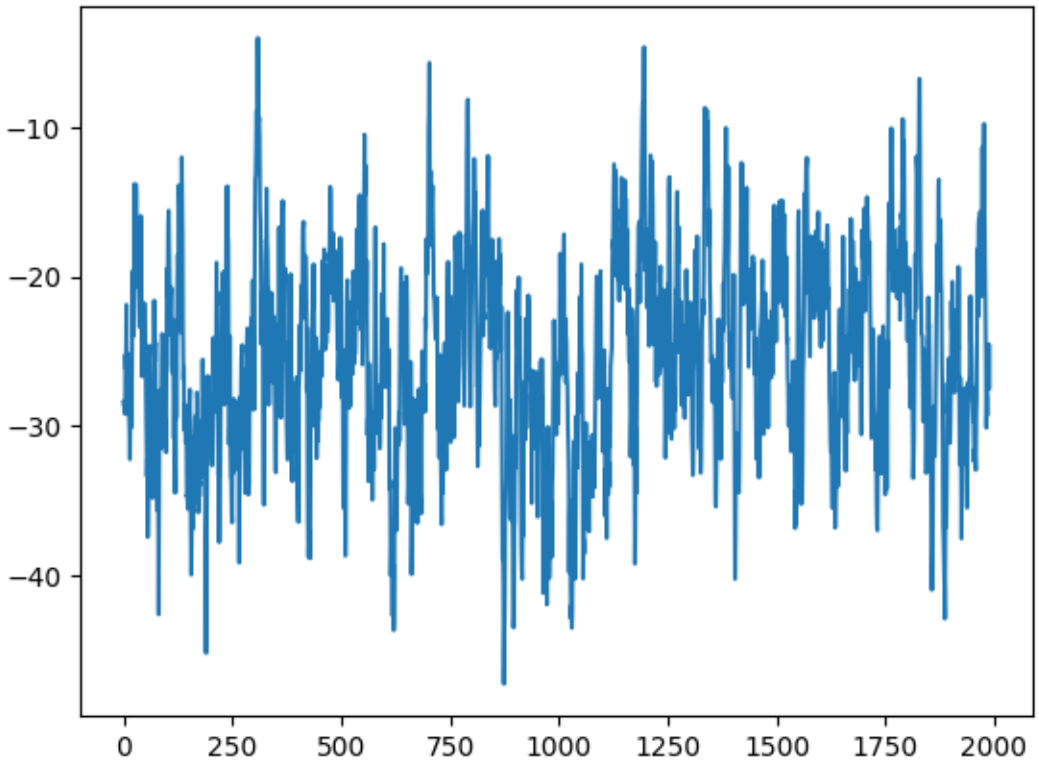
The plot commands below are good practice. It is good practice to inspect the sample runs. You should see a stochastic process where the entries are not overly correlated with each other (the process should look noisy and not overly smooth), and, where it appears to have mixed over its stationary distribution. This means that, the empirical distribution of values on any smaller interval, selected at random, should be close to the empirical distribution over the full sample. In other words,

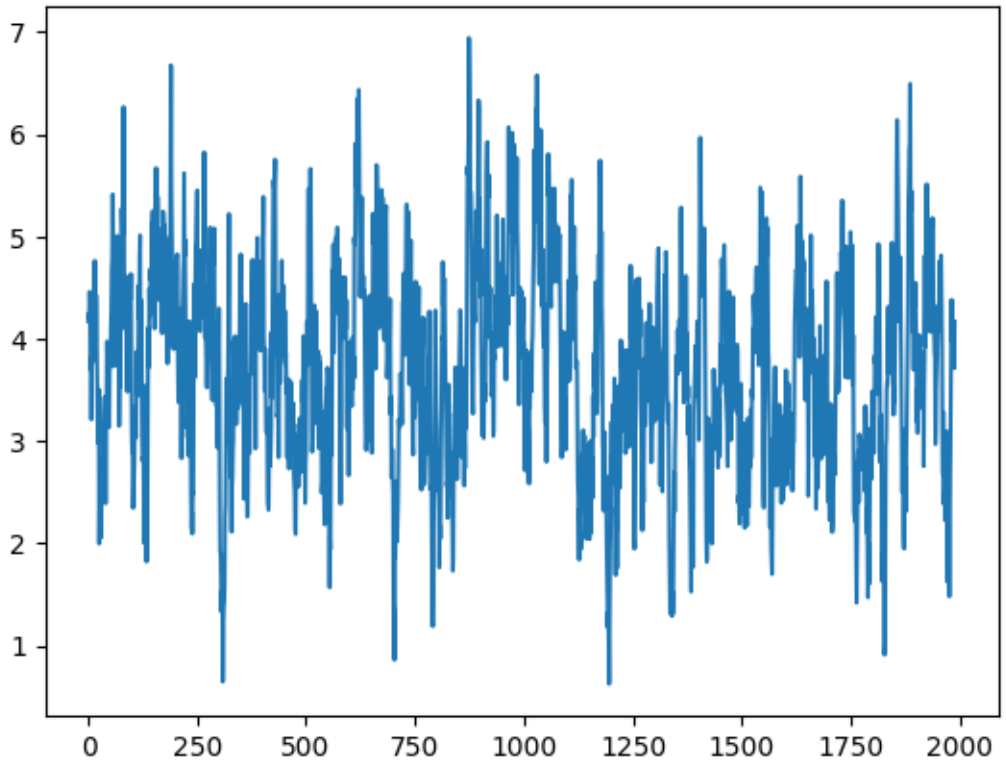
the process should look like a stationary noise process. It should not show a general trend in any direction, and you should not be able to change the distribution of samples significantly by truncating, or zooming in on an interval of roughly the same length as the full sample (say, an interval 1/5th the length of the full interval).

If you want to see this sampler fail (and how a check of this kind can detect failures), try changing the sampling parameters for a prior with a weak regularizing effect (small σ, large μ, small η). In particular, try decreasing the `step_size` parameter, or, changing the Gaussian distribution used to perturb the samples so that the scales parameters are `[1,1]` instead of `[7,1]`.

[12]:
```
# inspect behavior of the run to check MCMC parameters
plt.plot(post_samples[:,0])
plt.show()

plt.plot(post_samples[:,1])
plt.show()
```





[13]:
```
# compute posterior samples for x_*
x_stars = np.exp(-post_samples[:,0]/post_samples[:,1])
```

Posterior Summary:

The code below returns approximations to the: 1. **posterior mean** estimator for each entry of β 2. **95% credible intervals** for each entry of β
using the sampled regression coefficients.

[14]:
```
betas_posterior_means = post_samples.mean(axis=0)
betas_credible_intervals = np.percentile(post_samples, [2.5, 97.5], axis=0)

print("The posterior mean for the intercept is: {}".format(np.round(betas_posterior_means[0],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals[:,0],1)))

print("")

print("The posterior mean for the slope is: {}".format(np.round(betas_posterior_means[1],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals[:,1],1)))
```

```
The posterior mean for the intercept is: -25.3
with 95 percent credible interval: [-38.6 -12.2]

The posterior mean for the slope is: 3.7
with 95 percent credible interval: [1.8 5.6]
```

The code below returns approximations to the: 1. **posterior mean** estimator for $x_*(\beta)$ 2. **95% credible intervals** for $x_*(\beta)$ using the sampled regression coefficients.

[15]:
```
x_stars_posterior_means = x_stars.mean(axis=0)
x_stars_credible_intervals = np.percentile(x_stars, [2.5, 97.5], axis=0)
```

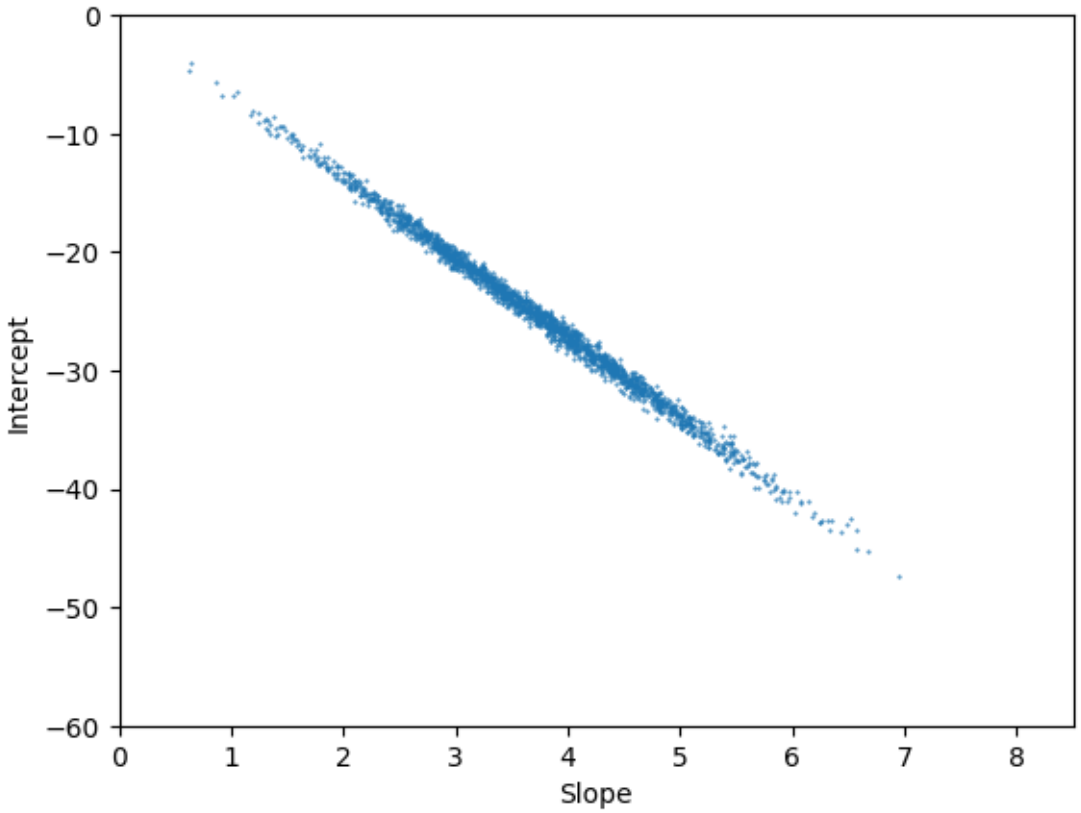
```
print("The posterior mean for the 50 percent lethality dose is: {}".format(np.
  ↪round(x_stars_posterior_means,1)))
print("with 95 percent credible interval: {}".format(np.round(x_stars_credible_intervals,1)))
```

```
The posterior mean for the 50 percent lethality dose is: 922.4
with 95 percent credible interval: [ 692.5 1262.8]
```

Posterior Visualization:

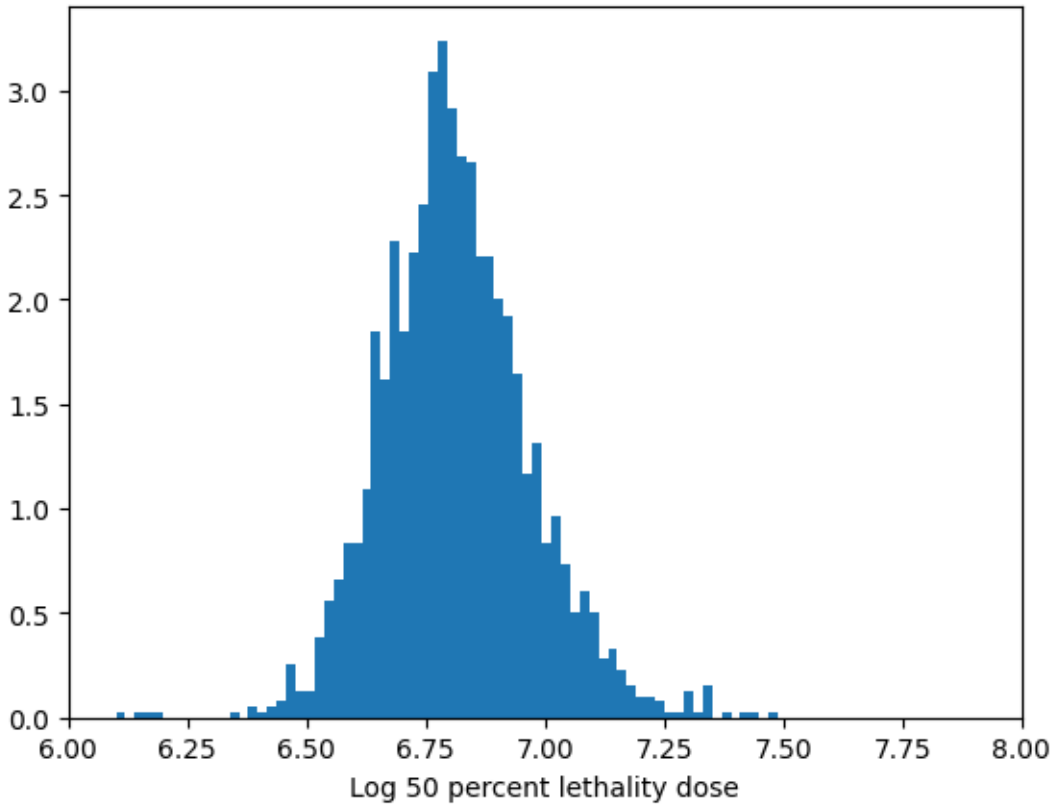
First, let's look at the resulting posterior distribution on the regression coefficients β. This is approximated below using 2,000 samples drawn according to the code provided above.

[16]:
```
plt.scatter(post_samples[:,1],post_samples[:,0],s = 0.2)
plt.ylim([-60,0])
plt.xlim([0,8.5])
plt.xlabel("Slope")
plt.ylabel("Intercept")
plt.show()
```



Next, let's visualize the posterior for $\log(x_*(\beta))$, the log concentration at which the compound is expected to kill 50% of subjects exposed to it. For consistency with the analysis in Racine, this posterior should put most of its mass between 6.5 and 7, with a peak near 6.75.

[17]:
```
plt.hist(np.log(x_stars),70,density = True)
plt.xlim([6,8])
plt.xlabel("Log 50 percent lethality dose")
plt.show()
```



Instructions for Experimentation:

For each prior specification below, rerun the sampling and posterior visualization code. Inspect the change in each posterior summary, then reason with the change in the posterior. Which regularization do you think is most reasonable as an inferential argument? How would you argue your choice to a regulator (think about which biases are conservative, and the differential impacts of under and overestimation of the 50 percent lethal dose)? Which set would you trust? What is the range of inferences you would be willing to return as the possible inferences that are plausible under a range of plausible prior choices?

Uniform

Start with a near to uniform prior. This can be achieved by setting σ very large, μ very large, and ν very small. Try $\sigma = 400$, $\mu = 1000$ and $\nu = 1$. This is a very flat prior. You should notice that, since the data is "well separated", the likelihood alone places essentially no upper bound on the unknown parameters. This is a case where the MLE would be highly sensitive to changes in the data, and may be unbounded above.

[32]:
```
# Set the prior parameters
sigma = 400 # default 0.1 (reasonably close to the expert provided prior)
mu = 1e3 # default 10 (weak regularization)
nu = 1 # defualt 40 (close to Gaussian tails)

# Intialize to the expert prior expectation
beta = np.array([-17.31,2.57])

# Run MCMC sampling ... this may take a while
prior_param = np.array([sigma, mu, nu])
post_samples_new = metropolis_hastings(beta, toxicity_data, prior_param)

# compute posterior samples for x_*
```

```
x_stars_new = np.exp(-post_samples_new[:,0]/post_samples_new[:,1])
```

[33]:
```
# print summaries
betas_posterior_means_new = post_samples_new.mean(axis=0)
betas_credible_intervals_new = np.percentile(post_samples_new, [2.5, 97.5], axis=0)

print("The posterior mean for the intercept is: {}".format(np.round(betas_posterior_means_new[0],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,0],1)))

print("")

print("The posterior mean for the slope is: {}".format(np.round(betas_posterior_means_new[1],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,1],1)))

x_stars_posterior_means_new = x_stars_new.mean(axis=0)
x_stars_credible_intervals_new = np.percentile(x_stars_new, [2.5, 97.5], axis=0)

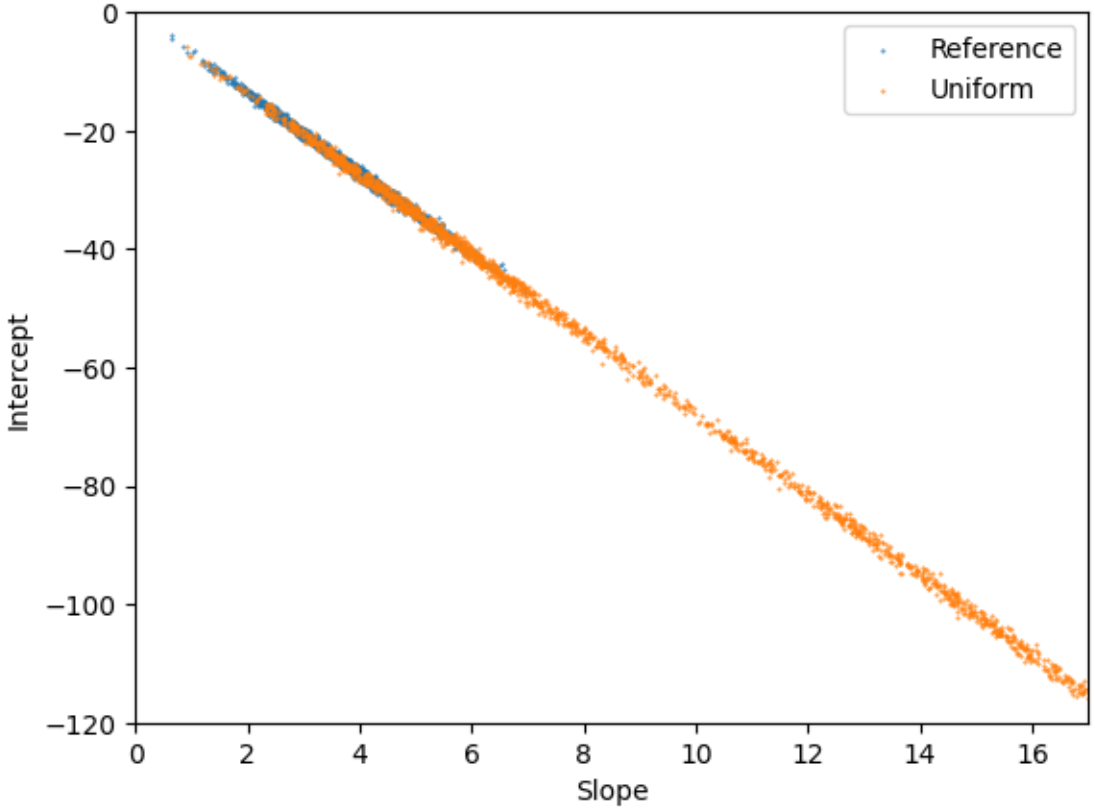
print("The posterior mean for the 50 percent lethality dose is: {}".format(np.
 ↪round(x_stars_posterior_means_new,1)))
print("with 95 percent credible interval: {}".format(np.round(x_stars_credible_intervals_new,1)))

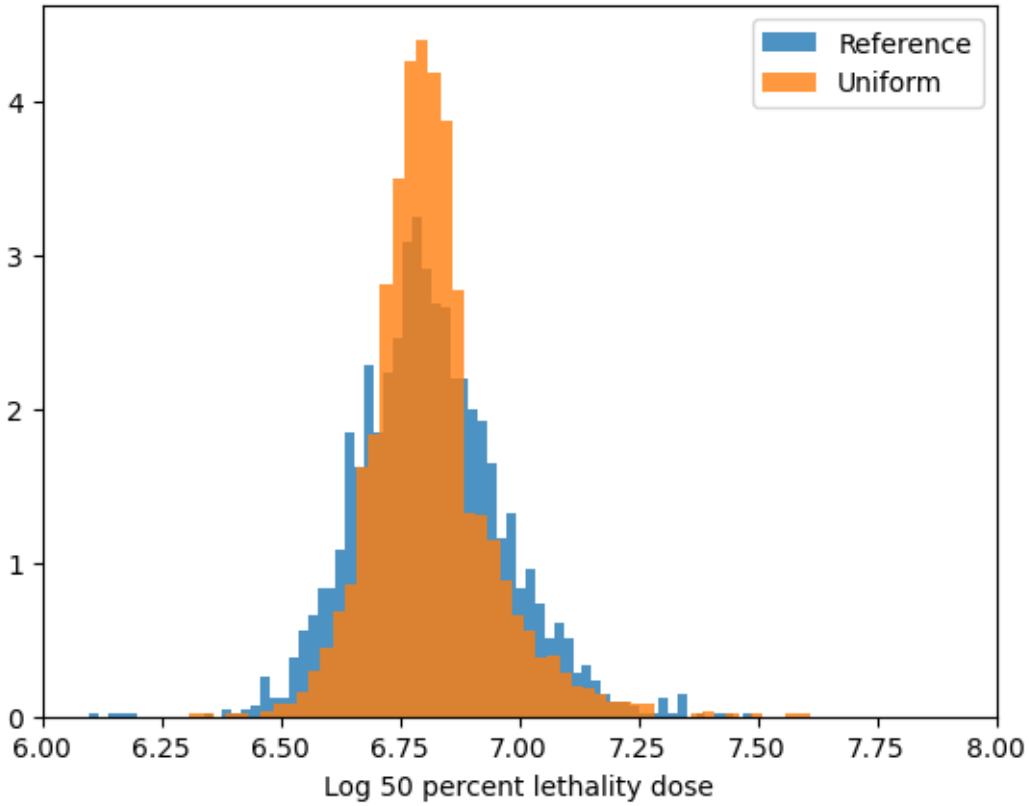
# plot
plt.scatter(post_samples[:,1],post_samples[:,0],s = 0.2)
plt.scatter(post_samples_new[:,1],post_samples_new[:,0],s = 0.2)
plt.ylim([-120,0])
plt.xlim([0,17])
plt.xlabel("Slope")
plt.ylabel("Intercept")
plt.legend(['Reference','Uniform'])
plt.show()

plt.hist(np.log(x_stars),70,alpha = 0.8,density = True)
plt.hist(np.log(x_stars_new),70, alpha = 0.8,density = True)
plt.xlim([6,8])
plt.xlabel("Log 50 percent lethality dose")
plt.legend(['Reference','Uniform'])
plt.show()
```

```
The posterior mean for the intercept is: -61.4
with 95 percent credible interval: [-125.4  -17. ]

The posterior mean for the slope is: 9.0
with 95 percent credible interval: [ 2.4 18.5]
The posterior mean for the 50 percent lethality dose is: 919.2
with 95 percent credible interval: [ 736.8 1228.7]
```


Reference
Uniform
Intercept
Slope
0
−20
−40
−60
−80
−100
−120
0
2
4
6
8
10
12
14
16


Reference
Uniform
Log 50 percent lethality dose
0
1
2
3
4
6.00
6.25
6.50
6.75
7.00
7.25
7.50
7.75
8.00

Expert

Now, adopt the expert prior. Set $\sigma = 0.1$, $\mu = 1000$, and $\nu = 40$. Does this succeed in constraining the unknown parameters? Pay attention to the axis scales when you read the plots below. We've zoomed in.

[20]:
```
# Set the prior parameters
sigma = 0.1 # default 0.1 (reasonably close to the expert provided prior)
mu = 1e3 # default 10 (weak regularization)
nu = 40 # defualt 40 (close to Gaussian tails)

# Intialize to the expert prior expectation
beta = np.array([-17.31,2.57])

# Run MCMC sampling ... this may take a while
prior_param = np.array([sigma, mu, nu])
post_samples_new = metropolis_hastings(beta, toxicity_data, prior_param)

# print summaries
betas_posterior_means_new = post_samples.mean(axis=0)
betas_credible_intervals_new = np.percentile(post_samples, [2.5, 97.5], axis=0)

# compute posterior samples for x_*
x_stars_new = np.exp(-post_samples_new[:,0]/post_samples_new[:,1])
```

[21]:
```
# print summaries
betas_posterior_means_new = post_samples_new.mean(axis=0)
betas_credible_intervals_new = np.percentile(post_samples_new, [2.5, 97.5], axis=0)

print("The posterior mean for the intercept is: {}".format(np.round(betas_posterior_means_new[0],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,0],1)))

print("")

print("The posterior mean for the slope is: {}".format(np.round(betas_posterior_means_new[1],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,1],1)))

x_stars_posterior_means_new = x_stars_new.mean(axis=0)
x_stars_credible_intervals_new = np.percentile(x_stars_new, [2.5, 97.5], axis=0)

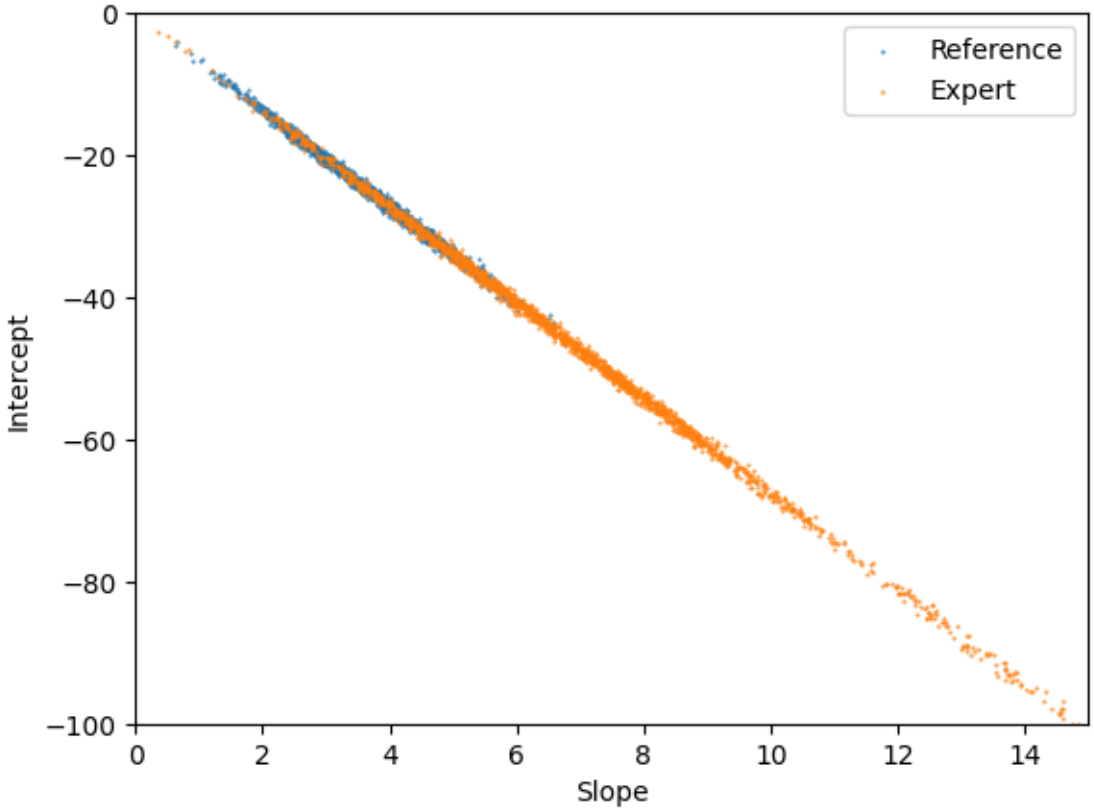
print("The posterior mean for the 50 percent lethality dose is: {}".format(np.
 ↪round(x_stars_posterior_means_new,1)))
print("with 95 percent credible interval: {}".format(np.round(x_stars_credible_intervals_new,1)))

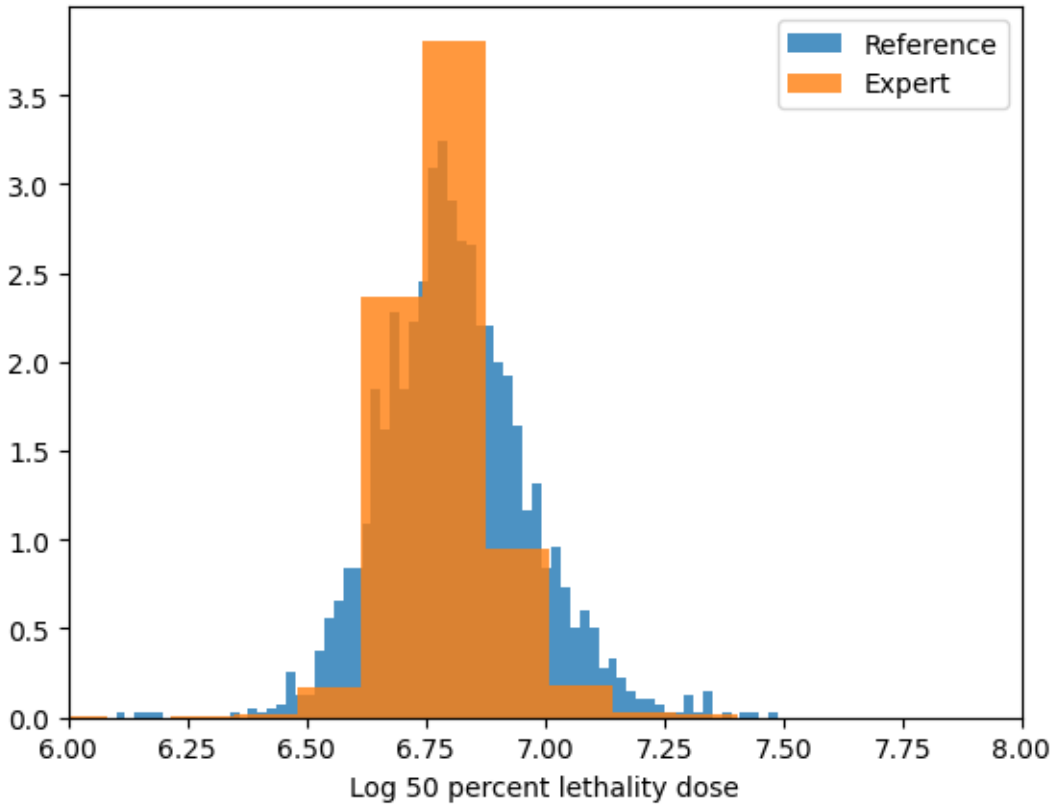
# plot
plt.scatter(post_samples[:,1],post_samples[:,0],s = 0.2)
plt.scatter(post_samples_new[:,1],post_samples_new[:,0],s = 0.2)
plt.ylim([-100,0])
plt.xlim([0,15])
plt.xlabel("Slope")
plt.ylabel("Intercept")
plt.legend(['Reference','Expert'])
plt.show()

plt.hist(np.log(x_stars),70,alpha = 0.8,density = True)
plt.hist(np.log(x_stars_new),70, alpha = 0.8,density = True)
plt.xlim([6,8])
plt.xlabel("Log 50 percent lethality dose")
plt.legend(['Reference','Expert'])
plt.show()
```

```
The posterior mean for the intercept is: -49.6
with 95 percent credible interval: [-112.3  -16.8]

The posterior mean for the slope is: 7.3
with 95 percent credible interval: [ 2.5 16.6]
The posterior mean for the 50 percent lethality dose is: 890.2
with 95 percent credible interval: [ 740.9 1124.7]
```





Regularized

Regularize by restricting the variance. Try setting $\mu = 5$, $\mu = 1$, and $\mu = 0.2$. Leave $\sigma = 0.1$ and $\nu = 40$.

[22]:
```
# Set the prior parameters
sigma = 0.1 # default 0.1 (reasonably close to the expert provided prior)
mu = 5 # default 10 (weak regularization) << CHANGE ME
nu = 40 # defualt 40 (close to Gaussian tails)

# Intialize to the expert prior expectation
beta = np.array([-17.31,2.57])

# Run MCMC sampling ... this may take a while
prior_param = np.array([sigma, mu, nu])
post_samples_new = metropolis_hastings(beta, toxicity_data, prior_param)

# compute posterior samples for x_*
x_stars_new = np.exp(-post_samples_new[:,0]/post_samples_new[:,1])
```

[23]:
```
# print summaries
betas_posterior_means_new = post_samples_new.mean(axis=0)
betas_credible_intervals_new = np.percentile(post_samples_new, [2.5, 97.5], axis=0)

print("The posterior mean for the intercept is: {}".format(np.round(betas_posterior_means_new[0],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,0],1)))

print("")

print("The posterior mean for the slope is: {}".format(np.round(betas_posterior_means_new[1],1)))
```

```
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,1],1)))

x_stars_posterior_means_new = x_stars_new.mean(axis=0)
x_stars_credible_intervals_new = np.percentile(x_stars_new, [2.5, 97.5], axis=0)

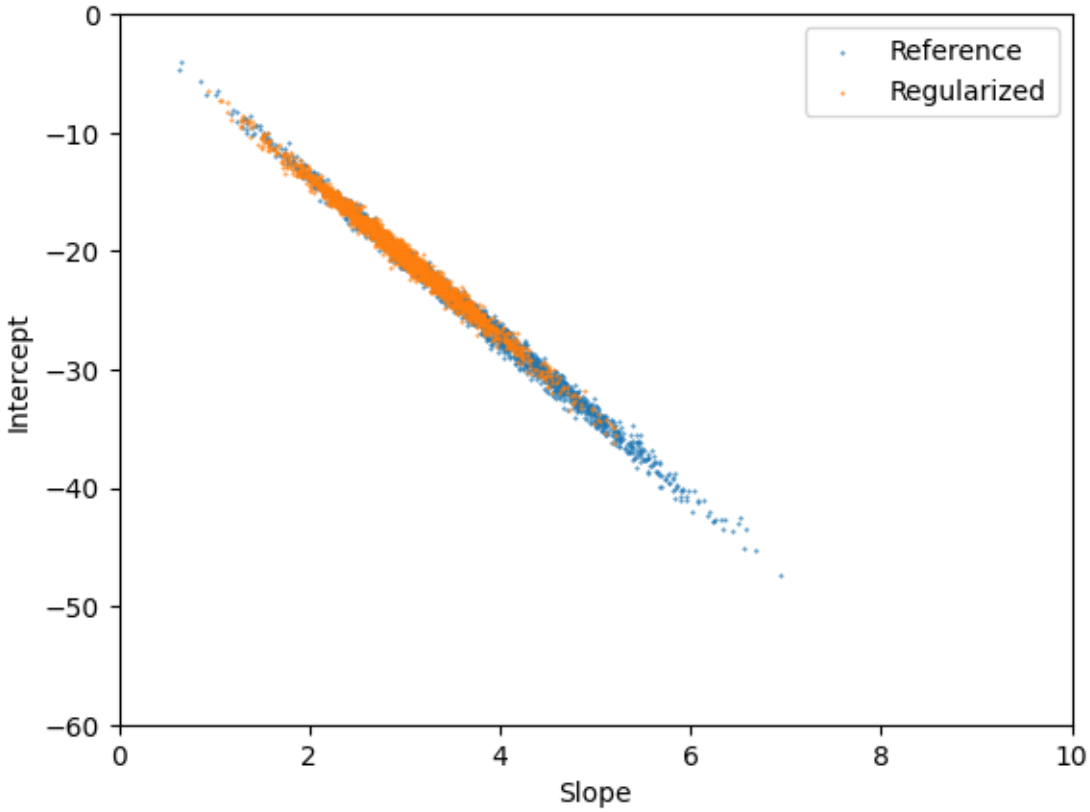
print("The posterior mean for the 50 percent lethality dose is: {}".format(np.
  ↪round(x_stars_posterior_means_new,1)))
print("with 95 percent credible interval: {}".format(np.round(x_stars_credible_intervals_new,1)))

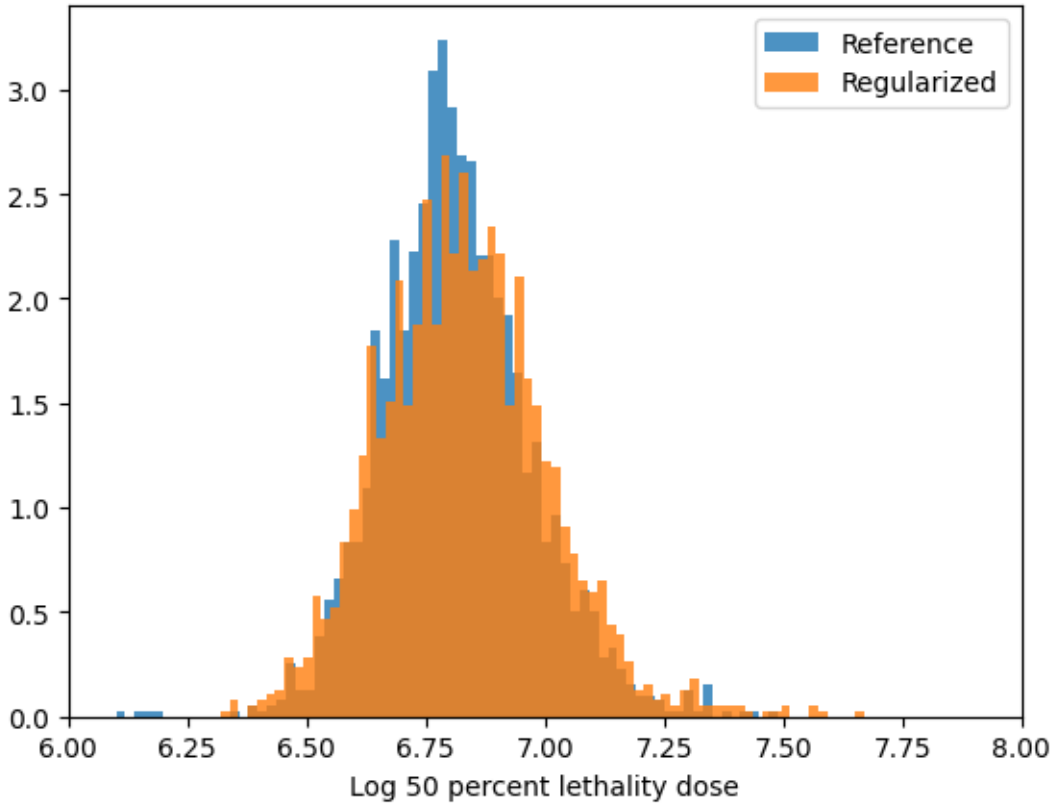
# plot
plt.scatter(post_samples[:,1],post_samples[:,0],s = 0.2)
plt.scatter(post_samples_new[:,1],post_samples_new[:,0],s = 0.2)
plt.ylim([-60,0])
plt.xlim([0,10])
plt.xlabel("Slope")
plt.ylabel("Intercept")
plt.legend(['Reference','Regularized'])
plt.show()

plt.hist(np.log(x_stars),70,alpha = 0.8,density = True)
plt.hist(np.log(x_stars_new),70, alpha = 0.8,density = True)
plt.xlim([6,8])
plt.xlabel("Log 50 percent lethality dose")
plt.legend(['Reference','Regularized'])
plt.show()
```

```
The posterior mean for the intercept is: -20.6
with 95 percent credible interval: [-30.4 -11.8]

The posterior mean for the slope is: 3.0
with 95 percent credible interval: [1.7 4.5]
The posterior mean for the 50 percent lethality dose is: 941.9
with 95 percent credible interval: [ 674.7 1326.7]
```


Reference
Regularized
Intercept
Slope
0
−10
−20
−30
−40
−50
−60
0
2
4
6
8
10


Reference
Regularized
Log 50 percent lethality dose
3.0
2.5
2.0
1.5
1.0
0.5
0.0
6.00
6.25
6.50
6.75
7.00
7.25
7.50
7.75
8.00

Robust

Set $\sigma = 0.1$ and $\mu = 5$. Relax the tails of the prior by setting $\nu = 4$, then $\nu = 2$, then $\nu = 1$. Pay attention to the outlying samples.

[24]:
```
# Set the prior parameters
sigma = 0.1 # default 0.1 (reasonably close to the expert provided prior)
mu = 5 # default 10 (weak regularization)
nu = 2 # defualt 40 (close to Gaussian tails) << CHANGE ME

# Intialize to the expert prior expectation
beta = np.array([-17.31,2.57])

# Run MCMC sampling ... this may take a while
prior_param = np.array([sigma, mu, nu])
post_samples_new = metropolis_hastings(beta, toxicity_data, prior_param)

# compute posterior samples for x_*
x_stars_new = np.exp(-post_samples_new[:,0]/post_samples_new[:,1])
```

[25]:
```
# print summaries
betas_posterior_means_new = post_samples.mean(axis=0)
betas_credible_intervals_new = np.percentile(post_samples, [2.5, 97.5], axis=0)

print("The posterior mean for the intercept is: {}".format(np.round(betas_posterior_means_new[0],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,0],1)))

print("")

print("The posterior mean for the slope is: {}".format(np.round(betas_posterior_means_new[1],1)))
print("with 95 percent credible interval: {}".format(np.round(betas_credible_intervals_new[:,1],1)))

x_stars_posterior_means_new = x_stars.mean(axis=0)
x_stars_credible_intervals_new = np.percentile(x_stars, [2.5, 97.5], axis=0)

print("The posterior mean for the 50 percent lethality dose is: {}".format(np.
 ↪round(x_stars_posterior_means_new,1)))
print("with 95 percent credible interval: {}".format(np.round(x_stars_credible_intervals_new,1)))

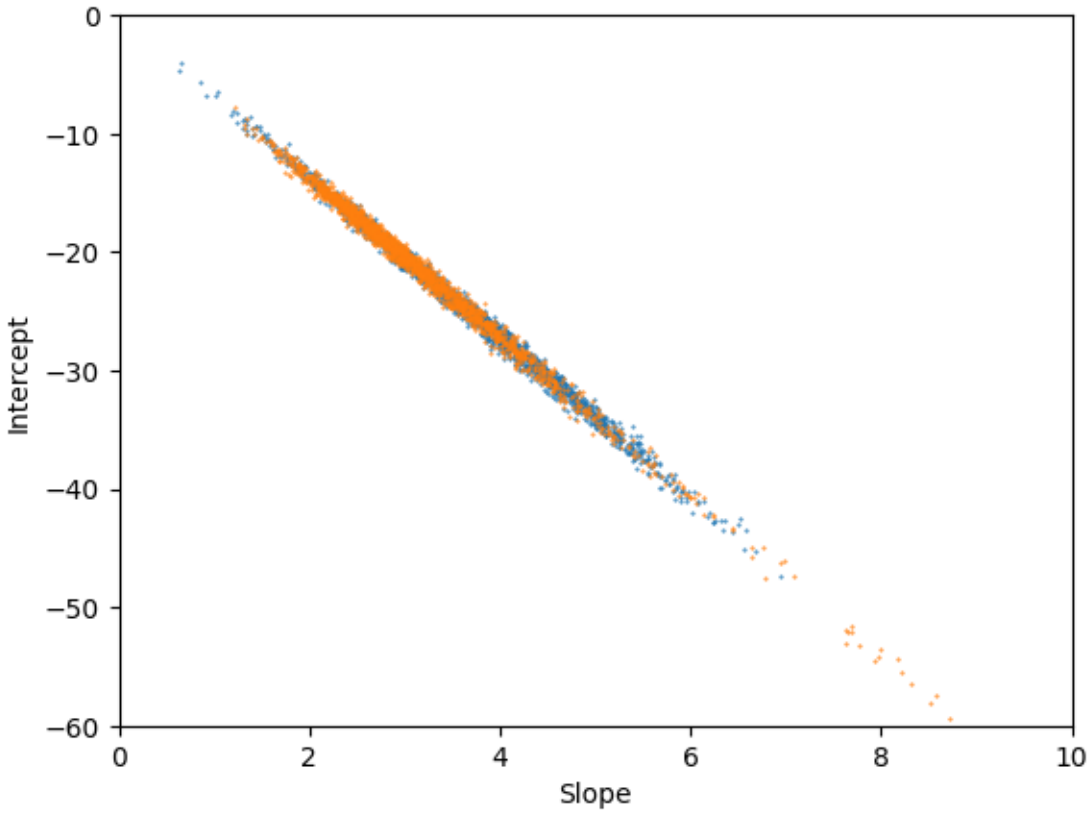
# plot
plt.scatter(post_samples[:,1],post_samples[:,0],s = 0.2)
plt.scatter(post_samples_new[:,1],post_samples_new[:,0],s = 0.2)
plt.ylim([-60,0])
plt.xlim([0,10])
plt.xlabel("Slope")
plt.ylabel("Intercept")
plt.show()

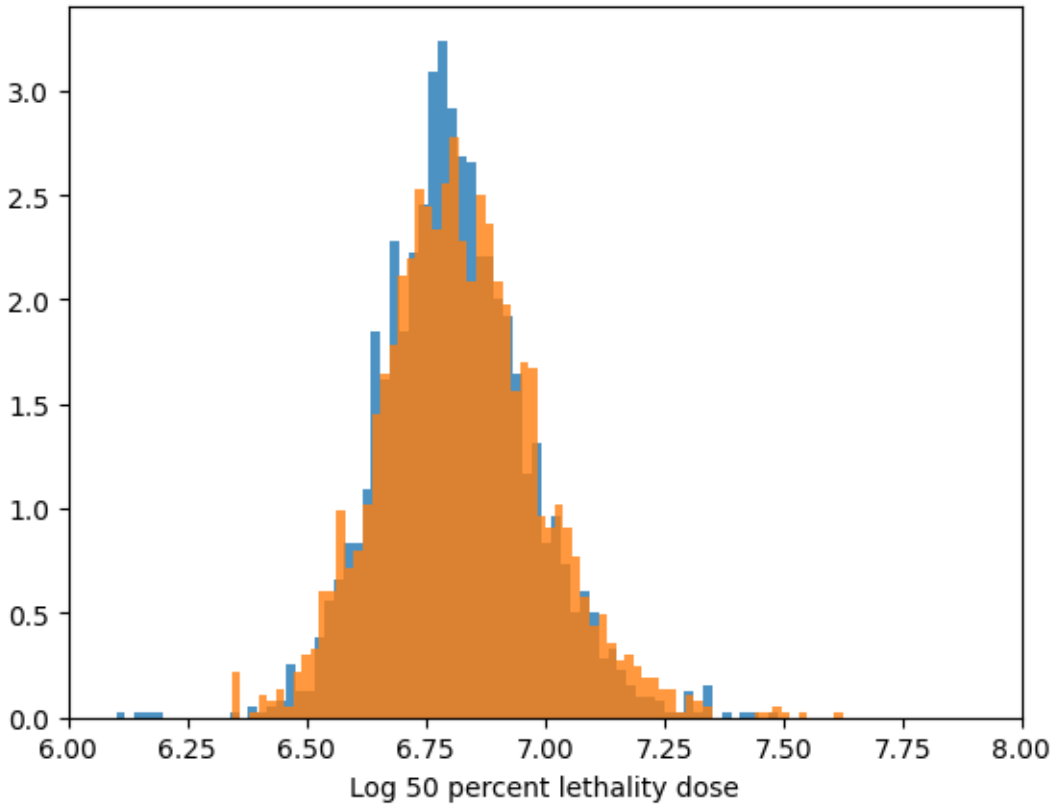
plt.hist(np.log(x_stars),70,alpha = 0.8,density = True)
plt.hist(np.log(x_stars_new),70, alpha = 0.8,density = True)
plt.xlim([6,8])
plt.xlabel("Log 50 percent lethality dose")
plt.show()
```

```
The posterior mean for the intercept is: -25.3
with 95 percent credible interval: [-38.6 -12.2]

The posterior mean for the slope is: 3.7
with 95 percent credible interval: [1.8 5.6]
The posterior mean for the 50 percent lethality dose is: 922.4
```

```
with 95 percent credible interval: [ 692.5 1262.8]
```





Discussion

Based on your observations, make an argument to yourself about what prior (or set of priors) you would pick if trying to infer the 50 percent lethal dose for use in writing regulation. Think about:

- How to motivate your choice of prior
- Whether you can anticipate the types of errors it would induce if (as is certain) it is misspecified
- The degree to which it balances induced bias for reduced variance (in posterior and in sampling over new data sets)
- Whether you could justify the errors induced as misspecification as errors in the "right" direction (towards overestimation for safety, towards underestimation for experimentation and research)
- How you would explain to an audience your range of plausible conclusions, and how you would disambiguate within model uncertainty (posterior variance), from modeling uncertainty.

Lab 7: MAP Estimation in Bayesian Hierarchical Models

Lab Date: Wednesday, Mar 12
Lab Due: Wednesday, Apr 2

Instructions

Work with your lab group to complete the following notebook. Your work will be reviewed by your peers in three weeks (Wednesday, April 2nd)

In this lab, you will: 1. Understand a Bayesian hierarchical model in the context of image reconstruction. 2. Apply the Fourier transform as a transformation matrix in an inverse problem. 3. Implement Maximum A Posteriori (MAP) estimation using: - Coordinate Ascent Optimization - Gradient-Based Optimization (L-BFGS-B) 4. Compute and interpret Laplace Approximation to estimate posterior uncertainty. 5. Compare the effectiveness of different optimization methods in Bayesian inference.

This lab applies Bayesian sparse regression to the MNIST dataset, where images of handwritten digits are processed and reconstructed using probabilistic models.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

Set Up:

Run the cells below to load necessary packages and to load the example data set.

[30]:
```
# If torchvision does not load for you, uncomment the line below and install torchvision
# %pip install torch torchvision
```

[31]:
```
import numpy as np
# import matplotlib.pyplot as plt
import torchvision.transforms as transforms
from torchvision import datasets
import scipy
from scipy.linalg import inv
from scipy.fft import fft, fftfreq
from scipy.stats import invgamma
from scipy.optimize import minimize

# My plotting settings
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'''
\usepackage{newpxtext}
\usepackage[vvarbb]{newpxmath}
\usepackage{amsmath}
\DeclareMathOperator*{\E}{\mathbb{E}}
\DeclareMathOperator*{\Var}{Var}
'''
rcParams["font.family"] = "Palatino"
rcParams['figure.dpi'] = 300
```

[32]:
```
# Load an MNIST digit as an example
transform = transforms.Compose([transforms.ToTensor(), transforms.Resize((28, 28))])
```

```
mnist_dataset = datasets.MNIST(root="./data", train=True, download=True, transform=transform)

# Select an image from the dataset
img, label = mnist_dataset[0]  # Get the first image
img_array = img.squeeze().numpy()  # Convert from PyTorch tensor to NumPy
x_true = img_array.flatten()  # Flatten the image for processing
```

Introduction

Image reconstruction is a common problem in fields such as medical, geophysical, and astronomical imaging. In this lab, we will explore an application of a Bayesian hierarchical model to reconstruct an image that has been blurred by a low-pass filter (a filter that damps high-frequency components of the image) and that has been corrupted by noise.

We assume that the true image x is transformed by an operator A, and we observe $\tilde{x}$:

$$\tilde{x} = Ax + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

where A represents the low-pass filter.

Any low pass filter can be decomposed $A = F^* D F$ where F is a Fourier matrix (the matrix that implements a discrete Fourier transform), and D is a diagonal matrix with decreasing non-negative entries. Since the discrete Fourier transform is orthonormal, its conjugate transpose, F^*, is F^{-1}, so D is the matrix of eigenvalues of the low-pass filter, and F contains its eigenvectors. Since F is unitary, $\zeta = F\epsilon \sim \mathcal{N}(0, \sigma^2 F F^*) = \mathcal{N}(0, \sigma^2 I)$.

Since most signal processing pipelines begin by converting the original received signal, $\tilde{x}$, to its Fourier transform, $y = F\tilde{x}$, we can assume that the problem takes the form:

$$\text{Estimate } x \text{ given:} \quad y = Ax + \zeta, \quad A = DF, \qquad \zeta \sim \mathcal{N}(0, \sigma^2 I).$$

To recover x, we will try a Bayesian hierarchical model. First, we assume that the elements of x are exchangeable, so they may be modeled as conditionally i.i.d. Here, we assume that the entries of the true image are, conditional on an unknown scale parameter, drawn i.i.d. from a Gaussian prior:

$$x_i \sim \mathcal{N}(0, \tau^2)$$

We place an Inverse-Gamma hyperprior on τ^2:

$$\tau^2 \sim \text{Inv-Gamma}(\alpha, \beta)$$

Our goal is to estimate x using MAP estimation and compare different optimization methods. We will estimate x and τ jointly. This will, in effect, place an L_2 regularization on the image recovery problem where the regularization parameter is unknown and optimized in tandem with the image x.

Part I: Model Construction

With your group, make a diagram representing the hierarchical model. Then, answer the questions below.

Q 1.1: Find the marginal prior distribution for each x_i by marginalizing over τ^2. Show that this mixture produces a standard distribution. Hint: it will help to know that, an inverse-gamma distributed random variable is the reciprocal of a Gamma distributed random variable with the same parameters. What does this result suggest about the regularizing effect of the hierarchical model?

Marginal Distribution

Let $X \mid \tau^2 \sim N(0, \tau^2)$, $\quad \tau^2 \sim \text{InvGamma}(\alpha, \beta)$.

Recall that if $\tau^2 \sim \text{InvGamma}(\alpha, \beta)$, then $\frac{1}{\tau^2} \sim \text{Gamma}(\alpha, \beta)$.

The likelihood and prior densities are given by

$$\begin{aligned} p(x \mid \tau^2) &= \frac{1}{\sqrt{2\pi\,\tau^2}} \exp\!\left(-\tfrac{x^2}{2\tau^2}\right) \\ p(\tau^2) &= \frac{\beta^\alpha}{\Gamma(\alpha)} (\tau^2)^{-\alpha-1} \exp\!\left(-\frac{\beta}{\tau^2}\right) \end{aligned}$$

The marginal distribution of X is found by integrating out τ^2:

$$p(x; \alpha, \beta) = \int_0^\infty p(x \mid \tau^2)\, p(\tau^2)\, d\tau^2.$$

Substituting the expressions above,

$$p(x;\alpha,\beta) \;=\; \int_0^\infty \underbrace{\frac{1}{\sqrt{2\pi\,\tau^2}}\exp\!\Big(-\tfrac{x^2}{2\,\tau^2}\Big)}_{\text{Normal}} \;\times\; \underbrace{\frac{\beta^\alpha}{\Gamma(\alpha)}\,(\tau^2)^{-\alpha-1}\exp\!\Big(-\tfrac{\beta}{\tau^2}\Big)}_{\text{Inv-Gamma}}\; d\tau^2.$$

Let $s = \frac{1}{\tau^2}$, so $\tau^2 = \frac{1}{s}$ and $d\tau^2 = -\frac{1}{s^2}\,ds$. After some algebra, we have

$$p(x;\alpha,\beta) \;\propto\; \int_0^\infty s^{\alpha-\frac{1}{2}}\,\exp\!\big(-s\,(\tfrac{x^2}{2}+\tfrac{1}{\beta})\big)\,ds.$$

Observe that this integral is (up to constant factors) the kernel of a gamma density in the variable s. In particular, if we set

$$\alpha' \;=\; \alpha+\tfrac{1}{2}, \quad \beta'(x) \;=\; \Big(\tfrac{x^2}{2}+\tfrac{1}{\beta}\Big)^{-1},$$

then

$$\int_0^\infty s^{\alpha'-1}\exp\!\Big(-\tfrac{s}{\beta'(x)}\Big)\,ds \;=\; \Gamma(\alpha')\,\big[\beta'(x)\big]^{\alpha'} \;\propto\; \big[\beta'(x)\big]^{\alpha'} = \Big[\tfrac{1}{\beta}+\tfrac{1}{2}x^2\Big]^{-(\alpha+1/2)}$$

$$p(x;\alpha,\beta) \propto \big[1+\tfrac{1}{2}\beta x^2\big]^{-\left(\frac{2\alpha+1}{2}\right)}$$

This is a Student-t distribution (location-scale t distribution)

$$\boxed{X \;\sim\; \text{Student's} - t\big(\text{scale} = \beta^{-1/2}, \text{shape} = 2\alpha\big)}$$

Regularization effect:
A Student-t has fatter tails than a Gaussian, so this hierarchal model admits solutions of larger magnitude and is therefore a less informative prior.

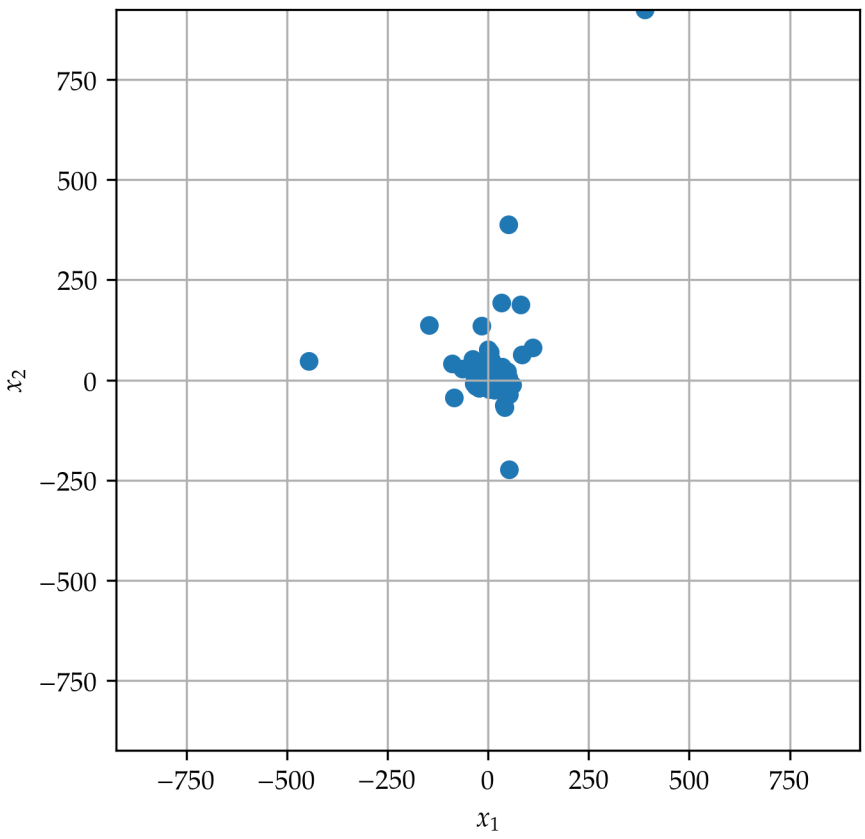
Q 1.2: Under this model, are the entries of the true image i.i.d.? Are they correlated?

Given τ^2, the entries $x_i \mid \tau^2$ are i.i.d. but marginal entries x_i are correlated as they all shared the same parameter τ^2.

Q 1.3: Write a code that can sample x_1, x_2 from the hierarchical model. Produce 1,000 samples, $\{x_1(i), x_2(i)\}_{i=1}^{1,000}$, and make a scatter plot showing the position of the samples. Make sure to match the aspect ratio of the axes in your plot.

[33]:
```
# Insert your sampling code
from scipy.stats import invgamma
alpha = 1
tau = invgamma.rvs(alpha, size=1000)
x1  = np.random.normal(0, tau)
x2  = np.random.normal(0, tau)

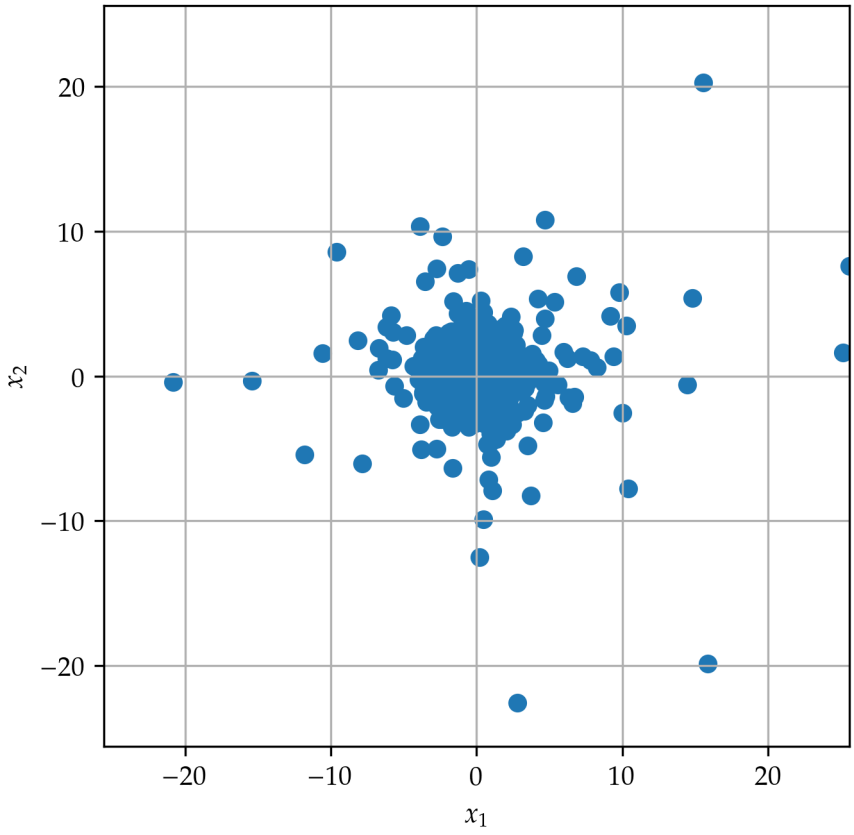
plt.figure(figsize=(5,5))
plt.scatter(x1,x2)
bbox = np.max(np.concatenate([np.abs(x1), np.abs(x2)]))
plt.xlim([-bbox, bbox])
plt.ylim([-bbox, bbox])
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.grid(True)
plt.show()
```



[ ]:
```
from scipy.stats import invgamma
def sample_students_t(alpha, beta, n_samples=1000):
    x1_samples = np.zeros(n_samples)
    x2_samples = np.zeros(n_samples)
    for i in range(n_samples):
        tau2 = invgamma.rvs(a=alpha, scale=beta)
        x1_samples[i] = np.random.normal(0, np.sqrt(tau2))
        x2_samples[i] = np.random.normal(0, np.sqrt(tau2))
    return x1_samples, x2_samples

alpha = 1
beta  = 1
n     = 1000
# x_1 = sample_students_t(alpha, beta, n)
# x_2 = sample_students_t(alpha, beta, n)
x_1, x_2 = sample_students_t(alpha, beta, n)

plt.figure(figsize=(5,5))
plt.scatter(x_1,x_2)
bbox = np.max(np.concatenate([np.abs(x_1), np.abs(x_2)]))
plt.xlim([-bbox, bbox])
plt.ylim([-bbox, bbox])
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.grid(True)
plt.show()
```



Q 1.4: Note any patterns you observe in the scatter plot. In particular, comment on how x_1 and x_2 are coupled by the hierarchical model. In what sense do they share information?

We notice that they shared the same standard deviation along the directions x_1 and x_2. Essentially we sample the standard deviation for all directions i.e., $\tau \sim \text{Inv} - \text{Gamma}(\alpha, \beta)$, and then we sample from a normal with that standard deviation, i.e., $x_i \mid \tau^2 \sim \mathcal{N}(0, \tau^2)$. This is why we see the circular pattern.

Q 1.5: Suppose we modified the hierarchical model so that, instead of drawing a global scale parameter τ^2, shared by all x_i, we drew a scale for each x_i, τ_i^2, i.i.d. from an inverse gamma with parameters α and β. This is sometimes called "local regularization". Sketch the ensuing hierarchical model, then discuss how switching from global to local regularization would change the way information is shared across the model, and change the character of solutions promoted by the prior if we optimize over x and τ jointly.

This would ensure that each x_i for a vector x are truly independent, as each element as independent variance, and is sampled independently from a normal distribution.

Construct the Low-Pass Filter

[34]:
```
def fourier_transform_matrix(size):
    """Create a Fourier transformation matrix."""
    F = scipy.linalg.dft(size)/np.sqrt(size) # Build orthonormal discrete-Fourier transform matrix in
one dimension
    return F

# Define the low-pass filter
def low_pass_filter(size):
    """Create the low pass filter."""
    frequencies = fftfreq(size, 1) # Get the frequencies for the Fourier transform
    d = 10**(-2*np.abs(frequencies)) # Laplace/exponential in frequency domain = blurring against a
Cauchy kernel
    D = np.diag(d) # diagonal matrix of eigenvalues
    F = fourier_transform_matrix(size) # Fourier transform matrix
```

```
    A1 = D @ F # low-pass filter in one dimension
    A = np.kron(A1, A1) # Kronecker product to create a 2D low-pass filter
    return A
```

[35]:
```
# Visualize transforms
plt.figure(figsize=(25, 5))

F = fourier_transform_matrix(12)
A = low_pass_filter(12)

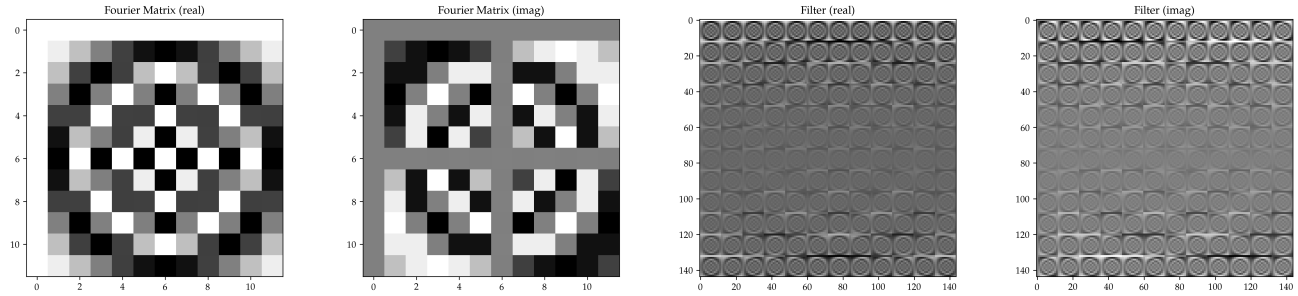
plt.subplot(1, 4, 1)
plt.imshow(F.real, cmap="gray")
plt.title("Fourier Matrix (real)")

plt.subplot(1, 4, 2)
plt.imshow(F.imag, cmap="gray")
plt.title("Fourier Matrix (imag)")

plt.subplot(1, 4, 3)
plt.imshow(A.real, cmap="gray")
plt.title("Filter (real)")

plt.subplot(1, 4, 4)
plt.imshow(A.imag, cmap="gray")
plt.title("Filter (imag)")

plt.show()
```



Simulate the Observations

1. Select an MNIST image and flatten it into a vector x_{true}.
2. Create the transformation matrix A.
3. Generate observations by applying A to x_{true}, and then by adding noise to the transformed image.

[36]:
```
# Define noise level
noise_std = 0.05  # Standard deviation of Gaussian noise

# Apply the transformation matrix and add noise
A = low_pass_filter(28)
y = A @ x_true + np.random.normal(0, noise_std, size=x_true.shape)  # Noisy observations
```

Part II: Point Estimation

To try and recover x, from y, we will try to find the joint MAP estimator $\{\hat{x}(y), \hat{\tau}^2(y)\}$.

Q 2.1: With your group members discuss whether the joint MAP problem, find $\{\hat{x}(y), \hat{\tau}^2(y)\}$ that maximize $p_{x,\tau|y}$, is distinct from the marginal MAP problem, find $\{\hat{x}(y)\}$ that maximize $p_{x|y}$. Does the choice to maximize with, or marginalize

over, the nuisance parameter τ change the nature of the estimator? Comment on which of these two procedures would be easier to implement computationally for general hierarchical models.

The joint map estimator my not equal the marginal MAP problem for xˆhat, which means marginalizing over tau may change the nature of the estimator. However, estimating over the joint is usually easier because we can avoid the marginalizing integral.

Coordinate-Ascent

First, we will consider a coordinate ascent scheme. Coordinate ascent is an optimization technique that iteratively optimizes distinct subsets of the variables while keeping others fixed. For example, we might optimize x_1 holding all other variables fixed, then optimize x_2 holding all other variables fixed, and so on. The general algorithm is available in https://en.wikipedia.org/wiki/Coordinate_descent.

Coordinate ascent is commonly used in problems where the full optimization problem is hard to solve directly but can be broken down into simpler subproblems. This is often the case for hierarchical models built out of tractable components. For example, it is common to use conditionally Gaussian hierarchical models, with independent elementwise priors on the parameters of the Gaussian. Then, each step that pools over related variables can be optimized in closed form via the linear algebra of the normal, normal model. The remaining variables may be optimized independently, in parallel. This breaks the problem into a recursive procedure that solves a sequence of easy optimize problems (e.g. least squares for pooling information in multiple dimensions, then a series of one-dimensional optimization problems). This approach retains expressivity through the mixtures encoded in the hierarchical model. For example, L_1 regularization may be represented as the MAP solution to a hierarchical model, for which the standard LASSO solution is produced using a coordinate ascent algorithm on the individual x_i jointly with a local scale parameter. The order, and subsets of variables optimized simultaneously, are up to the designer to pick. Usually, these should be chosen to mirror the architecture of the hierarchical model.

In our case, we seek to maximize the posterior distribution of x, τ given y. We will, alternately, optimize x given τ, then optimize τ given x.

1. Updating x In the first stage of coordinate ascent, we fix τ, and optimize over x.

Q 2.2: Show that, finding the conditional MAP estimator of x given τ is equivalent to minimizing:

$$f(x|\tau, y) = \|Ax - y\|^2 + \frac{\sigma^2}{\tau^2}\|x\|^2$$

where a_i is the i-th column of A.

The conditional MAP is given by

$$\hat{x}_{\text{MAP}}(\tau, y) = \arg\max_x \{p(x \mid \tau^2, y)\}$$

We note that the conditional distribution is given by

$$p(x \mid \tau^2, y) \propto p(x, \tau^2, y) = p(y \mid x, \tau^2)\, p(x, \tau^2) = p(y \mid x)\, p(x \mid \tau^2)\, p(\tau^2)$$

as $p(y \mid x, \tau^2) = p(y \mid x)$. Therefore the conditional MAP is given by

$$\hat{x}_{\text{MAP}}(\tau, y) = \arg\max_x \{p(y \mid x, \tau^2)\} = \arg\min_x \{ -\log(p(y \mid x)) \ -\log(p(x \mid \tau^2)) \ -\log(p(\tau^2))\}$$

$$\hat{x}_{\text{MAP}}(\tau, y) = \arg\min_x \left\{ \frac{(y - Ax)^*(y - Ax)}{2\sigma^2} \ + \frac{x^*x}{2\tau^2} \ + \frac{1}{2}\log(\tau^2) \ + (\alpha + 1)\log(\tau^2) \ + \frac{\beta}{\tau^2} \right\}$$

$$\hat{x}_{\text{MAP}}(\tau, y) = \arg\min_x \left\{ \frac{\|Ax - y\|_2^2}{2\sigma^2} \ + \frac{\|x\|_2^2}{2\tau^2} \right\} = \arg\min_x \left\{ \|Ax - y\|_2^2 + \frac{\sigma^2}{\tau^2}\|x\|_2^2 \right\} \quad \checkmark$$

Q 2.3: Show that the conditional MAP estimator of x given τ is given by (check HW 3, or the normal, normal exercises in lab 3):

$$\hat{x}(\tau, y) = \left(A^*A + \frac{\sigma^2}{\tau^2}I\right)^{-1} A^*y$$

The conditional MAP $\hat{x}(\tau, y)$ is given by

$$\frac{\partial}{\partial x}\left\{ -\log(p(x \mid \tau^2, y))\right\} = \frac{A^*Ax - A^*y}{\sigma^2} + \frac{x}{\tau^2} = 0 \quad \Longrightarrow \quad A^*Ax + \frac{\sigma^2}{\tau^2}x = A^*y$$

$$\implies \quad \hat{x}(\tau, y) = \left(A^*A + \frac{\sigma^2}{\tau^2} I\right)^{-1} A^* y \quad \checkmark$$

This suggests an automatic update step for x given τ. If $\{(x^{(t)}, \tau^{(t)})\}_{t=0}^{\cdots}$ is a sequence of guesses for the MAP estimator that iteratively approach the MAP via coordinate ascent, then, when updating the image x, given the scale τ, we should set:

$$x^{(t+1)} = \left(A^*A + \frac{\sigma^2}{\tau^{2(t)}} I\right)^{-1} A^* y$$

2. Updating τ^2 In the second stage, we optimize over τ, holding x fixed. That is, we aim to solve analytically for the conditional MAP estimator of τ given x.

Q 2.4: Using the structure of the hierarchical model, argue that the MAP estimator for τ given x is independent of y.

From the hierarchical model, we have that $y \mid x \perp\!\!\!\perp \tau^2 \mid x$. It then follows that $p(\tau^2 \mid x) = p(\tau^2 \mid x, y)$ as we have

$$p(\tau^2 \mid x, y) = \frac{p(\tau^2, x, y)}{p(x, y)} = \frac{p(\tau^2, y \mid x)\, p(x)}{p(y \mid x)\, p(x)} = \frac{p(\tau^2 \mid x)\, p(y \mid x)\, p(x)}{p(y \mid x)\, p(x)} = p(\tau^2 \mid x)$$

showing that the MAP estimator for τ given x is independent of y.

The MAP estimator is given by

$$\hat{\tau}^2_{\text{MAP}}(x) = \arg\max_{\tau^2} \{p(\tau^2 \mid x, y)\} = \arg\max_{\tau^2} \{p(\tau^2 \mid x)\} \quad \checkmark$$

Q 2.5: To find the MAP estimator for τ given x, show that $\tau^2|x$ has posterior density:

$$p(\tau^2|x) \propto (\tau^2)^{-(\alpha+n/2+1)} \exp\left(-\frac{\beta + 0.5\|x\|^2}{\tau^2}\right)$$

which is maximized at:

$$\hat{\tau}^2_{\text{MAP}}(x) = \frac{\beta + \frac{1}{2}\|x\|^2}{\alpha + \frac{1}{2}n + 1}$$

where n is the number of entries in x.

Recall the densities of τ^2 and $x \mid \tau^2$:

$$p(\tau^2) \propto (\tau^2)^{-\alpha-1} \exp\left(-\frac{\beta}{\tau^2}\right) \quad \text{and} \quad p(x \mid \tau^2) \propto \frac{1}{(\tau^{2n})^{1/2}} \exp\left(-\frac{\|x\|_2^2}{2\tau^2}\right) = (\tau^2)^{-n/2} \exp\left(-\frac{\|x\|_2^2}{2\tau^2}\right)$$

Then $\tau^2|x$ has posterior density:

$$p(\tau^2 \mid x) \propto p(x \mid \tau^2)\, p(\tau^2) = (\tau^2)^{-n/2} \exp\left(-\frac{\|x\|_2^2}{2\tau^2}\right) (\tau^2)^{-\alpha-1} \exp\left(-\frac{\beta}{\tau^2}\right)$$

$$p(\tau^2 \mid x) \propto (\tau^2)^{-(\alpha+n/2+1)} \exp\left(-\frac{\beta + 0.5\|x\|^2}{\tau^2}\right) \quad \checkmark$$

Then the MAP estimator for τ^2 is given by

$$\hat{\tau}^2_{\text{MAP}}(x) = \arg\max_{\tau^2} \{p(\tau^2 \mid x)\} = \arg\min_{\tau^2} \{-\log(p(\tau^2 \mid x))\}$$

$$\hat{\tau}^2_{\text{MAP}}(x) = \arg\min_{\tau^2} \left\{(\alpha + n/2 + 1)\log(\tau^2) + \frac{\beta + 0.5\|x\|^2}{\tau^2}\right\}$$

$$\frac{\partial}{\partial \tau^2}\left\{-\log(p(\tau^2 \mid x))\right\} = 2\left(\frac{\alpha + n/2 + 1}{\tau^2}\right)\tau - 2\left(\frac{\beta + 0.5\|x\|^2}{\tau^3}\right) = 0$$

$$\implies \quad \hat{\tau}^2_{\text{MAP}}(x) = \frac{\beta + \frac{1}{2}\|x\|^2}{\alpha + \frac{1}{2}n + 1} \quad \checkmark$$

Q 2.6: Refer back to Part I. Does this updating procedure match your intuition about how information should be passed through the model?

Yes it does. Starting with the $x^{(t+1)}$ update, we see that it is a ridge regressor where we regularize by the signal to noise ratio, i.e., if $\sigma^2 \ll \tau^2$ then we would expect the prediction $x^{(t+1)}$ to rely more heavily on the prior and vice versa if $\sigma^2 \gg \tau^2$. Additionally, the $\tau^{2(t+1)}$ update makes sense as we essentially predict τ based on how large our $\|x^{(t+1)}\|^2$ is. This is consistent with our observations in Part I as large x are very improbable when τ^2 is small.

This suggests the recursive step:

$$\tau^{2(t+1)} = \frac{\beta + \frac{1}{2}\|x^{(t+1)}\|^2}{\alpha + \frac{1}{2}n + 1}$$

Q 2.7: How would this update step change if we adopted the second hierarchical model described in Part I (that is, used an independent scale parameter for each x_i)?

This would change as we would now update x_i and τ_i independently for all i. For example, if x_1 grows, then τ_1 would grow proportional to its magnitude and the growth rates among the x_i's would be independent and uncoupled.

3. Implementation The full recursion is now:

$$(x^{(t)}, \tau^{2(t)}) \longrightarrow x^{(t+1)} = \left(A^*A + \frac{\sigma^2}{\tau^{2(t)}}I\right)^{-1} A^*y \longrightarrow \tau^{2(t+1)} = \frac{\beta + \frac{1}{2}\|x^{(t+1)}\|^2}{\alpha + \frac{1}{2}n + 1} \longrightarrow (x^{(t+1)}, \tau^{2(t)})$$

This is an example of an *Iteratively Reweighted Least Squares* Algorithm (IRLS). IRLS algorithms repeatedly solve a least squares problem, and use the solution of each least squares problem to update the weights used for the next least squares problem.

[37]:
```python
# Implement Coordinate Ascent
def coordinate_ascent_fourier(A, y, sigma=noise_std, alpha=2.0, beta=2.0, tol=1e-6, max_iter=300):
    """
    Perform MAP estimation for Bayesian hierarchical model using Coordinate Ascent.
    - X has a Gaussian prior: X_i ~ N(0, tau^2)
    - tau^2 has an Inverse-Gamma hyperprior with parameters alpha and beta
    """
    n = A.shape[1]
    x = np.zeros(n)  # Initialize X
    tau2 = 0.5  # Initial value of tau2

    AstarA = A.conj().T @ A
    Astary = A.conj().T @ y

    for iter in range(max_iter):
        x_old = x.copy()  # Store previous x for convergence check

        # The code below updates x using a coordinate ascent scheme.
        #for i in range(n):
        #    ai = A[:, i]
        #    residual = Y - A @ X + ai * X[i]
        #    X[i] = (ai @ residual) / (ai @ ai + 1/tau2 + 1e-6)  # Ensure numerical stability

        # Update x given tau2 (note: it is good numerical practice to avoid forming inverses.
        # There are stabler and faster ways to solve linear systems. These are implemented in most
solvers)
        x = np.linalg.solve(AstarA + ((sigma**2)/tau2)*np.eye(n), Astary)

        # Update tau2 given x
        tau2 = (beta + 0.5 * np.sum(x**2)) / (alpha + n/2 + 1)

        # Check for convergence
        if np.linalg.norm(x - x_old) < tol:
            break
```

```
    return x, tau2
```

Gradient-Based Optimization (L-BFGS-B)

Alternately, we could optimize the posterior with respect to x and τ using a standard gradient-based optimizer (e.g. gradient descent, Newton's method, etc).

Q 2.8: Show that finding $\hat{x}_{\text{MAP}}(y), \hat{\tau}^2_{\text{MAP}}(y)$ is the same as finding $\hat{x}$, $\hat{\tau}^2$ that jointly maximize the objective:

$$f(x, \tau^2|y) = \frac{1}{2\sigma^2}\|Ax - y\|^2 + \frac{1}{2\tau^2}\|x\|^2 + \left(\frac{n}{2} + \alpha + 1\right)\log(\tau^2) + \frac{\beta}{\tau^2}.$$

We have

$$p(x, \tau^2|y) \propto p(x, \tau^2, y) = p(y \mid x, \tau^2)\, p(x, \tau^2) = p(y \mid x)\, p(x \mid \tau^2)\, p(\tau^2)$$

where

$$p(y \mid x) \propto \exp\left(-\frac{\|Ax - y\|_2^2}{2\sigma^2}\right),$$

$$p(x \mid \tau^2) \propto (\tau^2)^{-n/2} \exp\left(-\frac{\|x\|_2^2}{2\tau^2}\right), \quad \text{and}$$

$$p(\tau^2) \propto (\tau^2)^{-\alpha-1} \exp\left(-\frac{\beta}{\tau^2}\right)$$

It then follows that

$$\{\hat{x}(y), \hat{\tau}^2(y)\} = \arg\max_{x,\tau^2}\{p(x, \tau^2|y)\} = \arg\min_{x,\tau^2}\{-\log(p(y \mid x)) - \log(p(x \mid \tau^2)) - \log(p(\tau^2))\}$$

$$\{\hat{x}(y), \hat{\tau}^2(y)\} = \arg\min_{x,\tau^2}\left\{\frac{\|Ax - y\|_2^2}{2\sigma^2} + \frac{\|x\|_2^2}{2\tau^2} + \left(\frac{n}{2} + \alpha + 1\right)\log(\tau^2) + \frac{\beta}{\tau^2}\right\} \quad \checkmark$$

The first term in the answer above is often called the "fidelity" term, since it is responsible for keeping the estimator loyal to the data. The second term is the "regularizer." Often, the regularizer is expressed by replacing τ with $\hat{\tau}_{\text{MAP}}(x)$ so that the regularizing term can be expressed only as a function of x.

Q 2.9: As is often true for hierarchical models built out of tractable parts, all derivatives of the log posterior are available in closed form. These can be passed as functions to an optimizer (e.g. pass a gradient and a Hessian for Newton type methods). As an example, compute the gradient of f with respect to x and τ explicitly.

$$\frac{\partial f}{\partial x} = \frac{A^*Ax - A^*y}{\sigma^2} + \frac{x}{\tau^2}$$

$$\frac{\partial f}{\partial \tau} = 2\left(\tfrac{n}{2} + \alpha + 1\right)\frac{1}{\tau} - 2\left(\tfrac{1}{2}\|x\|^2 + \beta\right)\frac{1}{\tau^3} \qquad \frac{\partial f}{\partial(\tau^2)} = \left(\tfrac{n}{2} + \alpha + 1\right)\frac{1}{\tau^2} - \frac{\frac{1}{2}\|x\|_2^2 + \beta}{(\tau^2)^2}$$

The code below minimizes this objective with a standard, gradient-based optimizer.

[38]:
```
# Define the objective function
def objective_function(params, A, y, sigma=noise_std, alpha=2.0, beta=2.0):
    """Objective function for MAP estimation using a gradient-based optimizer."""
    n = A.shape[1]
    x = params[:-1]  # Extract X from parameter vector
    tau2 = np.exp(params[-1])  # Ensure tau2 is positive by optimizing its log value

    loss = np.linalg.norm(A @ x - y) ** 2 / (2*sigma**2) + np.sum(x**2) / (2*tau2) + (n/2)*np.
↪log(tau2) # fidelity + conditionally gaussian prior on x
    loss += (alpha + 1) * np.log(tau2) + beta / tau2  # Inverse-Gamma prior on tau2
    return loss
```

```python
# Optimize using L-BFGS-B
def gradient_based_optimizer(A, y):
    """Optimize the MAP estimate using a gradient-based optimizer, jointly optimizing X and tau2."""
    n = A.shape[1]
    x_init = np.zeros(n)
    tau2_init = np.log(0.5)  # Optimize log(tau2) to enforce positivity

    params_init = np.concatenate([x_init, [tau2_init]])

    # Optimize both X and tau2
    result = minimize(objective_function, params_init, args=(A, y), method='L-BFGS-B')

    x_opt = result.x[:-1]
    tau2_opt = np.exp(result.x[-1])  # Convert back from log space
    return x_opt, tau2_opt
```

Part III: Laplace Approximation for Uncertainty Estimation

Suppose that we've run either coordinate-ascent or BFGS to convergence, and they have returned an estimator $\hat{x}_{\text{MAP}}(y)$, $\hat{\tau}^2_{\text{MAP}}(y)$.

Q 3.1: Derive the Hessian of the log-posterior density about $\hat{x}, \hat{\tau}$. Name it $H(\hat{x}, \hat{\tau})$, and show it only depends on the data y through its arguments $\hat{x}, \hat{\tau}$.

$$H(\hat{x}, \hat{\tau}) = -\begin{pmatrix} \frac{\partial^2}{\partial x\, \partial x^*} & \frac{\partial^2}{\partial x\, \partial \tau} \\ \frac{\partial^2}{\partial \tau\, \partial x^*} & \frac{\partial^2}{\partial \tau^2} \end{pmatrix} \Big[\log p(x, \tau \mid y)\Big]$$

$$H(\hat{x}, \hat{\tau}) = \begin{pmatrix} -\frac{A^*A}{\sigma^2} - \frac{1}{\tau^2} I & -\frac{1}{(\tau^2)^2}\, x \\ -\frac{1}{(\tau^2)^2}\, x^\top & \left(\frac{n}{2} + \alpha + 1\right) \frac{1}{(\tau^2)^2} + -\left(\|x\|^2 + 2\beta\right) \frac{1}{(\tau^2)^3} \end{pmatrix} \Bigg|_{(x,\tau)=(\hat{x},\hat{\tau})}$$

Q 3.2: Argue that, under a joint Laplace approximation to the posterior over x and τ, the Laplace approximation to the marginal posterior distribution of x (marginalizing over τ) is also normal, with covariance given by dropping one row and column from $H(\hat{x}, \hat{\tau})$.

Recall the Laplace approximation:

$$p(x, \tau \mid y) \approx \mathcal{N}\Big(\hat{z},\, H(\hat{x}, \hat{\tau})^{-1}\Big)$$

$$\text{where} \quad \hat{z} = \begin{pmatrix} \hat{x} \\ \hat{\tau} \end{pmatrix}$$

$$\{\hat{x}, \hat{\tau}\} = \arg\max_{x,\tau} \{p(x, \tau \mid y)\}$$

$$H(\hat{x}, \hat{\tau}) = -\nabla^2 \big[\log p(x, \tau \mid y)\big]\Big|_{(x,\tau)=(\hat{x},\hat{\tau})}$$

Therefore the marginal of x is just the portion of the covariance $-\frac{d^2 f}{dx dx^*}$, which equivalently is dropping the row/column for τ from H when inverting.

Q 3.3: Complete the code below to return the covariance of the Laplace approximation to the posterior given a joint MAP estimator $\hat{x}, \hat{\tau}^2$.

[39]:
```python
# Compute Laplace Approximation based on posterior
def laplace_approximation_covariance(A, x_map, tau2, sigma2=1.0, alpha=2.0, beta=2.0):
    """Compute the Laplace approximation to the posterior."""
    # Compute blocks of the Hessian
    H_1_1 = (-1/sigma2) * A.T @ A - (1/tau2)*np.eye(len(x_map))
    H_1_2 = x_map/(tau2**2) # n x 1 vector
    H_2_1 = x_map.T/(tau2**2) # 1 x n vector
    H_2_2 = -np.sum(x_map**2)/(tau2**3) + (alpha+len(x_map)/2+1)/(tau2**2) - (2*beta)/(tau2**3)
```

```
    # Reshape vectors to proper dimensions
    H_1_2 = H_1_2.reshape(-1, 1) # Make sure it's n x 1
    H_2_1 = H_2_1.reshape(1, -1) # Make sure it's 1 x n
    # Create the block matrix
    H = np.block( [ [H_1_1, H_1_2],
                    [H_2_1, np.array(H_2_2)]])
    # Add small regularization and invert
    covariance = -1 * np.linalg.inv(H+1e-6*np.eye(H.shape[0]))
    return x_map, covariance
```

Part IV: Visualization and Comparison

Let's compare our results. First, we will plot the MAP estimates for the images produced by both algorithms. Then, (*optional*) we will estimate the uncertainty in those estimates using our Laplace approximation.

[40]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import inv

# Compute MAP estimates using Bayesian hierarchical model with Fourier transform
x_map_fourier, tau2_estimate = coordinate_ascent_fourier(A, y)
x_map_gradient, tau2_gradient = gradient_based_optimizer(A, y)

# Compute Laplace Approximation with posterior-based covariance
x_map, covariance = laplace_approximation_covariance(A, x_map_fourier, tau2_estimate)

# Reshape for visualization
x_map_fourier_img = x_map_fourier.reshape(28, 28)
x_map_gradient_img = x_map_gradient.reshape(28, 28)

# --- Visualization ---
plt.figure(figsize=(25, 5))

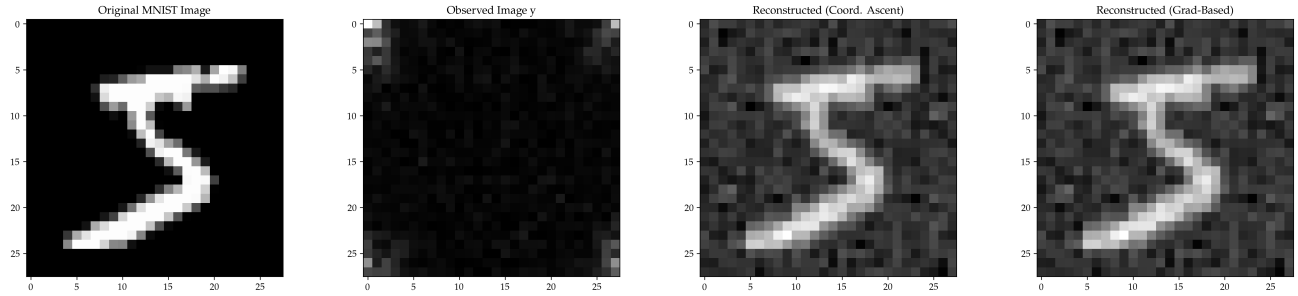
# Original MNIST Image
plt.subplot(1, 4, 1)
plt.imshow(img_array, cmap="gray")
plt.title("Original MNIST Image")

# Fourier Transformed Image (Log Magnitude for Better Visualization)
plt.subplot(1, 4, 2)
plt.imshow(np.log(1 + np.abs(y.reshape(28, 28))), cmap="gray")
plt.title("Observed Image y")

# Reconstructed Image using Coordinate Ascent
plt.subplot(1, 4, 3)
plt.imshow(x_map_fourier_img.real, cmap="gray")
plt.title("Reconstructed (Coord. Ascent)")

# Reconstructed Image using Gradient-Based Optimization
plt.subplot(1, 4, 4)
plt.imshow(x_map_gradient_img.real, cmap="gray")
plt.title("Reconstructed (Grad-Based)")

plt.show()
```



It is also often worth checking whether the "nuisance" parameters introduced in the hierarchy are well resolved. The scale parameter is more abstractly related to the data, since it is the random variable farthest removed from the observations. Their are many intermediate degrees of freedom (the entire image x) that can adapt to fit the data conditional on any choice of τ. The quality of those fits depend on τ, but often do so loosely, so do not produce a sharply peaked marginal likelihood for τ per observation. On the other hand, τ influences all the data, so every observation has the potential to carry information about τ. This is often the basic trade-off balanced by a hierarchical model. As you move up the hierarchy, unknowns become more loosely related to the observations, but, influence more data points, so more data points can be used to estimate the deeper unknowns.

Q 4.1: In the space below report the MAP estimator for the nuisance parameter τ^2, and a 95% interval estimate for τ^2 based on the Laplace approximation to the posterior, for both optimization procedures. Are your answers consistent? If not, propose a reason the optimizers may have returned different answers.

[41]:
```
# Print the MAP estimates for tau^2 from both methods
tau2_estimate = tau2_estimate.real
print(f"MAP estimate for tau^2 (Coordinate Ascent): {tau2_estimate:.6f}")
print(f"MAP estimate for tau^2 (Gradient-Based): {tau2_gradient:.6f}")

tau2_std_fourier = np.sqrt(1 / covariance[-1, -1]).real
tau2_ci_lower_fourier = tau2_estimate - 1.96 * tau2_std_fourier
tau2_ci_upper_fourier = tau2_estimate + 1.96 * tau2_std_fourier
print(f"\n95% CI for tau^2 (Coordinate Ascent): [{tau2_ci_lower_fourier:.6f}, {tau2_ci_upper_fourier:
  ↪.6f}]")

_, covariance_gradient = laplace_approximation_covariance(A, x_map_gradient, tau2_gradient)
tau2_std_gradient = np.sqrt(1 / covariance_gradient[-1, -1]).real
tau2_ci_lower_gradient = tau2_gradient - 1.96 * tau2_std_gradient
tau2_ci_upper_gradient = tau2_gradient + 1.96 * tau2_std_gradient
print(f"95% CI for tau^2 (Gradient-Based): [{tau2_ci_lower_gradient:.6f}, {tau2_ci_upper_gradient:.
  ↪6f}]")
```

```
MAP estimate for tau^2 (Coordinate Ascent): 0.099814
MAP estimate for tau^2 (Gradient-Based): 0.108471

95% CI for tau^2 (Coordinate Ascent): [-3.661065, 3.860692]
95% CI for tau^2 (Gradient-Based): [0.108471, 0.108471]
```

Q 4.2: *Optional.* To visualize the uncertainty in x, do the following:

1. Make a heatmap image showing the standard deviation in each entry of x using the Laplace approximation associated with each approximation to the MAP.
2. Compute the SVD of the covariance of each Laplace approximation. The singular vectors of the covariance are the principal components. Make a series of 6 heatmap images, 3 per optimization method, each showing the 1st, 2nd, and 3rd principal components of the Laplace covariance. To do this, extract the first three singular vectors of the covariance, then reshape each to make an image. You may borrow some of the code written above to help visualize the results. Comment on the directions of principal uncertainty in the posterior.

[42]:
```
# Insert your code here
```

Q 4.3: Comment on the MAP estimators for the images and scale, and, if completed, the directions of principal posterior uncertainty about the estimator. How do you expect your results would change if you used a set of unknown local scale parameters $\{\tau_i\}_{i=1}^{\cdots}$ instead of a shared, global scale parameter? What signals x would be promoted by this prior model? Would posterior estimation remain stable with twice as many unknowns as data points and as many nuisance parameters as parameters of inferential interest?

Lab 8: Sampling with Metropolis–Hastings on the Ising Model

Lab Date: Wednesday, Apr 2
Lab Due: Wednesday, Apr 16

Instructions

Work with your lab group to complete the following notebook. Your work will be reviewed by your peers in two weeks (Wednesday, April 16)

In this lab, you will: 1. Implement the Metropolis–Hastings algorithm for the 2D Ising model. 2. Compare different proposal distributions (independent spin flips vs. block updates) and understand their impact on convergence. 3. Test for the end of the burn-in period using visual diagnostics and an automated method. 4. Analyze mixing times.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

Set Up

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(123)
```

Model

The Ising model is a simplified model of ferromagnetism. It was developed to explain the phase transition between a magnetized and de-magnetized state of metals (e.g. iron) as the temperature of a material increases. It is relevant to us for historical interest (MCMC and Metropolis-Hastings were first developed for problems in statistical mechanics and thermodynamics), is a standard, well-understood pedagogical example, and is an introduction to more advanced models that are areas of active research (spin-glasses, neuron activity, etc). It is a useful pedagogical example since it is easy to specify and simulate, justifies some of the language used in Monte Carlo approaches (e.g. "burn-in", "annealling", "tempering"), and offers a sandbox setting for comparing MCMC methods.

The 2-D Ising model considers a collection of atoms arranged in a grid. Each atom, i, has a magnetic dipole moment or spin s_i. The Ising model makes the simplifying assumption that, at any time, each atom has spin "up", $s_i = +1$, or spin "down", $s_i = -1$. A bulk magnetic field arises from a combination of the magnetic moments of all the atoms. If the magnetic moments are arranged uniformly at random then most of the individual magnetic moments cancel out, so the sheet does not act as a magnet. If the magnetic moments align, then the fields combine constructively, so the sheet exhibits magnetism. The average spin across all atoms is typically used to measure magnetization:

$$M(\mathbf{s}) = \frac{1}{N^2} \sum_i s_i.$$

To model the development of magnetization, the Ising model assumes that the sheet of atoms are at thermodynamic equilibrium, so the collection of spins is sampled according to a Boltzmann distribution. That is:

$$\Pr(S = s) \propto \exp\left(-\frac{1}{\tau} E(s)\right)$$

where S are the randomly drawn spins for all the atoms, τ is the temperature (formally, the Boltzmann constant times the temperature), and $E(s)$ is the potential energy stored in the magnetic field when the atoms are arranged with spins s. That is, the distribution of spins is exponential in the energy stored in the spin arrangement, with a preference for low energy arrangements. The cooler the material, the stronger this preference. At absolute zero, the metal must adopt a set of spins that minimize the energy E. At high temperature, the spins are essentially arranged uniformly.

Notice that, the Boltzmann distribution can be expressed $\Pr(S = s) \propto \exp(-E(s))^{1/\tau} \propto p(s)^{1/\tau}$ for some fixed distribution $p(s)$ (the Boltzmann with $\tau = 1$). Varying the temperature in this expression from hot to cold is analogous to annealling, a procedure by which a material is heated then cooled. Heating sets τ large, so raises the distribution $p(s)$ to a power near zero. This makes it very close to uniform. As τ decreases, the power increases, so the distribution concentrates about its peaks, eventually collapsing into a combination of deltas at the spin arrangements with minimal energy. *Simulated annealling* procedures (e.g. BDA 12.3) take advantage of this idea. They marry a simulation at a desired temperature to simulations at a higher temperature, where the higher temperature distribution acts like a proposal that covers the desired target distribution.

When sufficiently cool, the Ising model will exhibit a phase transition from a de-magnetized to a magnetized state. This transition occurs since $E(s)$ is smaller when the spins of nearby atoms align. The Ising model adopts a particularly simple approximation to the actual field interactions between nearby atoms. It assumes that the energy can be expressed as a sum of the interaction energy between each pair of atoms, $E_{ij}(s)$, that $E_{ij}(s)$ equals zero if the atoms are not neighbors in the grid, and $E_{ij}(s) \propto -s_i s_j$ with a positive proportionality constant shared by all neighboring pairs. The product $-s_i s_j$ is positive if s_i and s_j have opposite signs (misaligned spins) and is negative if they share the same sign (aligned spins). Therefore, to minimize energy, the Ising model will attempt to align spins between neighboring atoms.

The energy for the full arrangement is then:

$$E(\mathbf{s}) = -\frac{J}{2} \sum_{\langle i,j \rangle} s_i s_j$$

where the summation runs over all nearest-neighbor pairs, $J > 0$ is the strength of the magnetic interaction between neighboring atoms, and we divide by 2 to avoid double counting.

Our aim is to sample from this model. That is, to sample S from the distribution:

$$\Pr(S = s) \propto \exp\left(\frac{J}{2\tau} \sum_{\langle i,j \rangle} s_i s_j \right)$$

using Markov Chain Monte Carlo.

Q 0: Suppose that we used the Boltzmann distribution as a prior, and measure the magnetization up to some measurement error. Suppose that we observed magnetization $m = M(\mathbf{s}) + \zeta$ where $\mathbf{s}$ is the true set of spins for the material, and where $\zeta \sim \mathcal{N}(0, \sigma^2)$. Write out the posterior distribution for S given m in exponential family form (up to proportionality). How does the observation change the "energy" function $(-\tau \log(\Pr(S = s))$ as a function of s)?

Note: this is an example of posterior inference using a "first-principles" prior. That is, a prior derived from a model for the process generating the unknown.

We have

$$p(S \mid m) \propto p(m \mid S)\, p(S) \propto \exp\left(-\frac{1}{2} \left(\frac{m - M(s)}{\sigma} \right)^2 \right) \exp\left(\frac{J}{2\tau} \sum_{\langle i,j \rangle} s_i s_j \right)$$

$$p(S \mid m) \propto \exp\left(\frac{J}{2\tau} \sum_{\langle i,j \rangle} s_i s_j - \frac{1}{2\sigma^2} \left(m - \frac{1}{N^2} \sum_i s_i \right)^2 \right)$$

We note that

$$-\tau \log(\Pr(S = s)) \propto -\tau \log\left(\exp\left(\frac{J}{2\tau} \sum_{\langle i,j \rangle} s_i s_j \right) \right) = -\frac{J}{2\tau} \sum_{\langle i,j \rangle} s_i s_j$$

$$\implies E(\mathbf{s}) \overset{+c}{=} -\tau \log(\Pr(S = s))$$

Initialize the Ising Grid

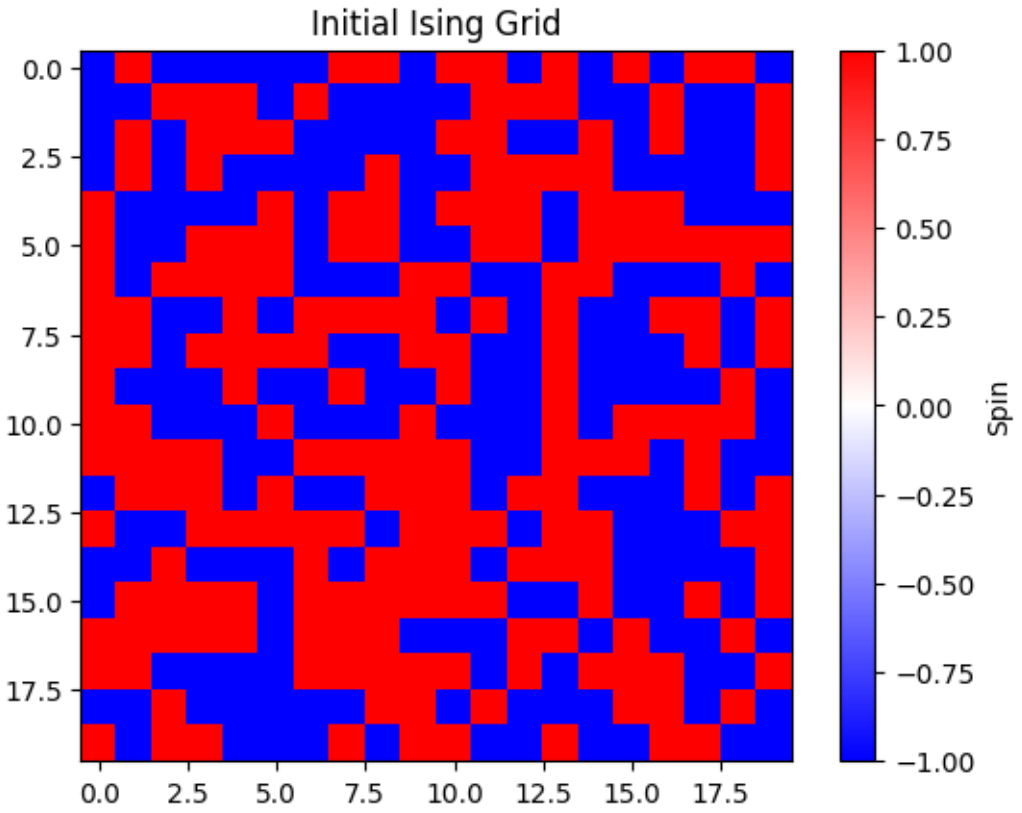
The function `initialize_grid(N)` generates an $N \times N$ grid where each site is randomly assigned a spin +1 or −1. This produces an initial state $\mathbf{s}_0$ by sampling each spin uniformly from $\{+1, -1\}$.

[2]:
```
def initialize_grid(N):
    """
    Initialize an N x N grid with spins randomly set to +1 or -1.

    Each spin s_i is sampled independently from {+1, -1}.
```

```
    """
    return np.random.choice([-1, 1], size=(N, N))

# Example: Initialize a 20x20 grid and display it.
N = 20
grid = initialize_grid(N)
plt.imshow(grid, cmap="bwr")
plt.title("Initial Ising Grid")
plt.colorbar(label="Spin")
plt.show()
```



Compute the Energy

The function `compute_energy(grid, J=1)` calculates the energy using periodic boundary conditions. The energy is given by:

$$E(\mathbf{s}) = -\frac{J}{2}\sum_{\langle i,j\rangle} s_i s_j,$$

where the summation is over all nearest-neighbor pairs and we divide by 2 to avoid double counting. Periodic boundaries ensure neighbors "wrap around."

[3]:
```
def compute_energy(grid, J=1):
    """
    Compute the energy of the grid using the Ising Hamiltonian with periodic boundaries.

    Mathematical Formulation:
      H(s) = -J * sum_{<i,j>} s_i * s_j,
    with each nearest-neighbor pair counted twice.
```

```
    """
    N = grid.shape[0]
    energy = 0
    for i in range(N):
        for j in range(N):
            s = grid[i, j]
            # Nearest neighbors with periodic boundary conditions.
            neighbors = (grid[(i+1) % N, j] +
                         grid[(i-1) % N, j] +
                         grid[i, (j+1) % N] +
                         grid[i, (j-1) % N])
            energy += -J * s * neighbors
    return energy / 2  # Correct for double counting

# Example: Compute and display the energy of the initial grid.
energy0 = compute_energy(grid)
print("Initial Energy:", energy0)
```

```
Initial Energy: 0.0
```

Markov Chain Monte Carlo

We will use the Metropolis-Hastings (MH) algorithm and Gibbs Sampling to generate samples from the Ising model.

Metropolis-Hastings: The MH algorithm is an example of a Markov Chain Monte Carlo (MCMC) method. It specifies a Markov chain whose steady state distribution equals a desired target distribution.

MH proceeds in two stages per step. Given an initial state, it proposes a perturbation to that state using a tractable proposal (or jump) distribution. Then, it decides whether to accept or reject the candidate state. The probability of acceptance is determined by comparing the probabilities of the original state and the candidate state.

Let $p_r(s \to s')$ denote the probability that the proposal (or jump) distribution proposed a candidate state s' given an initial state s. Let $\tilde{p}_*(s)$ denote the unnormalized target distribution. Then, the acceptance probability for the jump is set to:

$$\min\left\{\frac{\tilde{p}_*(s')}{\tilde{p}_*(s)}\frac{p_r(s' \to s)}{p_r(s \to s')}, 1\right\}$$

Review chapter 11.2 in BDA for a full specification. For an example, see the Lab 6 demo.

Note that, if we use a symmetric proposal, then $p_r(s' \to s) = p_r(s \to s')$, so MH reduces to the Metropolis algorithm. Further, if we set $\tilde{p}_*(s)$ to the Boltzmann distribution, then the target odds can be expressed:

$$\frac{\tilde{p}_*(s')}{\tilde{p}_*(s)} = \exp\left(-\tfrac{1}{\tau}(E(s') - E(s))\right) = \exp\left(-\tfrac{1}{\tau}\Delta E(s, s')\right)$$

Gibbs Sampling: Gibbs sampling is an MCMC algorithm that iteratively resamples a subset of the unknowns (here, a subset of the spins), conditional on the rest (here, conditional on the remaining spins). For details, see BDA Chapter 11.1. Gibbs sampling is a special instance of the MH algorithm, although, for implementation it is often easier to construct separately. Gibbs sampling works well for target distributions with tractable conditionals, and, with approximately independent subsets of variables. This is a common feature in hierarchical models built from conditionally conjugate distributions. It is also a feature of the Ising model.

Proposals:

Proposal I: Independent Spin Flips **Q 1:** In the space below, complete the function `proposal_independent(grid, flip_prob)` that updates each spin independently by flipping it with probability p = `flip_prob`. This update represents a local change, where each spin's new value is given by:

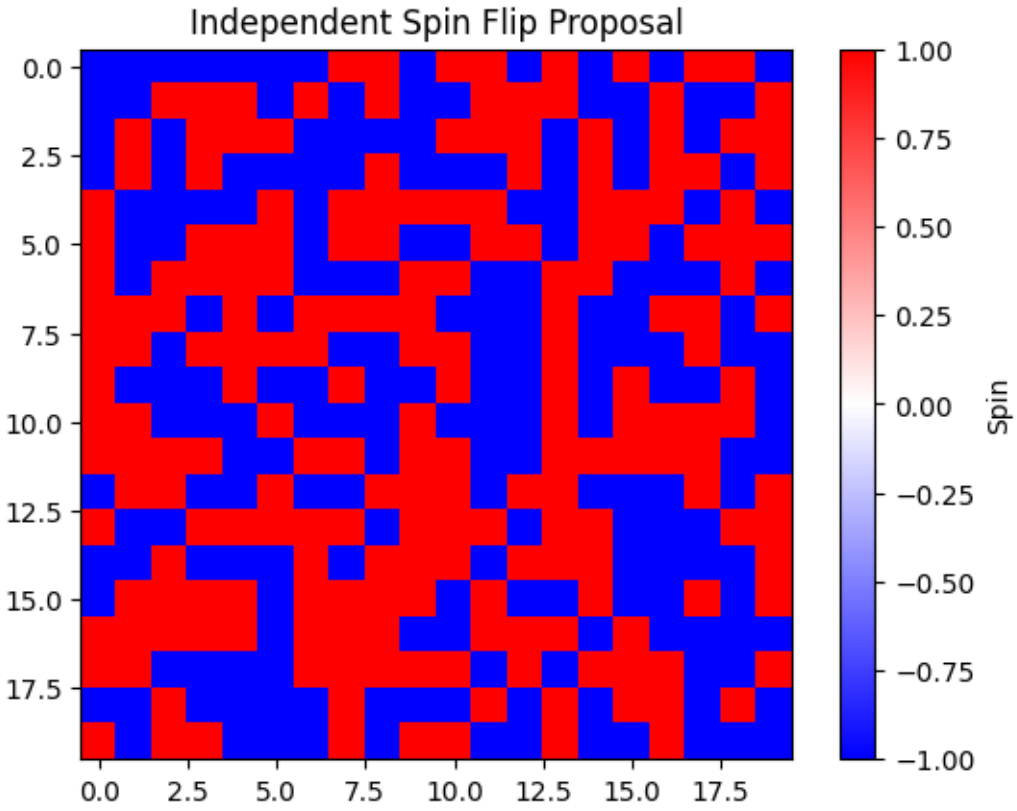
$$s_i \to -s_i \quad \text{with probability } p.$$

Because the proposal is symmetric, the MH acceptance probability depends only on the energy difference.

[8]:
```python
def proposal_independent(grid, flip_prob):
    """
    For each spin in the grid, flip its sign with probability flip_prob.

    This implements a local update with a symmetric proposal.
    """
    new_grid = grid.copy() # set the new grid of spins equal to the old grid of spins
    N = grid.shape[0] # extract the side-length
    for i in range(N):
        for j in range(N):
            if np.random.rand()<flip_prob:
                new_grid[i,j] *= -1
    return new_grid

# Example: Apply the independent spin flip proposal with a 5% flip probability.
flip_prob = 0.05
grid_proposed = proposal_independent(grid, flip_prob)
plt.imshow(grid_proposed, cmap="bwr")
plt.title("Independent Spin Flip Proposal")
plt.colorbar(label="Spin")
plt.show()
```



Proposal II: Block Updates In general, the efficiency of a MH scheme depends on the choice of proposal (or jump) distribution. Ideally, the proposal should be chosen by studying the target distribution. Since the Ising model tends to produce clusters of atoms with matching spins, it may help to use an update that acts locally and preserves magnetization within sub-blocks of the overall grid.

As a comparison, we've implemented a function `proposal_block_update(grid, block_size, update_fraction)` that partition the grid into $k \times k$ blocks and randomly scramble the spins in a fraction p = `update_fraction` of these

blocks. More formally, it divides the grid into blocks. Then, given M blocks, approximately $p \times M$ blocks are chosen to update. Within each chosen block, the spins are randomly permuted (uniformly over all possible arrangements), ensuring a symmetric proposal.

Think carefully about the advantages and disadvantages of this proposal relative to the first proposal.

[9]:
```
def proposal_block_update(grid, block_size, update_fraction):
    """
    Partition the grid into blocks of size block_size x block_size.
    Randomly select a fraction update_fraction of blocks and scramble their spins.

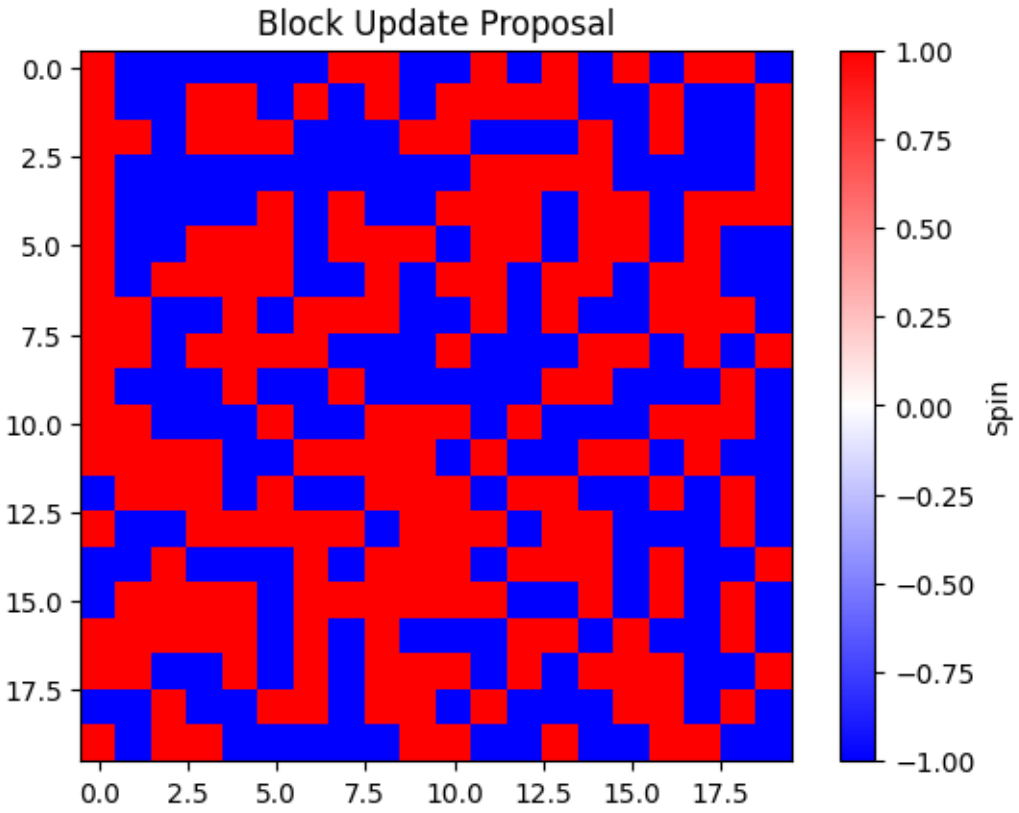
    Assumption: Grid size N is divisible by block_size.
    """
    N = grid.shape[0]
    new_grid = grid.copy()
    num_blocks = N // block_size  # number of blocks per dimension
    total_blocks = num_blocks * num_blocks
    num_to_update = int(np.ceil(update_fraction * total_blocks))

    # List of block indices.
    blocks = [(i, j) for i in range(num_blocks) for j in range(num_blocks)]
    selected_indices = np.random.choice(len(blocks), size=num_to_update, replace=False)

    for idx in selected_indices:
        block_i, block_j = blocks[idx]
        start_i = block_i * block_size
        start_j = block_j * block_size
        block = new_grid[start_i:start_i + block_size, start_j:start_j + block_size]
        block_flat = block.flatten()
        np.random.shuffle(block_flat)
        new_grid[start_i:start_i + block_size, start_j:start_j + block_size] = block_flat.
↪reshape(block_size, block_size)

    return new_grid

# Example: Apply block update with block size 4 and 30% of blocks updated.
block_size = 4
update_fraction = 0.3
grid_proposed_block = proposal_block_update(grid, block_size, update_fraction)
plt.imshow(grid_proposed_block, cmap="bwr")
plt.title("Block Update Proposal")
plt.colorbar(label="Spin")
plt.show()
```



Updates:

Metropolis Update Step: **Q 2:** In the space below, complete the function `metropolis_step(grid, proposal_func, temperature, **proposal_kwargs)` that carries out one MH update. It should calculate the energy $E(\mathbf{s})$ of the current state, propose a new state $\mathbf{s}'$, compute $\Delta E(\mathbf{s}, \mathbf{s}') = E(\mathbf{s}') - E(\mathbf{s})$, and accept $\mathbf{s}'$ with probability

$$\alpha(\mathbf{s}, \mathbf{s}') = \min\left\{\exp\left(-\tfrac{1}{\tau}\Delta E(\mathbf{s}, \mathbf{s}')\right), 1\right\}.$$

This acceptance probability ensures detailed balance and drives the chain towards the equilibrium distribution.

[20]:
```
def metropolis_step(grid, proposal_func, temperature, **proposal_kwargs):
    """
    Perform one Metropolis-Hastings update step.

    Steps:
      1. Compute current energy H(s).
      2. Generate proposed state s' using proposal_func.
      3. Compute DH = H(s') - H(s).
      4. Accept s' with probability: min(1, exp(-DH/T)).

    Returns:
      new_grid: the updated configuration.
      accepted: True if the move is accepted; otherwise, False.
    """
    current_energy = compute_energy(grid)

    proposed_grid = proposal_func(grid, **proposal_kwargs)

    proposed_energy = compute_energy(proposed_grid)
```

```
    delta_E = proposed_energy-current_energy # compute the change in energy

    if np.random.rand() <= min(np.exp((-1/temperature)*delta_E),1): # implement the MH acceptance
rule here
        return proposed_grid, True
    else:
        return grid, False

# Test one MH step using the independent proposal.
temperature = 2/(np.log(1+np.sqrt(2)))
grid, accepted = metropolis_step(grid, proposal_independent, temperature, flip_prob=flip_prob)
print("MH Step accepted?", accepted)
```

```
MH Step accepted? False
```

Gibbs Update Step: **Q 3:** For each atom i, find a set of atoms $\mathcal{N}(i)$ such that S_i is conditionally independent of S_k for any $k \notin \mathcal{N}(i)$, given the value of the spins of all atoms $j \in \mathcal{N}(i)$.

The set of atoms $\mathcal{N}(i)$ is given by its nearest neighbors, e.g., every atom has four neighbors, $\mathcal{N}(i) = \{i_{\text{up}}, i_{\text{down}}, i_{\text{left}}, i_{\text{right}}\}$. Note that this includes the atoms along the edges and in the corners by the periodic boundary conditions.

Q 4: Using your answer to question 3, find the exact conditional distribution for S_i as a function of the spins in its neighborhood, $\mathcal{N}(i)$, and the parameters τ, J.

$$\Pr\left(S_i \mid S_{\mathcal{N}(i)}\right) = \exp\left(\frac{J}{2\tau} \sum_{j \in \mathcal{N}(i)} s_i s_j\right)$$

Q 5: Using your answer to question 3, show that, if the total number of grid cells along each side of the grid is even, then the grid can be partitioned into two sets of atoms where all the spins in the first set are conditionally independent of one another given all the spins in the second set and visa-versa. (*Hint: it may help to draw the grid and neighborhoods*). Use your result to argue that, given even side-lengths, we can implement a Gibbs sampling procedure that uses two blocks of spins, and, that when resampling each block, we can draw all the spins within a block independently from each other.

If we split the atoms into a checkerboard pattern, then they will be conditionally independent and therefore enable Gibbs sampling. Let our 2D grid of atoms be indexed by $\mathbf{S} = S_{ij}$ for $i, j \in \{1, 2, \ldots, N\}$. Then let $\mathcal{R}$ and $\mathcal{B}$ be the two sets (i.e., the red and black squares on a checkerboard) where

$$\mathcal{R} := \{(i, j) \mid i + j \text{ is even}\} \quad \text{and} \quad \mathcal{B} := \{(i, j) \mid i + j \text{ is odd}\}$$

Q 6: Based on questions 3 - 5, write a pseudocode outlining a Gibbs sampling procedure for sampling from the Ising model:

1. Random select the set of atoms to update, $\mathcal{R}$ or $\mathcal{B}$. Let our randomly selected set be $\mathcal{U}$ and the one that we did not select be $-\mathcal{U}$.
2. Sample candidate Y from $\Pr(S_{\mathcal{U}} \mid S_{-\mathcal{U}})$.
3. Update sample with candidate, $S_{\mathcal{U}} \leftarrow Y$.

Q 7: In the space below implement a function that performs a single pair of Gibbs sampling steps (completes one resample of every spin). Your code should imitate the format of the `metropolis_step` function implemented above.

[ ]:
```
def gibbs_step(grid, temperature, J=1):
    N = grid.shape[0]
    new_grid = grid.copy()
    red_or_black = np.random.choice([0,1])
    for i in range(N):
        for j in range(N):
            if (i+j)%2==red_or_black:
                s = grid[i, j]
                # Nearest neighbors with periodic boundary conditions.
                neighbors = (grid[(i+1) % N, j] +
                             grid[(i-1) % N, j] +
```

```
                            grid[i, (j+1) % N] +
                            grid[i, (j-1) % N])
                energy = J * s * neighbors


    current_energy = compute_energy(grid)


    proposed_energy = compute_energy(proposed_grid)

    delta_E = proposed_energy-current_energy # compute the change in energy

    if np.random.rand() <= min(np.exp((-1/temperature)*delta_E),1): # implement the MH acceptance
rule here
        return proposed_grid, True
    else:
        return grid, True
```

Simulation

The function `run_MH_simulation(initial_grid, proposal_func, temperature, n_steps, **proposal_kwargs)` runs the MH algorithm for a fixed number of iterations. It also records both the energy and the magnetization at each step. Tracking $E(\mathbf{s})$ and $M(\mathbf{s})$ will help to monitor convergence.

Q 8: Modify the function below to track the acceptance rate (fraction of accepted steps), and to stop on the total number of accepted steps instead of the total number of proposed steps.

[ ]:
```
def run_MH_simulation(initial_grid, proposal_func, temperature, n_steps, **proposal_kwargs):
    """
    Run the MH simulation for n_steps iterations.

    Records:
      - Energy: H(s)
      - Magnetization: M = (1/N^2) * sum(s)

    Returns:
      grid: Final configuration.
      energies: Array of energy values over iterations.
      magnetizations: Array of magnetization values over iterations.
    """
    grid = initial_grid.copy()
    energies = []
    magnetizations = []
    acceptance_rate = 0 # Initialize acceptance rate

    for step in range(n_steps):
        grid, ... = metropolis_step(grid, proposal_func, temperature, **proposal_kwargs)
        energies.append(compute_energy(grid))
        magnetizations.append(np.mean(grid))
        acceptance_rate = ... # modify to track the acceptance rate

    return grid, np.array(energies), np.array(magnetizations)
```

Q 10: Experiment with both MH algorithms below. Vary the parameters for the proposal distributions until you are satisfied with the acceptance rates.

[ ]:
```
# Example: Run simulation with the independent spin flip proposal.
n_steps = 10000
# Independent Spin Flips Simulation
grid0_indep = initialize_grid(N)
```

```
grid_final_indep, energies_indep, magnetizations_indep = run_MH_simulation(
    grid0_indep, proposal_independent, temperature, n_steps, flip_prob=flip_prob
)
```

[ ]:
```
# Example: Run simulation with the block updates proposal
grid0_block = initialize_grid(N)
grid_final_block, energies_block, magnetizations_block = run_MH_simulation(
    grid0_block, proposal_block_update, temperature, n_steps, block_size=block_size,
update_fraction=update_fraction
)
```

Q 11: In the space below write a `run_Gibbs_simulation` function that implements your Gibbs sampling procedure. As before, imitate the structure of the `run_MH_simulation` as closely as possible (in particular, match the format for the outputs and relevant inputs). Then, generate a simulation run as an example.

[ ]:
```
# write your function here
```

Visual Diagnostics and Burn-In Detection

MCMC methods depend on an asymptotic guarantee. Given *long enough* the sample trajectories will produce states sampled from the target distribution. The samples produced by the initial stages of MCMC are usually drawn from an ensemble that is far from the stationary ensemble produced by initializing from the target. This early phase is usually called the "burn-in" phase, and is typically discarded from the sample run.

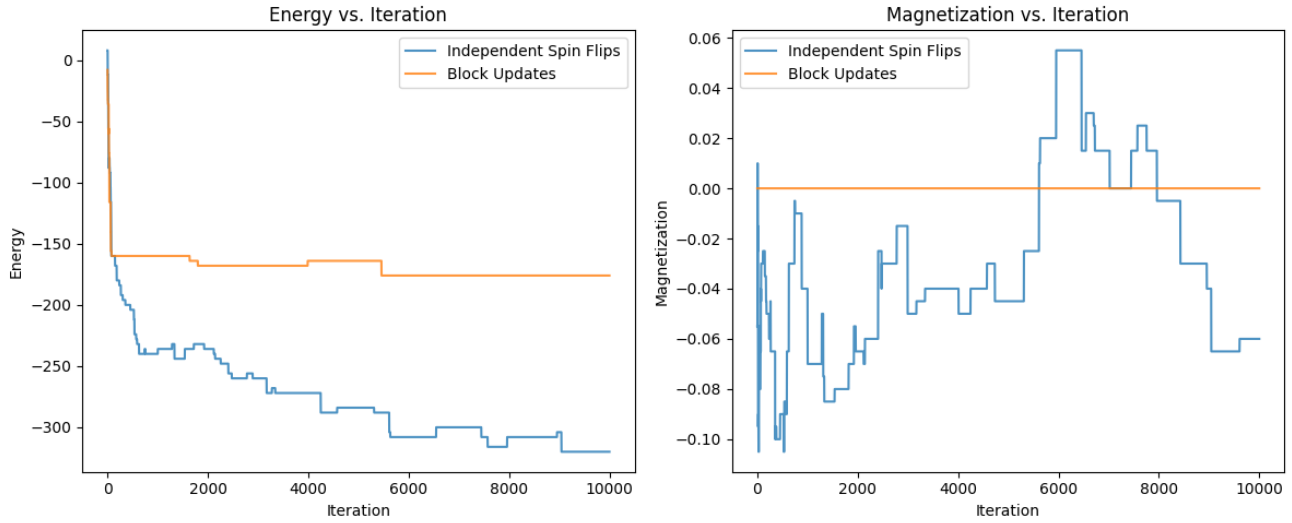
Q 12: Plot the energy and magnetization along each simulation versus iteration to visually assess convergence.

[ ]:
```
plt.figure(figsize=(12,5))

# Plot energy for both proposals.
plt.subplot(1,2,1)
plt.plot(energies_indep, label="Independent Spin Flips", alpha=0.8)
plt.plot(energies_block, label="Block Updates", alpha=0.8)
... # add the output of your Gibbs simulation here
plt.xlabel("Iteration")
plt.ylabel("Energy")
plt.title("Energy vs. Iteration")
plt.legend()

# Plot magnetization for both proposals.
plt.subplot(1,2,2)
plt.plot(magnetizations_indep, label="Independent Spin Flips", alpha=0.8)
plt.plot(magnetizations_block, label="Block Updates", alpha=0.8)
... # add the output of your Gibbs simulation here
plt.xlabel("Iteration")
plt.ylabel("Magnetization")
plt.title("Magnetization vs. Iteration")
plt.legend()

plt.tight_layout()
plt.show()
```



Q 13: Inspect the plots above and compare the convergence behavior of the three simulation methods. Which converges fastest? Why? Do all of the methods appear consistent with each other? If not, try to explain why. Pay close attention to the characteristics of the proposals used.

Write your answer here.

Q 14: Based on your observations, suggest an alternate proposal distribution for MH. Explain why (a) it is valid, and (b) why you think it might lead to better performance.

Write your answer here.

Q 15: Discuss how your answer would change if we had observed (a) the exact magnetization, or (b) a subset of spins exactly, and aimed to sample from the posterior over possible spin arrangements.

Write your answer here.

Q 15: In the space below, write a function to automatically detect the end of the burn-in period using the sequence of observed energies and magnetizations. Check BDA 11.4 for advice on assessing convergence. Explain the rationale for your procedure in the comment section of the code below.

[ ]:
```
def detect_burn_in_energy(...):
    """
    Estimate the burn-in period using energy and magnetization as the observable by ...

    Parameters:
      energy_series : 1D numpy array of energy values.
      magnetization_series : 1D numpy array of magnetization values.
      ...

    Returns:
      The estimated burn-in iteration.
    """

    return ...

# Detect end of burn-in period for all three simulations
burn_in_energy_indep = detect_burn_in_energy(...)
burn_in_energy_block = detect_burn_in_energy(...)
burn_in_energy_block = ... # repeat for Gibbs here


print("Independent Spin Flips: Estimated burn-in (energy) ends at iteration:", burn_in_energy_indep)
print("Block Updates: Estimated burn-in (energy) ends at iteration:", burn_in_energy_block)

# Plot energy with burn-in detection for Independent Spin Flips.
```

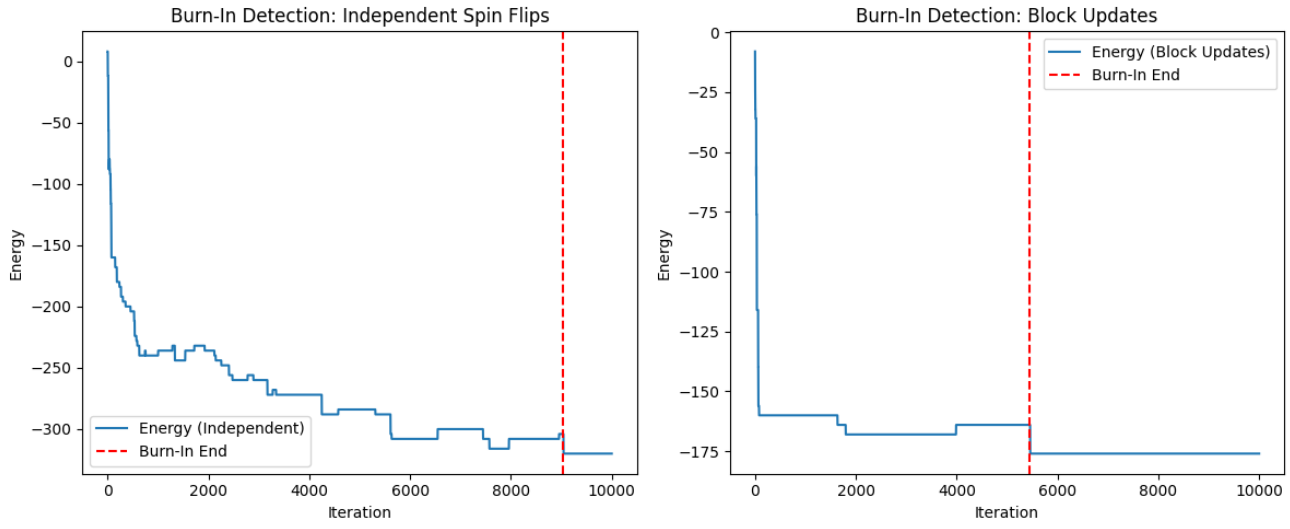
```
plt.figure(figsize=(12,5))
plt.subplot(1,2,3)
plt.plot(energies_indep, label="Energy (Independent)")
plt.axvline(burn_in_energy_indep, color='red', linestyle='--', label="Burn-In End")
plt.xlabel("Iteration")
plt.ylabel("Energy")
plt.title("Burn-In Detection: Independent Spin Flips")
plt.legend()

# Plot energy with burn-in detection for Block Updates.
plt.subplot(1,2,3)
plt.plot(energies_block, label="Energy (Block Updates)")
plt.axvline(burn_in_energy_block, color='red', linestyle='--', label="Burn-In End")
plt.xlabel("Iteration")
plt.ylabel("Energy")
plt.title("Burn-In Detection: Block Updates")
plt.legend()

# Plot energy with burn-in detection for Gibbs.
plt.subplot(1,3,3)
... # repeat for Gibbs
plt.xlabel("Iteration")
plt.ylabel("Energy")
plt.title("Burn-In Detection: Block Updates")
plt.legend()

plt.tight_layout()
plt.show()
```

```
Independent Spin Flips: Estimated burn-in (energy) ends at iteration: 9032
Block Updates: Estimated burn-in (energy) ends at iteration: 5450
```



Mixing Times

The efficiency of an MCMC scheme depends primarily on its mixing times. These control the length of the burn-in period, and the autocorrelation of the generated samples. Note, these two times need not be the same.

Q 16: In the space below, write a code that, after discarding the burn-in period, estimates the effective number of samples generated based on the autocorrelation in the time series for the energy and the magnetization (see BDA 11.5). Use the smaller of the two numbers as a measure of the effective sample size.

```
[ ]: # Write your code here
```

Q 17: Based on this measure, which of the *valid* methods mixes the fastest (has the largest effective sample size)?
Write your answer here.
Q 18: *Optional* Select this method, and use it to complete problem 11.6 (b) in BDA.

```
[ ]: # Write your code here.
```

Q 19: *Optional* The Ising model is often introduced since it provides a useful toy environment for exploration and experimentation. I've outlined an experiment below that could be adapted as part of a mini-project.

Select the fastest method, and use it to explore the mixing time (total number of samples divided by effective number of samples), as a function of gridsize and temperature. I'd suggest using a logarithmic spacing for both, a coarse grid size sampling (say sizes 4, 16, 64, 256), and a fine temperature spacing.

In particular, if you plot the average absolute magnetization (after burn-in) as a function of temperature for each grid size, you should see a transition between a demagnetized and a magnetized state as the material cools. This transition should sharpen, as a function of temperature, as the grid size increases. The critical temperature for the material is, roughly speaking, the temperature at which this transition occurs. We can estimate it hueristically by either selecting the temperature at which the average absolute magnetization crosses $1/2$ its peak value (value at absolute zero), or, where it changes the quickest as a function of temperature.

Estimate the critical temperature for each grid size, then make a plot showing the mixing time as a function of grid-size at three different temperatures: near absolute zero, at the critical temperature per grid size, and at a very high temperature. Pick your axis scales appropriately (e.g. linear vs log). How does the mixing time depend on the temperature? Is it slowest at criticality?

```
[ ]:
```

Lab 9: Hamiltonian Monte Carlo, Posterior Predictive Checks, and Empirical Bayes with PyMC

Lab Date: Wednesday, Apr 16
Lab Due: Wednesday, Apr 30

Instructions

Work with your lab group to complete the following notebook. Your work will be reviewed by your peers in two weeks (Wednesday, April 30)

In this lab, you will: 1. **Sampling from an intractable posterior using Hamiltonian Monte Carlo (HMC):** Use PyMC to sample from a Bayesian logistic regression model where the posterior is analytically intractable.

2. **Coverage:** Check the accuracy of your posterior sampling procedure by performing a coverage check.

3. **Posterior Predictive Checks (PPC):**
   Use the posterior samples to perform model checking.

If you are new to working in python, or in a Jupyter notebook, please ask your lab members for help. If you notice a lab member struggling, and have experience, please offer your help.

Please see this Ed post for corrections, questions, and discussion. If you would rather work with your own copy of the files, I have uploaded a zip folder there with the lab materials.

Corrections to the lab will be pushed directly to this notebook. We will only push corrections to the text, which is set to read only to prevent merge conflicts. In the event of a merge conflict, save your notebook under a different name, and click the link that launches the lab from the schedule on the stat238 homepage again. Then, check for discrepancies. If you can't find them, or resolve the conflict, contact us.

Problem Setup: Bayesian Logistic Regression

We consider a binomial logistic regression model. For groups $j = 1, \ldots, J$, we observe:

- $y_j \sim \text{Binomial}(n_j, \theta_j)$
- $\theta_j = \text{logistic}(\alpha + \beta x_j)$

Here: - x_j is a known covariate for group j - α, β are regression coefficients - $\text{logistic}(z) = \frac{1}{1+e^{-z}}$ is the inverse logit function

Prior Distributions

We place weakly informative priors: - $\alpha \sim t_4(0, 2^2)$ (Student-t with 4 degrees of freedom, mean 0, scale 2) - $\beta \sim t_4(0, 1)$

These priors allow for heavier tails than the normal distribution, making them robust (c.f. lab 6).

1. Import Required Packages and Define Helper Functions

[1]:
```
import numpy as np
np.set_printoptions(precision=4)
import pandas as pd
import pytensor
import os
os.environ["PYTENSOR_FLAGS"] = "cxx="
pytensor.config.cxx = ""
import pymc as pm
import arviz as az
import scipy.stats as st
from scipy.special import expit  # logistic (inverse-logit) function
from tqdm import tqdm

# Set a random seed for reproducibility
np.random.seed(123)
```

```
# My plotting settings
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import rcParams
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'''
\usepackage{newpxtext}
\usepackage[vvarbb]{newpxmath}
\usepackage{amsmath}
\DeclareMathOperator*{\E}{\mathbb{E}}
\DeclareMathOperator*{\Var}{Var}
'''
rcParams["font.family"] = "Palatino"
rcParams['figure.dpi'] = 300
```

2. Simulate the Dataset

Q 2.1: Simulate a dataset using the model:

$y_j \sim \text{Binomial}(n_j, \text{logit}^{-1}(\alpha + \beta\, x_j))$

where $J = 10$, the covariates x_j are drawn from Uniform$(-1, 1)$, and the sample sizes n_j are drawn from a Poisson with $\lambda = 5$ (reject any zeros). Use $\alpha_{\text{true}} = 0.5$ and $\beta_{\text{true}} = -1.0$ for simulation.

[2]:
```
# Set simulation parameters
J = 10
alpha_true = 0.5
beta_true = -1.0

# Draw x_j from Uniform(-1, 1)
x = np.random.uniform(-1, 1, size=J)

# Draw n_j from a truncated Poisson (reject zeros)
n = np.empty(J, dtype=int)
for j in range(J):
    nj = 0
    while nj < 1:
        nj = np.random.poisson(lam=5)
    n[j] = nj

# Compute the probabilities theta_j = logistic(alpha_true + beta_true * x_j)
theta = expit(alpha_true+beta_true*x)

# Draw y_j ~ Binomial(n_j, theta_j)
y = st.binom.rvs(n, theta)

print("x:", np.round(x, 2))
print("n:", n)
print("y:", y)
```

```
x: [ 0.39 -0.43 -0.55  0.1   0.44 -0.15  0.96  0.37 -0.04 -0.22]
n: [3 4 8 4 8 7 5 9 4 3]
y: [1 3 7 3 3 5 2 5 3 1]
```

3. Build and Sample the Model in PyMC

PyMC is a powerful, open-source library for probabilistic programming in Python that simplifies the process of building and fitting Bayesian models. It provides an intuitive interface for defining complex statistical models and performing inference using state-of-the-art sampling algorithms such as NUTS. If you're new to PyMC or Bayesian modeling, check out the official PyMC documentation for tutorials, examples (e.g. GLM Linear Regression), and comprehensive guidance.

If you get stuck in the documentation, check with the TA during lab or ask for help in office hours.
Q 3.1: In the space below:

- Define the Bayesian model using PyMC.
- Use Student-t priors with 4 degrees of freedom for α (scale = 2) and β (scale = 1).
- Define the likelihood: $y_j \sim \text{Binomial}(n_j, \theta_j)$ with $\theta_j = \text{logit}^{-1}(\alpha + \beta\, x_j)$.
- Sample from the posterior using the No-U-Turn Sampler (NUTS), a type of HMC.

[3]:
```
with pm.Model() as model:
    # Priors for alpha and beta using Student-t distributions
    alpha = pm.StudentT("alpha", nu=4, mu=0, sigma=2) # an example call to pymc defining a
distributional model component
    beta  = pm.StudentT("beta" , nu=4, mu=0, sigma=1)

    # Logistic regression: theta = logit^{-1}(alpha + beta * x)
    # use the pm.Deterministic function to fix a deterministic model component
    theta = pm.Deterministic("theta", pm.math.sigmoid(alpha + beta * x))

    # Likelihood: y_j ~ Binomial(n_j, theta_j)
    # use the observed = argument to fix the observation
    y_obs = pm.Binomial("y_obs", n=n, p=theta, observed=y)

    # Sample using NUTS (HMC)
    # use the pm.sample function
    sample_trace = pm.sample(draws=200, tune=200, progressbar=False)

# Display the posterior summary for alpha and beta
print(az.summary(sample_trace, var_names=["alpha", "beta"]))
```

```
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta]
Sampling 4 chains for 200 tune and 200 draw iterations (800 + 800 draws total)
took 2 seconds.

        mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_bulk  ess_tail  \
alpha  0.520  0.300  -0.030    1.048      0.014    0.011     483.0     448.0
beta  -1.108  0.579  -2.114    0.025      0.022    0.016     694.0     662.0

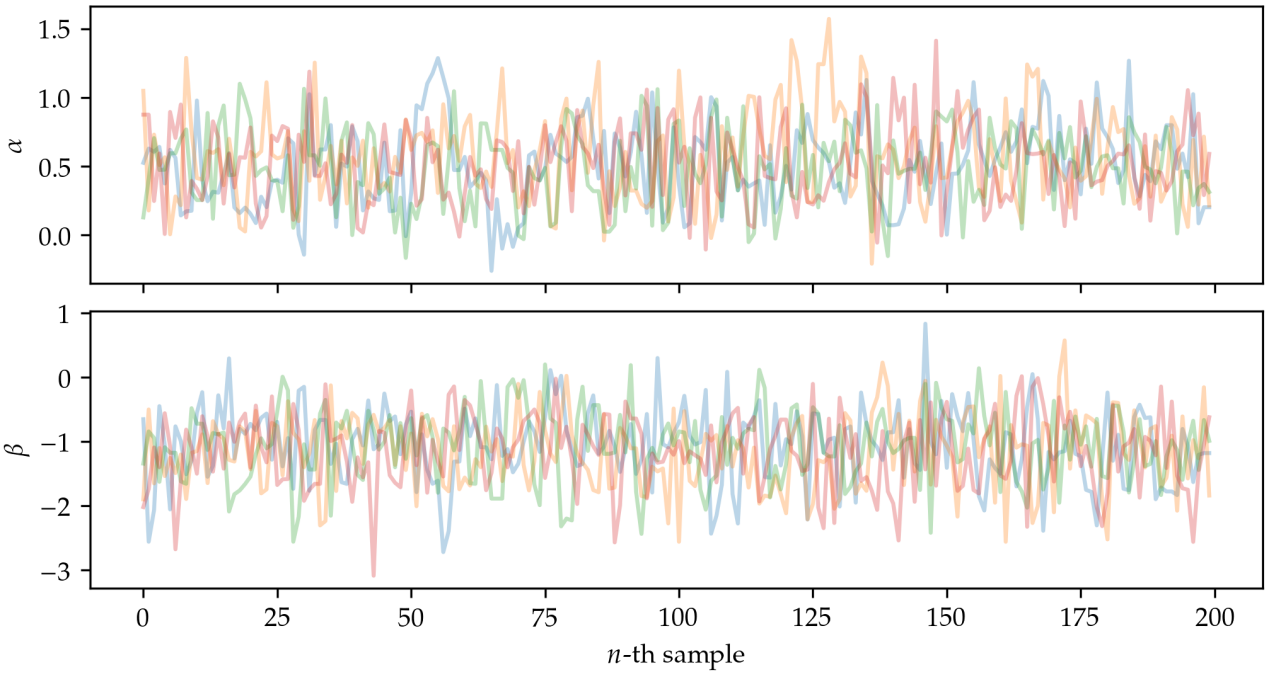
       r_hat
alpha   1.01
beta    1.00
```

Q 3.2: Adjust the sampling parameters (primarily, the number of steps taken) until you are convinced that the chain has mixed, and that you have enough samples to reliably represent the posterior. Explain the criteria you used in the space below.

[4]:
```
alpha_samples = sample_trace.posterior["alpha"]
beta_samples  = sample_trace.posterior["beta"]

fig, (ax1, ax2) = plt.subplots(nrows=2, figsize=(8, 4), sharex=True)
plt.subplots_adjust(hspace=0.1)

for alpha_c, beta_c in zip(alpha_samples, beta_samples):
    ax1.plot(alpha_c, alpha=0.3)
    ax1.set_ylabel(r"$\alpha$")
    ax2.plot(beta_c, alpha=0.3)
    ax2.set_ylabel(r"$\beta$")
    ax2.set_xlabel("$n$-th sample")
```



I plotted the 4 HMC chains for both α and β and found that they both mix fairly after about 200 samples. The effective sample size is just shy of 800 for both α and β (4 chains of 200 samples each).

4. Coverage

Q 4.1: Do your 50% posterior intervals for α and β (25th to 75th percentiles) contain the true values?
Yes! The posterior intervals are given by

Parameter	95% CI	Posterior Mean
α	$[0.13, 0.89]$	0.51
β	$[-1.49, -0.42]$	-0.82

[ ]:
```
alpha_ci = np.percentile(alpha_samples, [25, 50, 75])
beta_ci  = np.percentile(beta_samples, [25, 50, 75])

print("alpha 50% CI:", np.round(alpha_ci, 2))
print("beta  50% CI:", np.round(beta_ci, 2))

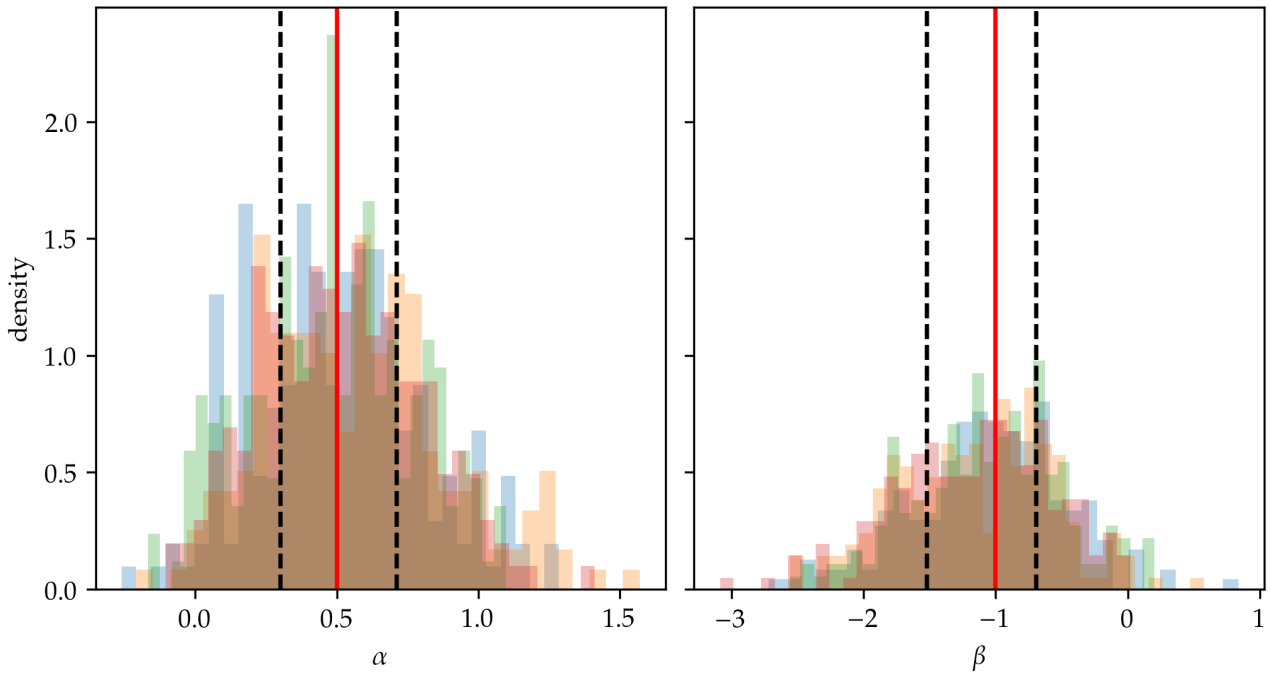
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(8, 4), sharey=True)
plt.subplots_adjust(wspace=0.05)

for alpha_c, beta_c in zip(alpha_samples, beta_samples):
    # Alpha
    ax1.hist(alpha_c, alpha=0.3, bins=30, density=True)
    ax1.axvline(alpha_ci[0], color="black", linestyle="--")
    ax1.axvline(alpha_ci[2], color="black", linestyle="--")
    ax1.axvline(alpha_true, color="red")
    ax1.set_xlabel(r"$\alpha$")
    ax1.set_ylabel("density")

    # Beta
    ax2.hist(beta_c, alpha=0.3, bins=30, density=True)
```

```
    ax2.axvline(beta_ci[0], color="black", linestyle="--")
    ax2.axvline(beta_ci[2], color="black", linestyle="--")
    ax2.axvline(beta_true, color="red")
    ax2.set_xlabel(r"$\beta$")
```

```
alpha 95% CI: [0.3  0.51 0.71]
beta  95% CI: [-1.52 -1.08 -0.69]
```



Our posterior inferences are approximate since they depend on an MCMC procedure which only samples from the exact posterior given infinitely long to run. To check that your sampling procedure reliably represents the true posterior, perform a coverage check.

Q 4.2: That is:

1. Generate a series of m replicate data sets sampled from the full data generating model (sample α, β from the prior, resample x and n using 10 observations as before (alternately, you can check coverage conditional on the same x and n from the main test case), then sample a series of new y's.
2. For each, run your sampling code to generate approximate samples from the posterior. For each set of samples build 65%, 80%, 90%, and 95% credible intervals for α and β.
3. Track the fraction of replicates for which the sampled α and β land in each posterior interval. Return a table comparing the fraction of the time that the replicates should have been contained in the interval (e.g. 65% for the 65% intervals), and the fraction that actually were contained inside the intervals. Choose m large enough so that the observed fractions are sufficiently stable for evaluation.
4. Report the expected standard deviation in the observed fraction of posterior intervals containing the truth. If our posterior sampling procedure is exact, and we use an interval that is chosen to contain the truth p percent of the time, and we run m replicates, then the observed percent of replicates containing the truth is C/m where $C \sim \text{Binomial}(m, p)$. Use the standard deviation of C/m for each p to choose m.
5. Based on steps 3 and 4, evaluate whether the discrepancy between the observed fraction contained in each interval, and the predicted fraction, is plausibly explained by randomness in the trials, or is evidence of systematic error in your posterior sampling procedure.

Note that $\text{Var}(C/m) = \frac{1}{m^2}\text{Var}(C) = \frac{mp(1-p)}{m^2} = \frac{p(1-p)}{m} \implies \text{SD}(C/m) = \sqrt{\frac{p(1-p)}{m}}$. Suppose we want to cap our $\text{SD}(C/m)$ by 0.05. Then we need to set m to at least 100 for credible intervals of $[65, 80, 90, 95]$.

[6]:
```
# Define model for coverage check
with pm.Model() as coverage_model:

    # Turn off logging
    import logging
    logger = logging.getLogger("pymc.sampling")
    logger.propagate = False

    # Define the y as a pm.Data
    y_data = pm.Data("y_data", y)

    # Priors for alpha and beta using Student-t distributions
    alpha = pm.StudentT("alpha", nu=4, mu=0, sigma=2) # an example call to pymc defining a
distributional model component
    beta  = pm.StudentT("beta" , nu=4, mu=0, sigma=1)

    # Logistic regression: theta = logit^{-1}(alpha + beta * x)
    # use the pm.Deterministic function to fix a deterministic model component
    theta = pm.Deterministic("theta", pm.math.sigmoid(alpha + beta * x))

    # Likelihood: y_j ~ Binomial(n_j, theta_j)
    # use the observed = argument to fix the observation
    y_obs = pm.Binomial("y_obs", n=n, p=theta, observed=y_data)

coverages = [65, 80, 90, 95]
intervals = {c:[(100-c)/2, 100-(100-c)/2] for c in coverages}
within    = {c:{'alpha': 0, 'beta': 0} for c in coverages}
M = 100
for k in tqdm(range(M), desc='Running coverage check'):

    # Sample a alpha and beta
    alpha_k = st.t.rvs(df=4, loc=0, scale=2)
    beta_k  = st.t.rvs(df=4, loc=0, scale=1)
    theta_k = expit(alpha_k+beta_k*x)
    y_k     = st.binom.rvs(n, theta_k)

    # Posterior sampling
    with coverage_model:
        pm.set_data({"y_data": y_k})
        idata  = pm.sample(progressbar=False)
        alphas = idata.posterior['alpha'].values.flatten()
        betas  = idata.posterior['beta'].values.flatten()

    # Check coverage
    for i, interval in intervals.items():
        alpha_ci = np.percentile(alphas,interval)
        beta_ci  = np.percentile(betas ,interval)
        within[i]['alpha'] += 1 if alpha_ci[0] <= alpha_k and alpha_k <= alpha_ci[1] else 0
        within[i]['beta']  += 1 if  beta_ci[0] <= beta_k  and beta_k  <=  beta_ci[1] else 0

table = pd.DataFrame({
    'CI': [c/100 for c in coverages],
    'alpha approx CI': [round(within[c]['alpha']/M,2) for c in coverages],
    'beta approx CI' : [round(within[c]['beta'] /M,2) for c in coverages],
    'std. dev. approx CI': [np.sqrt((c/100*(1-c/100))/M) for c in coverages]
})
print(table)
```

```
     CI  alpha approx CI  beta approx CI  std. dev. approx CI
0  0.65             0.65            0.68             0.047697
```

```
1  0.80            0.77            0.81            0.040000
2  0.90            0.91            0.92            0.030000
3  0.95            0.94            0.98            0.021794
```

Q 4.3: In the space below, reflect on the role of coverage checking in a Bayesian analysis when we don't know the true data generating model. Does a coverage check validate the model? If not, what aspect of the Bayesian process is validated using coverage checks?

The role of coverage checking in Bayesian analysis is to check if the model pipeline is capable of returning accurate posterior inferences/summaries? By accurate, we mean that it is capable of capturing the true parameters of the model. A coverage check does not validate the model, as many models are capable of passing coverage checks without being the true unknown model.

Q 4.4: In the space below, explain why we need many replicates from the data generating model to perform a coverage check (why wasn't Q4.1 sufficient by itself)?

Q4.1 is simply a coverage check on 1 replicate. We need many replicates as we need to see if the model is able to cover different true values sampled from the prior. In our case, we have $\alpha \sim t_4(0, 2^2)$ and $\beta \sim t_4(0, 1)$ and we need many replicates as we want to see if we cover possible samples from this prior.

5. Posterior Predictive Checks (PPC) and Plotting the Fit

Now, generate posterior predictive samples to perform model checking.

Q 5.1: In the space below:

- Generate m posterior predictive samples conditional on the observed data, sample sizes n and covariates x. That is, resample α, β from the posterior (use your posterior samples), then generate replicate $y's$ given α, β, x and n.
- Create a plot comparing the proportions y_j/n_j in the observed data set and the proportions predicted by replication. I would suggest a scatter plot whose horizontal coordinate is the observed proportion, and whose vertical coordinate is the replicated proportion for each replicate. For each j, compute an interval that contains 95% of the replicate proportions, then overlay the interval on your scatter plot. Finally, add the line of parity (line of slope one) to check agreement.

Vary m until you believe you have enough replicates to answer the question, "could my data have plausibly been generated by this model?".

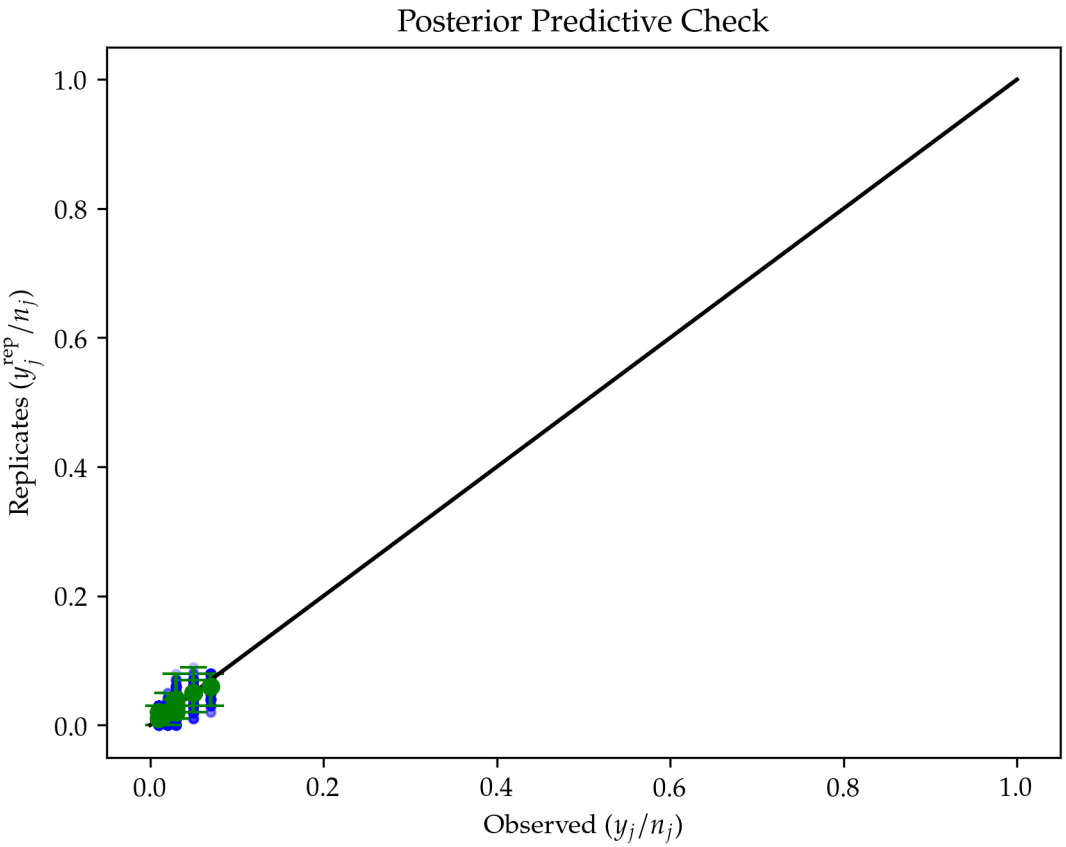
[21]:
```
# Sample many replicates
with model:
    # Turn off logging
    import logging
    logger = logging.getLogger("pymc.sampling")
    logger.propagate = False
    # Pull samples from posterior predictive distribution
    idata        = pm.sample(draws=1000, tune=1000, progressbar=False)
    y_rep_xarray = pm.sample_posterior_predictive(idata, var_names=['y_obs'], progressbar=False)
    y_reps       = y_rep_xarray.posterior_predictive['y_obs'].values.reshape(-1, J)

M = y_reps.shape[0]

true_prop = y/n
rep_prop  = y_reps/n
intervals = np.percentile(rep_prop, [5,50,100], axis=0)

_, ax = plt.subplots()
for true_p, rep_p, ci in zip(true_prop, rep_prop.T, intervals.T):
    ax.scatter(np.full(M,true_p),rep_p, color='b', s=10, alpha=0.1)
    lower, median, upper = ci
    error_color = 'green' if lower < true_p and true_p < upper else 'red'
    ax.errorbar(true_p, median, yerr=np.array([[median - lower, upper - median]]).T,
color=error_color,fmt='o', capsize=5)
ax.plot([0,1],[0,1],color='black',label='Line of parity')
```

```
ax.set_xlabel(r'Observed ($y_j/n_j$)')
ax.set_ylabel(r'Replicates ($y^\text{rep}_{j}/n_j$)')
ax.set_title('Posterior Predictive Check')
plt.show()
```



Q 5.2: To show that a PPC can reject mispsecified models, regenerate the data and rerun the PPC, but use a different model for generating the data than you do for posterior inference (e.g. try using two different prior parameters, one when you sample the data, and one when you run posterior inference). Vary the models until your PPC procedure would reject the model you used for inference.

[22]:
```
# Increase J
J = 100

# Prior parameters for the model used to generate data (y, x, n)
alpha_gen_params = {'df':100, 'loc':40, 'scale':1}
beta_gen_params  = {'df':100, 'loc':40, 'scale':1}
alpha_gen        = st.t.rvs(**alpha_gen_params)
beta_gen         = st.t.rvs(**beta_gen_params)

# Prior parameters for the model used for posterior inference
alpha_inf_params = {'nu':100, 'mu':-40, 'sigma':1}
beta_inf_params  = {'nu':100, 'mu':-40, 'sigma':1}

# Sample new data (y, x, n)
def sample_data(alpha_gen_params, beta_gen_params):
    alpha_true = st.t.rvs(**alpha_gen_params)
    beta_true  = st.t.rvs(**beta_gen_params)
    # alpha_true = 10
    # beta_true  = 10
```

```
    x = np.random.uniform(-1, 1, size=J)
    n = np.empty(J, dtype=int)
    for j in range(J):
        nj = 0
        while nj < 1:
            nj = np.random.poisson(lam=5)
        n[j] = nj
    y = st.binom.rvs(n, expit(alpha_true+beta_true*x))
    return y, x, n

# Sample data from the generating model
y_gen, x_gen, n_gen = sample_data(alpha_gen_params,beta_gen_params)

# Define the posterior inference model
with pm.Model() as inf_model:
    # Turn off logging
    import logging
    logger = logging.getLogger("pymc.sampling")
    logger.propagate = False
    alpha = pm.StudentT("alpha", **alpha_inf_params)
    beta  = pm.StudentT("beta" , **beta_inf_params)
    theta = pm.Deterministic("theta", pm.math.sigmoid(alpha + beta * x_gen))
    y_obs = pm.Binomial("y_obs", n=n_gen, p=theta, observed=y_gen)

# Pull y_reps from posterior predictive distribution
with inf_model:
    idata        = pm.sample(draws=100, tune=100, progressbar=False)
    y_rep_xarray = pm.sample_posterior_predictive(idata, var_names=['y_obs'], progressbar=False)
    y_reps       = y_rep_xarray.posterior_predictive['y_obs'].values.reshape(-1, J)

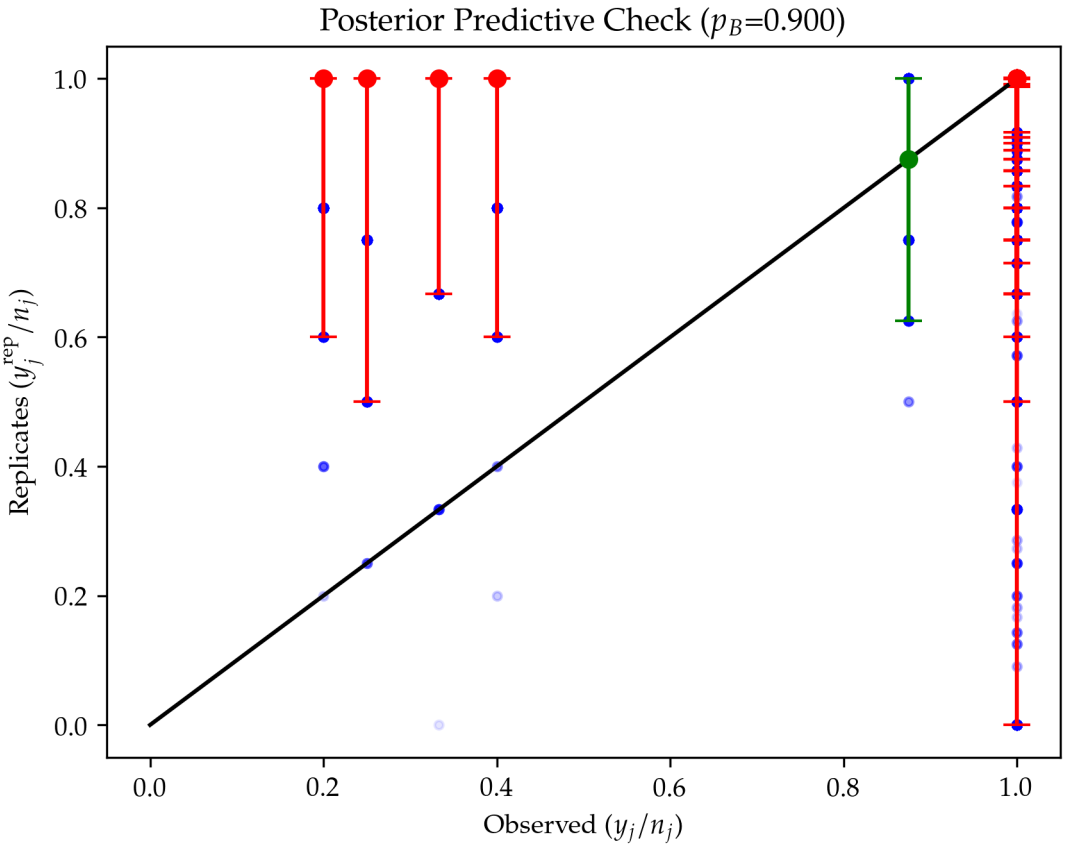
true_prop = y_gen/n_gen
rep_prop  = y_reps/n_gen
intervals = np.percentile(rep_prop, [5,50,100], axis=0)

# Calculate p-value
m, n = rep_prop.shape
s_geq = 0
for true_p, rep_p in zip(true_prop, rep_prop.T):
    s_geq += np.sum(rep_p>=true_p)
p_value = s_geq/(m*n)

_, ax = plt.subplots()
for true_p, rep_p, ci in zip(true_prop, rep_prop.T, intervals.T):
    ax.scatter(np.full(m,true_p),rep_p, color='b', s=10, alpha=0.1)
    lower, median, upper = ci
    error_color = 'green' if lower < true_p and true_p < upper else 'red'
    ax.errorbar(true_p, median, yerr=np.array([[median - lower, upper - median]]).T,
color=error_color,fmt='o', capsize=5,)
ax.plot([0,1],[0,1],color='black',label='Line of parity')
ax.set_xlabel(r'Observed ($y_j/n_j$)')
ax.set_ylabel(r'Replicates ($y^\text{rep}_{j}/n_j$)')
ax.set_title(f'Posterior Predictive Check ($p_B$={p_value:0.3f})')
plt.show()
```

```
There were 1 divergences after tuning. Increase `target_accept` or
reparameterize.
The rhat statistic is larger than 1.01 for some parameters. This indicates
problems during sampling. See https://arxiv.org/abs/1903.08008 for details
The effective sample size per chain is smaller than 100 for some parameters.  A
higher number is needed for reliable rhat and ess computation. See
```

```
https://arxiv.org/abs/1903.08008 for details
```



[12]:
```
m, n = rep_prop.shape
print(m,n)
s_geq = 0
for true_p, rep_p in zip(true_prop, rep_prop.T):
    s_geq += np.sum(rep_p>=true_p)
    # print(true_p, rep_p)
    # print(np.sum(rep_p>=true_p))

p_value = s_geq/(m*n)
print(p_value)
```

```
4000 100
0.7245475
```

Q 5.3: Explain how you would formalize the graphical PPC described above as a hypothesis test, and, based on your experience in Q5.2 whether your PPC procedure would make a sensitive (e.g. a powerful) test for model misspecification with respect to the parameter you chose. Explain why you believe your procedure would form a sensitive, or insensitive, test for the chosen parameter.

Much like a coverage check, we can perform a posterior predictive check as follows:

1. Sample posterior given the observed data $\{\theta_j^{\text{post}} \sim p(\Theta \mid Y_j = y_j^{\text{obs}})\}_{j=1}^{n}$
2. Sample likelihood given posterior samples $\{y_{jk}^{\text{rep}} \sim p(Y_j \mid \Theta = \theta_j^{\text{post}})\}_{k=1}^{m}$
3. Calculate summary statistics $s_{jk}^{\text{rep}} = T(y_{jk}^{\text{rep}}, \theta_j^{\text{post}})$ and $s_j^{\text{obs}} = T(y_j^{\text{obs}}, \theta_j^{\text{post}})$
4. Perform posterior predictive check: Are the summary statistics generated by the model plausible? Or are they extreme values? That is, are the replicates s_{jk}^{rep} almost always smaller than the observed summary statistic s_j^{obs}?

5. Calculate posterior predictive p-value:

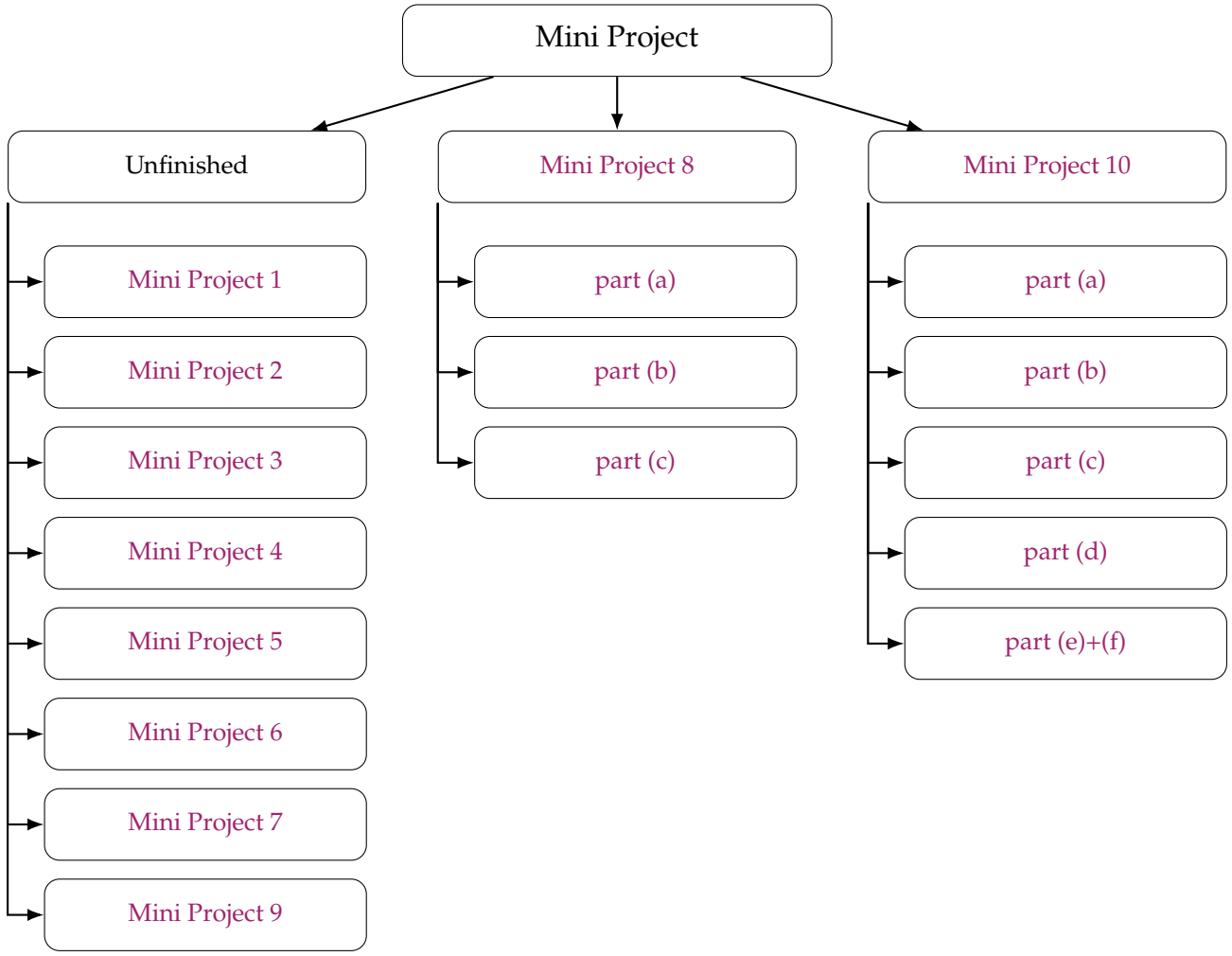
$$p_B = \mathbb{P}\left(s^{\text{rep}} \geq s^{\text{obs}} \mid y^{\text{obs}}\right) = \mathbb{P}\left(T\left(y^{\text{rep}}, \theta^{\text{post}}\right) \geq T\left(y^{\text{obs}}, \theta^{\text{post}}\right) \mid y^{\text{obs}}\right)$$

$$p_B = \int\int \mathbb{1}\left\{T\left(y^{\text{rep}}, \theta\right) \geq T\left(y^{\text{obs}}, \theta\right)\right\} p(y^{\text{rep}} \mid \theta) p(\theta \mid y^{\text{obs}}) dy^{\text{rep}} d\theta \approx \frac{1}{n}\frac{1}{m}\sum_{j=1}^{n}\sum_{k=1}^{m} \mathbb{1}\left\{s_{jk}^{\text{rep}} \geq s_j^{\text{obs}}\right\}$$

For example, if half of the replicates are larger than the observed data and half are smaller then $p_B \approx 0.5$ implying that the replicates are plausible. However, if almost all of the replicates are smaller than the observed data then $p_B \approx 0.01$. This implies that our replicates are not plausible and it's possible that our model is misspecified.

The power of the test depends on the sensitivity of the model. I have to change the parameters somewhat significant before I want able to see a failure. It also depends on the test statistic. In our case, we choose $T(y) = y/n$ which is a rather sparse space, especially with $n \sim \text{Poisson}^{+}(5)$ as we rarely see $n > 10$, so $T(y)$ can only be one of 10 values.

Mini Projects



Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **11:00 am** Tuesday, February 25, 2025

Submission Instructions

Project minis should be uploaded to Gradescope under the appropriate mini-assignment. If you work in a group (eligible projects are marked), you should submit one assignment, and link your partners. Please do not submit duplicated assignments.

Please read the project description in full before starting with your group. You do not need to read all the logistical details if you are not interested in this project, simply wait for a new one.

Mini Project 1: Dead Horse Talk

Is Bayesian statistics a valid epistemology?

The validity of Bayesian inference as a scientific epistemology (theory of knowledge) is one of the oldest and most acrimonious debates in statistics and the philosophy of science. It is often summarized as the "Bayesian vs. Frequentist" debate, and is, in many ways, a dead horse topic that most modern statisticians have decided to stop beating. Indeed, this class will follow the zeitgeist, which says "Oh, pha! (throws up hands) Enough with that albatross. We'll proceed with what works and leave the debate to the philosophers."

Does it matter?

Nevertheless, as members of a Bayesian stats class, you should be able to speak intelligently on this topic. In particular, you should be educated enough to clearly voice an opinion. After all, what is statistics, at its core, if not a philosophy of knowledge? To put it bluntly, what else do we really talk about when we talk about statistics?

A downside of the technocratic approach (let's only look at methods), is that it fails to prepare students to speak up and to defend their opinion regarding methods.[1] At worst, it can leave practitioners unequipped, or unwilling, to rebut criticism from "deeper" or more "rigorous" thinkers. It would be a real failure for Bayesian statisticians to cede their voice, and the value of their experience, in debates over the philosophy of their practice.

[1] In part, the trend here is cultural. Scientific training tends to avoid topics and exercises that are more traditionally associated with the humanities - in part - from a misplaced sense of exceptionalism or taboo.

So, while we will not discuss this debate at length in class, it is my hope that, if you are interested in it, you have the opportunity to learn about it, and to define your position. This project mini aims to provide that space. In this project, you will debate the validity of Bayesian statistics as a philosophy of science, as well as the relevance of philosophies of science to scientific practice. There are deep questions here regarding the effectiveness of mathematical models, the meaning of valid inference, the substance and mediums of scientific "progress", the existence and "know-ability" of ground truth, etc.

(a) **Group Formation:** This is a group discussion project. You may form groups of up to four. To find partners, either talk to your friends in class, your lab group, or, post in the Ed megathread associated to this project. If you intend to complete this project, you must have formed a group by the end of this week (Sunday, February 2nd).

(b) Consider the following extreme poles of opinion:

- **The cynic** argues against the validity of Bayesian statistics.

  Science advances deductively via falsification. We can only disprove hypotheses. Prior modeling encodes belief, but mixes it with apparently objective mathematics that obscure the introduced biases. Beware the modeler's safety blanket, "All models are wrong, some models are useful ... so it's ok if my model is wrong." Probability is only concretely meaningful in the frequency sense, with respect to variables that are, in principle observable (contrast with unknowns that are never directly observable, such as regression coefficients). Thus, in practice, and in theory, Bayesian statistics is mathematically consistent, but insidiously subjective.

  At worst, this opinion relapses into risk-averse pedantry that eschews the possibility of any positive theory of knowledge.

- **The believer** argues for the validity of Bayesian statistics.

  If it is valid to model observables as random given unknowns, then it seems inconsistent to insist that it is never valid to model the unknowns as random. Frequentist statistics feign objectivity, but don't discuss the subjectivity of likelihood modeling, or the reductionism of test frameworks. Arguing that we cannot reliably model prior knowledge argues, in its extreme, that we cannot, or do not, ever know things about our problems, that no unknowns are drawn from simple priors, or, that it is impossible to model complex distributions. Probability is an intuitive, self-consistent, and decision-theory optimal mathematical framework for expressing uncertainty and information. Science advances inductively by example. Information is acquired by conditioning on observations.

  At worst, this opinion relapses into recklessly subjective speculation - "my model told me so!" - or a tautology.

- **The pragmatist** argues that scientific practice should be shaped by utility/results.

  Philosophies of science are not relevant to scientific practice. The two communities are largely independent. Rather, science is as science does. Science does what worked. The astounding success of the scientific community need not be debated. Have you seen holograms? Lasers? Nuclear energy? Received an MRNA vaccine? Used plastic materials? Worked on a computer that uses microchips? Watched a video on a screen lit by LED's? *Blue* LED's? Gotten lost in the last fifteen years while synched to GPS? Need we debate this again? Let's get on with the work.

At worst, this opinion reduces into a statement of faith or into a small-mided exercise in drawing ever narrower boundaries around the problems we are willing to discuss.

- **The philosopher,** argues that scientific practice should be shaped by philosophical concerns regarding knowledge.

  All pragmatists are in the thrall of a dead methodologist. All scientific practices inherently advance an epistemology, implict or explicit. Applying methods without addressing their foundations surrenders a claim to the pursuit of truth.

  At worst, this opinion reduces to hopeless abstraction divorced from the lived practice of the communities of science, or from reasonably rich characterization of the real standards of evidence. Overly formal models of scientific progress are usually shabby fits to the real thing.

These poles have been arranged to caricature major sides of the debate. Imagine drawing a chart with two axes, and position the opinions on these axes. Take a moment, and work out, roughly, where you would place yourself in this space.

(c) **Positions:** For the sake of debate, each member of your group should adopt to represent a different corner of this space. Try to align your choices to your general beliefs, but, make sure that you are separated enough to hold a substantive debate.

(d) **Forum:** Create a time stamped Ed discussion channel with your group members. To do so, message the instructional staff stating your intention to complete this project and your team members. Email zhexiaolin@berkeleu.edu with the header *[STAT 238] - Mini Project 1 Group.* We will create the channel for you.

The debate will proceed in stages. Please check the italicized timeline associated with the stages below. This timeline is to ensure that you have time to read, digest ideas, and substantively respond to each other. Make sure everyone in your group has read these instructions before starting. We may dock points from groups whose work veers too far from the suggested schedule. In particular, if all the work is done in the last week, then the debate will not be accepted.

(e) **Format:** Please abide by the format outlined below. Discussions that vary too far from the provided format may not be accepted until corrected. *For all stages except 5 (open discussion), please start your posts with* ***[Stage <insert stage number>].***

(f) **Etiquette and Group Problems:** Ed is an extension of the formal course, so the course behavior expectations applied. In particular, since this project is posed as a debate, be careful to remain respectful. You are encouraged to challenge each other intellectually, but must remain collaborative, open, and decent.

If your group is experiencing unresolveable issues, please contact us (see the equivalent statements in the lab group policy).

(g) **Materials:** Each participant must substantively cite at least two relevant sources. Collectively, the group must cite at least four sources.

Your comments must show genuine engagment with your sources. *Reading at least two opinions in this space is required for this project.* I encourage you to suggest reading to each other, and point your partners to sources you find/found interesting. Sources

must satisfy the usual standards of quality (e.g. no blog posts except by petition). Course resources and required reading may be referenced, but will not count to your two sources. If you are unsure about a source, come ask us. If you are unsure where to start, we can point you to the course librarian. As much as possible, your arguments should either be sourced, or rooted in your experience.

For inspiration, look to the resources posted here and sources therein. The bibliographical note at the end of the first chapter of BDA is a good starting point. As is the bibliography in the required week 2 reading on philosophy and practice (Gelman). Other influential voices here include Thomas Kuhn (see, *The Structure of Scientific Revolutions*), Karl Popper (search *falsification*), David Hume (see his *problem of induction*), and, for the really enthusiastic, Schafer and Shapin (on the origins of empiricism, see *The Leviathan and the Air Pump*).[2] I was also recently pointed to Aubrey Clayton's work (see *Bernoulli's Fallacy*).

On submission, please collate and attach a bibliography, and update your posts to reference sources by number. If you want to make a bibliography post at the start, you are welcome to.

(h) **Submission:** One group member should create a pdf print-out of the Ed discussion and upload it to Gradescope before the deadline. As with all project minis, you will have the opportunity to resubmit based on instructor feedback for your final project portfolio.

Go to the next page to see the debate format

[2]This last source is a bit tangential, but is great if you want to learn about the formative debates that shaped the scientific method. It speaks more to the second pole than the first.

Debate and Discussion Format

(a) **Stage 0:** Before beginning the debate, draft a clearly defined summary of your true position (4 - 8 sentences). Post this privately in the group.

(b) **Stage 1: Leading Statements** Then, in turns, each state your assigned opinion in 2 - 3 sentences. I encourage you to state it as strongly and emphatically as responsibly possible (i.e. without reducing to a strawmen).

(c) **Stage 2: Argument and Elaboration** Read your partners leading (this is to help position your argument), then argue your position (don't rebut yet). Elaborate on your leading statement. Elaboration should be a paragraph to three paragraphs. You may use the reply function on your own leading statements. *Stages 0 - 2 should be completed within a week of forming your group, and at least two weeks before submission.*

(d) **Stage 3: Rebuttal** In turns, respond to your group member's positions. Please do not use the reply function. Instead, make successive posts so that the channel runs in order. You may use links to reference specific posts. Rebuttals should be a paragraph to three long.

(e) **Stage 4: Defense and Elaboration** Respond to your group members comments. Again, post linearly and link for reference. Your defense should not exceed four to five paragraphs.

(f) **Stage 5: Open Discussion** You may make short, open replies to your partners. These need not be oppositional/argumentative. This stage is an attempt to reach consesus, cede points, and to balance opinions. For this stage, you may use the reply function to nest your comments under specific posts. *Stages 3 - 5 should be completed at least one week before submission.*

(g) **Stage 6: Conclusions** Make a final post summarizing how your (true) position has, or has not changed, based on the discussion. Please be honest and reflective. You may use this space to cite comments a partner made that impacted your position, to suggest a question or source you would like to follow up on, how your opinion has evolved as you've learned more through the course, etc. This section must include a reference to your original position statement (private, stage 0). *Stage 6 should be completed in the last week before submission.*

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, March 7, 2025

Mini Project 2: Consistency, Inconsistency, Ishconsistency

In class we sketched the general asymptotic theory for posterior inference under "mild" regularity conditions. We've only discussed these informally.

In this project, you will be asked to identify and test the precise regularity conditions needed for the key theorems in BDA Chapter 4 and Appendix B. These theorems are, loosely:

1. *"Posterior point estimators (e.g. MAP) are consistent and asymptotically unbiased."*
2. *"The posterior concentrates about the truth, when available, and about a best fit to the truth when misspecified."*
3. *"The posterior distribution is asymptotically normal."*
4. *"The posterior distribution is asymptotically exponential in the KL divergence between the true data generating distribution and the conditional distribution proposed for the data under the model."*[1]

In this project, you will explore the limits of these statements.

Policies

- You may work with up to **2 partners**. To find partners, talk to your friends in class, or post on the Ed thread associated to this project mini.
- All submissions must be properly type-set and sourced.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment. If you work in a group (eligible projects are marked), you should submit one assignment, and link your partners. Please do not submit duplicated assignments.

Part I: Regularity Conditions

(a) Find, and state precisely, the minimal, sufficient, regularity guarantees needed for each of the four statements provided above. Make sure you clearly define any element that has been left loose in class, or that is new. Plainly cite the sources you use.

[1] (see Gelman and Shalizi, Philosophy and Practice of Bayesian Statistics.)

(b) Comment on the plausibility of each regularity condition. If you disagree as a group, record some of your debate. Try to identify an example where each might usually fail.[2]

(c) Elaborate (1 to 2 paragraphs) on one of your proposed examples. You will be asked to try and demonstrate this example at the end of the project.

Part II: Misspecification Gaurantees

The generic asymptotic theory applies in both the well and misspecified case. However, when misspecified, the posterior cannot concentrate about the truth. Instead, it concentrates about a model that is as hard to distinguish from the truth as possible.

In this part, you will explore the sense in which the posterior concentrates about the "best fit" parameters, and, whether the resulting "best fit" guarantees ensure that inferences made using misspecificed models will have small errors. In most tellings, the "best fit" model is "best fit" in the sense that it minimizes a Kullbach-Liebler (KL) divergence. Let's try to make sense of this notion.

(a) Is concentrating to the minimizer of the KL equivalent to concentrating to the MLE? Do the Frequentist and Bayesian inference procedures still agree in an infinite data limit?

(b) Relate the KL divergence between distributions p and q to the expected value of the log-likelihood-ratio test statistic used in a binary hypothesis test between p and q. In what sense does your answer imply that, if the KL divergence between two distributions is large, then they are easy to distinguish?

(c) The asymptotic theory states that the posterior concentrates about the parameters θ_* which minimize the KL divergence between the true data generating distribution, f_Y, and the modeled distribution $p_{Y|\theta}$. Use the Chernoff-Stein Lemma to translate this result into a conclusion regarding the distinguishability of the two distributions in a hypothesis test. Specifically, identify the sense in which minimizing the KL divergence selects the model $p_{Y|\theta}$ which is hardest to distinguish from the f_Y.

(d) Since we can always test for the difference between two distributions by comparing a hypothesized sampling distribution for a test statistics to the observed test statistic, argue that, if we consider a sequence of nested model families that approach a family including the true model, then, as the model families expand, the amount the sampling distribution of any test statistic under the true data generating process differs from its sampling distribution under the KL minimizing model, must vanish.

(e) Put your arguments together to argue that, if we use the KL between $p_{Y|\theta_*}$ and the f_Y to measure the degree of misspecification, then mild misspecification will induce mild errors in inference in the sense that the proposed generating distributions for the data, and sampling distribution for any tests statistic, are hard to distinguish.[3]

[2] Note: consistency is an asymptotic guarantee. We never collect an infinite amount of data, so the actual utility of an asymptotic theory follows from its usefulness in providing approximations that hold for sufficiently large data sets. The closer a problem gets to violating the regularity conditions, the more samples are usually needed before the asymptotic theory provides useful approximations.

[3] As usual, beware the allure of asymptotics... really we should want to know what KL divergence is *small enough* to ensure all other errors are small.

(f) Does this guarantee also imply that the best fit parameters are close to the true parameters if the model could be extended to include a parameterization of the f_Y? Try to identify a counter-example if your answer is no. You will be asked to try and test this idea using your counter-example in Part III. Discuss when errors in the inferred unknowns do or do not matter.[4]

(g) Assume, for this part, that θ_* is on the interior of the set of possible parameters. Taylor expand the KL divergence between $p_{Y|\theta}$ and the f_Y to second order in θ about θ_*. Then substitute into statement 4 on page 1 to derive an asymptotic approximation to the posterior. Does this approximation agree with our standard asymptotic approximation?

Part III: Exploration

Now that we have a rich idea for the theory, let's inspect it through example. Propose two Bayesian models and use them to demonstrate some of the ideas you discussed above.

- One should be a conjugate model, where, to study misspecification, you fix a subset of the prior parameters to the wrong values, and let the rest vary. Your model should be a member of a broader conjugate class which does include the truth. For example, consider a conjugate Bayesian model with a normal likelihood, normal prior on the mean, and with fixed, but incorrect covariance.
- One should be a non-conjugate model. It helps, as before, to choose a model parameterization that could include the truth, but, when you perform inference, you incorrectly fix some parameters. For example, you could try a logistic generalized linear model where you exclude necessary predictor variables, or, a Gaussian mixture model where you use the wrong number of mixture modes.

(a) Propose a pair of models and explain why you think they are interesting.

(b) Propose some characteristics of your posterior distributions that you might care about, or might return as inferential summaries (e.g. the point estimators, a notion of an interval estimator, a notion of spread). You can use discrepancies in these characteristics as both measures of the difference in two distributions, and, as practical measures of the inferential error incurred when using the wrong distribution.

(c) Implement code that can (a) generate data from your models, (b) perform posterior inference, and (c) compute or approximate the quantities you identified as your posterior summaries. You are welcome to use existing packages to implement your models.

(d) Try to illustrate the importance of the regularity condition you identified as suspect in Part I. Was your concern justified in the case you implemented? How strongly does your example violate the asymptotic theory? If it doesn't violate the asymptotic theory exactly, but is close to violating the asymptotic theory, discuss how many samples are needed before the asymptotic theory gives reasonable approximations.[5]

(e) Try to illustrate the biasing effects of misspecification. Does a small misspecification (in the sense of small KL) imply that the ensuing errors in posterior inference are small?

[4] Is your focus inference or prediction?

[5] See, for example, HW 2 problem 1.

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, March 21st, 2025

Mini Project 3: Bayes Optimal Coding, Search, and Experimental Design

Policies

- You may work with up to **1 partner**. To find partners, talk to your friends in class, or post on the Ed thread associated to this project mini.
- All submissions must be properly type-set and sourced.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment. If you work in a group (eligible projects are marked), you should submit one assignment, and link your partners. Please do not submit duplicated assignments.
- Please submit a single pdf. Properly export any notebook or code files to pdf so that their contents fit within the assigned pages.

Background

In class, we've seen a few examples of Bayesian decisions problems where we aim to select an optimal measurement scheme. These problems arise frequently in optimal experimental design (what experiment should we run?), search (what sequence of questions should we ask?), and coding (how can we optimally compress the codes used to represent a set of possible objects?). These problems are all related by a standard framing.

Imagine that you are asked to resolve an unknown Θ drawn from a prior distribution p_Θ. You have access to a set of possible measurements, $\mathcal{M}$, that you could make. Let $m \in \mathcal{M}$ denote a possible measurement. For experimental design, we might use $\mathcal{E}$ for a set of possible experiments we could run, and for search we might use $\mathcal{Q}$ for a set of possible questions we could ask. Since this assignment will mostly work with categorical unknowns drawn from a finite set, we will adopt $\mathcal{Q}$ for the set of possible questions we could ask, and $q \in \mathcal{Q}$ for a particular question.

If you select a particular question to ask, q, then you recieve back an answer $A \sim p_{A|\theta,q}$ where θ is the true value of the unknown, q is the question asked, and $p_{A|\theta,q}$ is the sampling distribution for the answer given the question and the true unknown. The sampling distribution fixes our likelihood function. Each distinct question we could ask produces a distinct likelihood function.

When you receive an answer, $A = a$, you condition on it to update your beliefs regarding the unknown. By combining the information you observed, a, and your prior over the unknown, you produce a posterior distribution for the unknown that incorporates the answer you received. Typically, we consider an answer informative if it leads to a large change in our beliefs (change from prior to posterior), or, a large reduction in our uncertainty regarding the unknown.

Often, we are allowed to ask questions sequentially. At each stage we can adopt our posterior following the n^{th} question, answer pair (q_n, a_n) as our prior when asking the $n+1^{st}$ question. Since your posterior will depend on both the questions previously asked, and the answers previously given, a recursive procedure for selecting questions will specify a decision tree that guides the sequence of questions asked based on the sequence of answers received.

Viewing the search procedure as the specification of a decision tree reveals a duality with coding. Suppose that you can only ask Yes/No questions. If you fix your decision tree over the questions you ask, and observe a string of answers that point to a specific unknown (e.g. "Y, N, Y, Y, N, N, N , Y" imply $\Theta = \theta$) you could assign that unknown the string "YNYYNNNY" that specifies the path through the tree pointing to it. Then, the tree specifies an encoding function c that, when given a value of the unknown, returns the string of answers that would specify it, e.g. $c(\theta) = YNYYNNNY$. If you adopt any set of d-ary questions (questions with d possible answers), then the string may be converted into a d-ary sequence. For example, we could use 1 for Y and 0 for N, then assign $c(\theta) = 10110001$. This might correspond to the sequence: "Is an animal. Is not a mammal. Is a bird. Is flightless. Does not live in Antarctica. Does not live in Africa. Does not share a name with a fruit. Is an emu."

Complete information codes that aim for optimal compression attempt to find an encoding that minimizes the expected length of the code $c(\Theta)$ when $\Theta \sim p_\Theta$. Since the length of the code is the number of questions asked, this is the same as searching for a question procedure that is expected to terminate as quickly as possible.

In this project you will experiment with two essential heuristics for designing optimal, or near-to-optimal codes/search procedures. These are outlined below:

- **Fano Code:** "*At each stage, select the question whose answer is expected to yield the most information.*"

  This is a greedy search procedure that builds the code forward from the root by iteratively selecting the maximally informative question. It aims to maximize the mutual information between the answer to each question, and the unknown. It is mildly suboptimal.

- **Huffman Code:** "*At any given stage, never ask the question that is expected to yield the least information.*"

  Given a finite set of possible unknowns, this amounts to the heuristic, "*always ask the lest informative question last.*" Once a question has been posed, the sets of unknowns distinguished by the question can be grouped, and the heuristic can be iterated. This procedure builds the tree recursively from the leaves, iteratively grouping the least likely remaining sets of outcomes.

The Huffman code is information-theoretic optimal for finite sets of unknowns and questions with deterministic answers conditional on the unknown. When the answer is non-deterministic, no set of outcomes can be exactly grouped or split by a particular question,

and most finite question sequences cannot terminate in exact recovery of the unknown with complete certainty. In this case, it is impossible to build the tree from its leaves without coarse graining the space of unknowns (converting from an infinite set of unknowns to a finite set by rounding), or by defining a sufficient certainty at which the process stops. As a result, Fano's heuristic is more popular given an infinite set of unknowns, or imprecise measurements. Measurement procedures designed to maximize the expected information gained by an observation are popular in scientific settings where there is a large space of possible experiments to run, and a well defined statistical model relating experimental outcomes to the unknown.

For a deeper introduction to these problems, and examples, see Chapter 5 from Thomas and Cover's *Elements of Information Theory* (Second Edition).

Example Problems

For this project, you will practice implementing both Fano and Huffman codes for three toy search problems. The first two are associated with famous word games. The third is the Kalman filtering problem introduced in Lab 4. The two games are outlined below:

- **Twenty-Questions:** Your goal is to guess an unknown word drawn from a fixed dictionary. You ask questions in stages about the unknown word. All questions must be Yes/No questions, and will return answers Yes or No deterministically given the unknown word. You aim to resolve the unknown as quickly as possible (in fewer than 20 questions). For this project, we will assume you have a prior distribution over the possible words, and aim to select a decision tree over questions that minimizes the expected number of questions needed to discover the secret word.

- **Wordle:** Wordle is a word guessing-game owned and hosted by the New York Times. In Wordle, the player aims to guess an unknown, five-letter word. Their set of questions consist of other five-letter words. When they propose a five-letter word, they are told whether a given letter appears in the true word, and, if it appears in the true word in the place it appeared in the proposed word. For a description of the rules, see this NYT article. To test a Wordle strategy, go to Wordledictionary.com.

  Our goal is to select a decision tree over five-letter words that resolves the unknown word as quickly as possible. For a primer on optimal strategies in Wordle see this webpage, or 3blue1brown's video on Wordle.

Resources:

In the Zip folder uploaded on Ed, you will find:

- **Unigram_freq.csv:** A comma separated array containing two columns: (1) a list of 330,000 words sorted in decreasing order of frequency (in internet use), (2) a list of counts of each word (in the associated trawl of the internet).

- **Worlde_word_list.txt:** A text document containing the standard Wordle dictionary. This dictionary is available here. A larger list that differentiates the list of possible search words and possible true words is available here.

You may also find it helpful to look up packages containing standard word-embedding packages (for instance, see this primer on standard word embeddings).

Prompts

Complete **three of the following four** tasks. You *must* complete the first and fourth tasks (*Twenty Questions: Unconstrained Questions* and *Kalman: Noisy Observations.*) You may choose whether to complete the second or third prompt.

Twenty-Questions: Unconstrained Questions

First, let's practice designing an optimal search/coding procedure for the twenty-questions game with no constraints on the set of possible questions. If we place no constraints on the set of possible questions, than any $q \in \mathcal{Q}$ can be specified by selecting a set of words, $\mathcal{Y}$, and its complement $\mathcal{N}$, such that $A|\theta, q = 1$ if $\theta \in \mathcal{Y}$, and 0 otherwise.

(a) (5 points) Trim the Unigram_freq.csv array, isolating the most common 3,000 words in the list. Then construct:

- A Huffman code by iteratively grouping the two least common words in the list. Proceed by first grouping the two least common items. Then group these into a set, and assign the set the frequency equal to the sum of the frequencies of the two grouped words. Assign the set a key, remove the two words from the list, add the key to the list, then resort the list in descending order. Repeat this process iteratively to build the decision tree. See Thomas and Cover Chapter 5.6 for reference.
- A Fano code by sorting the words in descending order, then splitting the list into two halves. This can be done by normalizing the word frequencies, then computing the cumulative distribution function for the words when sorted in order of decreasing frequency. Find the division point such that the CDF is as close to 0.5 as possible. Once divided into two sets, repeat the procedure within each set. See Thomas and Cover Chapter 5.7 for reference.

(b) (1 point) How many Yes/No questions would it take, on average, to recover the unknown word if it is drawn from the top 3,000 words in the Unigram_freq.csv list with prior probabilities proportional to the observed word frequencies? How much less efficient is Fano's approach than Huffman's?

(c) (1 point) Compute the Shannon entropy (base 2) in the prior distribution over the top 3,000 words. This is a lower bound on the expected number of questions it would take any procedure to resolve an unknown word. This bound is achievable per encoded word for coding procedures that attempt to encode multiple independent samples from the prior at once. How far are your codes from optimal?

Twenty-Questions: Constrained Questions

The methods you developed above are practical for coding/look-up problems that are implemented in a computer, and that do not place realistic constraints on the ensemble of questions that can be asked.

Let's try implementing a heuristic procedures that attempts to imitate Fano's approach (greedily select the maximally informative question) while placing some constraints on the questions that we can ask. In the following exercises, you will use a word-embedding to map our dictionary into an embedding space. Then, we will consider all questions that can be

formed by performing a linear measurement (computing a dot-product) in the embedding space. If the embeddings assign meaningful coordinates to the embedded words, then these questions better approximate the sorts of questions available to humans.

(a) (3 points) Find and implement a word embedding that embeds based on cosine-similarity (see Word2Vec and GloVe for examples). You do not need to train your own. You can find trained versions available on github, python, or in some libraries (for instance: Matlab's implementation and embedding.

(b) (5 points) Write a code that:

1. Projects the top 3,000 words onto the unit sphere in the embedding space by normalizing their embeddings. Discard any words that cannot be embedded. Weight each word by its prior probability in the reduced list, using the frequencies in "Unigram-freq.csv" as the prior before eliminating words.
2. Iteratively identifies a hyperplane passing through the origin that divides the point cloud of embedded words into two sets, each containing roughly 50% of the current probability mass over possible words.[1]
3. Asks the question, "Is the unknown word on side A or B of the hyperplane?" by computing a dot product between the embedded word and the normal vector to the hyperplane, then checking its sign.
4. Drops all words from the list appearing on the wrong side of the hyperplane.
5. Iterates the search procedure until the unknown word is resolved.

(c) (3 points) Estimate, by sampling, the expected number of questions needed to resolve the unknown word. How much less efficient is this procedure? Propose an alternate scheme for selecting the hyerplane orientation (i.e. the linear measurement) which might lead to faster method, or, explain why you think your current scheme is near-to-optimal given the constraint on the space of questions allowed.

Wordle: Structured Observations

Twenty-Questions offers an easy observation model. Wordle offers a more interesting example by constraining the set of available questions to five-letter words, and providing a more structured likelihood model. Let's try to design a Wordle strategy based on iteratively selecting the most informative question.

(a) (1 point) First, create two prior distributions over the words in the Wordle dictionary. In the first, assume that the words are uniformly selected. In the second, assume that they are selected according to the frequencies provided in the Unigram_freq.csv. Look up the frequency of each word in the Wordle dictionary and drop, from both prior distributions, any word that does not appear in Unigram_freq.csv.

(b) (6 points) Write a code that can, given a prior distribution over the possible unknown words:

[1] A heuristic approch: compute the covariance in embedded coordinates, then select the hyperplane perpendicular to the direction of largest variance.

1. Search through all pairs of possible search words (questions) and unknown words we are trying to guess. Make sure your search is restricted to unknown words that are possible given the current prior. For each pair, compute the answer returned if the search word is proposed, and the unknown word is the true hidden word.
2. For each search word, identify all possible answers, the prior probability of each answer, and the collection of possible true, but unknown words that could have generated that answer.
3. For each (search word, answer) pair, compute the posterior distribution over the unknown true words given the prior and the returned answer.
4. Compute the Shannon entropy in the posterior given each possible search word, answer pair.
5. Compute the expected Shannon entropy in the posterior for each search term by averaging the posterior entropy after each answer over the possible answers.
6. Select the search term with the smallest expected posterior entropy.

(c) (1 point) Implement your code to derive an optimal search procedure for both priors.

(d) (2 points) Compute the expected number of search words needed to resolve a draw from both priors. Which prior makes for an easier game? Try to explain your observation.

(e) (1 point) Compare the first couple words in the decision tree proposed by both methods. Are they similar? If not, how do they differ? Does the difference sensibly reflect the difference in the priors?

Kalman: Noisy Answers

In many real-world experimental design problems our unknowns are continuously distributed, and our measurements are noisy. The best procedures in these settings typically involve selecting from a restricted set of possible questions in order to maximize the expected information gain per question asked. For this part, return to the Kalman filtering method introduced in Lab 4. This method assumes that, at each stage, the unknown is normally distributed, and, we can observe a noisy linear measurement whose noise is multivariate normal.

Assume that x is an unknown state vector drawn from a multivariate normal distribution and m is a measurement vector with as many entries as x. Assume that all measurements take the form $y = m \cdot x + \zeta$ where $\zeta \sim \mathcal{N}(0, s(m) \times \sigma_y^2)$ where $s(m)$ determines how the noise scales with the size of the measurement vector. The set of available questions $\mathcal{Q}$ consists of all measurements available when m is drawn from a set of available measurement vectors $\mathcal{M}$.

(a) (2 points) First, assume that $\mathcal{M}$ is the unit sphere and that $s(m) = \|m\|_2{}^2$ where $\|\cdot\|_2$ is the Euclidean norm. Write a code that:

1. Can compute the unit vector m which maximizes the expected information gained by each measurement (see HW 2).
2. Iteratively adopts the measurement that maximizes the expected information gained inside the Kalman filters used in Lab 4.

(b) (2 points) How much more effective is this scheme than the fixed measurement schemes used in Lab 4 originally? Try re-running your experiments from Lab 4 to compare the

time it takes the filter to converge (in the Harmonic oscillator case), and its accuracy (in the Lorenz example).

(c) (1 point) Compute the "entropy rate" of the process. This equals, to a reasonable approximation, the differential entropy in the process noise distribution plus the log determinant of the Jacobian used in each update step, averaged across the collection of steps. Compute the "information rate" of the measurement process, that is, the average information gain per measurement, averaged over all measurements taken. See HW 2 for the relevant formula. For both models, check whether the information rate is greater than or less than the entropy rate. Does your answer explain whether the filter converges?

(d) (3 points) Usually we are not free to make arbitrary linear measurements. Instead, we are restricted to a set of measurements drawn from a discrete set. Let $\mathcal{M}$ consist only of the the canonical basis vectors $e_1 = [1, 0, \ldots]$, $e_2 = [0, 1, \ldots]$, and their unit linear combinations $e_1 + e_2$, $e_1 + e_2 + e_3$, $e_1 - e_2$, etc. Assume that $s(m) = \|m\|_1$.

Write a code that can:

1. Compute the vector $m \in \mathcal{M}$ that maximizes the expected information gained by each measurement (see HW 2). The number of possible measurement vectors is small in all cases, so you can use brute force here.
2. Iteratively adopts the measurement that maximizes the expected information gained inside the Kalman filters used in Lab 4.
3. Reproduces the visuals used in Lab 4, but colors the scatter points representing observations according to the measurement vector used. Add a legend mapping scatter color to the choice of measurement vector.

(e) (2 points) Apply this scheme to the Kalman filter examples in Lab 4, and compare your results to the results you saw when allowing any vector drawn from the unit sphere. Try to explain your observations. It may help to repeat the entropy rate vs information rate comparison you made before, using the constrained set of measurement vectors.

(f) (1 point) How much less effective is this method if you do not allow unit linear combinations (you only allow the basis vectors $\pm e_j$)? How does the information rate vary with the space of available measurements?

(g) (1 point) Comment on any patterns you notice regarding the choice of measurement vector.

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, March 14th, 2025

Mini Project 4: Model Critique

Policies

- This is an individual project.
- All submissions must be properly type-set and sourced. This means providing formal citations and a complete bibliography.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment.

Prompts

This is a reading project. You will be asked to find, read, and prepare a summary report on 3 research papers. Your report should **not exceed 6 pages.** It should be structured according to the instructions listed below.

(a) (3 points) Select three research papers that present a Bayesian model for an inference problem of your interest. The three papers must present at least two different prior models for any inferred unknowns.

(b) (2 points) Explain your interest in your inference problem of choice. Clearly motivate its importance, and explain the relevance of Bayesian modeling in your context.

(c) (4 points) Provide a comparative overview of the papers, emphasizing their similarities and differences. Highlight their position in the broader literature, and relation to each other. Explain why you have selected these three papers.

(d) (2 points) Summarize the likelihood model used in each paper. Explain any differences in approach.

(e) (4 points) Summarize the prior models used in each paper. Clearly state each modeling assumption and its justification. Explain any differences in approach.

(f) (2 points) Describe how each paper fits for unknown prior parameters, or handles uncertainty with respect to the choice of prior parameters.

(g) (2 points) Is there a clear asymmetry in the difficulty of, or attention paid to, modeling the prior vs modeling the likelihood? If so, why?

(h) (2 points) Try to describe the regularizing effect of each prior assumption. Highlight any prior assumptions that cannot be grounded empirically and whose biasing effects are unclear.

(i) (3 points) Position each paper's modeling approach with respect to the three aims discussed in class (realism, simplicity, tractability). Compare your evaluation to the author's presentation of their models.

(j) (4 points) Reflect critically on the model formulation. Do you think that the model priorities are appropriate for the inference problem of interest? Which assumptions are best justified, and which are you uncertain about?

(k) (2 points) Propose a change to the prior model formulation, or argue for one prior model over another.

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, March 28th, 2025

Mini Project 5: Bayesian Optimization

Policies

- This is a group project. You may work with up to 2 partners.
- All submissions must be properly type-set and sourced. This means providing formal citations and a complete bibliography when sources are used. Please export any notebooks as a single pdf and merge all components into one file. Only one group member should submit to gradescope. They must tag the other members when they submit.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment.

Prompts

This is a coding project. You will be asked to test and extend the Bayesian optimization algorithms you developed in lab 5 on a broader suite of test functions, to select one based on those tests, and apply it to a real optimization problem.

(a) (8 points) Complete the Q 2.7 - 2.10 in Lab 5 for the Ackley and Michalwiscz functions.

(b) (4 points) Select a test function from an applied optimization problem. This test function should be selected to model a real scenario where blackbox optimization (in particular, Bayesian Optimization using a GP surrogate) is relevant. These are optimization problems that share the following features:

- The objective is very slow or expensive to evaluate (as for large computer simulations, physical experiments, or for the hyperparameters of models that are expensive to train and test).
- Objective evaluations are noisy. This is an optional, but common, feature.
- Low-dimensional space of design parameters. Bayesian optimization is usually performed in moderately low-dimensional spaces. This is an optional, but common, feature. It enables global function approximation. If the design space is too large, Bayesian optimization will spend most of its effort exploring.
- Smooth but complicated objective. In particular the objective is usually not convex, and is often highly multimodal. This motivates approaches based on global function

approximation. If the objective is simple, then standard iterative optimizers based on local approximations are sufficient. This is an optional, but common, feature.

- Constraints. If the objective is simple (e.g. convex), but the design parameters are constrained, then, restricted to valid inputs, the optimization problem may still be difficult to solve with standard methods. Bayesian optimization offers two advantages here: global function approximation (avoids local maxima and requires exploration), and, if the objective is defined outside of the region of allowed inputs, then it may be sampled outside the constrained region to improve estimation inside the constrained region. This is a common, but optional feature.

You are welcome, and encouraged, to choose an optimization problem relevant to your own research or field. I've included some resources below to help you find examples.

There are a variety of libraries and packages that implement objectives from real applications. I've listed some sources below.

- **CompModels.** This is an R package. You can find the package here. It is documented here. It includes approximately 10 blackbox functions, largely drawn from engineering problems. See the documentation for two examples (e.g. the pressure vessel example).
- **BayesianOptimization.** This is a package built on scikitlearn and implemented in python. It includes examples that focus on hyperparameter optimization for classifier models. It is available here.
- **SAMBO:** This is a more implementation focused optimization package based on surrogate modeling. You could look here for a reference blackbox optimizers. SAMBO is available here.

For example applied papers and examples therein see:

- Sauer et al. *Vecchia-Approximated Deep Gaussian Processes for Computer Experiments.* Journal of Computational and Graphical Statistics. 2023.
- Gramarcy, RB. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences.* Chapman and Hall. 2020.
- Binois et al. *A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization.* 2022.
- Rajaram et al. *Empirical Assessment of Deep Gaussian Process Surrogate Models for Engineering Problems.* Aerospace Research Central. 2020.

(c) (4 points) Propose at least two Gaussian Process (GP) priors for your surrogates that you believe to be reasonable choices, and that you expect to produce different optimization behavior.

Choosing a GP means adopting a mean function and a covariance function.

1. Explain your choices for each (why are they reasonable). You may consider any of the three modeling aims discussed in class as arguments in favor of a prior (veracity, simplicity, tractability). Note: most surrogate models are chosen to be weakly informative based on loose smoothness guarantees.

2. Explain the difference between the models, how this effects sample draws from each model, and how you expect this to effect the behavior of the optimizer.

If you are lost here, Chapter 4 from Rasmussen and Williams (see the reference texts list on Ed) is a helpful resource.

(d) (6 points) Select one acquisition function/Bayesian optimization procedure, and apply it to your chosen example for both choices of your prior. You do not need to implement the optimization procedure yourself, but, must fully explain any tool you use. It must accept a GP prior specification, objective function, and standard acquisition function (e.g. an upper credible bound).

To compare the behavior of the optimizer:

- Make a plot showing the sequence of sampled objective function values and their running minimum.
- Make a plot showing the distance between the n^{th} and $n-1^{st}$ sample point from $n=2$ up until convergence. This will help you evaluate how the procedure focuses in on a proposed minimizer.
- Make a plot showing the negative log posterior probability of each observed sample, $-\log(p(f(x_j)|f(x_{j-1}), f(x_{j-2}), \ldots, f(x_1))$. This is a measure of how surprising each new observation is. The decay in the surprise measures concentration of the surrogate distribution to the true objective along the sequence of sampled points.
- A scatter plot showing the sequence of sample inputs at different stages of the procedure. I suggest coloring each point by the stage at which it was selected, and, if you have more than 2 design parameters, running PCA on the full set of sample points, then plotting the sequence of sample points in the space spanned by the first two principal components.

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, April 4th, 2025

Mini Project 6: Structured Models

Policies

- This is a group project. You may work with up to 2 partners.
- All submissions must be properly type-set and sourced. This means providing formal citations and a complete bibliography when sources are used.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment.
- Please export any notebooks as a single pdf and merge all components into one file. Only one group member should submit to gradescope. They must tag the other members when they submit.

Prompts

This is a model and inference project. You will be asked to select an inference problem of interest to your group, then propose either a generalized linear model as a likelihood, or a hierarchical model as a prior. You are welcome to look at chapters 5 and 16 in BDA for examples, but should select your own model specification.

(a) (8 points) Select an inference problem to study. The inference problem must admit a reasonable Bayesian model with either a structured likelihood (e.g. a generalized linear model), or a structured prior (e.g. a hierarchical model). This model must be relevant to the example, and of interest. Interest can be motivated by, (1) necessity (e.g. the model has the minimal components needed to perform inference), (2) standard practice (if so, proceed skeptically), or (3) curiosity. You should be able to clearly articulate why the model is interesting to you, course relevant, and appropriate for the problem you selected.

You may look for examples in BDA, in Bayesian Methods for Data Analysis (Carlin and Louis), or the reference source list on Ed. You are also welcome, and encouraged, to use an inference problem from your own research work. If you get stuck, please contact us on Ed. We can suggest sources, or point you to the course librarian.

In your write-up please:

1. Clearly explain your inference problem, its importance/your interest in it, and its course relevance. Clearly motivate Bayesian methods.
2. Clearly cite your source.
3. Explain the statistical model used in the source, then motivate the assumptions made. Place each with respect to the three modeling aims discussed in class (veracity, simplicity, tractability). This is a chance to think critically.
   - If the model uses a structured likelihood, clearly identify the model components. For example, given a generalized linear model, identify the features, link/inverse link, and conditional distribution of the data given the parameter predicted by passing the linear combination of the features through the link. How were the features, link, and distribution motivated?
   - If the model uses a structured prior, clearly identify the components. Discuss any exchangeability assumptions made, or mixture models used. Explain how the model structure shares information between observations. How were the distributions at each stage of the model motivated? How did the authors choose when to stop adding priors? (Standard answers are: stop when you run out of exchangeable variables to group, stop when you reach a prior that you know empirically, stop when you reach an uninformative prior, or stop when you think that further modeling is counterproductive.)
4. Check that the data used for inference is available. If it is not, but you can find an equivalent data set, cite the alternate data source, and explain why any differences in format may be ignored, or accounted for with model adjustments.

(b) (6 points) Implement the model. You do not need to implement everything from scratch. You will likely find it easier to adopt a probabilistic programming language (see, for example, pymc3, bambi, arviz). You may not borrow code provided by the authors. You are welcome to use it as reference if attached as supplement to the source, but should attempt to build your own implementation.

(c) (6 points) Try to replicate an inferential analysis performed in the source you selected. Note: replication is often frustratingly difficult, even with extensive documentation. Try to make this work. If it doesn't work, document your efforts as clearly as possible. This part will be graded based on demonstrated effort.

Your write-up should discuss:

1. What you did to replicate the study. Discuss any information missing here, and what you did to fill in gaps in the documentation.
2. Whether or not you could replicate your source's results. If you couldn't, explain why you think your results differed from the published results.

(d) (4 points) Describe a follow-up study you would be interested in performing based on your experience in this project. Alternate studies could include changes to the model architecture, application to related data sets, or changes to the inferential aim. Clearly provide one research question in the style of a proposal abstract.

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, April 4th, 2025

Mini Project 7: Exponential Family Proofs

Policies

- This is an individual project.
- All submissions must be properly type-set and sourced. This means providing formal citations and a complete bibliography when sources are used.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment.

Prompts

This is an analysis. You will be asked to prove the two theorems stated at the end of lecture 10.

In Lecture 10 we stated two remarkable facts about exponential families. First, if we observe a set of moments of an unknown distribution, then the exponential family with features set equal to those moments is, in a sense, the least informative distribution with the known moments. This justifies using exponential families to build minimally informed priors. Second, given an exponential family, the choice of parameters that maximize the likelihood for a set of observed sufficient statistics are equivalent to the parameters which equate the expected value of the feature functions and the observed sufficient statistics.

(a) (8 points) *Exponential Families and Maximum Entropy Priors (Discrete Unknowns):*

Suppose that X is a discrete unknown taking on finitely many possible values in $\mathcal{X}$ (that is, $\mathcal{X} = \{x_1, x_2, \dots, x_{|\mathcal{X}|}\}$ contains $|\mathcal{X}| < \infty$ possible states). Then, all p_X may be represented as nonnegative vectors in $\mathbb{R}^{|\mathcal{X}|}$ with entries that add to one.

Next, we need a way to measure how informative a distribution is. Let, $H[X] = H(p_X)$ denote the Shannon entropy:

$$H[X] = H(p_X) = -\mathbb{E}_{X\sim p_X}[\log_d(p_X)]. \tag{1}$$

The Shannon entropy measures the expected number of d-ary questions (e.g. $d = 2$ corresponds to yes/no questions) needed to resolve $X \sim p_X$ under an optimal search procedure. As such, large entropy means that p_X leaves X uncertain. The larger the entropy, the more information is needed to resolve X, so the less informative p_X.

Suppose that p_X is unknown, but some of the moments of p_X are known. In particular, let $\vec{\phi}(\cdot)$ be a vector-valued function of x: $\vec{\phi}(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_m(x)] \in \mathbb{R}^m$. Then, suppose that $\mathbb{E}_{X\sim p_X}[\vec{\phi}(X)] = \vec{\alpha}$.

Theorem 1: Given X taking values in $\mathcal{X}$, with moments $\mathbb{E}_X[\vec{\phi}(X)] = \vec{\alpha}$, the distribution $p_*(\vec{\alpha})$ that maximizes $H(p)$ over all distributions supported on $\mathcal{X}$ with moments $\mathbb{E}_{Y\sim p}[\vec{\phi}(Y)] = \vec{\alpha}$, is an exponential family with feature functions $\vec{\phi}(\cdot)$ and base measure $f(x) = \mathbf{1}(x \in \mathcal{X})$.

1. Write out the theorem as the solution to a constrained optimization problem on $p \in \mathbb{R}^{|\mathcal{X}|}$ with objective $H(p) = -\sum_{x\in\mathcal{X}} p(x)\log(p(x))$. In particular, write out constraint functions that require each entry of p is nonnegative, the entries sum to one, and the moments of $\vec{\phi}(X)$ match $\vec{\alpha}$. You should have $|\mathcal{X}|$ linear inequality constraints and $m+1$ linear equality constraints. Argue that these constraints define a convex polytope.
2. The Shannon entropy is a convex, continuously differentiable function of the input distribution. The optimizer of a convex, continuously differentiable object on a convex domain must satisfy the Karush-Kuhn-Tucker (KKT) conditions. These generalize the Lagrange conditions used given equality constraints. Write out the KKT conditions p_* must satisfy.
3. Compute each of the gradients needed to satisfy the KKT conditions.
4. Rearrange the KKT conditions to prove the theorem statement.

(b) (4 points) *Exponential Families and Maximum Entropy Priors (Continuous Unknowns):*

Attempt to generalize your proof for discrete $\mathcal{X}$ to continuous $\mathcal{X}$ (e.g. $\mathcal{X} \subseteq \mathbb{R}^d$ with a non-empty interior). This can be done in much the same method, but replacing vector-calculus with functional calculus. The KKT conditions still apply, and specify the optimal distribution. Now you will have one non-negativity constraint for each $x \in \mathcal{X}$, and $m+1$ linear equality constraints. To compute functional gradients let $\nabla U(f) = g$ if $\partial_h U(f) = \lim_{\epsilon\to 0}(U(f+\epsilon h) - U(f))/\epsilon = \langle g, h\rangle = \int_{x\in\mathcal{X}} g(x)h(x)dx$. That is, the functional gradient of a functional of U at f is the function $g : \mathcal{X} \to \mathbb{R}$ such that the inner product between g and h returns the partial derivative of U at f along perturbations in the direction h. This extends the usual definition used in finite-dimensional spaces.[1]

(c) (4 points) *Justifying Standard Priors:*

Using this result, create a table showing the support $\mathcal{X}$, and set of known moments, for which the following priors are minimally informative:

1. A beta distribution
2. An exponential distribution
3. A normal distribution
4. A gamma distribution
5. A Student's t distribution

[1] This is a tricky problem to set up. If you are unfamiliar with functional calculus, but want to try it, come talk to us or ask on Ed. The basic proof machinery is identical to the machinery used in finite-dimensions, but requires functional calculus.

(d) (8 points) *Maximizing Exponential Families and Method of Moments:*

Consider an exponential family of the form:

$$p(x;\vec{\phi}) = \frac{f(x)}{Z(\vec{\phi})}\exp(\vec{\phi}\cdot\vec{u}(x)). \tag{2}$$

with known features and base measure but unknown parameters $\vec{\phi}$. Suppose that $Z(\vec{\phi})$ is finite for all $\vec{\phi} \in \Phi$ where Φ is a convex set with a non-empty interior. Suppose, in addition, that the feature functions are linearly independent.

Suppose we observe $X^{(n)} = \{X_1, X_2, \dots X_n\}$ drawn i.i.d. from the exponential family. Let $\hat{\phi}_{\text{MLE}}$. Let:

$$\vec{s}(X^{(n)}) = \frac{1}{n}\sum_{j=1}^{n}\vec{u}(X^{(n)}) \tag{3}$$

be the sufficient statistics. Then:

Theorem 2: The maximum likelihood estimator for $\vec{\phi}$, $\hat{\phi}_{\text{MLE}}(x^{(n)}) = \text{argmax}_{\vec{\phi}\in\Phi}\{\text{Lik}(\vec{\phi}|X^{(n)} = x^{(n)})\}$, is the unique solution to the moment matching problem:

$$\text{Find } \vec{\phi} \text{ such that } \mathbb{E}_{X\sim p(\cdot;\vec{\phi})}[\vec{u}(X)] = \vec{s}(X^{(n)}). \tag{4}$$

Prove Theorem 2 by following the sequence of steps outlined in the lecture notes for lecture 10. You should proceed by:

1. Showing that the log partition function $\log(Z(\vec{\phi})$ has gradient equal to the expected value of the feature functions and Hessian equal to their covariance when X is sampled from $p(\cdot;\vec{\phi})$.
2. Use this conclusion to prove that the log partition function is strictly convex in ϕ given linearly independent feature functions (argue by contradiction).
3. Argue that, any strictly convex, twice-differentiable function admits an invertible mapping between its arguments and its gradient.
4. Show that setting the gradient of the likelihood to zero is equivalent to matching the gradient of the log-partition to the sufficient statistics.

STAT238: Bayesian Statistics May 2nd, 2025

Mini Project 8: Advanced MCMC Methods

Reece D. Huff

Prompts

This is a mixed coding/reading project. It builds directly on Lab 8. You will be asked to complete the mixing time experiments outlined at the end of the lab, use these to identify a model setting where standard Metropolis-Hastings (MH) algorithms mix slowly, then implement two modifications to the MH algorithm which often speed mixing. These are *simulated tempering* (see BDA 12.3) and *ensemble MCMC* methods. Ensemble methods evolve a population of random walkers together in parallel rather than a single walker.

Model

Background & Notation

The 2-D Ising model considers a collection of atoms arranged in a N-by-N grid. We let s_i represent the magnetic dipole moment of atom $i = 1, 2, \ldots, N^2$. Each atom has spin "up", $s_i = +1$, or spin "down", $s_i = -1$. We let $M : \{+1, -1\}^{N \times N} \to \mathbb{R}$ be the *average spin* across all atoms as it is typically used to measure *magnetization*:

$$M(s) = \frac{1}{N^2} \sum_i s_i.$$

The Ising model assumes the collection of spins is sampled according to a Boltzmann distribution,

$$\mathbb{P}(S = s) \propto \exp\left(-\frac{1}{\tau} E(s)\right) \quad \text{where} \quad E(s) = -\frac{J}{2} \sum_{\langle i,j \rangle} s_i s_j \quad \text{and}$$

- τ is the temperature (formally, the Boltzmann constant times the temperature),
- $E(s)$ is the potential energy stored in the magnetic field when the atoms are arranged with spins s,
- $\langle i, j \rangle$ is the summation over all nearest-neighbor pairs (as such, we divide by 2 to avoid double counting), and
- $J > 0$ is the strength of the magnetic interaction between neighboring atoms.

Suppose that we observed magnetization $m = M(s) + \zeta$ where $\zeta \sim \mathcal{N}(0, \sigma^2)$. Then our joint model is given by

Prior: $p(s) \propto \exp\left(-\frac{1}{\tau} E(s)\right)$
Likelihood: $p(m \mid s) \sim \mathcal{N}\big(M(s), \sigma^2\big)$
Posterior: $p(s \mid m) \propto p(m \mid s)\, p(s) \propto \exp\left(-\frac{1}{2}\left(\frac{m - M(s)}{\sigma}\right)^2\right) \exp\left(-\frac{1}{\tau} E(s)\right) \propto \exp\left(\frac{J}{2\tau} \sum_{\langle i,j \rangle} s_i s_j - \frac{1}{2\sigma^2}\left(m - \frac{1}{N^2} \sum_i s_i\right)^2\right)$

Example

Consider an example configuration in Figure 8. The energy of s_i is given by

$$E(s_i) = -Js_i \sum_{j \in \mathcal{N}(i)} s_j = -Js_i(+1 - 1 - 1 - 1) = 2Js_i$$

such that the probability of S_i being $+1$ and -1 is given by

$$\mathbb{P}(S_i = +1) \propto \exp\left(-\frac{1}{\tau}E(s_i = +1)\right) \implies \mathbb{P}(S_i = +1) = \frac{\exp(-2J/\tau)}{\exp(+2J/\tau) + \exp(-2J/\tau)} = \frac{1}{1 + \exp\left(-2\left(-\frac{2J}{\tau}\right)\right)}$$

$$\mathbb{P}(S_i = -1) \propto \exp\left(-\frac{1}{\tau}E(s_i = -1)\right) \implies \mathbb{P}(S_i = -1) = \frac{\exp(+2J/\tau)}{\exp(+2J/\tau) + \exp(-2J/\tau)} = \frac{1}{1 + \exp\left(2\left(-\frac{2J}{\tau}\right)\right)}$$



FIGURE 8: An example configuration of the Ising model. We let ✓ and ✗ represent when s_i and its neighbor are aligned and misaligned, respectively.

Metropolis-Hastings Sampling

Suppose we take $\tilde{p}_*(s) = \mathbb{P}(S = s)$ as our target density and sample from it via the Metropolis algorithm. Let p_r represent a symmetric proposal distribution of sampling a new configuration s' given the current configuration s. Then the acceptance probability is given by

$$a(s') = \min(r, 1) \quad \text{where} \quad r = \frac{\tilde{p}_*(s')\, p_r(s' \to s)}{\tilde{p}_*(s)\, p_r(s \to s')} = \frac{\exp\left(-\frac{1}{\tau}E(s')\right)}{\exp\left(-\frac{1}{\tau}E(s)\right)} = \exp\left(-\frac{1}{\tau}\left[E(s') - E(s)\right]\right) = \exp\left(-\frac{1}{\tau}\left[\Delta E(s, s')\right]\right)$$

We note that since $\tau > 0$, the only candidates s' with an acceptance probability less than 1 are candidates that a proposing a lower energy, i.e., $\Delta E(s, s') > 0$. Additionally, we note that as the temperature increases, the acceptance probability increases. We display this relationship below:

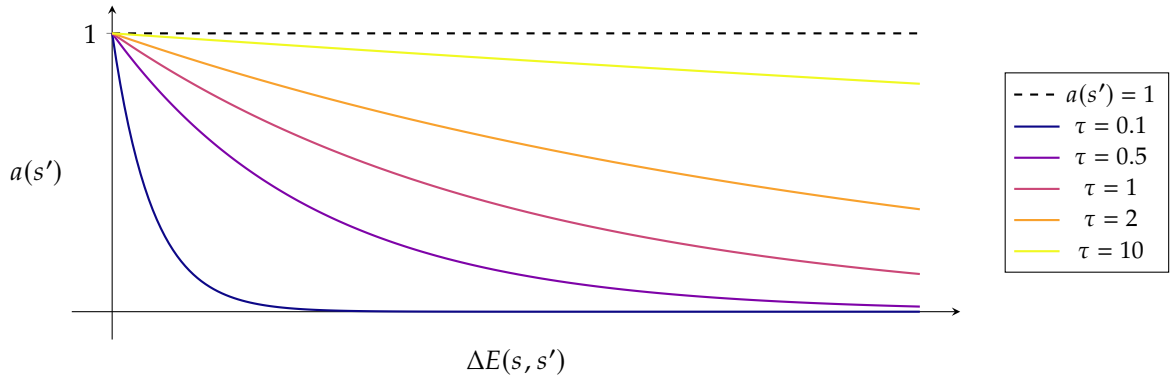


FIGURE 9: Metropolis-Hastings acceptance probability as a function of change in energy $\Delta E(s, s')$ for varied temperatures.

(a) Mixing Times for MH in the Ising Model

Complete the mixing time experiment marked optional at the end of Lab 8 using the Metropolis-Hastings algorithm with Proposal I (independent spin flips). Identify a setting (grid size and temperature) where the process mixes slowly (either, too slowly for MH, or, slowly enough the MH is barely feasible).

Report the results of your experiment. In particular, is mixing slowest near criticality, or at an extreme (very high or very low) temperature?

Experimental Results

- Mixing times are in Figure 10
- Average absolute magnetization, $\mathbb{E}[|M|]$, and estimated critical temperatures τ_{crit} are in Figure 11
- Energy and magnetization samples after warmup are in Figure 12

Discussion

Mixing times: We note that the mixing times largest when the temperature is low. This trend becomes especially clear for the larger grid sizes Figure 10. This is because the low temperature samples are more likely to be stuck in one of the two modes. More on why this is the case is discussed in part (c) 2.

Average absolute magnetization: The average absolute magnetization is largest when the temperature is low. This trend becomes especially clear for the larger grid sizes Figure 11. The critical temperature is estimated to when there is a phase transition in the average absolute magnetization and plotted in Figure 11. All of the estimated critical temperatures hover around the theoretical critical temperature of $\tau_{\text{crit}} = 2.269$ provided in Lab 8. This trend makes sense as when the temperature is low, the spins are more likely to be aligned (either all up or all down).

Energy and magnetization samples: The energy and magnetization samples after warmup are plotted in Figure 12. The nice trend emerges as the temperature changes. For example, when the grid size is 256, the magnetization samples are all close to 1 or -1 when the temperature is low while the hot samples seems to jitter around 0. This trend is also seen in the energy samples. The low temperature samples drive the energy to its minimum (all spins aligned) while the high temperature samples jitter around larger values.

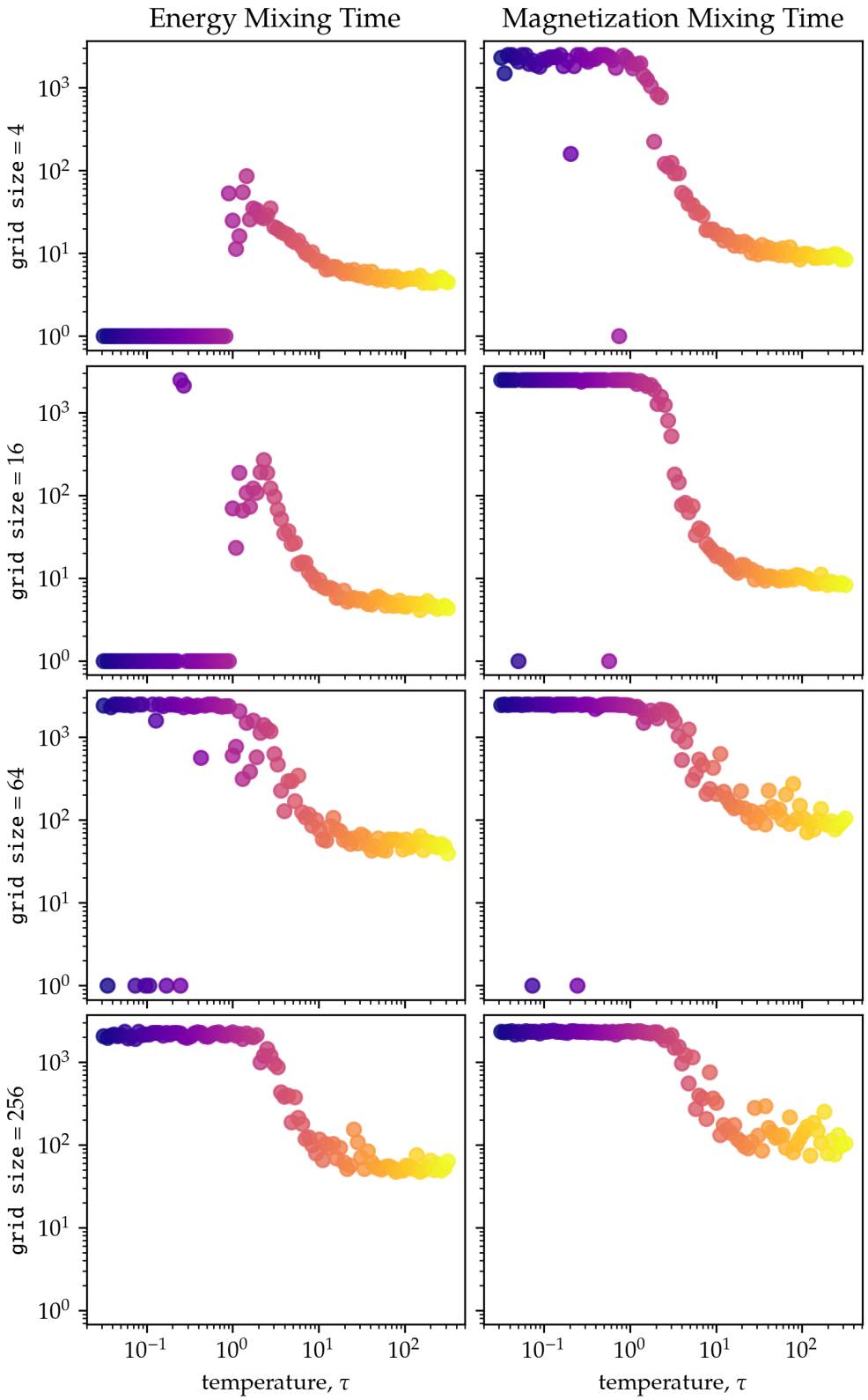


Figure 10: Energy $M(\mathbf{s})$ and magnetization $M(\mathbf{s})$ mixing time calculated as total number of samples divided by effective number of samples for grid sizes 4, 16, 64, and 256

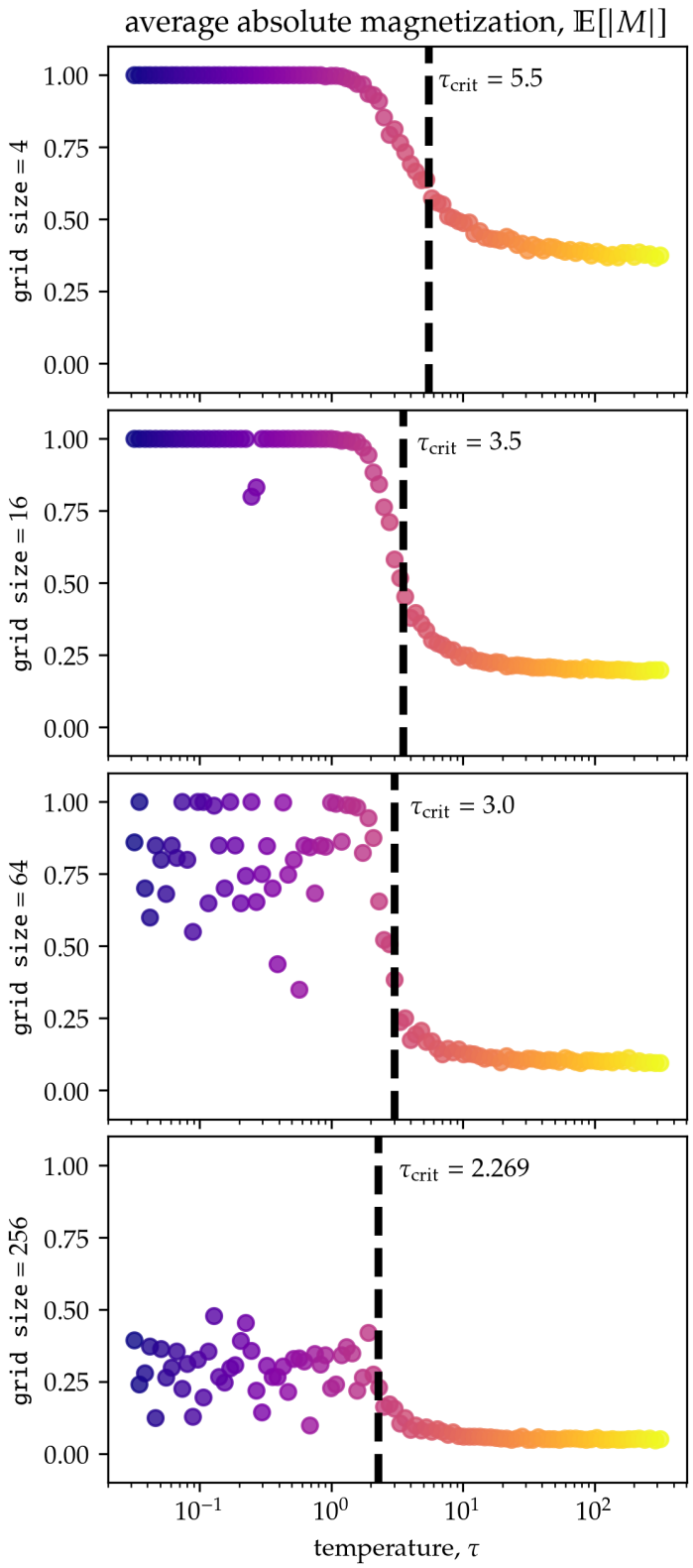


Figure 11: Average absolute magnetization, $\mathbb{E}[|M|]$, for grid sizes 4, 16, 64, and 256. The vertical black dashed lines are the estimated critical temperature τ_{crit}

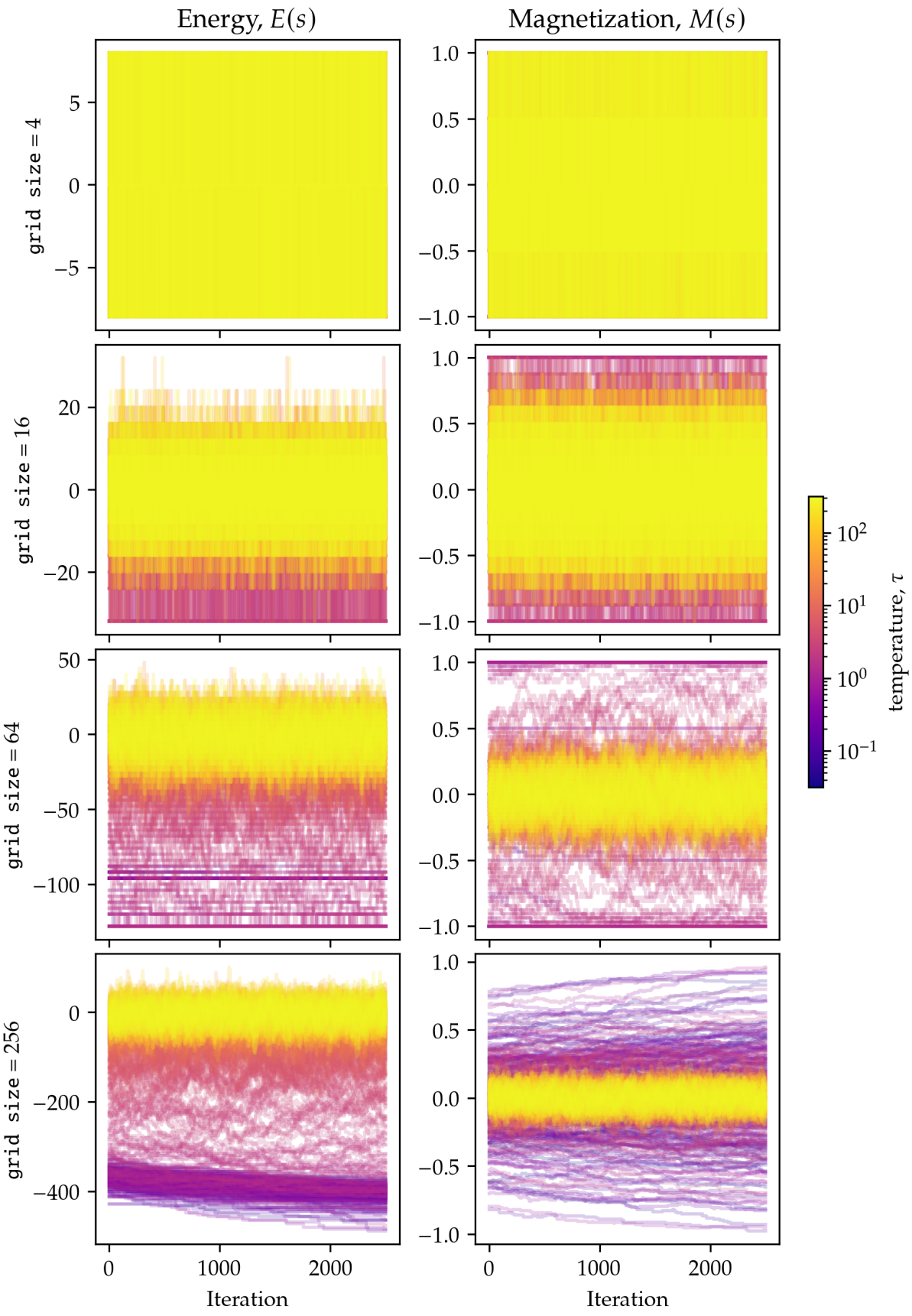


Figure 12: Energy and magnetization samples after burn in for grid sizes 4, 16, 64, and 256

(b) Simulated Tempering

1. Read BDA Chapter 12.3 on simulated tempering.

   Suppose our target density is our posterior $p(\theta \mid y)$. For simulated tempering, we define a set of $K+1$ distributions $p_k(\theta \mid y)$ for $k = 0, 1, 2, \ldots, K$ where $p_0(\theta \mid u) = p(\theta \mid y)$ and $p_1, \ldots, p_K$ are distributions with the same basic shape but are tempered for improved opportunities for mixing across modes. As in most cases, we don't have access to the normalized target, but we do have access to the unnormalized target $q_k(\theta)$.

   Simulated tempering works by setting a ladder of unnormalized densities $q_k(\theta)$

   $$q_k(\theta) = p(\theta \mid y)^{1/T_k} p_0(\theta)^{1-1/T_k} \quad \text{for} \quad k = 0, 1, 2, \ldots, K$$

   for a set of 'temperature' parameters $T_k > 0$. We set $T_0 = 1$ such that $q_0(\theta) = p(\theta \mid y)$. Large values of T_k smooth out the target density (e.g., produce less highly peaked mode), going in the limit to some 'base measure' p_0 that is some convenient prechosen distribution with high variance [BDA3]. The simulated tempering algorithm is as follows:

Algorithm 4: Simulated Tempering

Input:

- $q_k(\theta)$: set of K unnormalized target distributions
- k_t: an integer identifying the distribution used at iteration t
- p_r: proposal distribution for sample θ^* candidates
- J: jumping distribution for sampling k^* candidates

Initialize: $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**
  Draw a candidate sample from the proposal distribution: $\theta^* \sim p_r(\theta_t \to \theta^*)$
  Compute acceptance ratio and acceptance probability:

$$a(\theta^*) = \min(r, 1) \quad \text{where} \quad r = \frac{q_{k_t}(\theta^*)\, p_r(\theta^* \to \theta_t)}{q_{k_t}(\theta_t)\, p_r(\theta_t \to \theta^*)} \tag{58}$$

  if $U \le a(\theta^*)$ where $U \sim \text{Uniform}([0, 1])$ **then**
    Accept candidate θ^*, set $\theta_{t+1} = \theta^*$.
  else
    Reject candidate θ^*, set $\theta_{t+1} = \theta_t$.
  end
  if $k_t = 0$ **then**
    Append θ_{t+1} to $\mathcal{S}$
  end
  Draw a candidate sample from the jumping distribution: $k^* \sim J(k_t \to k^*)$
  Compute acceptance ratio and acceptance probability:

$$a(k^*) = \min(r, 1) \quad \text{where} \quad r = \frac{q_{k^*}(\theta_{t+1})\, J(k^* \to k_t)}{q_{k_t}(\theta_{t+1})\, J(k_t \to k^*)} \tag{59}$$

  if $U \le a(k^*)$ where $U \sim \text{Uniform}([0, 1])$ **then**
    Accept candidate k^*, set $k_{t+1} = k^*$.
  else
    Reject candidate k^*, set $k_{t+1} = k_t$.
  end
end
return $\mathcal{S}$

2. Show that, if we apply simulated tempering with p_0 uniform, then the artificial temperature parameter defined in simulated tempering can be absorbed into the choice of temperature in the Ising model. Argue that, simulated tempering for the Ising model can be viewed as a MCMC technique that generates coupled random walks for a series of parallel Ising processes each at a different temperature.

Suppose we wish to sample from the 2D Ising model where our target distribution is $p_*(s) = \mathbb{P}[S = s] \propto \exp\left(-\frac{1}{\tau}E(s)\right)$ with $E(s) = -\frac{J}{2}\sum_{\langle i,j\rangle} s_i s_j$. When using a uniform $p_0(s) \propto 1$, our unnormalized targets are given by

$$q_k(s) = p_*(s)^{1/T_k} p_0(s)^{1-1/T_k} \propto \exp\left(-\frac{1}{\tau}E(s)\right)^{1/T_k} = \exp\left(-E(s)\right)^{1/\tau T_k} = \exp\left(-E(s)\right)^{1/\beta_k} \quad \text{for} \quad k = 0, 1, 2, \ldots, K$$

where we define $\beta_k = \tau T_k$ such that the simulated tempering temperature parameter T_k is absorbed into Ising model temperature τ.

We can think of our simulated tempering approach with $\beta_k = \tau T_k$ as follows:

- We initialize K Ising processes with temperatures β_k
- We randomly select $\ell \in \{0, 1, \ldots, K\}$ to be the 'leader' of the parallel processes
- At each step...
  - (a.) We propose candidate from the leader's tempered target $s' \sim q_\ell(s)$
  - (b.) We accept or reject candidate s' via Metropolis-Hastings
  - (c.) We propose new leader from $\ell' \in \{0, 1, \ldots, K\}$ from temperature jumping distribution
  - (d.) We accept or reject new leader ℓ' via Metropolis-Hastings

We use 'leader' as it describes how the sampling will unfold. You can imagine the K Ising processes as a group of walkers *that move together*. They rotate who leads the group by proposing (and accepting or rejecting) a new leader at each step. And importantly each walker only save samples during their steps as the leader. In this way, simulated tempering for the Ising model can be viewed as a MCMC technique that generates coupled random walks for a series of parallel Ising processes each at a different temperature.

3. Adapt your MH algorithm (with independent spin flips) to incorporate simulated tempering. Choose p_0 to be the uniform distribution over spin arrangements. Use your exploratory analysis in part (a) to choose a temperature ladder.

We choose a symmetric jumping distribution such that our acceptance probability in Equation (59) becomes

$$a(k^*) = \min(r, 1) \quad \text{where} \quad r = \frac{q_{k^*}(s_{t+1})\, J(k^* \to k_t)}{q_{k_t}(s_{t+1})\, J(k_t \to k^*)} = \frac{p_*(s_{t+1})^{1/T_{k^*}}}{p_*(s_{t+1})^{1/T_{k_t}}} = \frac{\left(\frac{1}{Z_{k^*}}\right)\tilde{p}_*(s_{t+1})^{1/T_{k^*}}}{\left(\frac{1}{Z_{k_t}}\right)\tilde{p}_*(s_{t+1})^{1/T_{k_t}}} = \left(\frac{Z_{k_t}}{Z_{k^*}}\right)\tilde{p}_*(s_{t+1})^{\frac{1}{T_{k^*}} - \frac{1}{T_{k_t}}}$$

$$r = \left(\frac{Z_{k_t}}{Z_{k^*}}\right)\exp\left(-E(s_{t+1})\right)^{\frac{1}{\beta_{k^*}} - \frac{1}{\beta_{k_t}}} \quad \text{where} \quad Z_k := \int \exp\left(-E(s)\right)^{1/\beta_k}\, \mathrm{d}s$$

It is often more convenient computationally to compute the acceptance probability in log space:

$$a(k^*) = \exp\big(\min\big\{\log(r), 0\big\}\big) \quad \text{where} \quad \log(r) = \log\big(Z_{k_t}\big) - \log(Z_{k^*}) - \left(\frac{1}{\beta_{k^*}} - \frac{1}{\beta_{k_t}}\right)E(s_{t+1})$$

We select a ladder where $\beta_0 = \tau_{\text{hard}}$ (while normally β_0 is 1, β_0 is temperate the returns the target) corresponds to a difficult temperature (e.g., a cool temperature with a large mixing time) the remaining temperatures $\beta_1, \ldots, \beta_K$ include easier temperatures (e.g., hot temperatures with small mixing times). We space the ladder geometrically such that temperatures are clustered around β_0.

Tempering Ladder: We select a low temperature target distribution $\tau_{\text{hard}} = 0.1$ that is difficult to sample.

k	0	1	2
T_k	1	10	100
β_k	0.1	1	10

4. Is your algorithm the same as an importance sampling, rejection sampling, or importance resampling procedure that uses a high temperature Ising model as a proposal for a lower temperature target? Discuss the conceptual similarities and differences between this approach and the simulated tempering approach.

Consider simulated tempering with only two temperatures where β_0 corresponds to a low temperature distribution (the target) while β_1 corresponds to a high temperature distribution (proposal). Assuming our simulated tempering algorithm is sufficiently tuned, we will draw about half of samples from the target, cold distribution and half from the

proposal, hot distribution. We will then only keep the half corresponding to the target low temperature. Simulated tempering uses an MCMC algorithm to transverse the sample space. Therefore, the stationary distribution of our Markov chain will follow the target distribution (or at least they will in the limit). The tempering allows us to more readily discover modes and therefore may decrease mixing time.

Importance Sampling and Importance Resampling: Recall that importance sampling is a procedure in which we collect a set of m independent samples from the proposal $\{x_1, \ldots, x_m\}$. We then calculate the importance of each sample $\tilde{w}(x_j) = \frac{\tilde{p}_*(x_j)}{p_r(x_j)}$. We then select a subset of our samples via importance resampling. Specifically, we sequentially sampling x_j from $\{x_1, \ldots, x_m\}$ with probabilities proportional to the importance weights. This is done without replacement, so we rebalance the importance weights of the remaining samples at each step. Full summaries of importance sampling and importance resampling are provided.

Importance resampling with a high temperature proposal and a low temperature target is quite similar to simulated tempering. We first note a major difference: importance resampling only ever samples from the proposal, high temperature distribution; it never samples from the target, low temperature distribution directly. Consider the case where the proposal is extremely high temperature such that we essentially collect $\{x_1, \ldots, x_m\}$ uniformly at random. Then when we importance resample, we are effectively sampling from the target as $p_r \propto 1$ such that $\tilde{w}(x_j) \propto \tilde{p}_*(x_j)$. So in this sense, we essentially explore the space via the proposal (much like simulated tempering), and then keep important samples with high target, low temperature density.

Rejection Sampling: Recall that rejection sampling is the procedure in which we collect a set of n independent samples. At each step, we draw a candidate from the proposal $Y \sim p_r$ and accept or reject the candidate according to the acceptance probability $a(Y) = \frac{\tilde{p}_*(Y)}{Mp_r(Y)}$ where $M = \max_x \left\{\frac{\tilde{p}_*(x)}{p_r(x)}\right\}$ is a constant to ensure that $\frac{\tilde{p}_*(x)}{Mp_r(x)} \leq 1$ for all x. We repeat until we have accepted n candidates. A full summary of rejection sampling is provided.

Rejection sampling with a high temperature proposal and a low temperature target is quite similar to simulated tempering. Like importance sampling, rejection sampling only ever samples from the proposal, high temperature distribution and never directly from the target. However, we keep samples according to acceptance $\frac{\tilde{p}_*(Y)}{Mp_r(Y)}$. So while we only sample from the high temperature proposal, we are more likely accept candidates that have high-probability on the target, low temperature distribution.

5. Apply your simulated tempering algorithm to the Ising model example you identified as challenging in part (a). Is your simulated tempering method more or less computationally efficient (does it take more or less computational effort to achieve the same effective sample size)? Why or why not?

   We applied our simulated tempering algorithm to the Ising model example we identified as challenging in part (a) ($\tau_{\text{hard}} = 0.1$). For fairness, we ran the similar number of iterations for both the Metropolis-Hastings and simulated tempering algorithms. Specifically, we ran Metropolis-Hastings for 5000 with a burn-in of 2500 samples. We ran simulated tempering for 7500 iterations and used a fairly sparse temperature ladder with $K = 3$ temperatures such that the approximate number of samples from the target was 2500. With these settings, we ran each approach 100 times and calculated the effective sample size (ESS) for both algorithms. We summarize the quantiles of the ESS in Table 3 for both the energy and magnetization.

	Energy N_{eff}					Magnetization N_{eff}				
	2.5%	25%	50%	75%	97.5%	2.5%	25%	50%	75%	97.5%
Metropolis-Hastings	1.17	1.23	1.26	1.36	2.14	1.28	1.33	1.64	5.24	13.59
Simulated Tempering	4.88	23.90	31.23	40.28	54.09	1.89	7.01	11.18	14.41	24.54

TABLE 3: Effective sample size for energy and magnetization using Metropolis-Hastings and simulated tempering after 100 runs.

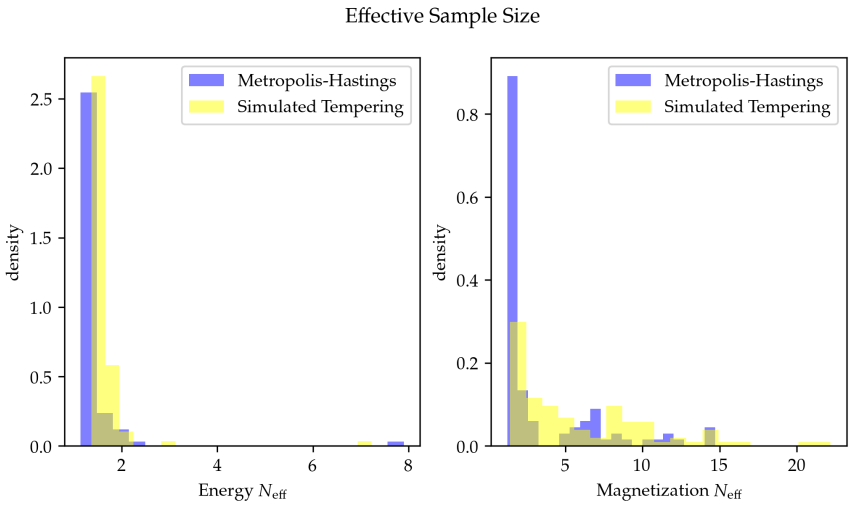


FIGURE 13: Effective sample size for energy and magnetization using Metropolis-Hastings and simulated tempering after 100 runs.

We also plot the samples from the Metropolis-Hastings and simulated tempering algorithms in Figure 14. We see that the Metropolis-Hastings samples often get stuck in either positive or negative magnetization states. This is expected as the Metropolis-Hastings algorithm is not able to easily traverse the sample space. In contrast, the simulated tempering samples are able to traverse the sample space more easily and therefore have a higher effective sample size.

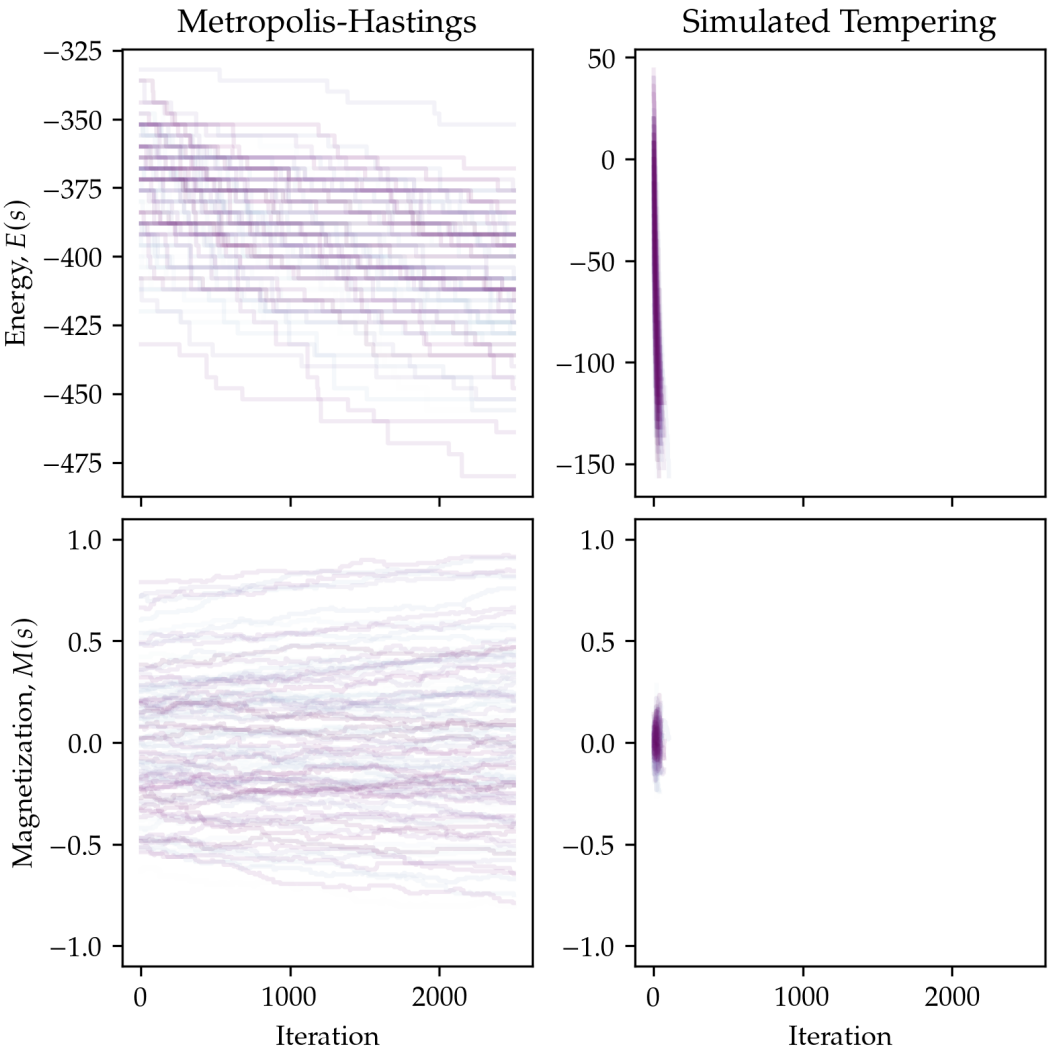


Figure 14: Energy and magnetization samples for Metropolis-Hastings and simulated tempering for 100 runs with $\tau_{\text{hard}} = 0.1$.

Importance Sampling and Importance Resampling

Let p_*, $\tilde{p}_*$, and p_r be the target, unnormalized target, and proposal distributions. Let $w(x) = \frac{p_*(x)}{p_r(x)}$ and $\tilde{w}(x) = \frac{\tilde{p}_*(x)}{p_r(x)}$ be the normalized and unnormalized importance weights such that

$$p_*(x) = \left(\frac{p_*(x)}{p_r(x)}\right) p_r(x) = w(x)\, p_r(x) \quad \text{and} \quad \tilde{p}_*(x) = \left(\frac{\tilde{p}_*(x)}{p_r(x)}\right) p_r(x) = \tilde{w}(x)\, p_r(x)$$
$$p_r(x) = \frac{p_*(x)}{w(x)} = \frac{\tilde{p}_*(x)}{\tilde{w}(x)} = \frac{Z p_*(x)}{\tilde{w}(x)} \quad \Longrightarrow \quad w(x) = \left(\frac{1}{Z}\right) \tilde{w}(x)$$

The goal of importance sampling is to estimate the expectation $\mathbb{E}_{x\sim p_*}[f(x)]$ using n iid samples from the proposal: We have

$$\mathop{\mathbb{E}}_{x\sim p_*}\big[f(x)\big] = \int_{\text{all } x} f(x)\, p_*(x)\, \mathrm{d}x = \int_{\text{all } x} f(x)\, w(x)\, p_r(x)\, \mathrm{d}x = \mathop{\mathbb{E}}_{x\sim p_r}\big[f(x)\, w(x)\big]$$

But we generally don't have access to the normalized target p_* so we therefore don't have access to the importance weights as $w(x) = \frac{p_*(x)}{p_r(x)}$. We will therefore estimate the importance weights as follows

$$\mathop{\mathbb{E}}_{x\sim p_r}[\tilde{w}(x)] = \int_{\text{all } x} \left(\frac{\tilde{p}_*(x)}{p_r(x)}\right) p_r(x)\, \mathrm{d}x = \int_{\text{all } x} \tilde{p}_*(x)\, \mathrm{d}x = Z \quad \Longrightarrow \quad w(x) = \frac{\tilde{w}(x)}{Z} = \frac{\tilde{w}(x)}{\mathbb{E}_{x\sim p_r}[\tilde{w}(x)]} \approx \frac{\tilde{w}(x)}{\frac{1}{n}\sum_{j=1}^n \tilde{w}(x_j)}$$

where $x_j \overset{\text{i.i.d.}}{\sim} p_r$ for $j = 1, \ldots, n$. With this approximation of $w(x)$, we can approximate $\mathbb{E}_{x\sim p_*}\big[f(x)\big]$. We have

$$\mathop{\mathbb{E}}_{x\sim p_*}\big[f(x)\big] = \mathop{\mathbb{E}}_{x\sim p_r}\big[f(x)\, w(x)\big] \approx \mathop{\mathbb{E}}_{x\sim p_r}\left[f(x)\left(\frac{\tilde{w}(x)}{\frac{1}{n}\sum_{j=1}^n \tilde{w}(x_j)}\right)\right] \approx \frac{\frac{1}{n}\sum_{j=1}^n f(x_j)\, \tilde{w}(x_j)}{\frac{1}{n}\sum_{j=1}^n \tilde{w}(x_j)} \quad \text{where } x_j \overset{\text{i.i.d.}}{\sim} p_r$$

Algorithm 5: Importance Resampling

Input: $\tilde{p}_*$ unnormalized target, p_r proposal distribution, and integers $m > n$
Initialize: $\mathcal{S} \leftarrow \emptyset$

Collect m independent samples from proposal $\mathcal{X} = \{x_1, \ldots, x_m\}$ where $x_j \overset{\text{i.i.d.}}{\sim} p_r$ and $|\mathcal{X}| = m$
for $i = 1, 2, \ldots, n$ **do**
  Compute and standardize unnormalized importance weights

$$\tilde{w}(x_j) = \frac{\tilde{p}_*(x_j)}{p_r(x_j)} \quad \text{for all } j = 1, \ldots, |\mathcal{X}| \quad \text{and} \quad \hat{w}(x_j) = \frac{\tilde{w}(x_j)}{\sum_{k=1}^{|\mathcal{X}|} \tilde{w}(x_k)} \quad \text{such that} \quad \sum_{j=1}^{|\mathcal{X}|} \hat{w}(x_j) = 1$$

  Sample x' from $\mathcal{X}$ where the probability of sampling $x_j \in \mathcal{X}$ is proportional to the weight $\hat{w}(x_j)$
  Append x' to $\mathcal{S}$ and remove x' from $\mathcal{X}$
end
return $\mathcal{S}$

Rejection Sampling

Let p_*, $\tilde{p}_*$, and p_r be the target, unnormalized target, and proposal distribution such that

$$\tilde{p}_*(x) := \frac{\tilde{p}_*(x)}{Z} \quad \text{where} \quad Z := \int_{\text{all } x} \tilde{p}_*(x)\, \mathrm{d}x \quad \text{and} \quad M p_r(x) \geq \tilde{p}_*(x) \quad \text{for all } x.$$

The goal of rejection sampling is to collect a set of n independent samples $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ from the target:

Algorithm 6: Rejection Sampling

Input: $\tilde{p}_*$ unnormalized target, p_r proposal distribution, and $M^\star = \max_x \{\tilde{p}_*(x)/p_r(x)\}$
Initialize: $\mathcal{S} \leftarrow \emptyset$

while $|\mathcal{S}| < n$ **do**
  Draw candidate sample from the proposal distribution: $Y \sim p_r$
  Compute acceptance probability: $a(Y) = \frac{\tilde{p}_*(Y)}{M^\star p_r(Y)} \in [0, 1]$
  if $U \leq a(Y)$ where $U \sim \text{Uniform}([0, 1])$ **then**
    Append Y to $\mathcal{S}$
  else
    Reject Y
  end
end
return $\mathcal{S}$

It follows that the probability that we accept a candidate sample is

$$\mathbb{P}[Y \text{ is accepted}] = \mathbb{P}[U \leq a(Y)] = \int_{\text{all } x} \mathbb{P}[U \leq a(Y) \mid Y = x]\ \mathbb{P}[Y = x]\, \mathrm{d}x = \int_{\text{all } x} \mathbb{P}[U \leq a(x)]\ \mathbb{P}[Y = x]\, \mathrm{d}x$$

Recall that $\mathbb{P}[U \leq c] = c$ for any $c \in [0, 1]$ implying that $\mathbb{P}[U \leq a(x)] = a(x)$. Therefore, we have

$$\mathbb{P}[Y \text{ is accepted}] = \int_{\text{all } x} \frac{\tilde{p}_*(x)}{M p_r(x)}\, p_r(x)\, \mathrm{d}x = \frac{1}{M} \int_{\text{all } x} \tilde{p}_*(x)\, \mathrm{d}x = \frac{Z}{M}.$$

To complete, we solve for $M^\star$, the constant that maximizes our acceptance probability. We have

$$M^\star = \underset{M}{\text{argmax}} \left\{ \mathbb{P}[Y \text{ is accepted}] \right\} = \underset{M}{\text{argmax}} \left\{ \frac{Z}{M} \right\} = \underset{M}{\text{argmax}} \left\{ \frac{1}{M} \right\} = \underset{M}{\text{argmin}} \{M\}$$

But M has to satisfy $M \geq \frac{\tilde{p}_*(x)}{p_r(x)}$ so we have that $\min_M \{M\} = \max_x \left\{ \frac{\tilde{p}_*(x)}{p_r(x)} \right\}$ implying that $\boxed{M^\star = \max_x \left\{ \frac{\tilde{p}_*(x)}{p_r(x)} \right\}}$

(c) Ensemble Methods

An ensemble method (or particle filtering method) is an MCMC method that evolves many chains in parallel rather than a single chain. The chains interact, with interaction rules chosen to speed convergence. At any time, an ensemble method consists of a population of particles, each assigned a state that updates according to an MCMC procedure.

For example, we could track the importance weights assigned to each particle had we used the collection of particles as a proposal. Particles with large weights have discovered regions where the target is under-represented by the ensemble, while particles with small weights occupy over-represented regions. This information can be used to move particles from over-represented regions to under-represented regions. In this sense, particles that discover an underexplored region of the target can rapidly recruit other particles to join their neighborhood. In general, these methods do not help the MCMC method discover new modes, but can drastically reduce the mixing time needed to balance mass between, and to fully explore, separate modes.

1. Read Lindsey, Weare and Zhang. *Ensemble Markov Chain Monte Carlo with Teleporting Walkers*. (2022). Briefly summarize the method proposed here, how it shares information between particles/walkers to speed convergence, and some example target distributions where it might be an effective tool.[23]

   Lindsey, Weare and Zhang propose an MCMC method where an ensemble of coupled walkers sample from a target distribution [LWZ22]. A summary of the algorithm is below:

   Suppose we are given a target (possible unnormalized) distribution $\pi(x)$ on a space $\mathcal{X}$. We wish to sample from the joint target of N walkers, i.e., $\mathbf{x} = (x_1, \dots, x_N) \sim \Pi(\mathbf{x}) = \prod_{i=1}^{N} \pi(x_i)$. While $(x_1, \dots, x_N)$ are independent on the joint density, the walkers are coupled such that walkers are able to 'teleport' to another walker's location (and thus will not produce N independent chains). Let $p_r(\cdot \to \cdot)$ represent the proposal distribution from we will sample candidates $\mathbf{x}'$. At each step our algorithm, we will accept or reject the candidate according to a Metropolis-Hastings acceptance probability. A full summary of the algorithm is below:

Algorithm 7: Ensemble Markov Chain Monte Carlo with Teleporting Walkers [LWZ22]

Input: $\pi(x)$ target distribution (possibly unnormalized), p_r proposal distribution, fixed N random walkers
Initialize: N random walkers with initial positions $\mathbf{x} = (x_1, \dots, x_N) \in \mathcal{X}^N$

for $t = 1, 2, \dots, T$ **do**
- Sample teleporter's arrival index $j \in \{1, 2, \dots, N\}$ uniformly at random and sample $z \sim p_r(x_j \to z)$
- Sample teleporter's starting index $i \in \{1, 2, \dots, N\}$ (possibly equal to j) according to importance weights

$$w_i(\mathbf{x}, z) := \frac{p_r(z \to x_i) + \sum_{k \neq i}^{N} p_r(x_k \to x_i)}{\pi(x_i)} \Bigg/ Z(\mathbf{x}, z) \quad \text{where } Z(\mathbf{x}, z) := \sum_{l=1}^{N} \frac{p_r(z \to x_l) + \sum_{k \neq l}^{N} p_r(x_k \to x_l)}{\pi(x_l)} \tag{60}$$

- Construct candidate $\mathbf{x}'$: set $\mathbf{x}' = (x'_1, \dots, x'_N) \leftarrow \mathbf{x} = (x_1, \dots, x_N)$ and teleport walker by overwriting $x'_i \leftarrow z$[†]
- Compute acceptance probability and acceptance ratio:

$$a(\mathbf{x}') = \min(r, 1) \quad \text{where} \quad r = \frac{\Pi(\mathbf{x}')\, p_r(\mathbf{x}' \to \mathbf{x})}{\Pi(\mathbf{x})\, p_r(\mathbf{x} \to \mathbf{x}')} = \frac{Z(\mathbf{x}, z)}{Z(\mathbf{x}', x_i)} \tag{61}$$

- **if** $U \leq a(\mathbf{x}')$ where $U \sim \text{Uniform}([0, 1])$ **then**
  - Accept candidate $\mathbf{x}'$, set $\mathbf{x}_{t+1} = \mathbf{x}'$
- **else**
  - Reject candidate $\mathbf{x}'$, set $\mathbf{x}_{t+1} = \mathbf{x}$
- **end**

end

[†]Conceptually, the candidate is the same as the current sample but the walker at x_i is teleported to x_j and takes one step $z \sim p_r(x_j \to z)$.

To appreciate how the algorithm works, let us examine and derive the acceptance probability in Equation (61). The

[23]For further reading on related methods see Lu, Lu, and Nolen. *Accelerating Langevin Sampling with Birth-death*. (2019).

probability of proposing $\mathbf{x}'$ given current sample $\mathbf{x}$ is

$$p_r(\mathbf{x} \to \mathbf{x}') = \mathbb{P}[\text{sampled index } i]\ \mathbb{P}[\text{sampled } z] = w_i(\mathbf{x}, z) \sum_{j=1}^{N} \mathbb{P}[z \mid x_j]\ \mathbb{P}[\text{sampled index } j] = w_i(\mathbf{x}, z)\, \frac{1}{N} \sum_{j=1}^{N} p_r(x_j \to z)$$

Noting that $\mathbf{x} = \mathbf{x}'$ for all indices except i where we have $x_i' = z$, our acceptance ratio r is given by

$$r = \frac{\Pi(\mathbf{x}')\, p_r(\mathbf{x}' \to \mathbf{x})}{\Pi(\mathbf{x})\, p_r(\mathbf{x} \to \mathbf{x}')} = \frac{\pi(x_i')}{\pi(x_i)} \frac{\prod_{k \neq i} \pi(x_k')}{\prod_{k \neq i} \pi(x_k)} \frac{w_i(\mathbf{x}', x_i)\, \frac{1}{N} \sum_{j=1}^{N} p_r(x_j' \to x_i)}{w_i(\mathbf{x}, z)\, \frac{1}{N} \sum_{j=1}^{N} p_r(x_j \to z)}$$

$$r = \frac{\pi(x_i')}{\pi(x_i)} \frac{Z(\mathbf{x}, z)}{Z(\mathbf{x}', x_i)} \frac{\left(\frac{p_r(x_i \to x_i') + \sum_{k \neq i}^{N} p_r(x_k' \to x_i')}{\pi(x_i')} \right)}{\left(\frac{p_r(z \to x_i) + \sum_{k \neq i}^{N} p_r(x_k \to x_i)}{\pi(x_i)} \right)} \frac{\sum_{j=1}^{N} p_r(x_j' \to x_i)}{\sum_{j=1}^{N} p_r(x_j \to z)} = \frac{Z(\mathbf{x}, z)}{Z(\mathbf{x}', x_i)} \frac{p_r(x_i \to z) + \sum_{k \neq i}^{N} p_r(x_k' \to z)}{p_r(z \to x_i) + \sum_{k \neq i}^{N} p_r(x_k \to x_i)} \frac{\sum_{j=1}^{N} p_r(x_j' \to x_i)}{\sum_{j=1}^{N} p_r(x_j \to z)}$$

$$r = \frac{Z(\mathbf{x}, z)}{Z(\mathbf{x}', x_i)} \frac{p_r(x_i \to z) + \sum_{k \neq i}^{N} p_r(x_k \to z)}{p_r(x_i' \to x_i) + \sum_{k \neq i}^{N} p_r(x_k' \to x_i)} \frac{\sum_{j=1}^{N} p_r(x_j' \to x_i)}{\sum_{j=1}^{N} p_r(x_j \to z)} = \frac{Z(\mathbf{x}, z)}{Z(\mathbf{x}', x_i)} \frac{\sum_{k=1}^{N} p_r(x_k \to z)}{\sum_{k=1}^{N} p_r(x_k' \to x_i)} \frac{\sum_{j=1}^{N} p_r(x_j' \to x_i)}{\sum_{j=1}^{N} p_r(x_j \to z)}$$

$$r = \frac{Z(\mathbf{x}, z)}{Z(\mathbf{x}', x_i)} \quad \checkmark$$

Ensemble MCMC can be well suited for multimodal target distributions as the walkers are able to explore the sampling space and jump between modes more easily. Consider the bimodal distribution in Figure 15. Suppose at time t, we randomly select walker x_j and pull candidate $z \sim p_r(x_j \sim z)$ (see pink dot in Figure 15). We next calculate the importance weights. For example, Figure 15 depicts the calculation of $w_i(\mathbf{x}, z) \propto p_r(z \to x_i) + \sum_{k \neq i}^{N} p_r(x_k \to x_i)$ where we sum the connections to x_i highlighted in blue with thicker lines representing higher transition probabilities. The importance weight of x_i be the largest as it has a strong connection to z, strong connections to the other particles, and $\pi(x_i)$ is small. Therefore, we are most likely to select x_i as the teleporting walker. Suppose that we in fact sample x_i and the candidate $\mathbf{x}'$ is accepted. Then x_i would teleport to z and we would keep the bottom row of particles in Figure 15. We note that by teleporting to z, the walker x_i is now closer to mode x_A than to mode x_B. This highlights how teleportation is able to overcome a deep valley in density and make larger transitions in the sample space.

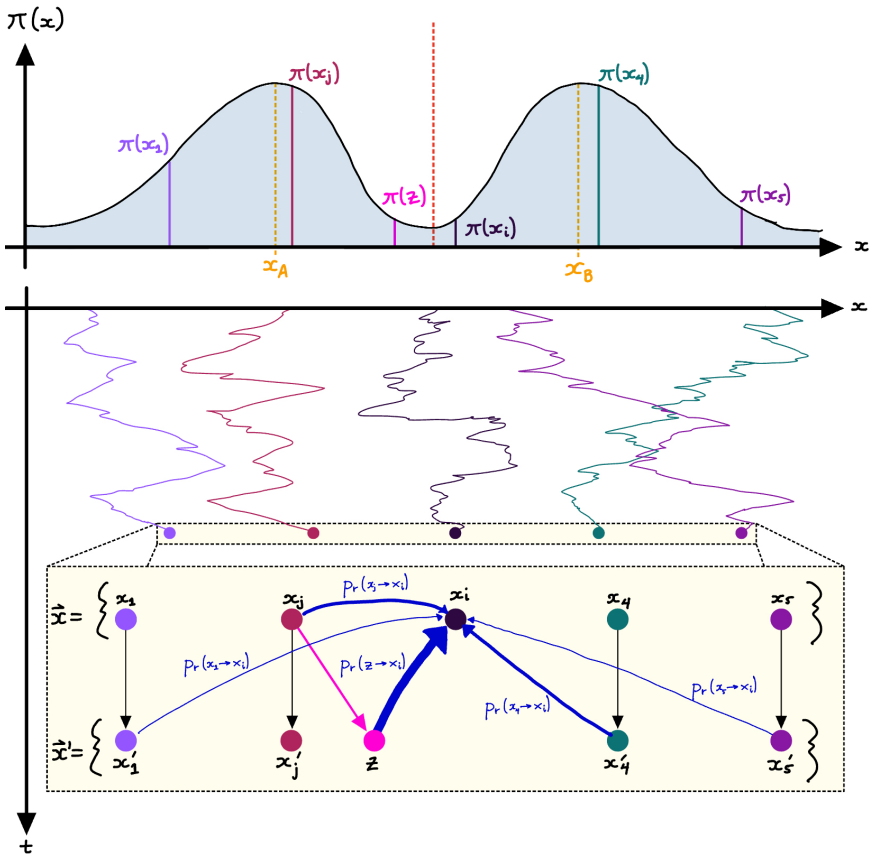


FIGURE 15: Ensemble MCMC with Teleporting Walkers

2. Propose, in pseudocode, an ensemble particle method for sampling from the Ising model. Are there parameter settings for the Ising model where you think an ensemble method might be useful?[24]

[24]Think about the symmetry of the model and the number of modes in the target distribution. Are there situations where the target is multimodal and standard MH fails to discover, or balance between, all the modes? Do you need an ensemble method to improve MH in this setting or are there simpler ways to exploit symmetry?

Algorithm 8: Ising Model + Ensemble MCMC with Teleporting Walkers [LWZ22]

Input: $p(\mathbf{s}) \propto \exp\left(-\frac{1}{\tau}E(\mathbf{s})\right)$ target distribution (unnormalized), p_r proposal distribution

Initialize: M random particles $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_M\}$ where each $\mathbf{s}_k \in \{+1, -1\}^{N \times N}$ is an N-by-N Ising model

for $t = 1, 2, \ldots, T$ **do**

Choose a particle $\mathbf{s}_j$ uniformly at random from $\mathbf{S}$

Propose a new configuration $\mathbf{z} \sim p_r(\mathbf{s}_j \to \mathbf{z})$[†]

Teleport particle $\mathbf{s}_i$ to $\mathbf{z}$, where $\mathbf{s}_i$ is selected from $\mathbf{S}$ according to weights[‡]

$$w_i(\mathbf{S}, \mathbf{z}) := \frac{\tilde{w}_i(\mathbf{S}, \mathbf{z})}{Z(\mathbf{S}, \mathbf{z})} \quad \text{where} \quad \tilde{w}_i(\mathbf{S}, \mathbf{z}) := \frac{p_r(\mathbf{z} \to \mathbf{s}_i) + \sum_{k \neq i}^{N} p_r(\mathbf{s}_k \to \mathbf{s}_i)}{p(\mathbf{s}_i)} \quad \text{and} \quad Z(\mathbf{S}, \mathbf{z}) := \sum_{\ell=1}^{N} \tilde{w}_\ell(\mathbf{S}, \mathbf{z})$$

Construct candidate by setting $\mathbf{s}_i = \mathbf{z}$ while keeping all other particles unchanged, $\mathbf{S}' = \{\mathbf{s}_1, \ldots, \mathbf{z}, \ldots, \mathbf{s}_M\}$

Compute acceptance probability and acceptance ratio:

$$a(\mathbf{S}') = \min(r, 1) \quad \text{where} \quad r = \frac{\Pi(\mathbf{S}')\, p_r(\mathbf{S}' \to \mathbf{S})}{\Pi(\mathbf{S})\, p_r(\mathbf{S} \to \mathbf{S}')} = \frac{Z(\mathbf{S}, \mathbf{z})}{Z(\mathbf{S}', \mathbf{s}_i)}$$

if $U \leq a(\mathbf{S}')$ where $U \sim \text{Uniform}([0, 1])$ **then**

Accept candidate $\mathbf{S}'$, set $\mathbf{S}_{t+1} = \mathbf{S}'$

else

Reject candidate $\mathbf{S}'$, set $\mathbf{S}_{t+1} = \mathbf{S}$

end

end

[†] we use a symmetric proposal where we independently flip all of the dipoles with probability p_{flip}

[‡] by construction, it is possible that $\mathbf{s}_i = \mathbf{s}_j$ in which case the particle does not teleport, rather it takes a standard Metropolis-Hastings step

Low temperature Ising models is where an ensemble method will be useful as the target is bimodal in magnetization. Without loss of generality, consider the case where our Ising model is initialized with more positive than negative spins such that $M(\mathbf{s}) > 0$. Recall the Metropolis acceptance probability

$$a(s') = \min(r, 1) \quad \text{where} \quad r = \exp\left(-\frac{1}{\tau}\left[E(s') - E(s)\right]\right) = \exp\left(-\frac{1}{\tau}\left[\Delta E(s, s')\right]\right)$$

We note that we will always accept if $\Delta E(s, s') \leq 0$ and we accept with probability $\exp\left(-\frac{1}{\tau}\left[\Delta E(s, s')\right]\right)$ when $\Delta E(s, s') > 0$. In other words, we will always accepted if the proposed energy is less than the current energy. However, especially in the lower temperature regime (see Figure 9), we will infrequently accept candidates with greater energy than the current. This implies that our system is biased towards accepting more and more positive spins driving the system towards almost all positive spins. It would require many steps with large changes in energy to flip the system to negative magnetization. This is where ensemble particle MCMC methods shine. We can initialize N walkers and expect that half fall on the positive magnetization and half on the negative magnetization side ensuring with see both modes. Additionally, teleporting walkers would make jumping between modes easier.

Another simpler strategy for exploring both modes would be flipping all of the spins occasionally as flipping them all results in the same energy $E(-\mathbf{s}) = E(\mathbf{s})$ and flips the magnetization $M(-\mathbf{s}) = -M(\mathbf{s})$.

Stat 238, Fall 2025
Project Minis

Alexander Strang
Due: by **5:00 pm** Friday, April 4th, 2025

Mini Project 9: Mixing Times in Sampling

Policies

- This is an individual project.
- All submissions must be properly type-set and sourced. This means providing formal citations and a complete bibliography when sources are used.
- Project minis should be uploaded to Gradescope under the appropriate mini-assignment.
- Please export any notebooks as a single pdf and merge all components into one file.

Prompts

This is a reading project.

Bayesian inference is typically performed via approximate posterior sampling. Accordingly, the boundaries of modern Bayesian inference are, given accepted models, largely computational. In particular, if a Bayesian model admits an efficient sampling scheme, then the model is, for all practical purposes, tractable, and may be considered a solved case. Otherwise, it merits further computational work. Since most sampling is performed iteratively via Markov Chain Monte Carlo (MCMC), the existence of an efficient sampling scheme can usually be reduced to the existence of an MCMC scheme that mixes sufficiently quickly.

In this project, you will be asked to read and summarize a subset of papers discussing a popular Bayesian inference problem that does not admit efficient sampling schemes. I've suggested a topic and a selection of papers below, but, you are welcome to substitute this topic with one of your own interest if it fits the subsequent discussion questions.

Example Topic: *Phylogenetic or Phylodynamic Inference*

Phylogenetic and phylodynamic inference concern the inference of phylogenies and effective population sizes from observed genetic data on a sample population. Examples include tracking the evolution of infectious diseases, reconstructing genealogies or ancient migration patterns, or reconstructing evolutionary trees. At its core, phylogenetic and phylodynamic inference problems aim to reconstruct a tree from sampled genetic data.

Bayesian phylogenetic inference is performed by sampling from the space of binary trees. These trees may or may not include a ranking associated with an ordering of the branching events in time, and labels associated with nodes and leaves. The space of such trees is

combinatorially large, and posterior distributions over the space of trees can be highly "multimodal" in the sense that, if we introduce a notion of similarity between trees and form a graph over trees by linking similar trees, then clusters of trees with high posterior probability may be disconnected, connected only by paths that pass through intermediate trees of low posterior probability. As a result, sampling from a posterior distribution over trees can be impractically slow.

Please read:

1. Mossel and Vigoda. *Phylogenetic MCMC Algorithms Are Misleading on Mixtures of Trees.* Science (2005).
2. Simper and Palacios. *An Adjacent-Swap Markov Chain on Coalescent Trees.* Journal of Applied Probability (2022).
3. Either:
   - Samyak and Palacios. *Statistical Summaries of Unlabelled Evolutionary Trees.* Biometrika (2024). or,
   - Lan et al. *An efficient Bayesian inference framework for coalescent-based nonparametric phylodynamics.* Bioinformatics (2015).
4. *Optional.* Either:
   - Diaconis and Holmes. *Random Walks on Trees and Matchings.* (2002), or
   - Aldous, David. *Mixing Times for a Markov Chain on Cladograms.* Combinatorics (2000).

Then write a 4 - 5 page essay that summarizes and comments on your reading. It should, at minimum, address the discussion prompts outlined below. If you choose to pursue a different example, make sure that your example allows interesting answers to each prompt. In particular, your example must be an inference problem for which sampling is challenging.

(a) (5 points) Summarize the inference problem of interest in your reading. Highlight:
   - The space of unknowns.
   - The relationship between the unknowns and the observables.
   - The standard priors used in this inference problem.

(b) (5 points) Explain, in detail, why this inference problem is difficult (e.g. why sampling from the associated posterior efficiently is hard). If available, give bounds on the mixing time for the chain as a function of relevant problem paramaters.

(c) (5 points) Summarize the sampling schemes proposed in the papers you read, contrasting any differences in approach. Carefully explain how the authors attempt to address the challenges you summarized in part (b). Highlight, in particular, the transition steps allowed in any MCMC method, and why the chosen transition steps were proposed by the authors. For example:
   - If the transition step is based on making a "small perturbation" in some metric over the space of unknowns, discuss why the chosen metric should lead to relevant proposals.
   - If the transition step is based on gradient information, discuss why the gradient information should produce relevant proposals.

STAT238: Bayesian Statistics May 13th, 2025

Mini Project 10: Variational Inference

Reece D. Huff

Prompts

Bayesian inference is typically performed via approximate posterior sampling. However, there are many situations where sampling is inefficient, especially if we want to perform high-dimensional inference. Variational inference substitutes sampling with optimization. Instead of drawing samples from the posterior (or an approximation to the posterior produced by running an MCMC scheme), variational inference searches for the closest approximation to the target posterior among all distributions in an (often infinite-dimensional) family of tractable distributions. Please read:

1. Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "*Variational inference: A review for statisticians*." Journal of the American statistical Association 112, no. 518 (2017): 859-877. [BKM17]
2. Liu, Qiang, and Dilin Wang. "*Stein variational gradient descent: A general purpose Bayesian inference algorithm*." Advances in neural information processing systems 29 (2016). [LW16]

Then write a 4-5 page essay that summarizes and comments on your reading. It should, at minimum, address the discussion prompts outlined below. For the second paper it may help to read:

- Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. "*Normalizing flows for probabilistic modeling and inference*." Journal of Machine Learning Research 22, no. 57 (2021): 1-64. [PNR+21]
- Wang, Yiwei, Jiuhai Chen, Chun Liu, and Lulu Kang. "*Particle-based energetic variational inference*." Statistics and Computing 31 (2021): 1-17. [WCLK21]

(a) Compare variational inference methods and MCMC procedures. Identify pros and cons for each framework, and propose at least one example setting where you prefer each over the other.

(b) Variational inference typically adopts the KL divergence between the variational distribution, q, and the target as its objective, p_*. Explain why we adopt $D_{\mathrm{KL}}(q||p_*)$ rather than $D_{\mathrm{KL}}(p_*||q)$. Explain how your interpretation of the objective would (or would not) change, if you exchanged the order of the distributions inside the KL divergence. What biases do you expect this to induce in the variational solution?

(c) Clearly summarize the CAVI algorithm for mean-field inference in conditionally conjugate models. In particular, show that each step in the iterative procedure is a descent step on the KL divergence. Try to relate the procedure to another method from the course (e.g. coordinate ascent, expectation-maximization).

(d) Discuss the biases induced by the mean-field assumption, and, given these biases, how you would use the variational distribution produced by optimizing over the mean field class.

(e) Clearly summarize the SVGD algorithm for mean-field inference. Then, answer the following questions:

- What is the variational family that SVGD optimizes over?[25]
- Give an intuitive description of the particle dynamics specified by the SVGD algorithm. Compare this algorithm to another ensemble algorithm from the class. What component of the algorithm ensures that the particles spread out? What is responsible for the variance in samples in related MCMC methods?
- Are you convinced by the "median-trick" for selecting the kernel bandwidth?[26] Propose an alternative procedure for adaptively selecting the bandwidth.

(f) SVGD, like other particle flow based methods, uses a flow map to transport particles drawn from a reference distribution (often normal). The transport map is chosen so that, after the transformation, the original distribution is as close as possible to the target. This is implemented by at each step moving a set of reference particles as if they obeyed a time inhomogeneous ODE. The vector field specifying the ODE is chosen to instantaneously decrease the KL as quickly as possible. Show that the functional gradient of the KL divergence with respect to a velocity field transporting $q(t)$ to $q(t + dt)$ can be calculated using only computationally available quantities (the unnormalized target, log densities, partial derivatives of log densities).

[25] This is a tricky question. It may help to start by asking, how would I sample from the distribution produced at the end of SVGD? The variational family will be all distributions that can be specified by this sampling procedure.

[26] Be skeptical here.

Notation

Let $\mathbf{z} = [z_1, \ldots, z_m]$ represent the unknown or "latent" variables of interest. Let $\mathbf{x} = [x_1, \ldots, x_n]$ be the known, observable variables. Suppose we are given a joint model $p(\mathbf{x}, \mathbf{z})$ such that $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid \mathbf{z})\ p(\mathbf{z})$ where $p(\mathbf{x} \mid \mathbf{z})$ represents the likelihood, and $p(\mathbf{z})$ represents the prior distribution. Let $p(\mathbf{z} \mid \mathbf{x}) = \tilde{p}(\mathbf{z} \mid \mathbf{x})/Z(\mathbf{x})$ represent the posterior where $\tilde{p}(\mathbf{z} \mid \mathbf{x}) = p(\mathbf{x} \mid \mathbf{z})\ p(\mathbf{z})$ and $Z(\mathbf{x}) = \int_{\text{all } \mathbf{z}} \tilde{p}(\mathbf{z} \mid \mathbf{x})\ \mathrm{d}\mathbf{z}$ is the troublesome normalizing constant.

(a) Variational Inference vs. Markov Chain Monte Carlo

A fundamental challenge in Bayesian statistics is approximating, computing, and sampling from challenging distributions. We begin by discussing two methods for approximating a target distribution, $p_*(\mathbf{z})$.

Markov Chain Monte Carlo. In Markov Chain Monte Carlo (MCMC), we approximate the target density p_* by defining a Markov chain whose stationary distribution is the target. Specifically, we simulate an *ergodic*[27] chain that obeys *detailed balance*[28] for long enough to approach stationarity. We then approximate the target with an empirical estimate constructed from (a subset of) the collected samples.

Variational Inference. In variational inference, we approximate the target density p_* by selecting the optimal distribution q^* from a *variational family* $\mathcal{Q}$. Specifically, we define a set of densities $q(\mathbf{z}) \in \mathcal{Q}$ meant to approximate the target distribution. The optimal density q^* is a member of $\mathcal{Q}$ that minimizes the Kullback-Leibler (KL) divergence to the target,

$$q^*(\mathbf{z}) := \underset{q(\mathbf{z}) \in \mathcal{Q}}{\operatorname{argmin}} \left\{ \mathtt{KL}\Big(q(\mathbf{z}) \,\Big\|\, p_*(\mathbf{z})\Big) \right\} \quad \text{where} \quad \mathtt{KL}\Big(q(\mathbf{z}) \,\Big\|\, p_*(\mathbf{z})\Big) := \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\cdot)} \left[\log\left(\frac{q(\mathbf{z})}{p_*(\mathbf{z})} \right) \right] = \int_{\text{all } \mathbf{z}} q(\mathbf{z})\ \log\left(\frac{q(\mathbf{z})}{p_*(\mathbf{z})} \right) \mathrm{d}\mathbf{z} \tag{VI}$$

In contrast with MCMC, we cast our approximation of the target distribution as an optimization problem. Additionally, we have a proper density q^* as an approximation of the target, rather than empirical estimates constructed from samples.

Comparing variational inference and MCMC. Variational inference and MCMC are both powerful techniques for density approximation. When should a statistician use one or the other? MCMC methods are generally more computationally expensive as it can take an extremely long time to approach stationarity. With this expense, MCMC is able to provide guarantees of producing *exact* samples from the target density (asymptotically). While variational inference does not enjoy such guarantees, it is generally much faster than MCMC (assuming a reasonable variational family). Therefore, variational inference is generally better suited for large datasets where speed is a higher priority than accuracy. We may prefer MCMC, for example, in a medical setting. Here, accuracy (e.g., in the tails) is of utmost importance, and therefore we are willing to collect many samples over years and years to be confident that our model is capable of making inferences with high accuracy. On the other hand, we may prefer variational inference if we, for example, are designing a recommendation algorithm at Netflix. Here, we have access to an enormous dataset and desire a model that is fast and does well in expectation and therefore are willing to sacrifice a bit of accuracy for speed.

(b) Why use $\mathtt{KL}(q \,||\, p_*)$ instead of $\mathtt{KL}(p_* \,||\, q)$ for variational inference?

A primary reason for utilizing $\mathtt{KL}(q \,||\, p_*)$ instead of $\mathtt{KL}(p_* \,||\, q)$ is that we often only have access to the unnormalized target density $\tilde{p}_*$. In this case, the variational inference objective $\mathtt{KL}(q \,||\, p_*)$ is equivalent to minimizing $\mathtt{KL}(q \,||\, \tilde{p}_*)$ as

$$\mathtt{KL}\Big(q(\mathbf{z}) \,\Big\|\, p_*(\mathbf{z})\Big) = \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\cdot)} \left[\log\left(\frac{q(\mathbf{z})}{p_*(\mathbf{z})} \right) \right] = \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\cdot)} \left[\log q(\mathbf{z}) - \log \tilde{p}_*(\mathbf{z}) + \log(Z) \right] = \mathtt{KL}\Big(q(\mathbf{z}) \,\Big\|\, \tilde{p}_*(\mathbf{z})\Big) + \text{constant}$$

On the other hand, the alternative objective requires an expectation over the target as $\mathtt{KL}(p_* \,||\, q) = \mathbb{E}_{\mathbf{z} \sim p_*(\cdot)}[\log(p_*(\mathbf{z})/q(\mathbf{z}))]$. To evaluate this expectation, we would have to calculate the troublesome normalizing constant which is often intractable. We could approximate the expectation with sampling (i.e., importance sampling) but this makes our optimization problem stochastic which is undesirable as two runs may result in different solutions.

Understanding the biases induced by both objectives is best seen through example. Suppose our target distribution is a two-dimensional Gaussian with a non-zero correlation between z_1 and z_2 (Figure 16). Suppose $q_1(\mathbf{z})$ and $q_2(\mathbf{z})$ on the left and right of Figure 16 represent optimal densities under objectives $\mathtt{KL}(q \,||\, p_*)$ and $\mathtt{KL}(p_* \,||\, q)$, respectively. To see why, consider the green points $\mathbf{z}_A$ and $\mathbf{z}_B$ in Figure 16:

[27] a chain that is both *irreducible* (it's possible to get from any state to any other state) and *aperiodic* (there's no fixed cycle or period in the transitions between states; the chain doesn't get stuck in a repeating pattern)

[28] $T(\mathbf{z} \to \mathbf{z}')\, p_S(\mathbf{z}) = T(\mathbf{z}' \to \mathbf{z})\, p_S(\mathbf{z}')$ for all $\mathbf{z}, \mathbf{z}'$

- Under variational objective $\mathrm{KL}(q \,||\, p_*)$: Notice that $\mathbf{z}_A$ plays a negligible role in $\mathrm{KL}(q_1 \,||\, p_*)$ as $q_1(\mathbf{z}_A) \approx 0$, whereas $\mathbf{z}_A$ plays a significant role in $\mathrm{KL}(q_2 \,||\, p_*)$ as $q_2(\mathbf{z}_A)$ is non-negligible and $\log(q_2(\mathbf{z}_A)/p_*(\mathbf{z}_A)) \gg 1$. Therefore, we select $q_1(\mathbf{z})$ instead of $q_2(\mathbf{z})$ under the variational objective.
- Under alternative objective $\mathrm{KL}(p_* \,||\, q)$: Notice that $\mathbf{z}_B$ plays a negligible role in $\mathrm{KL}(p_* \,||\, q_2)$ as $q_2(\mathbf{z}_B) \approx p_*(\mathbf{z}_B) \implies \log(p_*(\mathbf{z}_B)/q_2(\mathbf{z}_B)) \approx 0$, whereas $\mathbf{z}_B$ plays a significant role in $\mathrm{KL}(p_* \,||\, q_1)$ as $\log(p_*(\mathbf{z}_B)/q_1(\mathbf{z}_B)) \gg 1$. Therefore, we select $q_2(\mathbf{z})$ instead of $q_1(\mathbf{z})$ under the alternative objective.

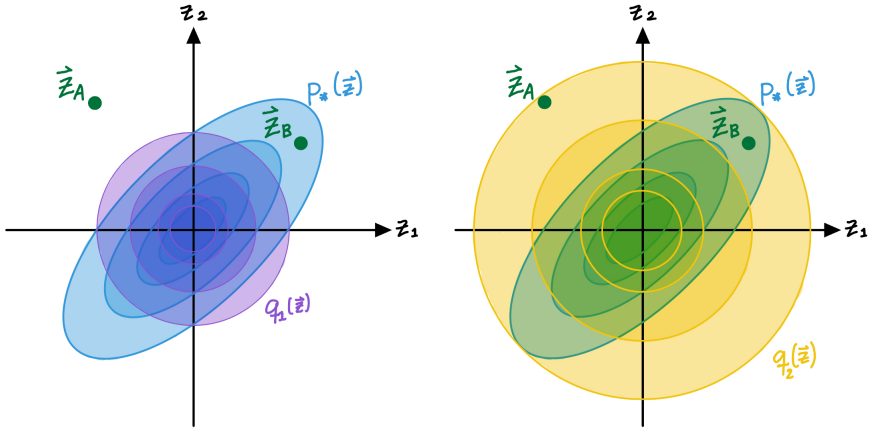


FIGURE 16: Example target two-dimensional Gaussian with a non-zero correlation between z_1 and z_2. Here, $q_1(\mathbf{z})$ on the left and $q_2(\mathbf{z})$ on the right represent optimal densities under objectives $\mathrm{KL}(q \,||\, p_*)$ and $\mathrm{KL}(p_* \,||\, q)$, respectively.

This example highlights a key behavior of each objective. Minimizing $\mathrm{KL}(q \,||\, p_*)$ tends to choose $q(z)$ that underestimates uncertainty whereas minimizing $\mathrm{KL}(p_* \,||\, q)$ will tend to lead to more conservative approximations that overestimate uncertainty, potentially becoming broader or heavier-tailed. In this sense, the variational inference objective is primarily focused on the mode while the alternative objective is focused on coverage. Note that $q_1(\mathbf{z})$ and $q_2(\mathbf{z})$ are results we would expect when optimizing over the *mean-field variational family* (as discussed in (c) and (d)), but our analysis is without loss of generality: the variational inference objective is biased towards the mode(s) while the alternative objective is biased towards coverage. Finally, while our example considers a unimodal target both objectives could in theory capture multi-modal distributions.

Variational inference to Bayesian models

From this point onward, we apply variational inference to a Bayesian model and take our target distribution p_* as the posterior, $p_*(\mathbf{z}) = p(\mathbf{z} \mid \mathbf{x}) \propto p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$. In this setting, the variational inference objective is

$$\mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z} \mid \mathbf{x})\Big) := \mathop{\mathbb{E}}_{\mathbf{z}\sim q(\cdot)}\left[\log\left(\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})}\right)\right] \overset{(29)}{=} \mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z})\Big) - \mathop{\mathbb{E}}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{x} \mid \mathbf{z})\big)\Big] + \log\big(p(\mathbf{x})\big). \tag{62}$$

We note that the log evidence $\log\big(p(\mathbf{x})\big)$ is constant with respect to q so we can remove it from the objective in Equation (VI). It follows that minimizing $\mathrm{KL}\big(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x})\big)$ is equivalent to maximizing the evidence lower bound, $\mathrm{ELBO}\big(q(\mathbf{z})\big)$[30]

$$q^*(\mathbf{z}) = \operatorname*{argmax}_{q(\mathbf{z})\in Q}\Big\{\mathrm{ELBO}\big(q(\mathbf{z})\big)\Big\} \quad \text{where} \quad \mathrm{ELBO}\big(q(\mathbf{z})\big) := \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{x} \mid \mathbf{z})\big)\Big] - \mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z})\Big) \tag{ELBO}$$

Writing the variational inference objective in terms of $\mathrm{ELBO}\big(q(\mathbf{z})\big)$ clarifies its meaning. We see that variational inference is

- maximizing the likelihood that explain the observed data in the first term $\mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{x} \mid \mathbf{z})\big)\Big]$
- minimizing the divergence between the approximate density and the prior in $p(\mathbf{z})$ in the second term $-\mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z})\Big)$

[29] $\mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z} \mid \mathbf{x})\Big) = \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(q(\mathbf{z})\big)\Big] - \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{z} \mid \mathbf{x})\big)\Big] = \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(q(\mathbf{z})\big)\Big] - \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{x} \mid \mathbf{z})\big)\Big] - \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{z})\big)\Big] + \mathbb{E}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{x})\big)\Big]$.

[30] Note that the nomenclature "evidence lower bound" is literal. By Equations (62) and (ELBO) and the fact that KL divergence is non-negative, we have

$$\mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z} \mid \mathbf{x})\Big) = \mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z})\Big) - \mathop{\mathbb{E}}_{\mathbf{z}\sim q(\cdot)}\Big[\log\big(p(\mathbf{x} \mid \mathbf{z})\big)\Big] + \log\big(p(\mathbf{x})\big) = \log\big(p(\mathbf{x})\big) - \mathrm{ELBO}\big(q(\mathbf{z})\big) \geq 0 \quad \implies \quad \mathrm{ELBO}\big(q(\mathbf{z})\big) \leq \log\big(p(\mathbf{x})\big)$$

(c) Coordinate ascent variational inference (CAVI) algorithm in conditionally conjugate models

Coordinate ascent variational inference (CAVI) algorithm for mean-field inference is a powerful approach for solving Equation (VI). CAVI optimizes over the *mean-field variational family* where the latent variables are mutually independent and each governed by a distinct density. Each member of the family has the form $q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$. In other words, we ignore possible correlations between variables and therefore approximate the target with a product of independent *variational factors*. CAVI works by updating variational factor $q_j(z_j)$ while holding all other variational factors constant.

We begin by deriving a single step of the CAVI algorithm. Let z_j and $q_j(z_j)$ represent the j-th latent variable and variational factor, respectively, and let $\mathbf{z}_{-j}$ and $q_{-j}(\mathbf{z}_{-j}) = \prod_{\ell \neq j} q_\ell(z_\ell)$ represent the remaining latent variables and variational factors, respectively. At each step, we update $q_j(z_j)$ while holding $q_{-j}(\mathbf{z}_{-j})$ fixed, and therefore we decompose the KL divergence in Equation (VI) into

$$\mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z} \mid \mathbf{x})\Big) \overset{(31)}{=} \mathbb{E}_{z_j \sim q_j(\cdot)}\Big[\log q_j(z_j) - \mathbb{E}_{\mathbf{z}_{-j} \sim q_{-j}(\cdot)}\left[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})\right]\Big] + \mathrm{KL}\Big(q_{-j}(\mathbf{z}_{-j}) \,\Big\|\, p(\mathbf{z}_{-j} \mid \mathbf{x})\Big).$$

where $p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})$ is the *complete conditional*, i.e., the density of the j-th latent variable z_j conditioned on all the other latent variables $\mathbf{z}_{-j}$ and the observable $\mathbf{x}$. Since we are updating $q_j(z_j)$ while holding $q_{-j}(\mathbf{z}_{-j})$ constant, we minimize the expression above by setting the terms within the expectation to zero. We have that

$$\log q_j(z_j) - \mathbb{E}_{-j}\left[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})\right] = 0 \quad \Longrightarrow \quad q_j^*(z_j) \propto \exp\left\{\mathbb{E}_{-j}\left[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})\right]\right\} \tag{CAVI}$$

Since $q_j^*(z_j)$ zeros the first term the CAVI step is in fact a descent step on the KL divergence. This step is the essence of the CAVI algorithm. A full description of the CAVI algorithm is provided below and highlights that at each iteration, we loop through the m variational factors and (i.) calculate the complete conditional (Line 3) and (ii.) use it to update variational factor $q_j(z_j)$ (Line 4). We repeat until the ELBO has converged (i.e., ELBO hasn't changed much between iterations).

Algorithm 9: Coordinate Ascent Variational Inference (CAVI) [BKM17]

Input: Joint model $p(\mathbf{x}, \mathbf{z})$ over latent variable (unknown) $\mathbf{z}$ and observable dataset $\mathbf{x}$
Output: A mean-field variational density $q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$
Initialize: Variational factors $q_j(z_j)$

1 **while** the ELBO has not converged **do**
2 **for** $j \in \{1, \ldots, m\}$ **do**
3 Calculate complete conditional $p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})$
4 Set $q_j(z_j) \propto \exp\left\{\mathbb{E}_{-j}\left[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})\right]\right\}$ via Equation (CAVI)
5 **end**
6 Calculate $\mathrm{ELBO}\big(q(\mathbf{z})\big) := \mathbb{E}_{\mathbf{z} \sim q(\cdot)}\left[\log p(\mathbf{x} \mid \mathbf{z})\right] - \mathrm{KL}\Big(q(\mathbf{z}) \,\Big\|\, p(\mathbf{z})\Big)$ via Equation (ELBO)
7 **end**
8 **return** $q(\mathbf{z})$

Complete Conditionals in the Exponential Family

We consider a class of Bayesian models that are well-suited for CAVI: models with complete conditionals $p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})$ belonging to the exponential family where we have

$$p(z_j \mid \mathbf{z}_{-j}, \mathbf{x}) = \frac{\tilde{p}(z_j \mid \mathbf{z}_{-j}, \mathbf{x})}{Z(\mathbf{z}_{-j}, \mathbf{x})} \quad \text{where} \quad \tilde{p}(z_j \mid \mathbf{z}_{-j}, \mathbf{x}) \propto h(z_j) \exp\left\{\eta_j(\mathbf{z}_{-j}, \mathbf{x})^\top z_j\right\} \quad \text{and} \quad Z(\mathbf{z}_{-j}, \mathbf{x}) := \int_{\text{all } z_j} \tilde{p}(z_j \mid \mathbf{z}_{-j}, \mathbf{x}) \, \mathrm{d}z_j$$

where $h(z_j)$ is a base measure, $\eta_j(\mathbf{z}_{-j}, \mathbf{x})$ is the natural parameter, z_j is its own sufficient statistic, and $Z(\mathbf{z}_{-j}, \mathbf{x})$ is the normalizing constant. When the complete conditional is in the exponential family, the CAVI step is

$$q_j^*(z_j) \propto \exp\left\{\mathbb{E}\left[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})\right]\right\} = \exp\left\{\mathbb{E}\left[\log h(z_j) + \eta_j(\mathbf{z}_{-j}, \mathbf{x})^\top z_j - \log Z(\mathbf{z}_{-j}, \mathbf{x})\right]\right\} \propto h(z_j) \exp\left\{\mathbb{E}_{-j}\left[\eta_j(\mathbf{z}_{-j}, \mathbf{x})\right]^\top z_j\right\}$$

The expression above reveals the simplicity of the CAVI update when the complete conditional is in the exponential family: $q_j(z_j)$ has the same parametric form as $p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})$ and therefore the CAVI step is simply updating its natural parameter to $\mathbb{E}_{-j}\left[\eta_j(\mathbf{z}_{-j}, \mathbf{x})\right]$.

[31] $\mathrm{KL}\Big(q \,\Big\|\, p\Big) = \mathbb{E}_j\left[\mathbb{E}_{-j}\left[\log q_j(z_j) + \log q_{-j}(\mathbf{z}_{-j}) - \log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x}) - \log p(\mathbf{z}_{-j} \mid \mathbf{x})\right]\right] = \mathbb{E}_j\left[\log q_j(z_j) - \mathbb{E}_{-j}\left[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})\right]\right] + \mathrm{KL}\Big(q_{-j}(\mathbf{z}_{-j}) \,\Big\|\, p(\mathbf{z}_{-j} \mid \mathbf{x})\Big)$

Conditional Conjugacy and Bayesian Models

An important case of exponential family models are *conditionally conjugate models* with local and global variables. Let $\vec{\beta} \in \mathbb{R}^K$ be a vector of *global latent variables* and $\mathbf{z}$ be a vector of *local latent variables*. The global latent variables govern all of the data, while local latent variables govern the i-th component of the observable $\mathbf{x}$. Figure 17 depicts the joint model $p(\vec{\beta}, \mathbf{z}, \mathbf{x})$:

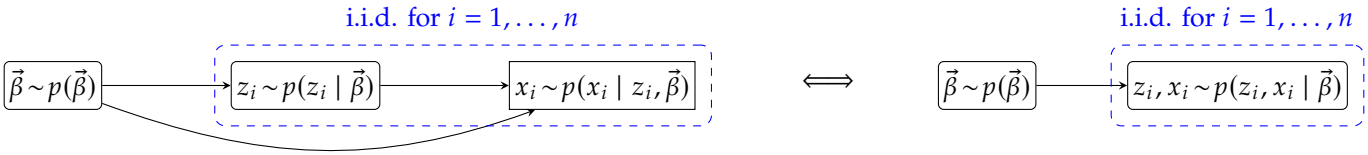


FIGURE 17: Hierarchical model $p(\vec{\beta}, \mathbf{z}, \mathbf{x})$ over global and latent variables $\vec{\beta}$, $\mathbf{z}$ and observable $\mathbf{x}$.

Under the model, we define a generic member of the mean-field variational family $q(\vec{\beta}, \mathbf{z}) \in \mathcal{Q}$ as $q(\vec{\beta}, \mathbf{z}) = q(\vec{\beta}) \prod_{i=1}^n q_i(z_i)$ such that we update $q(\vec{\beta})$ and all $q_i(z_i)$ at each CAVI step. We do so by calculating their respective complete conditionals.

CAVI step for $q(\vec{\beta})$. Consider the CAVI step for $q(\vec{\beta})$, where we hold local variables $\mathbf{z}$ and variational factors $\prod_{i=1}^n q_i(z_i)$ constant. The CAVI step for $q(\vec{\beta})$ is of the form

$$q(\vec{\beta}) \propto \exp\left\{\mathbb{E}_{\mathbf{z} \sim \prod_{i=1}^n q_i(z_i)}\left[\log p(\vec{\beta} \mid \mathbf{z}, \mathbf{x})\right]\right\} \quad \text{where} \quad p(\vec{\beta} \mid \mathbf{z}, \mathbf{x}) \propto p(\vec{\beta}, \mathbf{z}, \mathbf{x}) = p(\vec{\beta})\, p(\mathbf{z}, \mathbf{x} \mid \vec{\beta}) = p(\vec{\beta}) \prod_{i=1}^n p(z_i, x_i \mid \vec{\beta})$$

where $p(z_i, x_i \mid \vec{\beta})$ belongs to the exponential family such that

$$p(z_i, x_i \mid \vec{\beta}) = \frac{\tilde{p}(z_i, x_i \mid \vec{\beta})}{Z(\vec{\beta})} \quad \text{where} \quad \tilde{p}(z_i, x_i \mid \vec{\beta}) \propto h(z_i, x_i) \exp\left\{\vec{\beta}^\top t(z_i, x_i)\right\} \quad \text{and} \quad Z(\vec{\beta}) := \int_{\text{all } z_i, x_i} \tilde{p}(z_i, x_i \mid \vec{\beta})\, \mathrm{d}z_i\, \mathrm{d}x_i \tag{63}$$

where $h(z_i, x_i)$ is a base measure, $\vec{\beta}$ is the natural parameter, $t(z_i, x_i)$ is the sufficient statistic, and $Z(\vec{\beta})$ is the normalizing constant. We select the conjugate prior for the global latent variables as[32]

$$p(\vec{\beta}) = p(\vec{\beta}; \vec{\alpha}) = \frac{h(\vec{\beta})}{Z(\vec{\alpha})} \exp\left\{\left[\vec{\alpha}_{1:K}, \alpha_0\right]^\top \left[\vec{\beta}, -\log Z(\vec{\beta})\right]\right\} \quad \text{where} \quad Z(\vec{\alpha}) := \int_{\text{all } \vec{\beta}} h(\vec{\beta}) \exp\left\{\vec{\alpha}_{1:K}^\top \vec{\beta} - \alpha_0 \log Z(\vec{\beta})\right\} \mathrm{d}\vec{\beta}$$

where $\vec{\alpha} = \left[\vec{\alpha}_{1:K}, \alpha_0\right] \in \mathbb{R}^{K+1}$ are the natural parameters and $t(\vec{\beta}) = \left[\vec{\beta}, -\log Z(\vec{\beta})\right] \in \mathbb{R}^{K+1}$ are the sufficient statistics. By selecting a conjugate prior, the complete conditional $p(\vec{\beta} \mid \mathbf{z}, \mathbf{x})$ is given by

$$p(\vec{\beta} \mid \mathbf{z}, \mathbf{x}) = p(\vec{\beta}) \prod_{i=1}^n p(z_i, x_i \mid \vec{\beta}) \propto h(\vec{\beta}) \exp\left\{\left[\vec{\alpha}_{1:K} + \sum_{i=1}^n t(x_i, z_i)\ ,\ \alpha_0 + n\right]^\top \left[\vec{\beta}, -\log Z(\vec{\beta})\right]\right\}$$

implying that the CAVI step for $q(\vec{\beta})$ is given by

$$\boxed{q(\vec{\beta}) \propto \exp\left\{\mathbb{E}_{\mathbf{z} \sim \prod_{i=1}^n q_i(z_i)}\left[\log p(\vec{\beta} \mid \mathbf{z}, \mathbf{x})\right]\right\} \propto h(\vec{\beta}) \exp\left\{\vec{\lambda}^\top t(\vec{\beta})\right\} \quad \text{where} \quad \vec{\lambda} := \left[\vec{\alpha}_{1:K} + \sum_{i=1}^n \mathbb{E}_{z_i \sim q_i(z_i)}\left[t(x_i, z_i)\right]\ ,\ \alpha_0 + n\right]} \tag{64}$$

CAVI step for $q_i(z_i)$. Consider the CAVI step on $q_i(z_i)$, where we hold all other local variables $\mathbf{z}_{-i}$ and global latent variable $\vec{\beta}$ constant. We note that local latent variables are i.i.d. and independent of all other data such that $p(z_i \mid \mathbf{z}_{-i}, x_i, \mathbf{x}_{-i}, \vec{\beta}) = p(z_i \mid x_i, \vec{\beta})$. Therefore the CAVI step is of the form

$$q_i(z_i) \propto \exp\left\{\mathbb{E}_{\vec{\beta} \sim q(\vec{\beta})}\left[\log p(z_i \mid x_i, \vec{\beta})\right]\right\} \quad \text{where} \quad p(z_i \mid x_i, \vec{\beta}) \propto p(z_i, x_i \mid \vec{\beta})$$

[32] See bayesian estimation for conjugate distributions with $\boldsymbol{\chi} = \vec{\alpha}_{1:K}$, $\nu = \alpha_0$, $A(\boldsymbol{\eta}) = \log(Z(\vec{\alpha}_{1:K}, \alpha_0))$, and $f(\boldsymbol{\chi}, \nu) = \frac{h(\vec{\beta})}{Z(\vec{\alpha})}$.

where we assume $p(z_i \mid x_i, \vec{\beta})$ belongs to an exponential family such that[33]

$$p(z_i \mid x_i, \vec{\beta}) = \frac{\tilde{p}(z_i \mid x_i, \vec{\beta})}{Z(\eta_i(x_i, \vec{\beta}))} \quad \text{where} \quad \tilde{p}(z_i \mid x_i, \vec{\beta}) \propto h(z_i) \exp\left\{\eta_i(x_i, \vec{\beta})^\top z_i\right\} \quad \text{and} \quad Z(\eta_i(x_i, \vec{\beta})) := \int_{\text{all } z_i} \tilde{p}(z_i \mid x_i, \vec{\beta}) \, \mathrm{d}z_i$$

where $h(z_i)$ is a base measure, $\eta_i(x_i, \vec{\beta})$ is the natural parameter, z_i is its own sufficient statistic, and $Z(\eta_i(x_i, \vec{\beta}))$ is the normalizing constant. Therefore, the CAVI step for $q_i(z_i)$ is given by

$$\boxed{q_i(z_i) \propto \exp\left\{\mathbb{E}_{\vec{\beta}\sim q(\vec{\beta})}\left[\log p(z_i \mid x_i, \vec{\beta})\right]\right\} \propto \exp\left\{\varphi_i^\top z_i - \mathbb{E}_{\vec{\beta}\sim q(\vec{\beta})}\left[\log Z(\eta_i(x_i, \vec{\beta}))\right]\right\} \quad \text{where} \quad \varphi_i := \mathbb{E}_{\vec{\beta}\sim q(\vec{\beta})}\left[\eta_i(x_i, \vec{\beta})\right]} \tag{65}$$

ELBO for conditionally conjugate models. The final step in CAVI is calculating the evidence lower bound as it defines our stopping criteria. For conditionally conjugate models, we have that

$$\text{ELBO}\left(q(\vec{\beta}, \mathbf{z})\right) \overset{(34)}{=} \mathbb{E}_{\vec{\beta}, \mathbf{z}\sim q(\vec{\beta}, \mathbf{z})}\left[\log p(\vec{\beta}, \mathbf{z}, \mathbf{x}) - \log q(\vec{\beta}, \mathbf{z})\right]$$

$$\boxed{\text{ELBO}\left(q(\vec{\beta}, \mathbf{z})\right) = \left(\vec{\alpha}_{1:K} + \sum_{i=1}^{n} \mathbb{E}_{z_i\sim q_i(z_i)}\left[t(x_i, z_i)\right]\right)^\top \mathbb{E}_{\vec{\beta}\sim q(\vec{\beta})}\left[\vec{\beta}\right] - \mathbb{E}_{\vec{\beta}\sim q(\vec{\beta})}\left[\log Z(\vec{\beta})\right](\alpha_0 + n) - \mathbb{E}_{\vec{\beta}, \mathbf{z}\sim q(\vec{\beta}, \mathbf{z})}\left[\log q(\vec{\beta}, \mathbf{z})\right]} \tag{66}$$

where

$$\mathbb{E}_{\vec{\beta}, \mathbf{z}\sim q(\vec{\beta}, \mathbf{z})}\left[\log q(\vec{\beta}, \mathbf{z})\right] = \vec{\lambda}^\top \mathbb{E}_{\vec{\beta}\sim q(\vec{\beta})}\left[t(\vec{\beta})\right] - \log Z(\vec{\lambda}) + \varphi_i^\top \mathbb{E}_{z_i\sim q(z_i)}\left[z_i\right] - \log Z(\varphi_i)$$

where $Z(\vec{\lambda}) := \int_{\text{all } \vec{\beta}} h(\vec{\beta}) \exp\left\{\vec{\lambda}^\top t(\vec{\beta})\right\} \mathrm{d}\vec{\beta}$ and $Z(\varphi_i) := \int_{\text{all } z_i} h(z_i) \exp\left\{\varphi_i^\top z_i\right\} \mathrm{d}z_i$.

We now have all the pieces to implement CAVI for conditionally conjugate models. The CAVI algorithm is provided below and highlights that at each iteration, we (i.) update the global variational factor $q(\vec{\beta})$ (Line 2) and then (ii.) loop through the n variational factors and update the local variational factors $q_i(z_i)$ (Line 3). We repeat until the ELBO has converged (i.e., ELBO hasn't changed much between iterations).

Algorithm 10: Coordinate Ascent Variational Inference (CAVI) in Conditionally Conjugate Bayesian Models [BKM17]

Input: Joint model $p(\vec{\beta}, \mathbf{x}, \mathbf{z})$ over global and latent variables $\vec{\beta}, \mathbf{z}$ and observable $\mathbf{x}$ (Figure 17)
Output: A mean-field variational density $q(\vec{\beta}, \mathbf{z}) = q(\vec{\beta}) \prod_{i=1}^n q_i(z_i)$
Initialize: Variational factors $q(\vec{\beta})$ and $q_i(z_i)$
1 **while** the ELBO has not converged **do**
2 Update global variational factor $q(\vec{\beta})$ by setting its natural parameters to $\left[\vec{\alpha}_{1:K} + \sum_{i=1}^n \mathbb{E}_{q_i}\left[t(x_i, z_i)\right] \, , \, \alpha_0 + n\right]$ via (64) **for** $i \in \{1, \ldots, n\}$ **do**
3 Update local variational factor $q_i(z_i)$ by setting its natural parameters to $\mathbb{E}_{q(\vec{\beta})}\left[\eta_i(x_i, \vec{\beta})\right]$ via (65)
4 **end**
5 Calculate $\text{ELBO}\left(q(\vec{\beta}, \mathbf{z})\right)$ via (66)
6 **end**
7 **return** $q(\vec{\beta}, \mathbf{z})$

Relation to another algorithm in the class

The primary similarity of CAVI to other algorithms in the class is how it leverages conjugacy. Examples from the class include coordinate ascent, expectation maximization, and Gibbs sampling. All of these approaches are well suited for situations where conditioning on subsets of the latent variables may speed the optimization/sampling process. Such settings include

[33] Note that by assuming that the local likelihood $p(z_i, x_i \mid \vec{\beta})$ belongs to the exponential family, we have that $p(z_i \mid x_i, \vec{\beta})$ is in an exponential family as $p(z_i \mid x_i, \vec{\beta}) \propto p(z_i, x_i \mid \vec{\beta})$

[34] $\text{KL}\left(q(\vec{\beta}, \mathbf{z}) \,\|\, p(\vec{\beta}, \mathbf{z} \mid \mathbf{x})\right) = \mathbb{E}_{q(\vec{\beta}, \mathbf{z})}\left[\log\left(\frac{q(\vec{\beta}, \mathbf{z})}{p(\vec{\beta}, \mathbf{z} \mid \mathbf{x})}\right)\right] = \mathbb{E}_{q(\vec{\beta}, \mathbf{z})}\left[\log q(\vec{\beta}, \mathbf{z}) - \log p(\vec{\beta}, \mathbf{z}, \mathbf{x}) - \log p(\mathbf{x})\right] \implies \text{ELBO}(q(\vec{\beta}, \mathbf{z})) = -\text{KL}\left(q(\vec{\beta}, \mathbf{z}) \,\|\, p(\vec{\beta}, \mathbf{z} \mid \mathbf{x})\right) - \log p(\mathbf{x})$

conditionally conjugate models where conditioning a subsets of latent variables (e.g., complete conditional) results in a simple and fast update (e.g., updating the natural parameters of an exponential family). We argued in class that the speed ups from simple updates may make up for having to iterate over all of the latent variables. For example, Gibbs sampling is a procedure in which we sample along complete conditionals of the latent, and while we constrained our step to be along z_i, it may speed mixing times especially in settings when the target distribution is multi-modal and does not have strongly correlated subsets (e.g., $\text{Corr}(z_i, z_j) < 0.1$ for all $i \neq j$). Similarly, CAVI is well-suited for targets that do not have strongly correlated subsets as CAVI optimizes over the mean-field variational family.

(d) Biases induced by the mean-field assumption

To better understand the biases introduced by the mean-field assumption, let us revisit Figure 16 discussed in (b). The optimal $q^*(\mathbf{z}) = q_1(\mathbf{z})$ on the left is the result of optimizing over the mean-field variational family. We see that the resulting $q^*(\mathbf{z})$ is accurate, especially about the mode and marginal densities of the latent variables (i.e., along z_1 and z_2). However, it cannot capture correlation between the latents by construction. Thus the variances along the direction $z_1 + z_2$ are underestimated. This is a common theme in variational inference: the variational objective $\text{KL}(q \,||\, p_*)$ tends to underestimate the variance of the target and fails to capture the wide tails of the target distribution.

With this in mind, when is it appropriate to utilize the mean-field approximation given its biases? The mean-field approximation is most appropriate when the target distribution the latents are independent. In this case, the variational family is able to capture the mode and marginal densities of the latents with high accuracy. When the target has weak to moderate correlations between latents (e.g., $\text{Corr}(z_i, z_j) < 0.3$ for all $i \neq j$), the mean-field approximation can still be appropriate as long as the analysis is focused on the mode(s) and/or the latent marginals (i.e., along z_1 and z_2). In the case of strong correlations, the mean-field approximation is only appropriate for mode capture, and anything beyond that should be interpreted with caution as variational factors will collapse to a local modes as the correlation goes to 1.

(e) + (f) Stein variational gradient descent (SVGD)

Background. Let $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^m$ and let $\kappa(\mathbf{z}, \mathbf{z}') : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be a positive definite kernel. The reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ of $\kappa(\mathbf{z}, \mathbf{z}')$ comprises the linear combinations $\mathcal{H} = \left\{ f \;:\; f(\mathbf{z}) = \sum_{i=1}^N a_i \kappa(\mathbf{z}, \mathbf{z}_i),\; a_i \in \mathbb{R},\; N \in \mathbb{N},\; \mathbf{z}_i \in \mathcal{Z} \right\}$, endowed with an inner product $\left\langle f, g \right\rangle_{\mathcal{H}} = \sum_{i=1}^N \sum_{j=1}^N a_i b_j \kappa(\mathbf{z}_i, \mathbf{z}_j)$. Kernel $\kappa(\mathbf{z}, \cdot)$ in $\mathcal{H}$ satisfies the important "reproducing" property,

$$f(\mathbf{z}) = \left\langle f, \kappa(\cdot, \mathbf{z}) \right\rangle_{\mathcal{H}} = \left\langle \kappa(\mathbf{z}, \cdot), f \right\rangle_{\mathcal{H}} \quad \text{and} \quad \kappa(\mathbf{z}, \mathbf{z}') = \langle \kappa(\cdot, \mathbf{z}), \kappa(\cdot, \mathbf{z}') \rangle_{\mathcal{H}}$$

Let $\mathcal{H}^d$ represent the vector space $\mathbf{f} = [f_1, \ldots, f_d]^\top$ with $f_i \in \mathcal{H}$, equipped with inner product $\langle \mathbf{f}, \mathbf{g} \rangle_{\mathcal{H}^d} = \sum_{i=1}^d \left\langle f_i, g_i \right\rangle_{\mathcal{H}}$. More information about RKHS is provided in Appendix A.

Intuitive derivation of SVGD. For Stein variational gradient descent (SVGD), we consider a variational family of transformed random variables. Let $\mathbf{z} = \boldsymbol{\chi}(\mathbf{z}_0)$ where $\boldsymbol{\chi} : \mathcal{Z}_0 \to \mathcal{Z}$ represent the mapping from a tractable *base distribution* (e.g., Normal) $\mathbf{z}_0 \sim q_0(\mathbf{z}_0)$ to the target distribution $\mathbf{z} \sim p_*(\mathbf{z})$. We define the flow map as $\boldsymbol{\chi}(\mathbf{z}_0) = \mathbf{z}_0 + \boldsymbol{v}(\mathbf{z}_0)$ where $\boldsymbol{v}$ represents the *displacement field* (Figure 18). We note that the displacement field $\boldsymbol{v}$ is unknown and therefore we consider an infinitesimal flow $\mathbf{z}_{t+dt} = \boldsymbol{\psi}_t(\mathbf{z}_t) = \mathbf{z}_t + \varepsilon_t \boldsymbol{\omega}(\mathbf{z}_t)$. We view the infinitesimal displacement $\boldsymbol{\omega}(\mathbf{z}_t)$ as an ε_t-step that maps particle $\mathbf{z}_t$ to $\mathbf{z}_{t+dt}$. As result of this motion, the density q_t evolves to q_{t+dt}. Therefore, our goal is put the particles into a motion that drives the approximate density q towards the target p_*. Specifically, we set the infinitesimal displacement $\boldsymbol{\omega}(\mathbf{z}_t)$ such that our variational density approaches the target as quickly as possible. To illustrate this point, we select a transport map to move the reference particles as if they obeyed a time inhomogeneous ODE as we have

$$\mathbf{z}_{t+dt} = \mathbf{z}_t + \varepsilon_t \boldsymbol{\omega}(\mathbf{z}_t) \quad \Longrightarrow \quad \lim_{\varepsilon_t \to 0} \left\{ \frac{\mathbf{z}_{t+dt} - \mathbf{z}_t}{\varepsilon_t} \right\} = \boldsymbol{\omega}(\mathbf{z}_t) \quad \Longrightarrow \quad \frac{\partial \mathbf{z}}{\partial t} = \mathbf{v}(\mathbf{z}, t) = \boldsymbol{\omega}(\mathbf{z}_t)$$

where we see that the infinitesimal displacement is the instantaneous velocity of the particle $\mathbf{v}(\mathbf{z}, t)$.

We now deriving the optimal flow $\boldsymbol{\omega}^*$ under SVGD. While there a several ways to do so (see Appendix A.), an intuitive method is *functional gradient descent* on the KL divergence between the approximation distribution q and the target p_*. We treat $\text{KL}(q \| p_*)$ as a *functional* of the displacement field, $F[\boldsymbol{v}]$, and perform functional gradient descent. Much like canonical gradient descent where one iteratively calculates the gradient of an objective *function* to move through a *vector* space and ultimately arrive at a minimum of that *function*, functional gradient descent works by iteratively calculating the gradient of a *functional* to move through a function space and ultimately arrive at a minimum of that *functional*. Here, the function space is the space of possible velocity fields applied to the particles and given that our functional $F[\boldsymbol{v}]$ is the $\text{KL}(q \| p_*)$ under $\boldsymbol{v}$, our procedure will minimize the KL divergence at each step $\text{KL}(q_{t+dt} \,||\, p_*) \leq \text{KL}(q_t \,||\, p_*)$. We have that

$$\mathbf{z}^{(t+1)} \leftarrow \mathbf{z}^{(t)} - \varepsilon_t \left(\nabla_{\boldsymbol{v}} F[\boldsymbol{v}] \right) \Big|_{\boldsymbol{v}=\mathbf{0}} \quad \Longrightarrow \quad \mathbf{z}^{(t+1)} \leftarrow \mathbf{z}^{(t)} + \varepsilon_t \; \boldsymbol{\omega}^*(\mathbf{z}^{(t)}) \tag{67}$$

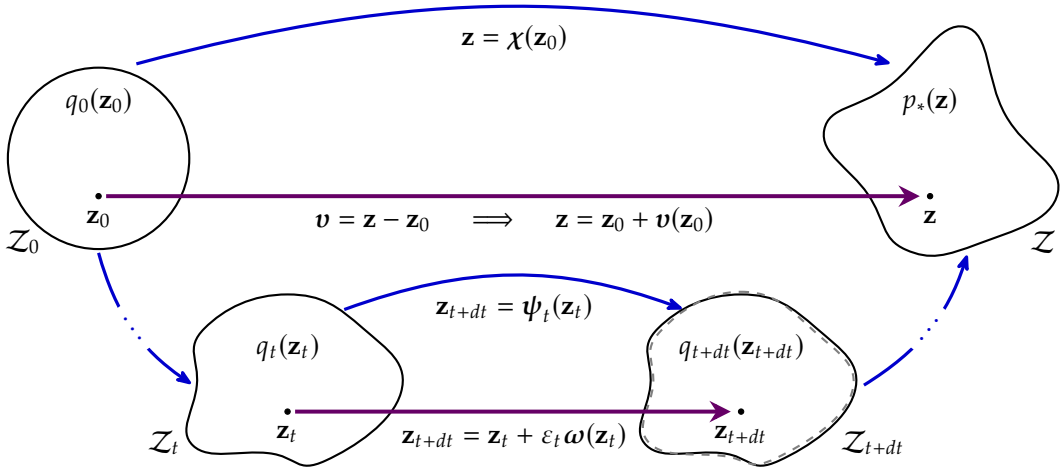


Figure 18: Flow map $\chi : \mathcal{Z}_0 \to \mathcal{Z}$ and infinitesimal flow $\psi_t : \mathcal{Z}_t \to \mathcal{Z}_{t+dt}$. The dashed gray line indicates the infinitesimal flow ψ_t at time t.

$$\boldsymbol{\omega}^*(\cdot) = \mathop{\mathbb{E}}_{\mathbf{z}\sim q(\cdot)}\left[\mathcal{A}_{p_*}\kappa(\mathbf{z},\cdot)\right] \quad \text{where } \mathcal{A}_{p_*} \text{ is the Stein operator} \quad \mathcal{A}_{p_*}\kappa(\mathbf{z},\cdot) := \kappa(\mathbf{z},\cdot)\,\nabla_{\mathbf{z}} \log p_*(\mathbf{z}) + \nabla_{\mathbf{z}}\kappa(\mathbf{z},\cdot) \tag{68}$$

and ε_t is the learning rate at t. A full derivation is provided in Appendix B. We note that we are able to calculate the Stein operator as the κ is predefined, and we have access to the unnormalized target $\tilde{p}$ such that $\nabla_{\mathbf{z}} \log p_*(\mathbf{z}) = \nabla_{\mathbf{z}} \log \tilde{p}_*(\mathbf{z})$. We approximate the expectation over q with sample averages by defining a set of N particles $\{\mathbf{z}_i\}_{i=1}^N$ to approximate the expectation

$$\boldsymbol{\omega}^*(\cdot) \approx \hat{\boldsymbol{\omega}}^*(\cdot) := \frac{1}{N}\sum_{j=1}^{N}\left[\kappa(\mathbf{z}_j,\cdot)\,\nabla_{\mathbf{z}_j} \log \tilde{p}_*(\mathbf{z}_j) + \nabla_{\mathbf{z}_j}\kappa(\mathbf{z}_j,\cdot)\right] \tag{SVGD}$$

Therefore, SVGD is a ensemble method that iteratively applies the (estimated) optimal flow $\boldsymbol{\omega}^*$ to a finite set of particles $\{\mathbf{z}_i\}_{i=1}^N$. After undergoing T iterations, we have a set of particles $\{\mathbf{z}_i^{(T)}\}_{i=1}^N$ that approximate the target distribution $p_*(\mathbf{z})$. Algorithm 11 provides a complete summary SVGD:

Algorithm 11: Stein Variational Gradient Descent (SVGD) [LW16]

Input: Target distribution $p_*(\mathbf{z})$ over latent variable $\mathbf{z}$ and set of N initial particles $\left\{\mathbf{z}_i^{(0)}\right\}_{i=1}^N$ where $\mathbf{z}_i^{(0)} \overset{\text{i.i.d.}}{\sim} q_0(\mathbf{z})$

Output: Set of particles $\left\{\mathbf{z}_i^{(T)}\right\}_{i=1}^N$ that approximates the target distribution $p_*(\mathbf{z})$

for $t \in \{0, 1, \ldots, T-1\}$ **do**

  Take SVGD step $\mathbf{z}_i^{(t+1)} \leftarrow \mathbf{z}_i^{(t)} + \varepsilon_t \hat{\boldsymbol{\omega}}^*\left(\mathbf{z}_i^{(t)}\right)$ where $\hat{\boldsymbol{\omega}}^*(\mathbf{z}) := \frac{1}{N}\sum_{j=1}^{N}\left[\kappa(\mathbf{z}_j^{(t)},\mathbf{z})\,\nabla_{\mathbf{z}_j^{(t)}} \log \tilde{p}_*(\mathbf{z}_j^{(t)}) + \nabla_{\mathbf{z}_j^{(t)}}\kappa(\mathbf{z}_j^{(t)},\mathbf{z})\right]$

end

Variational family for SVGD. The variational family for SVGD the set of all distributions that can be produced by pushing an initial particle distribution through a sequence of infinitesimal flows. This provides a rich and expressive set of distributions, but the procedure can be computationally expensive requiring many iterations to reach a reasonable solution. SVGD also requires careful selection of the hyperparameters (e.g., kernel bandwidth). After passing the particles through SVGD, one can sample from the approximate target by drawing one of the N final particles uniformly as particles will be concentrated near modes and sparse in valleys. Increasing the number of particles leads to smoother empirical estimates and as $N \to \infty$, our estimate of the target will be "continuous" in a proper sense [LW16].

Particle dynamics. The (estimated) optimal flow $\hat{\boldsymbol{\omega}}^*$ describes the how the particles evolve with time. The two terms in Equation (SVGD) play different roles: The first term $\kappa(\mathbf{z}_j,\mathbf{z})\,\nabla_{\mathbf{z}_j} \log \tilde{p}_*(\mathbf{z}_j)$ exhibits mode-seeking behavior as it draws particles toward regions of high target probability by following a *kernel-smoothed* gradient. The second term $\nabla_{\mathbf{z}_j}\kappa(\mathbf{z}_j,\mathbf{z})$ acts

as a repulsion force that spreads out particles and prevents them from collapsing onto the same mode. To highlight this point, consider the RBF kernel $\kappa(\mathbf{z}, \mathbf{z}') = \exp\left(-\frac{1}{h}\|\mathbf{z} - \mathbf{z}'\|_2^2\right)$. The second term simplifies to $\sum_j \frac{2}{h}(\mathbf{z} - \mathbf{z}_j)\kappa(\mathbf{z}_j, \mathbf{z})$ which pushes a particle $\mathbf{z}$ away from neighboring particles $\mathbf{z}_j$ inside its kernel radius.

Comparison to Teleporting Walkers. Like SVGD, Teleporting Walkers maintains an ensemble of particles, but rather than using a functional gradient based flow, its evolution uses MCMC [LWZ22]. In addition to allowing for the ensemble to explore the target distribution, the particles interact with each other such that they can "teleport" to each other. So in some sense, the teleportation step is an attractive force pushes particles close to each other. The variance of the chains is otherwise controlled by the proposal distribution as the chains move independently according to an MCMC sampler, e.g., Metropolis-Adjusted Langevin (noisy optimization). The Langevin proposal is given by

$$\mathbf{y} = \mathbf{z}_t + \Delta t\left(\nabla_{\mathbf{z}} \log \tilde{p}_*(\mathbf{z})\Big|_{\mathbf{z}=\mathbf{z}_t}\right) + \sigma\sqrt{\Delta t}\,\boldsymbol{\xi} \quad \text{where} \quad \boldsymbol{\xi} \sim \mathcal{N}(0, I) \quad \text{and} \quad \sigma \geq \sqrt{2}$$

and the candidate $\mathbf{y}$ is accepted with the usual Metropolis-Hastings acceptance probability. Similarly to SVGD, the Langevin proposal has a mode-seeking term that draws particles toward regions of high target probability. However, the second term is neither a repulsion nor an attraction term, but rather a diffusion term that allows the particles to explore the target distribution where σ controls the variance of the chains. Appendix C describes the Teleporting Walkers algorithm in detail.

Selecting kernel bandwidth. Liu, *et al.* [LW16] describe the "median trick" for RBF kernel $\kappa(\mathbf{z}, \mathbf{z}') = \exp\left(-\frac{1}{h}\|\mathbf{z} - \mathbf{z}'\|_2^2\right)$, they take the bandwidth to be $h = \texttt{med}^2/\log n$, where $\texttt{med}$ is the median of the pairwise distance between the current points $\{\mathbf{z}_i^{(t)}\}_{i=1}^N$; this is based on the intuition that $\sum_j \kappa(\mathbf{z}_i, \mathbf{z}_j) \approx n \exp\left(-\frac{1}{h}\texttt{med}^2\right) = 1$, so that for each $\mathbf{z}_i$ the contribution from its own gradient and the influence from the other points balance with each other [LW16].

While the median trick is reasonable in lower dimensional spaces, ℓ_2-norms are known to collapse in higher dimensions. For example, most of the mass of a multivariate Gaussian is not near the mean, but in an increasingly distant "shell" around it [Dom12]. In this case, the mode seeking terms would drive particles to the shell and thus the pairwise distances would grow larger and larger. The kernel term $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ will be either ≈ 0 (if $h \gg \texttt{med}^2$) or ≈ 1 (if $h \ll \texttt{med}^2$). The median trick is sensitive to the dimension and to heavy-tailed targets.

Alternatively, we can use a K-nearest neighbors bandwidth set a distinct bandwidth for each particle,

$$h_i = \frac{1}{K}\sum_{j \in \mathcal{N}_k(i)} \|\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)}\|_2^2, \qquad \kappa_{ij} = \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2^2}{\sqrt{h_i h_j}}\right),$$

where $\mathcal{N}_K(i)$ is the set of K nearest neighbors of $\mathbf{z}_i^{(t)}$. This will allow the particles to adaptively select the bandwidth based on the local density of the particles. The K-nearest neighbors bandwidth is less sensitive to the dimension and can be more robust in high-dimensional spaces.

Appendix A. Stein's Identify and Reproducing Kernel Hilbert Spaces

Suppose $p(\mathbf{z})$ is a continuously differentiable (i.e., smooth) density supported on $\mathcal{Z}$. Let $\boldsymbol{\phi}(\mathbf{z}) = [\phi_1(\mathbf{z}), \dots, \phi_d(\mathbf{z})]^\top$ be a smooth vector function that is in the *Stein class* of p such that *Stein's identity* holds:

$$\mathbb{E}_{\mathbf{z}\sim p(\cdot)}\left[\mathcal{A}_p\boldsymbol{\phi}(\mathbf{z})\right] = 0 \quad \text{where } \mathcal{A}_p \text{ is the Stein operator} \quad \mathcal{A}_p\boldsymbol{\phi}(\mathbf{z}) := \boldsymbol{\phi}(\mathbf{z})\,\nabla_{\mathbf{z}}\log p(\mathbf{z})^\top + \nabla_{\mathbf{z}}\boldsymbol{\phi}(\mathbf{z}) \tag{69}$$

Let $q(\mathbf{z})$ represent a different smooth density also supported on $\mathcal{Z}$. The expectation $\mathbb{E}_{\mathbf{z}\sim q(\cdot)}\left[\mathcal{A}_p\boldsymbol{\phi}(\mathbf{z})\right]$ would no longer equal zero for general $\boldsymbol{\phi}$ and therefore measures how different p and q are. Note that $\mathbb{E}_{\mathbf{z}\sim q(\cdot)}\left[\mathcal{A}_p\boldsymbol{\phi}(\mathbf{z})\right]$ is a matrix quantity and ideally would have a scalar metric for measuring how different p and q are. We therefore let the trace of this expectation represent the "violation of Stein's identity" for a given $\boldsymbol{\phi}$. We then define the Kernelized Stein discrepancy (KSD) $\mathbb{S}(q,p)$ as the "maximum violation of Stein's identity" over the unit ball $\mathcal{B}(\mathcal{H}^d) := \left\{\boldsymbol{\phi}\in\mathcal{H}^d \;:\; \|\boldsymbol{\phi}\|^2_{\mathcal{H}^d} = \langle\boldsymbol{\phi},\boldsymbol{\phi}\rangle_{\mathcal{H}^d} \le 1\right\}$

$$\mathbb{S}(q,p) := \max_{\boldsymbol{\phi}\in\mathcal{B}(\mathcal{H}^d)}\left\{\mathbb{E}_{\mathbf{z}\sim q(\cdot)}\left[\mathrm{Tr}\left(\mathcal{A}_p\boldsymbol{\phi}(\mathbf{z})\right)\right]^2\right\} \tag{70}$$

It has been shown [LLJ16] that the optimal solution of $\mathbb{S}(q,p)$ in (70) has an analytical form $\boldsymbol{\phi}(\mathbf{z}) = \boldsymbol{\phi}^*_{q,p}(\mathbf{z})/\|\boldsymbol{\phi}^*_{q,p}\|_{\mathcal{H}^d}$ where

$$\boldsymbol{\phi}^*_{q,p}(\cdot) = \operatorname*{argmax}_{\boldsymbol{\phi}\in\mathcal{B}(\mathcal{H}^d)}\left\{\mathbb{E}_{\mathbf{z}\sim q(\cdot)}\left[\mathrm{Tr}\left(\mathcal{A}_p\boldsymbol{\phi}(\mathbf{z})\right)\right]^2\right\} = \mathop{\mathbb{E}}_{\mathbf{z}\sim q(\cdot)}\left[\mathcal{A}_p\kappa(\mathbf{z},\cdot)\right] \quad \text{for which we have} \quad \mathbb{S}(q,p) = \|\boldsymbol{\phi}^*_{q,p}\|_{\mathcal{H}^d}. \tag{71}$$

Appendix B. Functional Gradient

For any functional $F[\boldsymbol{v}]$ of $\boldsymbol{v}\in\mathcal{H}^d$, its functional gradient $\nabla_{\boldsymbol{v}}F[\boldsymbol{v}]$ is a function in $\mathcal{H}^d$ such that $F[\boldsymbol{v}+\varepsilon\boldsymbol{\omega}] = F[\boldsymbol{v}] + \varepsilon\langle\nabla_{\boldsymbol{v}}F[\boldsymbol{v}],\boldsymbol{\omega}\rangle_{\mathcal{H}^d} + O(\varepsilon^2)$ for any $\boldsymbol{\omega}\in\mathcal{H}^d$ and $\varepsilon\in\mathbb{R}$. We define our functional $F[\boldsymbol{v}] = \mathtt{KL}\big(q_{[\boldsymbol{\chi}]}(\mathbf{z})\,\|\,p_*(\mathbf{z})\big)$ where $\boldsymbol{\chi}(\mathbf{z}_0) = \mathbf{z}_0 + \boldsymbol{v}(\mathbf{z}_0)$. It follows that

$$\mathtt{KL}\Big(q_{[\boldsymbol{\chi}]}(\mathbf{z})\,\Big\|\,p_*(\mathbf{z})\Big) = \int q_{[\boldsymbol{\chi}]}(\mathbf{z})\log\left(\frac{q_{[\boldsymbol{\chi}]}(\mathbf{z})}{p_*(\mathbf{z})}\right)\mathrm{d}\mathbf{z} = \int q(\boldsymbol{\chi}^{-1}(\mathbf{z}))\cancel{\left|\det\left(\nabla_{\mathbf{z}}\boldsymbol{\chi}^{-1}(\mathbf{z})\right)\right|}\log\left(\frac{q(\boldsymbol{\chi}^{-1}(\mathbf{z}))\left|\det\left(\nabla_{\mathbf{z}}\boldsymbol{\chi}^{-1}(\mathbf{z})\right)\right|}{p_*(\mathbf{z})}\right)\cancel{\left(\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\mathbf{z}_0}\right)}\mathrm{d}\mathbf{z}_0$$

$$\mathtt{KL}\Big(q_{[\boldsymbol{\chi}]}(\mathbf{z})\,\Big\|\,p_*(\mathbf{z})\Big) = \int q(\mathbf{z}_0)\log\left(\frac{q(\mathbf{z}_0)}{p_*(\boldsymbol{\chi}(\mathbf{z}_0))\left|\det\left(\nabla_{\mathbf{z}_0}\boldsymbol{\chi}(\mathbf{z}_0)\right)\right|}\right)\mathrm{d}\mathbf{z}_0 = \mathtt{KL}\Big(q(\mathbf{z}_0)\,\Big\|\,p_{*[\boldsymbol{\chi}^{-1}]}(\mathbf{z}_0)\Big) \text{ as } \frac{\partial\boldsymbol{\chi}^{-1}(\mathbf{z})}{\partial\mathbf{z}} = \left(\frac{\partial\mathbf{z}}{\partial\mathbf{z}_0}\right)^{-1} = \frac{1}{\nabla_{\mathbf{z}_0}\boldsymbol{\chi}(\mathbf{z}_0)}\,.$$

Then we have that

$$F[\boldsymbol{v}+\varepsilon\boldsymbol{\omega}] = \mathtt{KL}\Big(q(\mathbf{z}_0)\,\Big\|\,p_{*[\boldsymbol{\chi}^{-1}]}(\mathbf{z}_0)\Big) = \mathop{\mathbb{E}}_{\mathbf{z}_0\sim q(\cdot)}\left[\log q(\mathbf{z}_0) - \log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0)+\varepsilon\boldsymbol{\omega}(\mathbf{z}_0)) - \log\det\left(I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)+\varepsilon\nabla_{\mathbf{z}_0}\boldsymbol{\omega}(\mathbf{z}_0)\right)\right]$$

Next we have that

$$F[\boldsymbol{v}+\varepsilon\boldsymbol{\omega}] - F[\boldsymbol{v}] = -\Delta_1 - \Delta_2$$

where

$$\begin{aligned}\Delta_1 &= \mathbb{E}_q\left[\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0)+\varepsilon\boldsymbol{\omega}(\mathbf{z}_0))\right] - \mathbb{E}_q\left[\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0))\right]\\ \Delta_2 &= \mathbb{E}_q\left[\log\det\left(I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)+\varepsilon\nabla_{\mathbf{z}_0}\boldsymbol{\omega}(\mathbf{z}_0)\right)\right] - \mathbb{E}_q\left[\log\det\left(I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)\right)\right]\end{aligned}$$

For the terms in the above equation, we have

$$\begin{aligned}\Delta_1 &= \mathbb{E}_q\left[\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0)+\varepsilon\boldsymbol{\omega}(\mathbf{z}_0))\right] - \mathbb{E}_q\left[\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0))\right]\\ \Delta_1 &= \varepsilon\,\mathbb{E}_q\left[\nabla_{\mathbf{z}_0}\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0))\cdot\boldsymbol{\omega}(\mathbf{z}_0)\right] + O(\varepsilon^2)\\ \Delta_1 &= \varepsilon\,\mathbb{E}_q\left[\nabla_{\mathbf{z}_0}\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0))\cdot\langle\kappa(\mathbf{z}_0,\cdot),\boldsymbol{\omega}\rangle_{\mathcal{H}^d}\right] + O(\varepsilon^2)\\ \Delta_1 &= \varepsilon\left\langle\mathbb{E}_q\left[\nabla_{\mathbf{z}_0}\log p_*(\mathbf{z}_0+\boldsymbol{v}(\mathbf{z}_0))\cdot\kappa(\mathbf{z}_0,\cdot)\right],\boldsymbol{\omega}\right\rangle_{\mathcal{H}^d} + O(\varepsilon^2)\end{aligned}$$

and

$$\begin{aligned}\Delta_2 &= \mathbb{E}_q\left[\log\det\left(I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)+\varepsilon\nabla_{\mathbf{z}_0}\boldsymbol{\omega}(\mathbf{z}_0)\right)\right] - \mathbb{E}_q\left[\log\det\left(I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)\right)\right]\\ \Delta_2 &= \varepsilon\,\mathbb{E}_q\left[\nabla\log\det\left(I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)\right)\cdot\nabla_{\mathbf{z}_0}\boldsymbol{\omega}(\mathbf{z}_0)\right] + O(\varepsilon^2)\\ \Delta_2 &\overset{(35)}{=} \varepsilon\,\mathbb{E}_q\left[\left[I+\nabla_{\mathbf{z}_0}\boldsymbol{v}(\mathbf{z}_0)\right]^{-1}\cdot\nabla_{\mathbf{z}_0}\boldsymbol{\omega}(\mathbf{z}_0)\right] + O(\varepsilon^2)\end{aligned}$$

$$\Delta_2 = \varepsilon\, \mathbb{E}_q \left[[I + \nabla_{\mathbf{z}_0} \boldsymbol{v}(\mathbf{z}_0)]^{-1} \cdot \left\langle \nabla_{\mathbf{z}_0} \kappa(\mathbf{z}_0, \cdot), \boldsymbol{\omega} \right\rangle_{\mathcal{H}^d} \right] + O(\varepsilon^2)$$
$$\Delta_2 = \varepsilon \left\langle \mathbb{E}_q \left[[I + \nabla_{\mathbf{z}_0} \boldsymbol{v}(\mathbf{z}_0)]^{-1} \cdot \nabla_{\mathbf{z}_0} \kappa(\mathbf{z}_0, \cdot) \right], \boldsymbol{\omega} \right\rangle_{\mathcal{H}^d} + O(\varepsilon^2)$$

Therefore, we have

$$F[\boldsymbol{v} + \varepsilon \boldsymbol{\omega}] - F[\boldsymbol{v}] = \varepsilon \left\langle \mathbb{E}_q \left[\kappa(\mathbf{z}_0, \cdot)\, \nabla_{\mathbf{z}_0} \log p_*(\mathbf{z}_0 + \boldsymbol{v}(\mathbf{z}_0)) \right] + \mathbb{E}_q \left[[I + \nabla_{\mathbf{z}_0} \boldsymbol{v}(\mathbf{z}_0)]^{-1}\, \nabla_{\mathbf{z}_0} \kappa(\mathbf{z}_0, \cdot) \right], \boldsymbol{\omega} \right\rangle_{\mathcal{H}^d} + O(\varepsilon^2)$$

and since $F[\boldsymbol{v} + \varepsilon \boldsymbol{\omega}] - F[\boldsymbol{v}] = \varepsilon \langle \nabla_{\boldsymbol{v}} F[\boldsymbol{v}], \boldsymbol{\omega} \rangle_{\mathcal{H}^d} + O(\varepsilon^2)$, the functional gradient is given by

$$\nabla_{\boldsymbol{v}} F[\boldsymbol{v}] = -\mathbb{E}_q \left[\kappa(\mathbf{z}_0, \cdot)\, \nabla_{\mathbf{z}_0} \log p_*(\mathbf{z}_0 + \boldsymbol{v}(\mathbf{z}_0)) + [I + \nabla_{\mathbf{z}_0} \boldsymbol{v}(\mathbf{z}_0)]^{-1}\, \nabla_{\mathbf{z}_0} \kappa(\mathbf{z}_0, \cdot) \right]$$

Setting $\boldsymbol{v} = \mathbf{0}$ gives us the desired result

$$\boxed{\left(\nabla_{\boldsymbol{v}} F[\boldsymbol{v}]\right)\Big|_{\boldsymbol{v}=\mathbf{0}} = -\mathbb{E}_q \left[\mathcal{A}_{p_*} \kappa(\mathbf{z}_0, \cdot) \right] = -\mathbb{E}_q \left[\kappa(\mathbf{z}_0, \cdot)\, \nabla_{\mathbf{z}_0} \log p_*(\mathbf{z}_0) + \nabla_{\mathbf{z}_0} \kappa(\mathbf{z}_0, \cdot) \right]}$$

Therefore, we can take a step and then treat the transformed particles as $\mathbf{z}_0$ and repeat until convergence.

Appendix C. Ensemble Markov Chain Monte Carlo with Teleporting Walkers

Algorithm 12: Ensemble Markov Chain Monte Carlo with Teleporting Walkers [LWZ22]

Input: $\pi(z)$ target distribution (possibly unnormalized), p_r proposal distribution, fixed N number of random walkers
Output: A mean-field variational density $q(\mathbf{y}) = \prod_{j=1}^m q_j(y_j)$
Initialize: N random walkers with initial positions $\mathbf{z} = (z_1, \ldots, z_N) \in \mathcal{Z}^N$

for $j \in \{1, \ldots, m\}$ **do**

Sample teleporter's arrival index $j \in \{1, 2, \ldots, N\}$ uniformly at random and sample $y \sim p_r(z_j \to y)$

Sample teleporter's starting index $i \in \{1, 2, \ldots, N\}$ (possibly equal to j) according to importance weights

$$w_i(\mathbf{z}, y) := \frac{p_r(y \to z_i) + \sum_{k \neq i}^N p_r(z_k \to z_i)}{\pi(z_i)} \Big/ Z(\mathbf{z}, y) \quad \text{where} \quad Z(\mathbf{z}, y) := \sum_{l=1}^N \frac{p_r(y \to z_l) + \sum_{k \neq l}^N p_r(z_k \to z_l)}{\pi(z_l)}$$

Construct candidate $\mathbf{z}'$: set $\mathbf{z}' = (z_1', \ldots, z_N') \leftarrow \mathbf{z} = (z_1, \ldots, z_N)$ and teleport walker by overwriting $z_i' \leftarrow y$

Compute acceptance probability and acceptance ratio:

$$a(\mathbf{z}') = \min(r, 1) \quad \text{where} \quad r = \frac{\Pi(\mathbf{z}')\, p_r(\mathbf{z}' \to \mathbf{z})}{\Pi(\mathbf{z})\, p_r(\mathbf{z} \to \mathbf{z}')} = \frac{Z(\mathbf{z}, y)}{Z(\mathbf{z}', z_i)}$$

if $U \leq a(\mathbf{z}')$ where $U \sim \text{Uniform}([0, 1])$ **then**

Accept candidate $\mathbf{z}'$, set $\mathbf{z}_{t+1} = \mathbf{z}'$.

else

Reject candidate $\mathbf{z}'$, set $\mathbf{z}_{t+1} = \mathbf{z}$.

end

end

return $q(\mathbf{y})$

[35] $\nabla_{\mathbf{A}} \log\det \mathbf{A} = \mathbf{A}^{-1}$ for symmetric positive definite matrix $\mathbf{A}$ from Section A.4.1 in *Convex Optimization* [BV09]. Here, $I + \nabla_{\mathbf{z}_0} \boldsymbol{v}(\mathbf{z}_0)$ is symmetric positive definite as $I + \nabla_{\mathbf{z}_0} \boldsymbol{v}(\mathbf{z}_0) = \mathbf{F}$ and $J = \det \nabla_{\mathbf{z}_0} \boldsymbol{\chi} > 0$ by conservation of mass.

Quiz Practice

Bayes' Central Limit Theorem

Consider the Beta, Binomial Model where we have

- Prior: $\Theta \sim \text{Beta}(\alpha, \beta)$
- Likelihood: $S = s \mid \Theta \sim \text{Binomial}(n, \theta)$
- Posterior: $\Theta \mid S = s \sim \text{Beta}(s + \alpha, n - s + \beta)$

$$\text{Posterior}(\theta) \propto \text{Prior}(\theta) \cdot \text{Lik}(\theta; s) \quad \Longrightarrow \quad \log(\mathbb{P}(\Theta \mid S = s)) \propto \log(\mathbb{P}(\Theta)) + \log(\text{Lik}(\Theta; s))$$

$$\begin{aligned} \text{Lik}(\theta; s) \propto \theta^s (1-\theta)^{n-s} \quad &\Longrightarrow \quad \log(\text{Lik}(\Theta; s)) \propto s \log(\theta) + (n-s)\log(1-\theta) \\ &\log(\text{Lik}(\Theta; s)) \propto n\left[\frac{s}{n}\log(\theta) + \frac{n-s}{n}\log(1-\theta)\right] \end{aligned}$$

Let $\hat{\Theta}_{\text{MLE}}(s) = \theta^\star$. We are aiming to show that as $n \to \infty$, the posterior will concentrate at $\theta^\star$. We have

$$\log(\text{Lik}(\Theta; s)) \propto n\Big[\underbrace{\frac{s}{n}}_{\hat{\Theta}_{\text{MLE}}(s)=\theta^\star}\log(\theta) + \underbrace{\frac{n-s}{n}}_{1-\theta^\star}\log(1-\theta)\Big] = n\Big[\theta^\star \log(\theta) + (1-\theta^\star)\log(1-\theta)\Big]$$

Taken together, we have

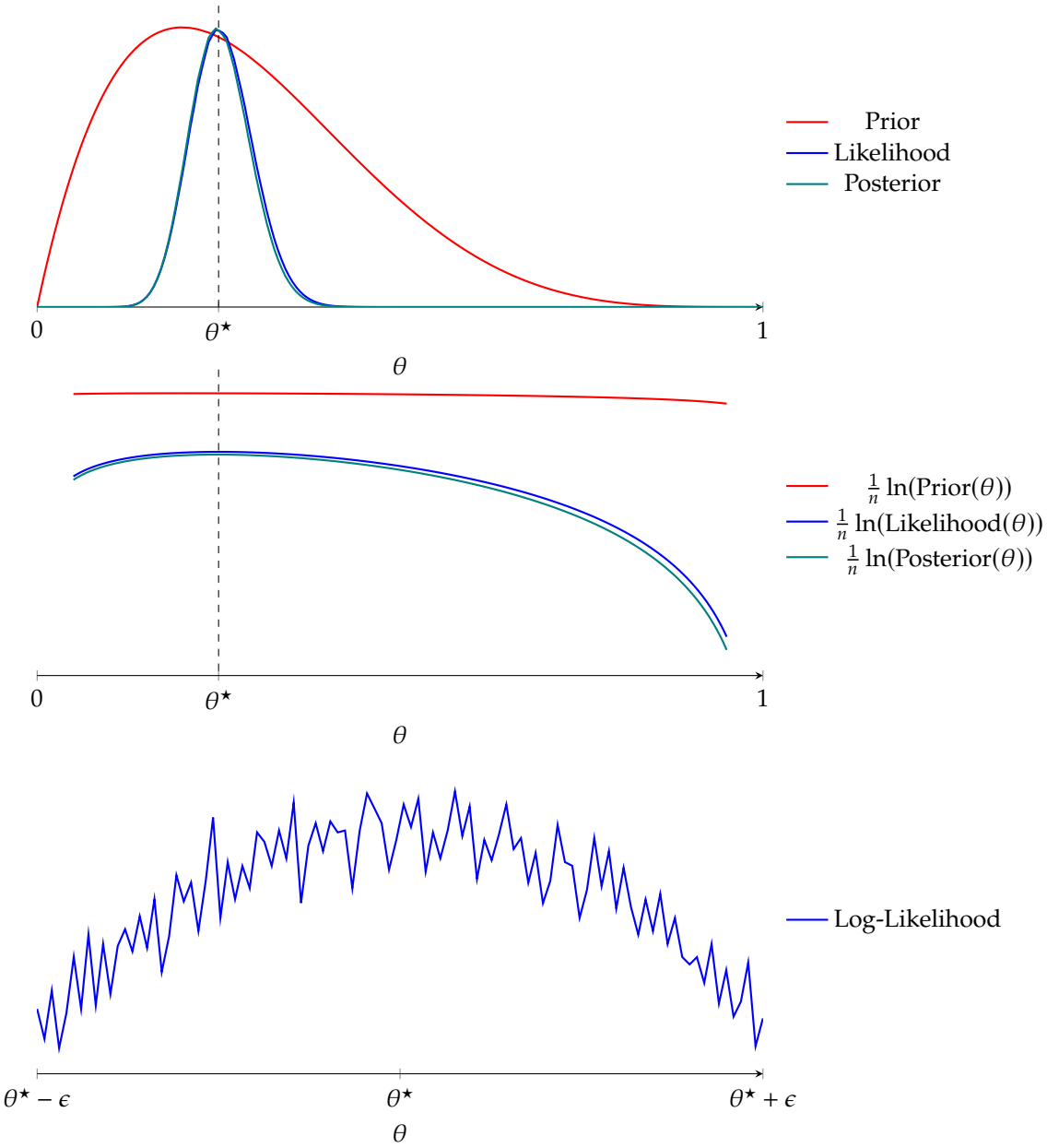
$$\begin{aligned} \frac{1}{n}\log(\mathbb{P}(\Theta \mid S = s)) &\propto \frac{1}{n}\log(\mathbb{P}(\Theta)) + \frac{1}{n}\log(\text{Lik}(\Theta; s)) \\ \frac{1}{n}\log(\mathbb{P}(\Theta \mid S = s)) &\propto \frac{1}{n}\log(\mathbb{P}(\Theta)) + \theta^\star \log(\theta) + (1-\theta^\star)\log(1-\theta) \end{aligned}$$

We note that $\frac{1}{n}\log(\mathbb{P}(\Theta)) \to 0$ as $n \to \infty$, so therefore

$$\frac{1}{n}\log(\mathbb{P}(\Theta \mid S = s)) \propto \theta^\star \log(\theta) + (1-\theta^\star)\log(1-\theta) \quad \text{as} \quad n \to \infty.$$

Therefore, we take a Tayler Series expansion of the $\frac{1}{n}\log(\mathbb{P}(\Theta \mid S = s))$ to show that

$$\frac{1}{n}\log(\mathbb{P}(\Theta \mid S = s)) \propto -\frac{\partial^2}{\partial \theta^2}\Big(\log(\text{Lik}(\Theta; s))\Big)(\theta - \theta^\star)^2$$


Prior
Likelihood
Posterior
0
$\theta^\star$
1
θ
$\frac{1}{n}$ ln(Prior(θ))
$\frac{1}{n}$ ln(Likelihood(θ))
$\frac{1}{n}$ ln(Posterior(θ))
0
$\theta^\star$
1
θ

Log-Likelihood
$\theta^\star - \epsilon$
$\theta^\star$
$\theta^\star + \epsilon$
θ

Conjugate Models via Exponential Families

Definition 1 (Exponential Family). A random quantity $X \in \mathcal{X}$ is drawn from an exponential family with parameters $\theta \in \Theta$,

$$p(x;\theta) \propto f(x)\exp\left\{\vec{\phi}(\theta)^\top \vec{u}(x)\right\} = f(x)\exp\left\{\sum_{j=1}^{m}\phi_j(\theta)u_j(x)\right\}$$

where $f(\cdot) : \mathcal{X} \to \mathbb{R}^+$ is the base measure
$\vec{u}(\cdot) : \mathcal{X} \to \mathbb{R}^m$ are a set of features
$\vec{\phi}(\cdot) : \Theta \to \mathbb{R}^m$ are a set of natural parameters

and $Z(\theta) = \int_{x\in\mathcal{X}} f(x)\exp\left\{\vec{\phi}(\theta)^\top\vec{u}(x)\right\}dx$ is the partition function or normalization function

From Theorem 1, we have

$$p(x;\theta) = \frac{f(x)}{Z(\theta)}\exp\left\{\vec{\phi}(\theta)^\top\vec{u}(x)\right\} = \exp\left\{\log\big(f(x)\big) + \vec{\phi}(\theta)^\top\vec{u}(x) - \log(Z(\theta))\right\}$$

and an alternative form of Theorem 1 is that an exponential family is a set of distributions that can be written in the form

$$\begin{aligned}\mathcal{P} = \big\{\text{all } p(\cdot;\theta) \text{ for which } Z(\theta) < \infty\big\} &= \big\{\text{all } p(\cdot\,;\theta) \text{ such that } \log\big(p(\cdot\,;\theta)\big) \in \text{span}\left\{[\vec{u}(\cdot), 1]\right\} + \log\big(f(\cdot)\big)\big\}\\ &= \big\{\text{all } \phi_1(\theta)u_1(x) + \cdots + \phi_m(\theta)u_m(x) - \phi_0(\theta) \text{ where } \phi_0(\theta) = -\log(Z(\theta))\big\}\end{aligned}$$

Example 2 (Exponential distribution). Let $X \sim \text{Exp}(\lambda)$ such that $p(x;\lambda) \propto \exp\{-\lambda x\}\,\mathbb{1}\{x \geq 0\}$.

- $f(x) = \mathbb{1}\{x \geq 0\}$
- $\vec{\phi}(\lambda) = \lambda$
- $Z(\lambda) = \int_{x\geq 0}\exp\{-\lambda x\}\,dx = \frac{-1}{\lambda}\exp\{-\lambda x\}\Big|_{x=0}^{\infty} = \frac{1}{\lambda}$
- $m = 1$
- $\vec{u}(x) = x$
- $\log(Z(\lambda)) = -\log(\lambda)$

Example 3 (Binomial distribution). Let $X \sim \text{Bin}(n,p)$ such that

$$p(x;n,p) = \binom{n}{x}p^x(1-p)^{n-x}\,\mathbb{1}\{x \in \{0,1,\ldots,n\}\} = \binom{n}{x}\exp\left\{x\log(p) + (n-x)\log\big(1-p\big)\right\}\,\mathbb{1}\{x \in \{0,1,\ldots,n\}\}$$

$$p(x;n,p) = \binom{n}{x}\exp\left\{x\log\left(\frac{p}{1-p}\right) + n\log\big(1-p\big)\right\}\,\mathbb{1}\{x \in \{0,1,\ldots,n\}\}$$

- $f(x) = \binom{n}{x}\mathbb{1}\{x \in \{0,1,\ldots,n\}\}$
- $\vec{\phi}(p) = \log\left(\frac{p}{1-p}\right)$
- $Z(p) = (1-p)^{-n}$
- $m = 1$
- $\vec{u}(x) = x$
- $\log\big(Z(p)\big) = -n\log\big(1-p\big)$

Example 4 (Normal distribution). Let $X \sim \mathcal{N}(\mu,\sigma^2)$ such that

$$p(x;\mu,\sigma^2) = \frac{\mathbb{1}\{x\in\mathbb{R}\}}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} = \mathbb{1}\{x\in\mathbb{R}\}\,\exp\left\{-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{\mu^2}{2\sigma^2} + \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)\right\}$$

$$p(x;\mu,\sigma^2) = \mathbb{1}\{x\in\mathbb{R}\}\,\exp\left\{-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \left(\frac{\mu^2}{2\sigma^2} + \log\left(\sqrt{2\pi\sigma^2}\right)\right)\right\}$$

- $f(x) = \mathbb{1}\{x\in\mathbb{R}\}$
- $\vec{\phi}(\mu,\sigma) = \begin{bmatrix}\frac{\mu}{\sigma^2} & -\frac{1}{2\sigma^2}\end{bmatrix}^\top$
- $Z(\mu,\sigma) = \sqrt{2\pi\sigma^2}\exp\left(\frac{\mu^2}{2\sigma^2}\right)$
- $m = 2$
- $\vec{u}(x) = \begin{bmatrix}x & x^2\end{bmatrix}^\top$
- $\log\big(Z(\mu,\sigma)\big) = \frac{\mu^2}{2\sigma^2} + \log\left(\sqrt{2\pi\sigma^2}\right) = \frac{1}{2}\frac{\mu^2}{\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$

Definition 5 (Sufficient Statistics). Given $X^{(n)} = \{X_1, \ldots, X_n\} \overset{i.i.d.}{\sim} p(x;\theta)$, the set of functions $\left\{s_j(X^{(n)})\right\}_{j=1}^{m}$ are sufficient if:

- Θ are conditional independent of $X^{(n)}$ given $\vec{s}(X^{(n)}) = [s_1(X^{(n)}), \ldots, s_m(X^{(n)})]$. Equivalently...
  - (i.) $\Theta \mid X^{(n)} \sim \Theta \mid X^{(n)'}$ if $\vec{s}(X^{(n)}) = \vec{s}(X^{(n)'})$.
  - (ii.) $\text{Lik}(\theta; X^{(n)}) = \text{Lik}(\theta; X^{(n)'})$ if $\vec{s}(X^{(n)}) = \vec{s}(X^{(n)'})$ at all $\theta \in \Theta$.
  - (iii.) all of the information in $X^{(n)}$ about θ is contained in $\vec{s}(X^{(n)})$, i.e., $I(\theta; X^{(n)}) = I(\theta; \vec{s}(X^{(n)}))$.

Let's apply Theorem 5 to the exponential family. We have

$$\text{Lik}(\theta; X^{(n)}) = \prod_{j=1}^{n} p(X_j;\theta) = \prod_{j=1}^{n} f(X_j) \exp\left\{\vec{\phi}(\theta)^\top \vec{u}(X_j) - \log(Z(\theta))\right\} = \exp\left\{\sum_{j=1}^{n} \vec{\phi}(\theta)^\top \vec{u}(X_j) - n\log(Z(\theta))\right\}$$

$$\implies \frac{1}{n}\log\Big(\text{Lik}(\theta; X^{(n)})\Big) = \frac{1}{n}\sum_{j=1}^{n} \vec{\phi}(\theta)^\top \vec{u}(X_j) - \log(Z(\theta)) = \vec{\phi}(\theta)^\top \left(\frac{1}{n}\sum_{j=1}^{n} \vec{u}(X_j)\right) - \log(Z(\theta))$$

$$\implies \vec{s}(X^{(n)}) = \frac{1}{n}\sum_{j=1}^{n} \vec{u}(X_j) \quad \text{are sample averages of the features } \vec{u}(X_j)$$

Building Conjugate Models for/from Exponential Families

Suppose that we have a likelihood of the form

$$p(y \mid \theta) \propto f(y) \exp\left\{\vec{\phi}(\theta)^\top \vec{u}(y)\right\}$$

Then we have $\mathbb{P}[Y^{(n)} = y^{(n)} \mid \theta] = \text{Lik}(\theta; Y^{(n)} = y^{(n)})$ is of the form

$$\text{Lik}(\theta; Y^{(n)} = y^{(n)}) = \exp\left\{\vec{\phi}(\theta)^\top \sum_{j=1}^{n} \vec{u}(y_j) - n\log(Z(\theta)) + \underbrace{\sum_{j=1}^{n} \log(f(y_j))}_{\text{const in } \theta\text{; ignore for posterior}}\right\} \propto \exp\left\{n\left[\vec{\phi}(\theta)^\top \vec{s}(y_j) - \log(Z(\theta))\right]\right\}$$

$$\implies \log\Big(\text{Lik}(\theta; Y^{(n)} = y^{(n)})\Big) = n\left[\vec{\phi}(\theta)^\top \vec{s}(y_j) - \log(Z(\theta))\right]$$

If we have a prior of the form $\Theta \sim p_\Theta(\theta) \in \mathcal{P}$, then we have

$$p_{\Theta \mid Y^{(n)} = y^{(n)}}(\theta) \propto p_\Theta(\theta) \cdot \text{Lik}(\theta; Y^{(n)} = y^{(n)})$$

$$\implies \frac{1}{n}\log\Big(p_{\Theta \mid Y^{(n)} = y^{(n)}}(\theta)\Big) \propto \frac{1}{n}\log\big(p_\Theta(\theta)\big) + \vec{\phi}(\theta)^\top \vec{s}(y_j) - \log(Z(\theta))$$

For conjugacy, we need $p_{\Theta \mid y^{(n)}}(\theta)\mathcal{P}$ for any $y^{(n)}$. This is equivalent to the following:

- need $\exp\left\{n\frac{1}{n}\log\Big(p_{\Theta \mid y^{(n)}}(\theta)\Big)\right\} \in \mathcal{P}$ for all $y^{(n)}$
- need $\frac{1}{n}\log\Big(p_{\Theta \mid y^{(n)}}(\theta)\Big) \in \mathcal{Q}$ for all $y^{(n)}$ where $\mathcal{Q} = \big\{\log(p) \mid p \in \mathcal{P}\big\}$

Ok so here is our idea:

$$\begin{aligned}
\text{Given:} \quad & \mathcal{L} = \left\{p(y;\theta) \propto f(y)\exp\left\{\vec{\phi}(\theta)^\top \vec{u}(y)\right\}\right\} \\
\text{choose:} \quad & \mathcal{P} = \left\{p(\theta, \vec{\alpha}, \beta) \propto g(\theta,\beta)\exp\left\{\vec{\alpha}^\top \vec{\phi}(\theta) + \alpha_0 \log(Z_Y(\theta))\right\}\right\} \\
\text{then:} \quad & \log\Big(p_{\Theta \mid y^{(n)}}(\theta)\Big) = \log\big(g(\theta,\beta)\big) + \vec{\alpha}^\top \vec{\phi}(\theta) + n\vec{s}(y^{(n)})^\top \vec{\phi}(\theta) - (1-\alpha_0)\log(Z_Y(\theta)) \\
& \log\Big(p_{\Theta \mid y^{(n)}}(\theta)\Big) = \left[\vec{\alpha} + n\vec{s}(\theta), \quad -(1-\alpha_0)\right]^\top \left[\vec{\phi}(\theta) \quad \log(Z_Y(\theta))\right] + \log\big(g(\theta,\beta)\big)
\end{aligned}$$

Sampling Procedures

Markov Chain Monte Carlo (MCMC)

Metropolis Algorithm

Algorithm 13: Metropolis Algorithm

Input: unnormalized target distribution $\tilde{p}_*$ and symmetric proposal distribution p_r
Initialize: Initialize empty set of samples $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**
  Draw a candidate sample from the proposal distribution: $Y \sim p_r(X_t \to Y)$
  Compute acceptance ratio and acceptance probability: $r = \dfrac{\tilde{p}_*(Y)}{\tilde{p}_*(X_t)}$ and $a(Y) = \min(r, 1)$
  Draw $U \sim \text{Uniform}([0, 1])$
  if $U \leq a(Y)$ **then**
    Accept candidate Y, set $X_{t+1} = Y$, and append Y to $\mathcal{S}$
  else
    Reject candidate Y, set $X_{t+1} = X_t$, and discard Y
  end
end
return $\mathcal{S}$

Metropolis-Hastings Algorithm

Algorithm 14: Metropolis-Hastings Algorithm

Input: unnormalized target distribution $\tilde{p}_*$ and not necessarily symmetric proposal distribution p_r
Initialize: Initialize empty set of samples $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**
  Draw a candidate sample from the proposal distribution: $Y \sim p_r(X_t \to Y)$
  Compute acceptance ratio and acceptance probability: $r = \dfrac{\tilde{p}_*(Y)\, p_r(Y \to X_t)}{\tilde{p}_*(X_t)\, p_r(X_t \to Y)}$ and $a(Y) = \min(r, 1)$
  Draw $U \sim \text{Uniform}([0, 1])$
  if $U \leq a(Y)$ **then**
    Accept candidate Y, set $X_{t+1} = Y$, and append Y to $\mathcal{S}$
  else
    Reject candidate Y, set $X_{t+1} = X_t$, and discard Y
  end
end
return $\mathcal{S}$

Gradient Based Sampling

Metropolis-Adjusted Langevin Algorithm

Algorithm 15: Metropolis-Adjusted Langevin Algorithm

Input: unnormalized target distribution $\tilde{p}_*$ with log-gradient $\nabla_x \log\big(\tilde{p}_*(x)\big)$. Note that $\nabla_x \log\big(\tilde{p}_*(x)\big) = \nabla_x \log\big(p_*(x)\big)$.
Initialize: Initialize empty set of samples $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**
  Draw candidate via noisy gradient ascent:

$$Y = X_t + \underbrace{\left(\nabla_x \log\big(\tilde{p}_*(x)\big)\Big|_{x=X_t}\right)\Delta t}_{\text{gradient ascent with } \Delta t \text{ step size}} + \underbrace{\sqrt{t}\sigma\xi}_{\text{noise}} \quad \text{where} \quad \sqrt{t}\sigma\xi \sim \mathcal{N}(0, \sigma^2 \Delta t I) \quad \text{such that}$$

$$Y \sim \mathcal{N}\left(X_t + \left(\nabla_x \log\big(\tilde{p}_*(x)\big)\Big|_{x=X_t}\right)\Delta t \ , \ \sigma^2 \Delta t I\right)$$

  Compute acceptance ratio and acceptance probability: $r = \dfrac{\tilde{p}_*(Y)\, p_r(Y \to X_t)}{\tilde{p}_*(X_t)\, p_r(X_t \to Y)}$ and $a(Y) = \min(r, 1)$
  Draw $U \sim \text{Uniform}([0, 1])$
  if $U \leq a(Y)$ **then**
    Accept candidate Y, set $X_{t+1} = Y$, and append Y to $\mathcal{S}$
  else
    Reject candidate Y, set $X_{t+1} = X_t$, and discard Y
  end
end
return $\mathcal{S}$

Hamiltonian Monte Carlo Algorithm

Algorithm 16: Hamiltonian Monte Carlo Algorithm

Input:

- Unnormalized joint target distribution $\tilde{p}_*(x, \phi) = \tilde{p}_*(x)\ p_*(\phi)$
- Target distribution log-gradient $\nabla_x \log\big(\tilde{p}_*(x)\big)$. Note that $\nabla_x \log\big(\tilde{p}_*(x)\big) = \nabla_x \log\big(p_*(x)\big)$
- Total mechanical energy, $E(x, \phi) := U(x) + \frac{1}{2}\phi^\top M^{-1}\phi$ where potential $U(x) := -\log\big(\tilde{p}_*(x)\big)$
- Iterations T, leapfrog steps L

Initialize: Initialize empty set of samples $\mathcal{S} \leftarrow \emptyset$

for $t = 1, 2, \ldots, T$ **do**
  Sample candidate by simulating Hamiltonian dynamics:
  Draw $\phi_0 \sim \mathcal{N}(0, M)$ and set $\phi(s) \leftarrow \phi_0$ and $X(s) \leftarrow X_t$
  for $s = 1, 2, \ldots, L$ **do**
    Simulating Hamiltonian dynamics via leapfrog method

$$\phi(s + \Delta s) \approx \phi(s) + \frac{\mathrm{d}}{\mathrm{d}s}\phi(s)\Delta s \qquad \frac{\mathrm{d}}{\mathrm{d}s}X(s) = \nabla_\phi E(x, \phi) = \nabla_\phi \left(U(x) + \frac{1}{2}\phi^\top M^{-1}\phi\right) = M^{-1}\phi$$

$$X(s + \Delta s) \approx X(s) + \frac{\mathrm{d}}{\mathrm{d}s}X(s)\Delta s \qquad \frac{\mathrm{d}}{\mathrm{d}s}\phi(s) = -\nabla_x E(x, \phi) = \nabla_\phi \left(-\log\big(\tilde{p}_*(x)\big) + \frac{1}{2}\phi^\top M^{-1}\phi\right) = \nabla_x \log\big(\tilde{p}_*(x)\big)\Big|_{x=X(s)}$$

  end
  Set candidate to end of Hamiltonian dynamics: $Y \leftarrow X_L$
  Compute acceptance ratio and acceptance probability: $r = \dfrac{\tilde{p}_*(Y)\ p_*(\phi(L))}{\tilde{p}_*(X_t)\ p_*(\phi_0)}$ and $a(Y) = \min(r, 1)$
  Draw $U \sim \text{Uniform}([0, 1])$
  if $U \leq a(Y)$ **then**
    Accept candidate Y, set $X_{t+1} = Y$, and append Y to $\mathcal{S}$
  else
    Reject candidate Y, set $X_{t+1} = X_t$, and discard Y
  end
end
return $\mathcal{S}$

Model Checking

Coverage Checking

Algorithm 17: Coverage Checking

Sample truth: $\left\{\theta_j^{\text{true}}\right\}_{j=1}^{n} \overset{\text{i.i.d.}}{\sim} p_\Theta$ such that we have summary statistics $\left\{S_j^{\text{true}} = (\theta_j^{\text{true}})\right\}_{j=1}^{n}$;

Sample data given truth: $\left\{Y_j \mid \theta_j^{\text{true}} \sim p_{Y|\Theta=\theta_j^{\text{true}}}\right\}_{j=1}^{n}$;

for $j = 1, 2, \ldots, n$ **do**

  Perform approximate posterior inference given $Y = Y_j$, i.e., collect m samples from posterior, e.g. using MCMC:

$$\left\{\theta_{j,k}\right\}_{k=1}^{m} \sim \tilde{p}_{\Theta|Y=Y_j} \quad \text{such that} \quad \left\{S_{j,k} = s(\theta_{j,k})\right\}_{k=1}^{m} \sim \tilde{p}_{S|Y=Y_j}$$

  Build approximate credible interval p%: $\hat{I}_p\left(\left\{S_{j,k}\right\}_{k=1}^{m}\right)$, e.g., `ci = np.percentile(samples, [5,95])`

  Check coverage: is the truth S_j^{true} in our approximate credible interval $\hat{I}_p\left(\left\{S_{j,k}\right\}_{k=1}^{m}\right)$?

end

Let $\hat{\tilde{p}} = \frac{1}{n}\sum_{j=1}^{n} \mathbb{1}\left\{S_j^{\text{true}} \in \hat{I}_p\left(\left\{S_{j,k}\right\}_{k=1}^{m}\right)\right\}$

if $\tilde{p}_{\Theta|Y} = p_{\Theta|Y}$ **then** `/* If posterior inference is exact */`

  $H_0 : p_{\text{test}} = \frac{X}{n}$ where $X \sim \text{Binomial}(n, p)$

end

return $\mathcal{S}$

Posterior Predictive Checking

Algorithm 18: Posterior Predictive Checking

Leave-One-Out Cross-Validation for Predictive Skill

Algorithm 19: Leave-One-Out Cross-Validation

Input: Data $\{y_i\}_{i=1}^n$, model $g(y \mid \theta)$ with prior $p(\theta)$

for $i = 1, 2, \ldots, n$ **do**

Define training set $y^{\text{train}} = \{ y_j : j \neq i\}$ and hold out y_i^{holdout}

Sample from posterior predictive distribution to estimate Skill:

$$\text{Skill}(g \mid f) = \mathop{\mathbb{E}}_{Y \sim f}[\log(g(Y))] = \int_{\mathcal{Y}} f(y)\, \log(g(y))\, dy$$

$$g_i(y) \;=\; \int g(y_i^{\text{holdout}} \mid \theta)\, p(\theta \mid y^{\text{train}})\, d\theta \approx \frac{1}{S} \sum_{s=1}^{S} g(y_i^{\text{holdout}} \mid \theta^s) \quad \text{where} \quad \theta^s \overset{\text{i.i.d.}}{\sim} p(\theta \mid y^{\text{train}})$$

Evaluate log-skill on the held-out point: $S_i \;=\; \log\, g_i(y_i)$

end

Compute overall LOO-Skill: $\text{LOO_Skill}(g) \;=\; \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} S_i \;=\; \dfrac{1}{n} \sum_{i=1}^{n} \log\, g_i(y_i)$

To compare two models g and g':

Compute $\Delta \;=\; \text{LOO_Skill}(g) \;-\; \text{LOO_Skill}(g')$.

if $\Delta > 0$ **then**

Conclude g is closer to the truth (smaller $D_{KL}(f \parallel g)$)

else

Conclude g' is closer to the truth

end

Quiz 1: Definitions & Beta Binomial Model

Full Name and SID: _____

Question 1 [4 points]. Suppose that Y is an observable that is jointly distributed with some unknown Θ. Suppose we observe $Y = y$. Then, select all true statements below:

- ☐ The prior at θ is the marginal probability that $\Theta = \theta$.
- ☐ The likelihood of θ is the conditional probability that $Y = y$ given $\Theta = \theta$.
- ☐ The posterior at θ equals the product of the prior and the likelihood, each evaluated at θ.
- ☐ The joint at (θ, y) equals (the posterior probability that $\Theta = \theta$ given $Y = y$) $\times$ (the evidence evaluated at y).

Answers: (a) True. (b) True. (c) False. Missing the normalization (the posterior is proportional to this product). (d) True. This says that $P(\Theta = \theta, Y = y) = P(\Theta = \theta | Y = y)P(Y = y)$.

Question 2 [5 points]. Suppose that X is drawn from a distribution that depends on an unknown parameter Θ. Suppose that Θ takes on values $\{1, 2, 3, 4\}$ with probabilities $P(\Theta = \theta) = \{0.6, 0.2, 0.1, 0.1\}$. Suppose that we observe $X = x$ and $P(X = x|\Theta = \theta) = \{0.2, 0.4, 0.3, 0.1\}$. Select all true statements below:

- ☐ The Maximum Likelihood Estimator, $\hat{\theta}_{\text{MLE}}$, equals 2 since $P(X = x|\theta)$ is maximized by $\theta = 2$.
- ☐ The Maximum a Posteriori Estimator, $\hat{\theta}_{\text{MAP}}$, equals 1 since $P(\Theta = \theta)$ is maximized at $\Theta = 1$.
- ☐ The posterior odds that $\Theta = \hat{\theta}_{\text{MAP}}$ rather than $\Theta = \hat{\theta}_{\text{MLE}}$ are $3/2 = 1.5$.
- ☐ The evidence equals 0.24.
- ☐ The posterior probability of sampling $\Theta = \hat{\theta}_{\text{MAP}}$ is 0.12.

Answers: (a) True. (b) False. The estimator is correct but the reasoning is wrong. The MAP is based on maximzing the posterior not the prior. (c) True. (d) True. (b) False. This is the joint probability. The posterior is $0.12/0.24 = 0.5$.

Full Name and SID: _____

Question 3 [3 points]. Select all true statements below regarding Bayesian inference:

- ☐ The posterior probability of the MAP estimator can be used as a standard for model checking (i.e. observing that the MAP estimator has a very small posterior probability is evidence against joint model over the unknowns and the observation).
- ☐ If the posterior odds that $\Theta = \theta$ vs $\Theta = \theta'$ equals $2/3$ and the prior odds equaled $1/2$, then the observed data must be 6 times more likely under θ than θ'.
- ☐ If (Θ, Y) are drawn jointly, then, the posterior variance in Θ after observing Y is expected to be (averaging over Y) less than the prior variance in Y.

Answers: (a) False. The posterior cannot be used to validate the model directly. This would not be a reasonable standard for evidence against the model, as it simply means that the unknown is not well resolved given the available observations. (b) False. The posterior odds equal the likelihood ratio times the prior odds. Solving would give a likelihood ratio of $4/3$. (c) True.

Question 4 [4 points]. Suppose that $\Theta \sim \text{Beta}(\alpha, \beta)$ and, $S|\Theta = \theta \sim \text{Binomial}(n, \theta)$. Then, complete each sentence below.

- The likelihood function is proportional to a **beta** distribution with parameters $s + 1, n - s + 1$.
- The posterior is a **beta** distribution with parameters $s + \alpha, n - s + \beta$.
- The posterior predictive distribution is a **beta-binomial** distribution with parameters **?,** $s + \alpha$**,** $n - s + \beta$. *Ignore the ?. I did not give enough information to specify this parameter.*
- The posterior predictive distribution returns the probability of **observing a new outcome, or new data, or** $S' = s'$ given **the observed data, or,** $S = S$.

Question 5 [4 points]. Suppose that $\Theta \sim \text{Beta}(\alpha, \beta)$ and, $S|\Theta = \theta \sim \text{Binomial}(n, \theta)$. Then, select each true statement below:

- ☐ Replacing α and β with $\alpha + 5$ and $\beta + 5$ has the same effect as using α, β but observing 10 more trials, of which 5 were successes.
- ☐ Holding α, n and s fixed, while increasing β will increase the posterior probability that $\Theta > a$ for any $a \in [0, 1)$.
- ☐ The posterior mode lies between the prior mode and the MLE, approaches the MLE when $\alpha + \beta \gg n$ and approaches the prior mode when $n \gg \alpha + \beta$.
- ☐ For large n, the posterior variance decays at rate $\mathcal{O}(n^{-1})$ (that is, proportional to n^{-1}).

Answers: (a) True. (b) False. This statement would hold for increasing α, or for any upper bound. (c) False. The limiting cases are reversed. The stronger the prior, the closer the MAP estimator is to the prior mode. The more data is collected, the closer it gets to the MLE. (d) True.

Quiz 2: Consistency & Decision Theory

Full Name and SID: _____

Disclaimer: all subsequent questions, proceed "under mild regularity conditions." You may assume that the generic behavior under the unstated regularity conditions is the same as the behavior we observed in the Beta, Binomial model. Equivalently, you may assume that the regularity conditions assumed in BDA chapter 4.1 - 4.4 and Appendix B hold.

Question 1 [5 points]. Let $\hat{\theta}(y)$ denote an estimator for the unknown Θ given the observation $Y = y$. Select all true statements below.

- ☐ The estimator is unbiased (w.r.t. the sampling distribution) if $\mathbb{E}_{Y|\theta}[\hat{\theta}(Y)] = \mathbb{E}_{\Theta}[\Theta]$.
- ☐ An estimator is jointly unbiased if $\mathbb{E}_{\Theta,Y}[\hat{\theta}(Y) - \Theta] = 0$.
- ☐ An estimator is consistent if, given sequential data $Y^{(n)} = \{Y_1, Y_2, ..., Y_n\}$ drawn i.i.d. from $p_{Y|\theta}$, $\hat{\theta}(Y^{(n)})$ converges, in probability, to the true value of the unknown, as $n \to \infty$.
- ☐ All consistent estimators are jointly unbiased.
- ☐ The posterior mean estimator may be jointly biased.

Answers: (a) True. [1 pt] (b) True. [1 pt] (c) True. [1 pt] (d) False. For example, the MAP. [1 pt] (e) False. [1 pt]

Question 2 [2 points]. Suppose that $Y \sim q_Y$ for some true distribution q_Y. Pose a Bayesian model of the form $(\Theta, Y) \sim p_{\Theta,Y}$ by fixing a prior, p_Θ and a conditional distribution $Y|\Theta = \theta \sim p_{Y|\theta}$. Select the correct definition for a well-specified model:

- ◯ The Bayesian model is well-specified if the prior and likelihood define valid joint, and thus posterior, distributions.
- ◯ The Bayesian model is well specified if q_Y equals the marginal distribution for Y under the joint model, p_Y.
- ◯ The Bayesian model is well specified if q_Y equals the conditional distribution of Y under the joint model, for some parameter θ_*.

Answers: (c) [2 pts]. Give 1 pt for (b).

Question 3 [4 points]. Suppose that $Y^{(n)} = \{Y_1, Y_2, \ldots, Y_n\}$ where $Y_j \sim f_Y$ i.i.d. Adopt a joint model with an unknown parameter Θ, and assume $Y_j \sim p_{Y|\theta}$ if $\Theta = \theta$. Suppose that $f_Y = p_{Y|\theta_*}$ for some θ_* on the interior of the set of possible parameters.
Complete the bubbles below that best describe the asymptotic behavior of the posterior distribution

- In the limit as $n \to \infty$ the posterior distribution of $\Theta|Y^{(n)}$ is asymptotically **normal** [1 pt] **consistent** [0.5 pt],
- with mean θ_* [0.5 pt],
- and covariance proportional to n^{-1} [1 pt] $n^{-1/2}$ [0.5 pt] times the **the inverse of the Fisher information** [0.5 pt].
- The Fisher information equals $-\mathbb{E}_{Y\sim f_Y}[\textbf{(a)(b)}]$ where:
  - **(a)** is ∇^2_θ [0.5 pt], and
  - **(b)** is **the log-likelihood at** θ_* [0.5 pt] **the likelihood at** θ_* [0.25 pt].

Full Name and SID: _____

For questions 4 and 5, consider a decision problem with loss function $l(\theta, d(y))$ where $l(\theta, d(y))$ is the expected loss over possible outcomes X, incurred if an unknown takes the value θ, and we made the decision $d(y)$ based on some observed data $Y = y$. The outcomes X are drawn conditional on θ and $d(y)$.

Question 4 [3 points]. Match the definitions provided below to the three notions of risk available if (Θ, Y) are modeled as jointly random:

(a) $\mathbb{E}_{\Theta|Y=y}[l(\Theta, d(y))]$, **(b)** $\mathbb{E}_{Y|\Theta=\theta}[l(\theta, d(Y))]$, **(c)** $\mathbb{E}_{Y,\Theta}[l(\Theta, d(Y))]$

- The frequentist risk is **(b)** [1 pt] **(a)** [0.25 pt]
- The posterior risk is **(a)** [1 pt] **(b)** [0,25 pt]
- The Bayes risk is **(c)** [1 pt]

Question 5 [4 points]. Select each true statement below:

☐ The Bayes risk is only a function of the decision rule, loss function, and joint statistical model.

☐ The frequentist risk is a function of the sampled data Y.

☐ Averaging the frequentist risk over Θ must give the same answer as averaging the posterior risk over Y when Θ and Y are drawn from the joint model.

☐ Any rule that minimizes the posterior risk, over all possible sample data sets Y, will minimize the frequentist risk.

Answers: (a) True. [1 pt], (b) False. [1 pt] It is a function of the unknown. It cannot be a function of the data, since it averages over hypothetical resamples. (c) True. [1 pt], (d) False. [1 pt]

Question 6 [2 points]. Select each true statement below:

☐ All decision problems admit a decision rule that minimizes the frequentist risk over all possible unknowns θ.

☐ A decision rule is admissible, in the frequentist sense, if it there is no other rule that has smaller (or equal) frequentist risk for all θ, and smaller risk at some θ.

Answers: (a) False. [1 pt], (b) True. [1 pt]

Quiz 3: Normal Normal Model & Exponential Families

Full Name and SID: _____

Question 1 [5 points]. Consider a 1-dimensional normal, normal model: $x \sim \mathcal{N}(\hat{x}, \sigma_x^2)$ and $y = x + \zeta$ where $\zeta \sim \mathcal{N}(0, \sigma_y^2)$. Then, select all true statements below:

- ☐ The posterior distribution for $x|y$ is normal.
- ☐ The posterior standard deviation in x is $\sigma_{x|y}^2 = \sigma_x^2 + \sigma_y^2$.
- ☐ The signal-to-noise ratio, SNR, is σ_y^2/σ_x^2.
- ☐ The ratio of the posterior and prior variances is monotonically decreasing in the SNR, and is independent of the observation.
- ☐ The posterior mean, $\hat{x}_{|y}$ can be expressed as a convex combination of the prior mean, $\hat{x}$, and the observation, y, that converges to $\hat{x}$ as the SNR approaches 0.
- ☐ The posterior predictive distribution is *not* normal.

Answers: (a) True. [1.5 pt], (b) False. [1 pt], (c) False [0.5 pt], (d) True [0.5 pt], (e) True [1 pt], (f) False [0.5 pt]

Question 2 [2 points]. Select all true statements below regarding priors:

- ☐ All improper priors induce improper posteriors.
- ☐ Some improper priors may be derived as limits of proper distributions.
- ☐ All proper priors induce a proper posterior.

Answers: (a) False [1 pt], (b) True [0.5 pt], (c) True [0.5 pt]

Question 3 [3 points]. Select all true statements below regarding priors, transformations, and invariance:

- ☐ If f is an invertible, monotonic transformation, and $\Theta \sim \text{Unif}([a.b])$ then $f(\Theta) \sim \text{Unif}([f(a), f(b)])$.
- ☐ Highest density intervals are invariant to invertible, monotonic transformations $\Theta \to f(\Theta)$.
- ☐ Jeffery's invariance principle suggests selecting a prior $p(\theta) \propto I(\theta)^{1/2}$ where $I(\theta)$ is the Fisher information
- ☐ If θ is a scale parameter for an observable y, then the improper prior $\log(\theta) \sim \text{Uniform}((-\infty, \infty))$ ensures that the posterior density depends only on the ratio y/θ.

Answers: (a) False [1 pt], (b) False [0.5 pt], (c) True [1 pt], (d) True [0.5 pt]

Question 4 [2 points]. Let $\mathcal{F}$ denote a class of sampling distributions $p(y|\theta)$, and $\mathcal{P}$ denote a class of prior distributions $p(\theta)$. Then, select the correct definition below.

- ◯ $\mathcal{P}$ is conjugate for $\mathcal{F}$ if the posterior $p(\theta|y) \in \mathcal{F}$ for all priors $p(\theta) \in \mathcal{P}$, all possible observations y, and all conditional distributions $p(y|\theta) \in \mathcal{F}$.
- ◯ $\mathcal{P}$ is conjugate for $\mathcal{F}$ if the posterior $p(\theta|y) \in \mathcal{P}$ for all priors $p(\theta) \in \mathcal{P}$, all possible observations y, and all conditional distributions $p(y|\theta) \in \mathcal{F}$.

Answers: (b) [2 pt]

Full Name and SID: _____

Question 5 [4 points]. Suppose that $Y^{(n)} = \{Y_j\}_{j=1}^n$ are drawn i.i.d. from an exponential family:

$$p(y|\theta) \propto f(y)e^{\vec{\phi}(\theta)\cdot\vec{u}(y)} \tag{1}$$

with unknown parameters θ. Then, select all true statements below.

- ☐ f is the normalization (or, partition) function.
- ☐ $\vec{\phi}(\cdot) = [\phi_1(\cdot), \phi_2(\cdot), \ldots, \phi_m(\cdot)]$ are the natural parameters.
- ☐ The unknowns θ are conditionally independent of the data $Y^{(n)}$ given the sample averages of the feature functions.
- ☐ The sufficient statistics are sample averages of the natural parameters.

Answers: (a) False [1 pt], (b) True [1 pt], (c) True [1 pt], (d) False [1 pt]

Question 6 [2 points]. Consider a likelihood drawn from an exponential family of form (1). Then, complete the sentence below by selecting all terms that apply:

"The family of distributions $\mathcal{P}$ is a family of conjugate priors for $Y^{(n)}$ drawn i.i.d. according to (1) if it is posed as an exponential family in θ whose feature functions include ($\vec{\phi}$ [1 pt], $\log(Z)$ [0.5 pt]) for (**any** [0.5 pt]) base measures $g(\theta)$."

Question 7 [2 points]. Suppose that $Y^{(n)} = \{Y_j\}_{j=1}^n$ are drawn i.i.d. according to a gamma distribution of the form:

$$p(Y = y|\theta) = \frac{\theta_1^{\theta_2}}{\Gamma(\theta_2)} y^{\theta_2 - 1} e^{-\theta_1 y}. \tag{2}$$

where $\Gamma(\cdot)$ is the Gamma function. Write down, in exponential family form, a possible family of conjugate priors over $\theta_1 \in [0, 10]$, $\theta_2 \in [0, 3]$. You may use $\mathbf{1}$(insert condition) to denote an indicator function which equals 1 when its condition is satisfied (e.g. $\mathbf{1}(\theta_1 > 0, \theta_2 > 0)$).

Answer: The prior should take the form:

$$p(\theta; \alpha) \propto \mathbf{1}(\theta_1 \in [0, 10], \theta_2 \in [0, 3]) e^{\alpha_1(\theta_2 - 1) \pm \alpha_2\theta_1 + \alpha_0 \log(Z(\theta))}$$

where $Z(\theta) = \Gamma(\theta_2)\theta_1^{-\theta_2}$.

Quiz 4: Conjugate Models, de Finetti's Theorem, & Conditional Independence

Full Name and SID: _____

Question 1 [3 points]. True of False: All likelihoods admit a conjugate family of priors that can be expressed as an exponential family with features equal to the natural parameters and log partition of the likelihood. All such prior families admit a partition function known in closed form.

◯ True , ◯ False.

Answers: False. The partition function is not known in closed form for all exponential families. If it was, then the evidence would be known in closed form for all distributions, and we would be able to analytically integrate any nonnegative function. [3 pt]

Question 2 [3 points]. Suppose that $Y|x, \beta \sim \text{Poisson}(\lambda(x, \beta))$ where $\lambda(x, \beta) = \exp(\beta \cdot u(x))$ for some vector-valued feature function $u(x) = [1, u_1(x), x_2(x), \ldots, u_m(x)]$. Then, select all true statements below.

☐ The gamma family are conjugate priors for the regression coefficients β.

☐ The link function is the exponential.

☐ The link function is the logarithm.

☐ The number of feature functions may exceed the number of predictors (dimension of x).

☐ If the feature functions are linearly dependent, then, no matter the observed data, there will exist a nontrivial subspace of coefficients β that all share the same likelihood.

☐ It is possible to recover the regression coefficients β, given a data set of the form $Y^{(n)} = \{Y_j\}_{j=1}^n$ for sufficiently large n, using a consistent estimator.

Answers: (a) False [0.75 pt]. (b) False [0.25 pt]. (c) True [0.25 pt]. (d) True [0.5 pt]. (e) True [0.75 pt]. (f) False [0.5 pt] (give [0.25] for True)

Question 3 [2 points]. Suppose that $S|\beta, x \sim \text{Bin}(n, \theta(\beta, x))$ where $\theta(\beta, x) = \text{logistic}(\beta \cdot x)$. Suppose we observed a sequence of outcomes $\{(S_i, X_i) = (s_i, x_i)\}_{i=1}^n = (s^{(n)}, x^{(n)})$. Assuming that large $\beta \cdot x$ indicates a high success probability θ, select the correct form for the likelihood function below:

◯ $\text{Lik}(\beta|s^{(n)}, x^{(n)}) = \prod_{i=1}^n (1 + \exp(-\beta \cdot x_i))^{-s_i} (1 + \exp(+\beta \cdot x_i))^{-(n_i - s_i)}$

◯ $\text{Lik}(\beta|s^{(n)}, x^{(n)}) = \prod_{i=1}^n (1 + \exp(+\beta \cdot x_i))^{-s_i} (1 + \exp(-\beta \cdot x_i))^{-(n_i - s_i)}$

◯ $\text{Lik}(\beta|s^{(n)}, x^{(n)}) = \prod_{i=1}^n (1 + \exp(-\beta \cdot x_i))^{s_i} (1 + \exp(+\beta \cdot x_i))^{(n_i - s_i)}$

◯ $\text{Lik}(\beta|s^{(n)}, x^{(n)}) = \prod_{i=1}^n (1 + \exp(+\beta \cdot x_i))^{s_i} (1 + \exp(-\beta \cdot x_i))^{(n_i - s_i)}$

Answer: (a) [2 pts]. (b) [1 pt]. (c) [0.5 pt]

Question 4 [3 points]. Select all true statements below regarding exchangeability:

☐ A sequence of random variables $\{X_j\}_{j=1}^n$ is exchangeable if and only if it is i.i.d.

☐ A sequence of random events are exchangeable if the joint probability of observing any sequence of events does not depend on the order of the events in the sequence.

☐ A sequence of random variables $\{X_j\}_{j=1}^n$ associated with a sequence of units $j = 1, 2, ...n$ are exchangeable, or may be modeled as exchangeable, if, to our knowledge, the units are indistinguishable.

Answers: False [1 pt]. True [1 pt]. True [1 pt].

Full Name and SID: _____

Question 4 [3 points]. In the space below, state de Finetti's theorem as precisely as you can. You may assume that the random variables X_j take values in a complete space $\mathcal{X}$ where the theorem holds. Your answer should clearly state an equivalence between a sequence of random variables, under some specified conditions, and, the existence of a representation (sampling procedure) that would produce an identically distributed sequence.

Answer: An *infinite* [0.5 pt] sequence of random variables is *exchangeable* [1 pt] if *and only if* [0.25 pt]:

- its joint distribution may be represented as a *mixture of joint distributions for i.i.d. sequences* [1.25 pt], or, equivalently
- there exists a prior distribution p_Θ, and conditional sampling distribution $p_{Y|\Theta=\theta}$ such that, if $\Theta \sim_\Theta$, then $Y_j \sim p_{Y|\Theta}$ i.i.d., the sequence $\{Y_1, Y_2, \ldots\}$ has the same distribution as the sequence $\{X_1, X_2, \ldots\}$. (give [1 pt] for the construction broken into [0.5] for the prior, and [0.5] for the conditionally i.i.d. sampling, and [0.25 pt] for the equivalence in distribution.)

Question 5 [2 points]. Draw a graphical model that represents the conclusion of the theorem for a sequence of exchangeable random variables $\{X_j\}_{j=1}^{\cdots}$. Your graphical model should follow the convention used in class, arrows represent sampling conditional on the value of a variable assigned at the root of the arrow. Multiple arrows leaving a single root represent conditionally i.i.d. sampling, conditional on the value of the variables at the root.

Answer: The students should draw a graph with a single arrow pointing into a single node (optionally labeled Θ, or by whatever latent random variable they defined in Q 4) [1 pt], then, a sequence of arrows should point out of that node to a sequence of nodes (optionally: labeled $X_1, X_2, \ldots$) [1 pt].

Question 6 [4 points]. Consider a Bayesian mixture model where $\Phi \sim p_\Phi$, then $\Theta|\Phi \sim p_{\Theta|\Phi}$, then $Y|\Theta \sim p_{Y|\Theta}$ where only Y is observed. Complete each statement below:

- Φ and Y are *conditionally independent* [1 pt]
- The joint model over (Θ, Y) provided above is equivalent to sampling $\Theta \sim p_\Theta$ then $Y|\Theta \sim p_{Y|\Theta}$ where $p_\Theta(\theta)$ is $\int_{\text{all } \phi} p_{\Theta|\Phi}(\theta|\phi)p_\Phi(\phi)d\phi$ [1 pt].
- The model provided above is equivalent to a Bayesian model with the joint unknown (Φ, Θ) drawn from the joint prior $p_{\Phi,\Theta}(\phi, \theta)$ which is $p_\Phi(\phi)p_{\Theta|\Phi}(\theta|\phi)$ [1 pt].
- If we aim to estimate Φ from Y, then we could view the mixture model as a Bayesian model with prior p_Φ [0.5 pt] and likelihood formed by marginalizing $p_{Y|\Theta}$ [0.25 pt] over the nuisance parameter Θ [0.25 pt]. *Skip this one, give the everyone 1 pt for this part.*

Quiz 5: Laplace Approximation & Coordinate Ascent

Full Name and SID: _____

Question 1 [6 points]. Suppose that $(\Theta, Y^{(n)})$ are drawn jointly, where $\Theta \sim p_\Theta$ and $Y_j|\Theta \sim p_{Y|\Theta}$ i.i.d. Let $\hat{\theta}(y^{(n)})$ denote the function that returns the MAP estimator for Θ when $Y^{(n)} = y^{(n)}$. Complete each statement below ($\bigcirc$ = select one. $\square$ = select all that are true):

1. The MAP estimator is the argument maximizer of **the posterior.** [2 pt]
2. Let $f(\cdot)$ denote the function you selected to maximize in 1.1. To find the MAP estimator, it is common practice to optimize on $\pm\frac{1}{n}\log(f(\theta))$ [0.5 pt].
3. Assuming standard regularity conditions, the MAP estimator is **consistent** [0.75 pt] for a **well-specified** [0.75 pt] model.
4. If the space of possible Θ is finite, then using the MAP estimator is optimal in the sense that it minimizes **the Baye's risk** [0.5 pt] and **posterior risk given any observation** [0.5 pt] with respect to the **0/1 loss** [1 pt].

Question 2 [3 points]. Suppose that $(\Theta, Y^{(n)})$ are drawn jointly, where $\Theta \sim p_\Theta$ and $Y_j|\Theta \sim p_{Y|\Theta}$ i.i.d. Let $\hat{\theta}(y^{(n)})$ denote the function that returns the MAP estimator for Θ when $Y^{(n)} = y^{(n)}$. Assume that the usual regularity conditions hold. Then, select the correct definition for the Laplace approximation.

"The Laplace approximation to the posterior approximates the posterior with a multivariate normal distribution with:

- $\square$ mean θ_* and covariance $-(H[\theta_*])^{-1}$ where $H[\theta]$ is the Hessian of the log-posterior about θ."
- $\square$ mean $\hat{\theta}(y^{(n)})$ and covariance $-(H[\hat{\theta}(y^{(n)})])^{-1}$ where $H[\theta]$ is the Hessian of the log-posterior about θ."
- $\square$ mean θ_* and covariance $(nI[\theta_*])^{-1}$ where $I[\theta]$ is the Fisher information at θ."
- $\square$ mean $\hat{\theta}(y^{(n)})$ and covariance $(nI[\hat{\theta}(y^{(n)})])^{-1}$ where $I[\theta]$ is the Fisher information at θ."

Answer: (b) is correct [3 pt], (a) expands about the wrong location [1.5 pt], (d) uses the wrong covariance [1 pt], (c) is the asymptotic normal [0.5 pt]

Question 1 [3 points]. Select all essentially true statements below.

- $\square$ The asymptotic normal approximation is usually computable given the prior model, likelihood, and $Y^{(n)}$ for finite n.
- $\square$ The Laplace approximation is usually computable given the prior model, likelihood, $Y^{(n)}$ for finite n, a convergent optimizer, and a numerical method to estimate derivatives.
- $\square$ The Laplace approximation cannot be computed without first estimating the evidence.
- $\square$ The Laplace approximation is consistent (e.g. it converges, with probability 1 over sequences $Y^{(n)}$, in Hellinger distance, to the true posterior distribution, at rate $\mathcal{O}(n^{-1/2})$).
- $\square$ Sampling from the Laplace approximation typically gives accurate posterior estimates for variables that depend on the tails of the posterior distribution.

Answer: (a) is false, it requires the true value of the unknown [0.5 pt]. (b) is true [1 pt]. (c) is false [0.5 pt]. (d) is true [0.5 pt]. (e) is false [0.5 pt].

Full Name and SID: _____

Question 4 [2 points]. In the space below, give one example of a situation where the Laplace approximation does not hold. Either, pick an example where the approximation is inaccurate, despite large n, or, an example where the regularity conditions fail, so the Laplace approximation does not hold for any n. Specify in which sense the approximation fails in your example.

Answers: There are a variety of reasonable answers here. These include:

1. The true parameter value lies on the boundary of the space of unknowns, or very close to it.
2. The likelihood is flat along some directions, or very close to flat, and the prior is non-Gaussian (heavy-tailed or skewed).
3. The posterior is highly multimodal, where the likelihood only weakly prefers one mode to the others.

Question 5 [4 points]. Suppose that $(\Theta, Y^{(n)})$ are drawn jointly, where $\Theta \sim p_\Theta$ and $Y_j|\Theta \sim p_{Y|\Theta}$ i.i.d. Let $\hat{\theta}(y^{(n)})$ denote the function that returns the MAP estimator for Θ when $Y^{(n)} = y^{(n)}$.

Suppose that Θ is multivariate, and can be broken into two different blocks of unknowns, $\Theta = [X, Z]$ where X contains the unknowns we are primarily interested in estimating, and Z are nuisance parameters (e.g. latent variables in a hierarchical model).

- ☐ The joint MAP over X and Z is the choice of x and z that maximize the joint posterior over x and z given the data $y^{(n)}$.
- ☐ The marginal MAP over X and Z is the choice of x and z that maximize the joint posterior over x and z given the data $y^{(n)}$.
- ☐ The estimates returned for X under a joint MAP scheme may not match the estimates under a marginal scheme.
- ☐ The joint MAP can be recovered by iteratively optimizing over X over its expected log posterior, where the expectation is taken over Z conditional on the data and the current best guess at $\hat{x}_{\text{MAP}}$ (the current iterate).

Answers: (a) is true [1.25 pt]. (b) is false [1.25 pt]. (c) is true [1 pt]. (d) is false [0.5 pt].

Question 6 [2 points]. Write a pseudocode description for a coordinate-ascent algorithm for MAP estimation in the space below. Your algorithm should be written to apply to any Bayesian model of the form specified in Question 5. If you cannot write a general set of instructions, outline an example algorithm from the class and specify the model it maximizes.

Answer: Give full credit to any psuedo-code or algorithm description that iteratively optimizes over subsets of the unknowns, while holding the rest fixed, and that rotates the subsets as the iteration progresses. For example:

1. Initialize $\theta^{(0)}$
2. Fix a schedule $\{s(t)\}_{t=0}^{\cdots}$
3. For $t = 0$ until convergence:
   (a) Optimize the log-posterior with respect to θ_j for all $j \in s(t)$ holding all θ_j fixed if $j \notin s(t)$

Various schedules $\{s(t)\}_{t=0}^{\cdots}$ are acceptable (e.g., rotate through the unknowns one at a time).

Quiz 6: Importance & Rejection Sampling + MCMC

Full Name and SID: _____

Question 1 [4 points]. Suppose that (Θ, Y) are drawn jointly, where $\Theta \sim p_\Theta$ and $Y|\Theta \sim p_{Y|\Theta}$. Suppose that we observe $Y = y$. Suppose that Θ is multivariate, and can be broken into two different blocks of unknowns, $\Theta = [X, Z]$ where X contains the unknowns we are primarily interested in estimating, and Z are nuisance parameters (other unknowns in the model).

Complete each statement below regarding expectation-maximization (EM).

- Expectation-maximization algorithms seek the **marginal** [1 pt] MAP estimator for X, **marginalizing** [0.5 pt] over Z.
- The maximization step in EM maximizes an objective $Q(x|x_t) = \mathbb{E}_{\textbf{(a)}}[\textbf{(b)}]$ where:
  - **(a)** $Z|X = x_t, Y = y$ [1.25 pt] give [0.5 pt] for $Z|Y = y.$ and [0.25 pt] for Z
  - **(b)** $\log(p(X|Z, Y = y))$ [1.25 pt] give [0.5 pt] for $p(X|Z, Y = y)$ and [0.5 pt] for $\log(p(X|Y = y))$.

Question 2 [3 points]. Let p_* denote a target distribution that we wish to sample from. Suppose that p_* is only known up to normalization, $\tilde{p}_*$. Let p_r denote a proposal distribution.
Select the correct formula for an importance-weighted approximation below:

- ◯ $\tilde{w}_j = \frac{p_r(X_j)}{\tilde{p}_*(X_j)}, \quad \mathbb{E}_{X\sim p_*}[f(X)] \approx \frac{1}{n}\sum_{j=1}^n \tilde{w}_j f(X_j).$
- ◯ $\tilde{w}_j = \frac{\tilde{p}_*(X_j)}{p_r(X_j)}, \quad \mathbb{E}_{X\sim p_*}[f(X)] \approx \frac{1}{n}\sum_{j=1}^n \tilde{w}_j f(X_j).$
- ◯ $\tilde{w}_j = \frac{p_r(X_j)}{\tilde{p}_*(X_j)}, \quad \mathbb{E}_{X\sim p_*}[f(X)] \approx \frac{\sum_{j=1}^n \tilde{w}_j f(X_j)}{\sum_{j=1}^n \tilde{w}_j}.$
- ◯ $\tilde{w}_j = \frac{\tilde{p}_*(X_j)}{p_r(X_j)}, \quad \mathbb{E}_{X\sim p_*}[f(X)] \approx \frac{\sum_{j=1}^n \tilde{w}_j f(X_j)}{\sum_{j=1}^n \tilde{w}_j}.$

Answers: (d) [3 pt], give [1.5 pt] for (b) and (c)

Question 3 [3 points]. Let p_* denote a target distribution that we wish to sample from. Suppose that p_* is only known up to normalization, $\tilde{p}_*$. Let p_r denote a proposal distribution.
Select al essentially true statements regarding a rejection sampling scheme below.

- ☐ We can rejection sample with any p_r, provided the support of the proposal contains the support of the target.
- ☐ To use rejection sampling, we must be able to find some finite $M > 0$ such that $Mp_r(x) \geq \tilde{p}_*(x)$ for all x.
- ☐ Rejection sampling depends on an asymptotic guarantee for exactness (it is exact in a limit of many samples).
- ☐ Given p_r and p_*, the maximum possible acceptance probability is $\min_x \left\{\frac{p_r(x)}{p_*(x)}\right\}$.
- ☐ If our proposal is much more concentrated than our target (has faster decaying tails), then rejection sampling will be inefficient since we will discard most proposed samples.
- ☐ If our proposal is much more concentrated than our target, then importance weighting is often a sufficiently efficient replacement for rejection sampling since importance weighted averages use all samples.

Answer: (a) is false [0.5 pt], (b) is true [0.5 pt], (d) is false [0.5 pt], (e) is true [0.5 pt], (f) is true [0.5 pt], (g) is false since the distribution of weights will be heavy tailed [0.5 pt]

Full Name and SID: _____

Question 4 [3 points]. Suppose that $\{X(t)\}_{t=0}^{...}$ is a discrete-time stochastic process. Select all true statements below.

- ☐ The process is a Markov process if any future state $X(t+h)$, $h > 0$, is conditionally independent of any past state $X(t-s)$, $s > 0$, conditional on the present $X(t)$.
- ☐ The process is irreducible if and only if, for any pair of states x, x', the transition probability that $X(t+1) = x'$ given $X(t) = x$ is nonzero.
- ☐ A distribution over the possible states is a steady-state distribution for the Markov chain, p_s, if and only if, given $X(0) \sim p_s$, $p(X(t) = x) = p_s(x)$ for all $t \geq 0$.
- ☐ If a Markov chain is ergodic, then time averages over trajectories containing T samples are equivalent to i.i.d. sample averages from the steady state distribution using T samples.
- ☐ A process obeys detailed balance if, at steady state, the net probability flux into or out of any state equals zero.
- ☐ If an irreducible process obeys detailed balance then the steady state distribution is uniquely determined by the collection of transition ratios $\{T(x \to x')/T(x' \to x)\}_{x,x'}$, and the collection of transition ratios are uniquely specified by the unnormalized steady state.

Answers: (a) is true [0.5 pt], (b) [+0.5 pt to everyone] this answer was miswritten, (c) is true [0.5 pt], (d) is false [0.5 pt] but ambiguous, so give [0.25 pt] for true, (e) is false [0.5 pt], (f) is true [0.5 pt]

Question 5 [4 points]. Let p_* denote a target distribution known up to normalization, $\tilde{p}_*$. Let $p_r(x \to x')$ denote a proposal (or jump) distribution, where $p_r(x \to x')$ gives the probability of drawing a candidate $X = x'$ given $X = x$.
Suppose that $X(t) = x$. Complete the pseudocode below for a Metropolis-Hastings Step:

1. Propose a candidate Y drawn from **the proposal given** $X(t) = x$ [1 pt].
2. Compute the acceptance probability a (write your answer in the space below):

$$a = \min\left\{\left(\tilde{p}_*(x')p_r(x' \to x)\right) / \left(\tilde{p}_*(x)p_r(x \to x')\right), 1\right\}$$

   [1 pt] for correct ratio of targets, [-0.25] if written p_* instead of $\tilde{p}_*$ [1 pt] for correct ratio of p_r's
3. Then, accept the candidate, $X(t+1) = Y$ [0.25 pt], with probability a [0.25 pt], or reject the candidate $X(t+1) = X(t)$ [0.25 pt], with probability $1 - a$ [0.25 pt].

Question 6 [3 points]. Let p_* denote a target distribution known up to normalization, $\tilde{p}_*$, for a set of unknowns Θ that can be decomposed into blocks $\theta = [\theta^{(1)}, \theta^{(2)}, ..., \theta^{(m)}]$.
Select all essentially true statements regarding Gibbs sampling below:

- ☐ At each iteration, select a block of unknowns, b, and select $\Theta_{t+1}^{(b)}$ to maximize the conditional distribution of $\Theta^{(b)}$ given the remaining unknowns, $\Theta^{-(b)}$.
- ☐ Gibbs sampling is an instance of the MH algorithm with acceptance ratio 1.
- ☐ Gibbs sampling should not be used (or will be slow/inefficient) if the blocks of unknowns are highly coupled (strongly correlated) when drawn from p_*.
- ☐ Gibbs sampling often admits efficient subroutines (conditional sampling steps) in hierarchical Bayesian models when the blocks of unknowns are conditionally conjugate.

Answers: (a) is false [0.75 pt], (b) is true [0.75 pt], (c) is true [0.75 pt], (d) is true [0.75 pt].

Quiz 7: Metropolis-Adjusted & Hamiltonian Monte Carlo

Full Name and SID: _____

Throughout this quiz, let p_* denote a smooth target density. Suppose that p_* is only known up to normalization, e.g. we only know $\tilde{p}_* \propto p_*$.

Question 1 [2 points]. It is possible to compute any partial derivative of $\log(p_*)$, if defined, knowing only $\tilde{p}_*$.

◯ True ◯ False

Answers: True [2 pts].

Question 2 [3 points]. Suppose that $X(t)$ is a continuous-time stochastic process satisfying the SDE:

$$dX(t) = \nabla_x \log(\tilde{p}_*(x))|_{x=X(t)} dt + \sigma dW(t) \tag{1}$$

where $W(t)$ is a Wiener (Brownian motion) process and $dW(t)$ is white-noise, and where $\tilde{p}_*$ has connected support. Then, select all true statements regarding the process below:

☐ The steady state distribution of the process is, for all σ, equal to the target density.

☐ The steady state distribution of the process is, for all σ, proportional to the target density raised to a power that depends on the noise level σ.

☐ The process defined in Equation (1) is an idealized version (in the limit as Δt goes to zero) of a discrete-time noisy optimization process.

Answers: (a) False [1 pt]. (b) True [1 pt]. (c) True [1 pt].

Question 3 [2 points]. Select the correct implementation of a Metropolis-Hastings step in the Metropolis-Adjusted Langevin Algorithm given $X(t)$ using step-size Δt:

(1) Draw a candidate $Y \sim \mathcal{N}((a), (b))$ where (a) is:

$$X(t) + _____$$

Answer: $\nabla_x \log(p_*(x))|_{x=X(t)}$ [1.25 pt total]. Give [0.5 pt] for ∇_x, [0.5 pt] for $\log(p_*(x))$ or $\log(\tilde{p}_*(x))$, [0.25 pt] for partially correct version of $\log(\tilde{p}_*(x)$ (e.g. missing the logarithm), [0.25 pt] for evaluation at $X(t)$.

and (b) is:

◯ $\sigma^2 I$, ◯ $\sigma^2 \Delta t I$, ◯ $\sigma^2 \Delta t^2 I$

Answer: $\sigma^2 \Delta t I$ is correct [0.25 pt].

(2) Then:

◯ Accept the candidate since the Langevin process has steady state equal to the target.

◯ Accept the candidate with probability determined by the Metropolis rule since the proposal is symmetric.

◯ Accept the candidate with probability determined by the Metropolis-Hastings rule.

Answers: (c) is correct [0.5 pt].

Full Name and SID: _____

Question 4 [2 points]. Hamiltonian Monte Carlo samples both the state X and the momentum Φ from a joint target density. In the space below, describe the form of the joint target over X and ϕ, given that we want to draw X from a marginal target p_*. Fully explain any notation you introduce.

Answer: Hamiltonian Monte Carlo draws from a joint target density $p_*(x, \phi)$ for (X, ϕ) where X and ϕ are independent [0.5 pt], $X \sim$ the original target [0.75 pt], and $\phi \sim \mathcal{N}(0, M)$ [0.75 pt] where M is the "mass matrix."

Question 5 [5 points]. Select the best summary implementation of a HMC step given X_t, ϕ_t using mass matrix M:

(1) Sample the initial momentum ϕ_0 from:

- ◯ the marginal target distribution for ϕ conditional on X_t and ϕ_t
- ◯ the marginal target distribution for ϕ independently of X_t and ϕ_t
- ◯ the marginal target distribution for ϕ independently of X_t but conditional on ϕ_t

Answer: (b) [0.75 pt]

(2) Using a leapfrog method, simulate the Hamiltonian dynamic:

$$\frac{d}{ds}x(s) = _____, \quad \frac{d}{ds}\phi(s) = _____$$

initialized from $x(0), \phi(0) = X_t, \phi_0$, for $s = 0$ to T.

Answer: $\nabla_\phi E(x, \phi), -\nabla_x E(x, \phi)$ where E is the energy function: $E(x, \phi) = -\log(p_*(x)) + \frac{1}{2}\phi^\intercal M^{-1}\phi$ [1.75 pts], or $-\nabla_\phi \log(p_*(x, \phi)), \nabla_x \log(p_*(x, \phi))$ where p_* is the joint target [1.75 pts], or, most explicitly, $M^{-1}\phi(s), \nabla_x \log(p_*(x))|_{x=x(s)}$ [1.75 pts]. Give reasonable partial credit here (e.g. [1 pt] for correct answer missing the logarithm, [1.25 pt] for correct answer but with the wrong signs in front of $\nabla_\phi E(x, \phi)$, etc.).

(3) Set the candidate Y equal to $x(T)$ then accept the candidate with probability:

$$a = \min\{_____/_____, 1\}$$

Answer: Numerator: $p_*(x(T), \phi(T))$ or $p_*(X(T))\mathcal{N}(\phi(T); 0, M)$ [0.75 pt]. Denominator: $p_*(X_t, \phi_0)$ or $p_*(X_T)\mathcal{N}(\phi_0; 0, M)$ [0.75 pt]. Give [1 pt] overall if the terms are correct, but the numerator and denominator are swapped. Give [0.75 pt] if the expression is missing the normal density evaluated at the momenta, or is missing the target evaluated at the states.

Then, in the space below, explain why, if the Hamiltonian dynamic is simulated accurately, the acceptance probability a should be close to 1.

Answer: The Hamiltonian dynamic preserves the joint target density so, if simulated exactly, $p_*(x(T), \phi(T))$ equals $p_*(x(s), \phi(s))$ for any s. Therefore $p_*(x(T), \phi(T)) = p_*(x(0), \phi(0)) = p_*(X_t, \phi_0)$. Then, the ratio of the joint target at the end of the path and the start of the path must equal 1. [1 pt] Give [0.5 pt] for any answer that discusses conservation of the joint density, energy function, or Hamiltonian.

Full Name and SID: _____

Question 6 [2 points]. Consider a tempered distribution of the form $p_\tau(x) \propto p(x)^{1/\tau}$. Then, increasing τ makes:

◯ the tempered distribution closer to uniform.

◯ the tempered distribution more concentrated about its peaks.

Answer: (a) [2 points].

Question 7 [1 points]. Performing the MH step in simulated tempering requires an estimate to the normalizing constant for each of the tempered distributions.

◯ True ◯ False

Answer: True [1 pt].

Question 8 [3 points]. Select all true statements below regarding density estimators:

☐ A Kernel Density Estimator approximates the target density via a linear combination of the kernel functions, each centered at an observed sample, and weighted by $1/n$ where n is the number of samples.

☐ A k-Nearest Neighbor Density Estimator sets $\hat{p}(\theta)$ equal to k/n divided by the distance from θ to its k^{th} nearest neighbor in the set of samples.

☐ Decreasing the bandwidth of the kernel in a Kernel Density Estimator increases the bias and decreases the variance in the estimator.

☐ Increasing k in a k-Nearest Neighbor Density Estimator increases the bias and decreases the variance in the estimator.

Answers: (a) is true [0.75 pt] give [0.25 pt] for false (potential confusion regarding weighting). (b) is false [0.75 pt]. (c) is false [0.75 pt]. (d) is true [0.75 pt].

Quiz 8: Coverage Check, Posterior Predictive Check, & Cross-Validation

Full Name and SID: _____

Question 1 [3 points]. Select the procedure that is usually used to answer each question listed below:

- Will my model procedure provide accurate predictions for new data?
  (◯ *Coverage Check*, ◯ *Posterior Predictive Check*, ◯ *Cross Validation*)
- Would the observed data be plausible if my model is true?
  (◯ *Coverage Check*, ◯ *Posterior Predictive Check*, ◯ *Cross Validation*)
- Does my computational procedure return accurate (true to model) posterior inferences/summaries?
  (◯ *Coverage Check*, ◯ *Posterior Predictive Check*, ◯ *Cross Validation*)

Answers: Cross Validation [1 pt]. Posterior Predictive Check [1 pt]. Coverage Check [1 pt].

Question 2 [4 points]. In the space below, complete the pseudocode for performing a coverage check with a summary quantity $s(\theta)$, desired coverage percentage q, and for joint model $p_{Y,\Theta} \sim (p_\Theta, p_{Y|\Theta})$.

(1) Sample $\{\Theta_{0j}\}_{j=1}^n$ to produce $\{S_{0j} = s(\Theta_{0j})\}_{j=1}^n$

(2) Sample $\{Y_j \sim p_{Y|\Theta=\Theta_{0j}}\}_{j=1}^n$ [0.75 pt]

(3) For each Y_j, sample $\{\Theta_{j,k}\}_{k=1}^m$ where $\Theta_{j,k}$ are drawn, approximately, i.i.d. from **the posterior:** $p_{\Theta|Y=Y_j}$ [0.75 pt]

(4) For each j, compute $\{S_{j,k} = s(\Theta_{j,k})\}_{k=1}^m$, then form an interval $\hat{I}_q(\{S_{j,k}\}_{k=1}^m)$ that contains q [0.5 pt] percent of the sampled summaries.

(5) Compute the observed coverage proportion $\hat{q} = \frac{1}{n}\sum_{j=1}^n \mathbf{1}(S_{0j} \in \hat{I}_q(\{S_{j,k}\}_{k=1}^m))$ [0.75 pt] where $\mathbf{1}$ denotes an indicator function.

(6) Determine whether the discrepancy between the observed coverage proportion and desired coverage proportion, $\hat{q} - q$, is statistically significant by testing against the null hypothesis that $n\hat{q}$ is drawn from a **binomial** [0.25 pt] distribution with parameters n [0.25 pt] and q [0.25 pt].

(7) Revise if the discrepancy is **statistically and practically** [0.5 pt] significant.

Question 3 [2 points]. Let (Θ, Y) denote a pair of random quantities drawn from a joint distribution $p_{\Theta,Y}$ parameterized by a prior p_Θ and a likelihood $p_{Y|\Theta}$. As usual, assume that Θ is unobserved while Y is observable. Select all true statements below regarding predictive distributions:

- ☐ The prior predictive distribution is the marginal distribution of Y when drawn from the joint model $p_{\Theta,Y}$.
- ☐ The posterior predictive distribution depends on the observation since the observation is used to infer Θ, then Θ is used for predictions.
- ☐ The evidence is the posterior predictive probability of observing a replicate data set equal to the data we actually observed, conditioned on the data set we observed.
- ☐ To sample from the posterior predictive distribution we would, usually, draw sampled Θ from the posterior, then draw sampled data sets from the likelihood $p_{Y|\Theta}$ for each sampled Θ.

Answers: (a) is true [0.5 pt], (b) is true [0.5 pt], (c) is false [0.5 pt], (d) is true [0.5 pt].

Full Name and SID: _____

Question 4 [3 points]. In the space below, complete the pseudocode for performing a posterior predictive check with a summary quantity/test statistic $s(y)$ for the joint model $p_{Y,\Theta} \sim (p_\Theta, p_{Y|\Theta})$. Let $Y^{\text{obs}} = y^{\text{obs}}$ denote the observed data, and Y^{rep} denote a replication drawn with the same Θ as Y^{obs}.

(1) Compute $s^{\text{obs}} = s(y^{\text{obs}})$ [0.5 pt]

(2) Sample replicate statistics s by drawing replicate data sets from **the posterior predictive distribution** [0.5 pt]. In practice, this is implemented by:

(1.1) Draw $\{\Theta_j \sim p_{\Theta|Y=y^{\text{obs}}}\}_{j=1}^n$ [0.25 pt]

(1.2) Then, for each j, draw $Y_j^{\text{rep}} \sim p_{Y|\Theta=\Theta_j}$ [0.25 pt]

(1.3) Then, for each j, compute $S_j^{\text{rep}} = s(Y_j^{\text{rep}})$ [0.25 pt]

(3) Use the sampled distribution of replicate test statistics to compute the **posterior predictive** p**-value**. [0.75 pt]

(4) Compare to a preset threshold (e.g. $\alpha = 0.05$) to evaluate **statistical** [0.5 pt] significance.

Question 4 [2 points]. Suppose that the quantity computed in step **(3)** above is less than the threshold $\alpha = 0.05$. In plain language, explain what conclusion you would then draw, and the standard of evidence supporting your conclusion.

Answer: "Reject the model [1 pt] because, if the model was true, then the observed data set would be implausibly atypical (e.g. fewer than 5 in 100 replications sharing the same unknowns would produce a test statistic equally or more extreme than the statistic observed) [1 pt]."

Question 5 [1 point]. What would you conclude if your quantity was greater than the threshold $\alpha = 0.05$?

Answer: "The observed test is insufficiently unusual, under replication, to reject the model (e.g. fail to reject)." [1 pt] Give any answer of this kind 1 pt. Do not give credit for an answer that suggests we should accept the model.

Full Name and SID: _____

Question 5 [2 points]. Suppose that $Y \sim f$. Let g be a distribution we use for prediction (to model f) selected from a set of available predictive distributions $\mathcal{G}$. Cross-validation procedures quantify the predictive skill of models g by estimating the quantity:

$$\text{Skill}(g|f) = \mathbb{E}_{Y\sim f}[\log(g(Y))]. \tag{1}$$

Select all true statements below regarding the predictive skill.

- ☐ Selecting g to maximize the predictive skill is the same as choosing the predictive distribution that is as indistinguishable from f when sampling from f in the sense that it minimizes $D_{\text{KL}}(f||g)$.
- ☐ Selecting g to maximize the predictive skill is the same as selecting the model that maximizes the likelihood for a $Y^{(n)} = \{Y_j\}_{j=1}^n \sim f$ i.i.d., in the limit as $n \to \infty$.
- ☐ For unknown f, the value of the quantity $\text{Skill}(g|f)$ can be used to decide whether or not to accept an individual g without comparison to the skill of other values.
- ☐ The skill difference $\text{Skill}(g|f) - \text{Skill}(g'|f)$ can be used to compare the models g and g' without knowing f.

Answers: (a) is true [0.5 pt], (b) is true [0.5 pt], (c) is false [0.5 pt], (d) is true [0.5 pt].

Question 6 [3 points]. In the space below, complete the pseudocode that estimates the predictive skill of a model via leave-one-out cross-validation. Suppose that you are given $Y^{(n)} = \{Y_j\}_{j=1}^n$ where $Y_j \sim f$ i.i.d.

(1) For $j = 1$ to n, split the data into:

- A test/held-out/replicate set: $Y_j^{\text{rep}} = Y_j$ [0.5 pt]
- A training/observed set: $Y_j^{\text{obs}} = Y_{-j} = \{Y_k\}_{k\neq j}$ [0.5 pt]

(2) Estimate the predictive skill of the **posterior predictive distribution** [0.5 pt]:

$$\text{Skill} \approx \frac{1}{n}\sum_{j=1}^{n} \log\left(\frac{1}{m}\sum_{k=1}^{m} p(Y_j^{\text{rep}}|\Theta_{j,k})\right) \tag{2}$$

[0.5 pt] evaluating the likelihood of the replicate, [0.25 pt] given the unknowns

where $\{\Theta_{j,k}\}_{k=1}^m$ [0.25 pt] are drawn from:

- ◯ the posterior distribution given Y_j^{train}.
- ◯ the posterior distribution given Y_j^{test}.
- ◯ the posterior predictive distribution given Y_j^{train}.
- ◯ the posterior predictive distribution given Y_j^{test}.

Answer: (a) [0.5 pt].

Stat 238 - Bayesian Statistics - Lecture 1: Introduction - 01/21/25

- Course intro & policies
- Bayes rule (likelihood, prior, evidence, posterior, posterior odds)
- Overview of Bayesian data analysis cycle
- Binomial Example ✓
  - Edge cases in freq. inference ✓
  - posterior = β dist. for uniform prior ✓
  - mean, var, mode, ~~entropy~~ of β dist. ✓ → save for HW
  - interp. of parameters ✓
  - use as a conjugate prior (likelihood) ✓
  - Conjugacy and parameter updates ✓
  - effects of prior & observations
    - ~~mean~~ mode as convex combination
    - variance ↘ on average (but not always)
    - vanishing influence of prior as # of obs ↗
      mean & mode all tend to MLE, posterior variance vanishes
    - ~~regularizing effect of prior~~

} regularizing effect of prior & variance-bias trade off

save for next time...

HW ideas:

1. entropy of β and observ. ↘ H in IE
2. uninformative priors
   Jeffery's β prior ($\beta(1/2, 1/2)$)
3. for what obs is Var ↗

Course Intro:

- Instructor name and emails & OH
- basic course procedure
- point to syllabus

Bayes Rule: Find cond. dist. of unknown given data...

- Observe data & $Y = y$ where we assume $Y \sim p_{Y|\theta}(\cdot \mid \theta)$ or $p_Y(\theta)$
  (e.g.
- Assume θ is also random. $\theta \sim p_\theta(\cdot)$
- Want the conditional distribution of θ given $Y = y$...

- given $p_{Y|\theta}$ and p_θ have a well-defined joint distribution

$$p_{Y,\theta} = p_{Y|\theta}\, p_\theta \Rightarrow p_{Y,\theta}(y,\theta) = p_{Y|\theta}(y|\theta)\, p_\theta(\theta)$$

equivalently, can factor

$$p_{Y,\theta} = p_{\theta|Y}\, p_Y \Rightarrow p_{\theta,Y}(y,\theta) = p_{\theta|Y}(\theta|y)\, p_y(y)$$

so:

$$p_{\theta|Y}(\theta = \theta \,|\, Y = y) = \frac{p_{Y|\theta}(y|\theta)\, p_\theta(\theta)}{p_Y(y)} \propto p_{Y|\theta}(y|\theta)\, p_\theta(\theta)$$

($p_{\theta|Y}$: posterior; p_θ: prior; $p_Y(y)$: evidence; $p_{Y|\theta}(y|\theta)$: likelihood — not a dist. for θ)

- evidence: $p_Y(y) = \int_\theta p_{Y,\theta}(y,\theta)\, d\theta = \int_\theta p_{Y|\theta}(y|\theta)\, p_\theta(\theta)\, d\theta$

(implied... sometimes hard to evaluate). ⇒ "nice" parametric models

- posterior odds:

$$\frac{p_{\theta|Y}(\theta|y)}{p_{\theta|Y}(\theta'|y)} = \frac{p_\theta(\theta)}{p_\theta(\theta')} \cdot \frac{p_{Y|\theta}(y|\theta)}{p_{Y|\theta}(y|\theta')}$$

prior odds; likelihood ratio

turns out this is all we will really need to build computational approximations for posterior evaluation

+ discuss inductive view of Bayes's rule...
demands model for the full data generating process...

2

Bayesian Data Analysis Cycle: 0. Collect data & form research questions...

1. Build a full model of the data generating process. Model Construction.
   a. likelihood (same as freq. inference)
   b. prior (discuss types of unknowns)
      • objective • subjective • weakly informative
2. ~~Collect observations~~
2. Condition on data to produce the posterior. Posterior Inference.
   • use the posterior for inference or to answer research question
   • quantify stat. uncertainty
3. Evaluate the model's ~~inferential~~ inference. Model criticism.
   • e.g. would model pred. match new data / current data plausibly
   • how sensitive are answers to model choices

induction from data
deduction + def. form model
deduction validates
w/in model
w/out model
explore model uncertainty/bias
deduction falsification & severity testing

Binomial Example (today 1. & 2.)

Lecture 2 - 01/23/25

- Estimate an unknown prob. from observed (binary outcomes) .. ← Q: ask for examples
- $Y^{(n)} = \{Y_1, Y_2, \dots Y_n\} \overset{iid}{\sim} \text{Bernoulli}(\theta)$ for $\theta \in [0, 1]$, $Y_i \in \{0, 1\}$
- Likelihood: $p(Y^{(n)} = y^{(n)} \mid \theta) = \prod_{j=1}^{n} \theta^{y_i^{(n)}} (1-\theta)^{1 - y_i^{(n)}}$

$$= \theta^{\sum_i y_i^{(n)}} (1-\theta)^{n - \sum_i y_i^{(n)}} \quad \dots \sum_i y_i^{(n)} \text{ is sufficient}$$

- let $S = \sum_{i=1}^{n} Y_i^{(n)}$, $s = \sum_{i=1}^{n} y_i^{(n)}$ = # of successes (by i.i.d., trials are exchangeable, order of outcomes is irrelevant)
- then $S \sim \text{Binomial}(n, \theta)$

$$p(S = s \mid \theta) = (\#\text{ of perm. given } s \text{ successes in } n \text{ trials})\ \theta^s (1-\theta)^{n-s} = \binom{n}{s} \theta^s (1-\theta)^{n-s}$$

4

- ~~Prior:~~
  - ~~subjective... let's use uniform~~

- MLE: $\underset{\theta}{\operatorname{argmax}}\left(p_{S|\theta}(S=s|\theta)\right) = \underset{\theta}{\operatorname{argmax}}\left(\theta^s(1-\theta)^{n-s}\right)$

$$\log(\dots) \overset{+C}{=} s\cdot\log(\theta) + (n-s)\log(1-\theta)$$

$$\frac{d}{d\theta}\dots = s/\theta - \frac{n-s}{1-\theta} = 0 \Rightarrow s(1-\theta) = (n-s)\theta$$

$$\frac{\theta}{1-\theta} = \frac{s}{n-s} = \frac{s/n}{1-s/n}$$

$$\boxed{\hat{\theta}_{MLE}(s) = \frac{s}{n}}$$

- obviously problematic if n small, $s=0$ or 1
  - $\hat{\theta}$ highly variable to sample success count
  - argue this should be formalized as a prior

is $\theta = 0$ or 1 plausible? how uncertain are we about θ given s?

- Confidence Interval vs. Credible Interval notion of UQ...
  - ask for definition
  - (want single observ. guarantee on prob. of θ...)

let's try Bayesian inference...

- Prior: start subjective, uninformed... $\theta \sim Unif([0,1])$ ← ask for this

  then $p_\theta(\theta) = 1$ for any θ so

- Posterior:

$$p_{\theta|S}(\theta=\theta|S=s) \propto \theta^s(1-\theta)^{n-s}\cdot 1$$

- Posterior odds:

$$\frac{p_{\theta|S}(\theta=\theta|S=s)}{p_{\theta|S}(\theta=\theta'|S=s)} = \left(\frac{\theta}{\theta'}\right)^s\left(\frac{1-\theta}{1-\theta'}\right)^{n-s}$$

5

to evaluate the posterior absolutely (not relative), need the evidence

$$p_S(S=s) = \int_{\theta=0}^{1} \theta^s (1-\theta)^{n-s}\, d\theta = ?$$ ← HW?

$$= \frac{\Gamma(s+1)\,\Gamma(n+2)}{\Gamma(s+1)\,\Gamma(n-s+1)} = \frac{1}{B(s+1,\, n-s+1)}$$

"Beta function"

so: $p_{\theta|S}(\theta|s) = \frac{1}{B(s+1,\, n-s+1)}\,\theta^s (1-\theta)^{n-s}$ = Beta density w/ parameters $\alpha = s+1$, $\beta = n-s+1$

so: $\left[\!\left[\theta \mid S=s \sim \text{Beta}(s+1,\, n-s+1)\right]\!\right]$ ... notice, the evidence is implied, so we could recognize from the functional form.

- Properties of Beta distributions: $\left[\!\left[\text{Beta}(\alpha,\beta) \to \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\right]\!\right]$
  - Mean: $\frac{\alpha}{\alpha+\beta} \Rightarrow \frac{s+1}{n+2} = \mathbb{E}[\theta|S=s]$ (Laplace rule of succession)
  - Mode: $\frac{\alpha-1}{\alpha+\beta-2} \Rightarrow \frac{s}{n} = \arg\max_\theta (p_{\theta|S}(\cdot|s=s)) = \hat{\theta}_{MAP}(s)$
  - SD: $\left(\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}\right)^{1/2} = \frac{(\mathbb{E}[\theta](1-\mathbb{E}[\theta]))^{1/2}}{(\alpha+\beta+1)^{1/2}}$
  - Skew: symm. if $\alpha=\beta$, skewed otherwise (sketch) (mean more central than mode)
  - boundaries: if $\alpha>1$, $\beta>1$, $p_\theta(\theta) \to 0$ as $\theta \to 0$ or 1

(posterior SD: $SD[\theta|S=s] = \frac{(\mathbb{E}[\theta|s=s](1-\mathbb{E}[\theta|s=s]))^{1/2}}{(n-1)^{1/2}}$ ... $= \mathcal{O}(n^{-1/2})$, concentrates as usual ⇒ SD

compare to $SD[\hat{\theta}_{MLE}(S) \mid S \sim \text{Binomial}(n, \hat{\theta}_{MLE}(s))] = \left(\frac{\hat{\theta}_{MAP}(s)(1-\hat{\theta}_{MAP}(s))}{n}\right)^{1/2}$

save for HW or lab or for asy. consistency conversation)

- ask class for scenarios to sketch ($n=0, s=0$ ⇒ uniform posterior, $n=1, s=1$ ⇒ linear, $n=4, s=2$ ⇒ parabolic down, sym. centered, $n=3, s=2$ ⇒ skew, $n=10, s=8$ ⇒ ...)

- "objective" prior... what if we'd already seen s_0 past successes in n_0 past trials

then should use $s + s_0$ successes, $n + n_0$ trials... now posterior is

$$\theta \mid S = s, s_0, n_0, n \sim \text{Beta}(s + s_0 + 1, n + n_0 + 1)$$

is this incorporating prior information in a Bayesian sense? if so, for what prior?

- factor the posterior:

$$p_{\theta|S}(\theta = \theta \mid S = s) \propto \theta^{s+s_0}(1-\theta)^{(n+n_0)-(s+s_0)}$$

$$= \left[\theta^{s}(1-\theta)^{n-s}\right]\left[\theta^{s_0}(1-\theta)^{n_0-s_0}\right]$$

$$\propto \underbrace{p_{\text{Beta}}(\theta; s, n)}_{\text{bin. lik.}} \underbrace{p_{\text{Beta}}(\theta; s_0, n_0)}_{\text{Beta prior...}}$$

- "conjugate" prior:

$$\theta \sim \text{Beta}(s_0 \overset{+1}{\vee}, n_0 \overset{+1}{\vee})$$

then $$p_{\theta|S}(\theta = \theta \mid S = s) \propto \left[\theta^{s}(1-\theta)^{n-s}\right]\left[\theta^{s_0}(1-\theta)^{n_0-s_0}\right]$$

$$= \theta^{s+s_0}(1-\theta)^{(n+n_0)-(s+s_0)}$$

so $$\theta \mid S = s, n, n_0, s_0 \sim \text{Beta}((s+s_0)+1, (n+n_0-s-s_0)+1)$$

posterior & prior are in the same family! just update parameters...

- **Posterior Inference w/ Beta, Binomial Model:**

  Q: how else might we select an "objective" prior?

  - **Joint model:** $\theta \sim \text{Beta}(\alpha, \beta)$

    $S|\theta \sim \text{Binomial}(n, \theta)$

    hyperparameters: α, β

    obs. parameters: n

  then...

  - **Posterior:** $\theta | S_{obs}=s;\ n, \alpha, \beta \sim \text{Beta}(s+\alpha, n-s+\beta)$

  - use ↗ for inference...

  e.g.

  - $\hat{\theta}_{MAP}(s; n, \alpha, \beta) = \frac{s+\alpha-1}{n+\alpha+\beta}$, $\quad \mathbb{E}[\theta | S=s] = \frac{s+\alpha}{n+\alpha+\beta}$
  - $SD[\theta|S=s] = \left(\frac{\mathbb{E}[\theta|S=s](1-\mathbb{E}[\theta|S=s])}{n+\alpha+\beta+1}\right)^{1/2} \leq \frac{1}{2}(n+\alpha+\beta)^{-1/2}$

  so $\theta|S=s \in \frac{s+\alpha}{n+\alpha+\beta} \pm K \cdot \mathcal{O}(n^{-1/2})$ w/ high posterior prob. if K large

  ↑ choose to make failure prob. small

  $n+\alpha+\beta$

- **Observations:** two major questions... ← ask class for this
  1. how does the prior regularize inference?
  2. are the inferences consistent if n is large?

1. **Consider:** $\hat{\theta}_{MAP}(s; n, \alpha, \beta) = \frac{s+\alpha-1}{n+\alpha+\beta} = \frac{s+s_0}{n+n_0}$ $\quad (\alpha = s_0+1,\ \beta = n_0 - s_0 + 1)$

   bias $= \mathcal{O}(1/n)$

$$= \left(\frac{n}{n+n_0}\right)\frac{s}{n} + \left(\frac{n_0}{n+n_0}\right)\frac{s_0}{n_0} = \left(\frac{n}{n+n_0}\right)\frac{s}{n} + \left(1-\frac{n}{n+n_0}\right)\frac{s_0}{n_0}$$

Convex Combination... pulls est. toward prior

$$\hat{\theta}_{MAP}(s; n, \alpha, \beta) = \left(\frac{n}{n+n_0}\right)\hat{\theta}_{MLE}(s; n, 0, 0) + (1-\ldots)\,\hat{\theta}_{MAP}(s_0; n_0, 0, 0)$$

2. are inferences consistent as $n \to \infty$?

$\hat{\theta}_{MLE}(s;n) = \frac{s}{n} \xrightarrow[\text{in prob}]{\text{SLLN}} \theta_{\text{true}}$, ⇒ CLT

$SD[\hat{\theta}_{MLE}(s;n) \mid \theta] = \sqrt{\frac{\theta(1-\theta)}{n}}$

$\mathbb{E}[\hat{\theta}_{MLE}(s;n) \mid \theta] = \theta$

by CLT: standardized error $\hat{\theta}_{MLE}(s;n)$ converges, in distribution, to a $N(0,1)$ r.v. as $n \to \infty$

so, confidence interval for large n of the form:

$$\hat{\theta}_{MLE}(s;n) \pm K\sqrt{\frac{\hat{\theta}_{MLE}(1-\hat{\theta}_{MLE})}{n}}$$

all point estimators agree

$\hat{\theta}_{MAP}(s;n,\alpha,\beta) = \mathbb{E}[\theta \mid s;n,\alpha,\beta] + \mathcal{O}(\frac{1}{n})$

$= \hat{\theta}_{MLE}(s;n) + \mathcal{O}(1/n)$

so, for large n, $\hat{\theta}_{MAP} \xrightarrow[\mathcal{O}(n^{-1})]{} \hat{\theta}_{MLE}$, $\mathbb{E}[\theta \mid s] \xrightarrow[\mathcal{O}(1/n)]{} \hat{\theta}_{MLE}$, will have essentially the same sampling distributions conditional on θ. Will concentrate on θ. Consistent.

- the posterior will also approach the sampling dist. conditional on $\theta = \hat{\theta}$ for any of the three estimators...
  - Why? $\text{Beta}(\alpha,\beta) \to \mathcal{N}$ if $\alpha, \beta \to \infty$, $\frac{\alpha}{\alpha+\beta}$ converges to a value $\neq 0$ or 1
  - $SD[\theta \mid S=s;n,\alpha,\beta] = \left(\frac{\mathbb{E}[\theta|s](1-\mathbb{E}[\theta|s])}{n + \underbrace{\alpha+\beta+1}_{n_0-1}}\right)^{1/2} \xrightarrow[\mathcal{O}(1/n)]{} SD[\hat{\theta}(s;\ldots) \mid \hat{\theta} = \hat{\theta}]$

Outro Q's:

- if we had samples from prior, how should we fit it? ↳ lab/HW
- how could we fit w/out outside samples? ↳ lab
- how much error would we incur if our prior is wrong? how does it decay in n? ↳ project
- how would/could we check?? ↳ lab

Lecture 4:

- Prep estimator formula on board
- Things we want to know about the estimators??
- Consistency & sampling dist.
- Shape of the posterior

(i) Beta(α,β) widget // given time, drawing exercise (1,1), (2,2), (3,3), (4,2)

(ii) ok... but want to understand it

do it in log space

(iii) asymp. normality by drawing & Taylor expansion

$$\text{Beta}(\alpha,\beta) \xrightarrow[\text{in dist}]{} N\left(\underbrace{\frac{\alpha+1}{\alpha+\beta+1}}_{=\frac{\alpha}{\alpha+\beta}+\mathcal{O}(1/n)=\mathbb{E}[\theta]},\ \frac{\mathbb{E}[\theta](1-\mathbb{E}[\theta])}{n+\alpha+\beta+1}\right)$$

$$\frac{\theta-\mathbb{E}[\theta]}{SD[\theta]} \xrightarrow[\text{in dist}]{} N(0,1) \quad \dots \text{ imitates sampling dist} \dots$$

- Bayesian CLT: Suppose: • observe $Y=y$, want posterior for unknown $\theta \in \Omega \subseteq \mathbb{C}^d$, Ω w/ open interior

• $\hat{\theta} = \hat{\theta}_{MAP}(y) \in$ interior of parameter space $\text{int}(\Omega)$

Laplace Approx.

• the log lik. & prior are 2^{nd} diff. w/ Lipschitz 2^{nd} deriv. at $\hat{\theta}$

$$\log(p(\theta|y)) \simeq \log(p(\hat{\theta}|y)) + \frac{1}{2}(\theta-\hat{\theta})^T \underbrace{\left[\partial_\theta^2 \log(p(\theta|y))\right]_{\theta=\hat{\theta}}}_{\text{Hessian of posterior} = -\text{"observed info"}} (\theta-\hat{\theta})$$

w/ error $\mathcal{O}(\|\theta-\hat{\theta}\|^3)$ if $\frac{1}{n}$ p.d. ($\hat{\theta}$ unique)

Now, suppose $Y^{(n)} = \{Y_1, Y_2, \dots Y_n\}$, ~~$Y_j \overset{iid}{\sim} p_Y$~~ $(Y_j,\theta) \sim p_{Y,\theta}$, $p_{Y,\theta} \to p_\theta, p_{Y|\theta}$

~~(for now, well specified) ⇒~~ $Y_j \overset{iid}{\sim} p_Y^*$

• model: $(Y_j,\theta) \sim p_{Y,\theta} \nearrow p_\theta \searrow p_{Y|\theta}$, $\theta \in \Omega \subseteq \mathbb{C}^d$, Ω open interior

• well-specified: s.t. $p_Y^* = p_{Y|\theta^*}$ for some θ^*

where $\theta^* \in \text{int}(\Omega)$

• ~~mis-specified: p^* ... model $\theta^* = \text{argmin} \sum KL(p^*$~~

Some fun HW or challenge problem

- then: $\hat{\theta}^{*}_{MAP} \xrightarrow[\text{in prob.}]{} \theta^{*}$, (see Appendix B for the gen. statement)

$$-\left[\frac{d^2}{d\theta^2}\log(p(\theta|y^{(n)}))\right]_{\theta=\hat{\theta}} = -\left[\frac{d^2}{d\theta^2}\log(p(\theta))\right]_{\theta=\hat{\theta}_{MAP}} - \sum_{i=1}^{n}\left[\frac{d^2}{d\theta^2}\log(p(y_i|\theta))\right]_{\theta=\hat{\theta}_{MAP}}$$

$$\approx \text{constant} + n\,J(\theta^*)$$

so in the limit as $n\to\infty$ $\left[n\,J(\theta_*)\right]^{-1/2}(\theta|y^{(n)} - \hat{\theta}^*)$ is normally distributed, $\sim \mathcal{N}(0, I_{d\times d})$ w/ prob. 1 over sequences of observations

- mis-specified: now $\hat{\theta}_{MAP} \xrightarrow[\text{in prob}]{} \theta^*$ where $\theta^* = \underset{\theta\in\Omega}{\operatorname{argmin}}\{KL(p^*_Y \,\|\, \text{marginal } p_{Y|\theta})\}$

Formal:

- Definitions: $Y\sim f_Y$, $Y^{(n)}\sim f_{Y^{(n)}}$, model $Y\sim p_{Y|\theta}$, ~~marginal of~~ $p_{Y,\theta}\to(p_\theta, p_{Y|\theta})$

~~(1) Fisher Information: $\mathcal{I}(\theta) = \mathbb{E}_{Y\sim f_Y}[\nabla_\theta \log(p(Y^{(n)}|\theta))]$~~

~~f $p_Y = p_{Y|\theta}$~~

(1) KL-divergence: $D_{KL}(p\,\|\,q) = \mathbb{E}_{x\sim p}\left[\log\left(\frac{p(x)}{q(x)}\right)\right]$

$d(\theta) = \mathbb{E}$

Stat 238 - Bayesian Statistics - Lecture 5: Why be Bayesian? - Decision Theory - 02/04/2025

- **Learning Goals:**
  - ~~Recap Discussion on β-binomial~~ ✓
  - 15 min • Why context matters through the Bayesian CLT... ✓ ← reinforce theorem statement
    what problems are we solving? ✓
  - 10 min > • My philosophy, cards on the table... ✓
  - > • Example problems where I've found Bayes useful/compelling ✓
  - 30 min • Decision theory as a problem Bayes can solve ← define setting, inadequacy of admissibility & minimax, posterior risk calculations simple hypothesis test & choosing "α"
  - 35 40 • Decision theory as an alternate method for posing a prior ← • Can actually run backward posing a risk/loss model leads to a criteria on LR that is equivalent to selection using posterior odds

to HW or to Th.

- **Quiz 1 - 12:15**

β-binomial: discussion Q's...

- ~~What is~~ Summarize 1 idea you thought was important & position it in the BDA cycle.
- Summarize 1 idea you found surprising or confusing & position it.

1. Modeling
2. Posterior inference
3. Model Checking

on board at start

(1)

- "Bayesian Central Limit Theorem"

- Definitions: (i) $Y \overset{(n)}{\underset{iid}{\sim}} f_Y$ ← true (ii) model $(\Theta, P_\theta, P_{Y|\theta})$, $\theta \in \Omega$

  (·) well-specified if $f_Y = P_{Y|\theta_*}$ for some $\theta \in \Omega$, otherwise, misspecified

  (··) otherwise let $\theta_* = \underset{\theta\in\Omega}{\operatorname{argmin}} \{KL(f_Y, P_{Y|\theta})\}$ ~ to the MLE given ∞'ly many draws of Y

  $D_{KL}(f_Y \| P_{Y|\theta}) = D_{KL}(\theta)$.

  (···) Fisher Information:

$$J(\theta) = \mathbb{E}_{Y\sim f_Y}\left[\nabla_\theta \log(Lik(\theta|Y))\right] = -\mathbb{E}_{Y\sim f_Y}\left[\nabla_\theta^2 \log(Lik(\theta|Y))\right]$$

  Hessian: $\begin{bmatrix} \partial_{\theta_1}^2 & \partial_{\theta_2}\partial_{\theta_1} & \cdots \\ \partial_{\theta_1}\partial_{\theta_2} & \partial_{\theta_2}^2 & \\ \vdots & & \ddots \end{bmatrix}$

- Thm: (Posterior Concentration) If Ω is compact, and N is a neighborhood of θ_* w/ a $P_\theta(\theta \in N) \neq 0$ then $P(\theta \in N | Y^{(n)}_{*}) \to 1$ as $n \to \infty$.

- Thm: (Consistency of Posterior Point Estimators) Under mild regularity conditions, and compact Ω, then Thm ↑ implies $\hat\theta_{MAP}(Y^{(n)}) \xrightarrow[n\to\infty]{} \theta_*$ in prob.

  $\hat\theta_{Mean}(Y^{(n)}) \xrightarrow[n\to\infty]{} \theta_*$ ...

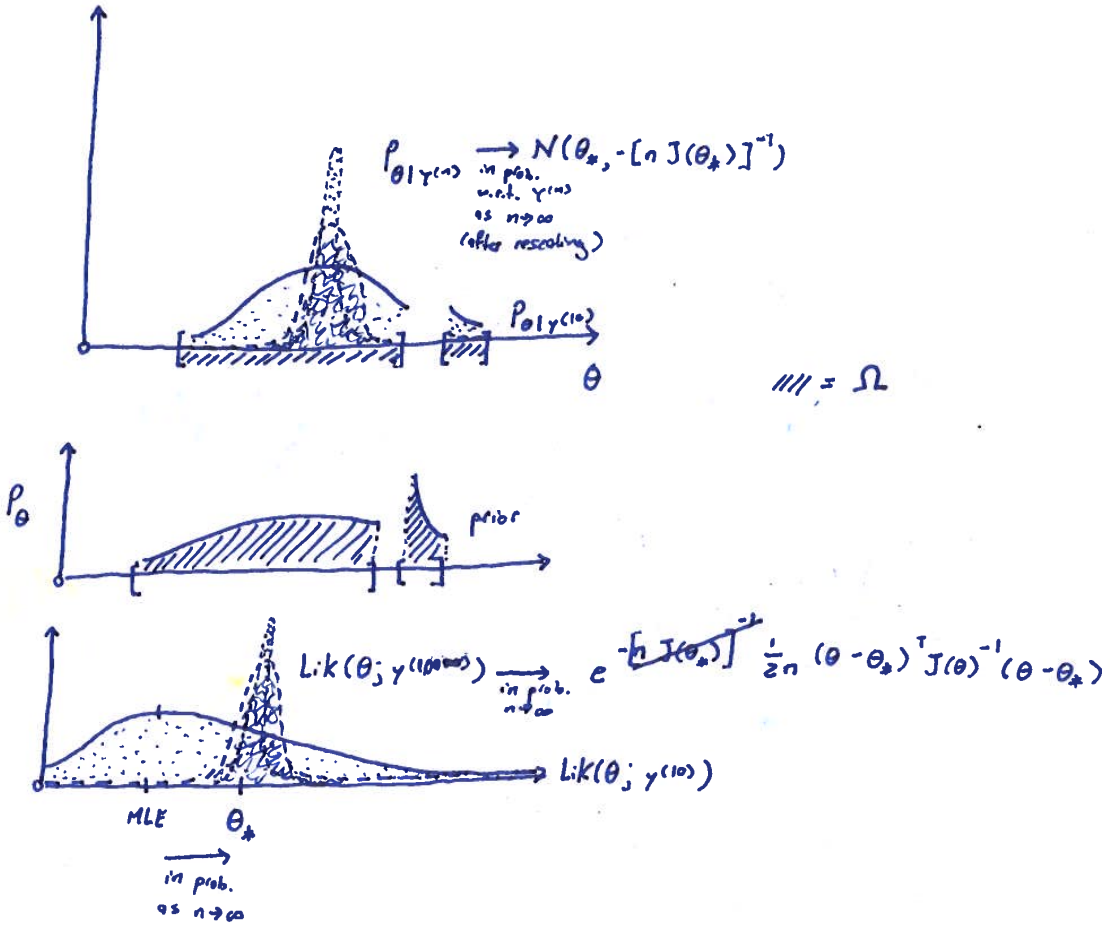
- Thm: (Asymptotic Normality) Under mild regularity conditions (notably $\theta^* \in int(\Omega)$)...

(i) $\theta|Y^{(n)} \xrightarrow[n\to\infty]{\text{in prob w.r.t. } Y,\ \text{(after scaling)}} \mathcal{N}\left(\theta_*, \hat\theta_{MAP}, -\left[\nabla_\theta^2 \log\left(p_{\theta|Y^{(n)}}(\theta)\right)\Big|_{\theta=\hat\theta_{MAP}}\right]^{-1}\right)$

(ii) $\theta|Y^{(n)} \xrightarrow[n\to\infty]{\text{in prob w.r.t. } Y,\ \text{(after scaling)}} \mathcal{N}\left(\theta_*, (n\, J(\theta_*))\right)$

(iii)* $\theta|Y^{(n)} \xrightarrow[n\to\infty]{\text{in prob w.r.t. } Y} \sim e^{-n(D_{KL}(\theta) - D_{KL}(\theta_*))}$

* more general.


$p_{\theta|Y^{(n)}} \to N(\theta_*, -[n J(\theta_*)]^{-1})$
in prob. w.r.t. $Y^{(n)}$ as $n \to \infty$ (after rescaling)
$p_{\theta|Y^{(10)}}$
θ
//// = Ω
p_θ
prior
$Lik(\theta; Y^{(1000)}) \to e^{-\frac{1}{2}n(\theta-\theta_*)^T J(\theta)^{-1}(\theta-\theta_*)}$
in prob. $n \to \infty$
$Lik(\theta; Y^{(10)})$
MLE
θ_*
in prob. as $n \to \infty$

but... all this asymptotic theory is based on the likelihood dominating the prior

reduces to a Bayesian restatement of the likelihood principle where the only role of the prior is rhetorical (can define $p_{\theta|T}$)...

- Example Problem: Medical Imaging

(e.g. CT, PET, MRI, ultrasound)

features:

i. per inference never enough data to resolve from likelihood alone...

ii. the unknown exists, could be measured ⇒ can aim for empirical priors w/ a freq. interpretation over randomization of subjects
... access to "hard" prior info, and even weak forms are helpful

iii. make a decision per subject (single sample inference)
but use the tool repeatedly...

iv. inferences lead to decisions w/ real consequences... can we optimize?
(inference → prediction, predictive accuracy $ validity matter)
since we use repeatedly, actually care about IE errors over the population (in addition to worst-case error guarantees cond. on ind, for any * most individuals)

v. engineers have control over the device physics... have control over what we measure/information gathered (what likelihood should you use?)
⇒ Bayesian search procedures.

- My philosophy:

- Dogmas usually follow from prioritizing self-consistency over utility or realism
often aim for "proof" or "simplicity"... rhetorical appeal
- Debates are positional and historically informed
- Almost all mathematics, once applied, are rhetorical devices $ rules for making arguments
  - in prob. regarding a formal model for assessing and rearranging uncertainty
  - axiomatic validity is only self consistency does not ensure accuracy (3.

- as such, most* prob. arguments rest on modeling, Freq. & Bayesians primarily differ in what they are willing to model
  - in part flowing from a philosophical diff. in acceptable definitions of prob.
  - but... freq. is more hypothetical than acknowledged
    - hyp. repitition - limited to falsification and thus sufficiency
  - convincing more from context of practice & modesty of aims
    - less comp. power available, harder to validate models, less data available
    - fewer real examples of automated decision making & repeated single sample inference...
    - Statistics was new, arguing for its own validity, took the most persuasive & defensible position
- be responsible for your arguments, whatever mode you choose... the burden of proof is on you & should be high
  - adv. of Bayes is it performs parts of arguments for us (induce from data, deduce from model)
  - only guarantees consistency w/in model
  - do the hard work to convince yourself and others of your model & argument

---

Bayesian Decision Theory:

- The end point of statistics is a choice not an inference
- Choices have consequences...
  - Want procedures that make "good" decisions
- Formalism:

1. collect data: Y → 2. use data to make a decision $d(Y)$ (often via intermediate inference on some unknown quantities Θ)

→ 3. decision induces an outcome $X \sim p_{X|d}$ → 4. Outcome has consequences, utility $U(X)$ or $U(X, \theta)$

- often, some "loss": $\ell_0(x, \theta)$ or $\mathbb{E}_{X|d}[\ell(x, \theta)] = \ell(d, \theta)$

e.g.
$X|d = d$, $d(Y) = \hat{\theta}(Y)$
$\ell_0(x, \theta) = \ell(\hat{\theta}, \theta) = \|\hat{\theta}(Y) - \theta\|$
(e.g. $\|\hat{\theta}(Y) - \theta\|_2^2$ = S.E.
$\|\hat{\theta}(Y) - \theta\|_1$ = abs. error

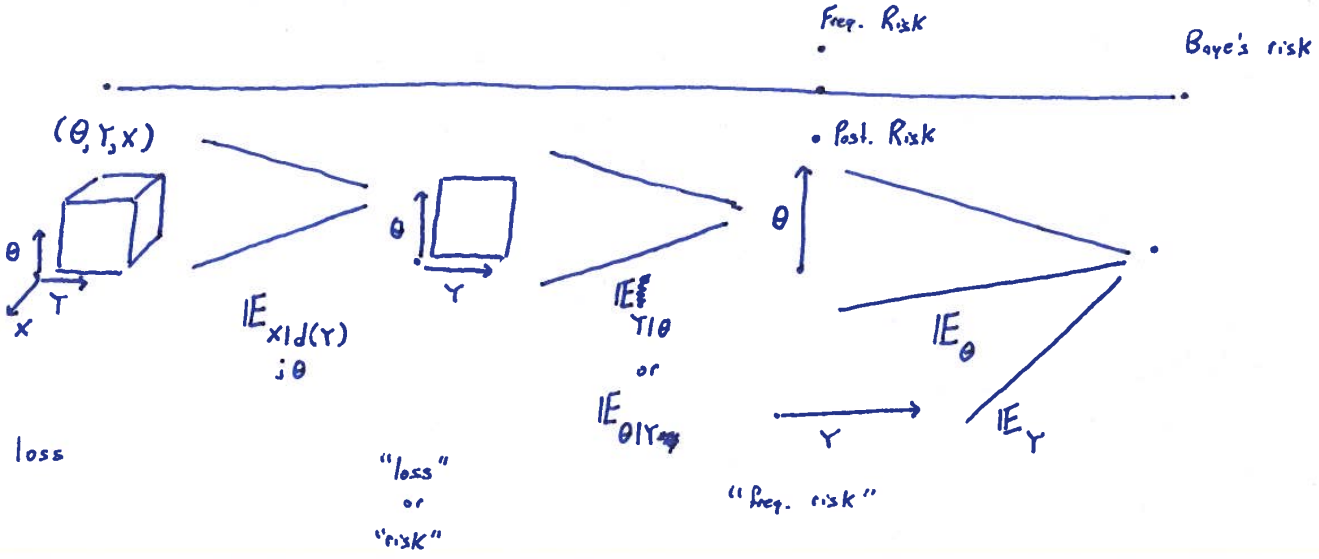
if we use the tool repeatedly, we (the tool user, not the individual) will see: $\mathbb{E}_{\theta, Y}[\text{utility}]$ or $\mathbb{E}[\text{loss}]$

(4)

- Contrast to the individual who only sees a single instance

↕

- what we are willing to average over vs. what we want to uniformly control depends on what we actually repeat



- **Def:** loss: $\ell(\theta, d(Y)) = \mathbb{E}_{X|d}[\ell_0(\theta, X)]$ → over. over decision outcomes X

freq. risk: $R(\theta, d) = \mathbb{E}_{Y|\theta}[\ell(\theta, d(Y))]$ → over. over sampling of data Y given θ

post. risk: $\rho(y, d) = \mathbb{E}_{\theta|y}[\ell(\theta, d(y))]$ → over. over unknowns given data

Baye's risk: $r(d) = \mathbb{E}_{\theta,Y}[\ell(\theta, d(Y))]$ → over. over unknowns and data.

↳ also depends on observation

$$r(d) = \begin{cases} \mathbb{E}_Y[\rho(Y, d)] \\ \mathbb{E}_\theta[R(\theta, d)] \end{cases} = \mathbb{E}_{\theta,Y}\left[\mathbb{E}_{X|d(Y)}[\ell_0(\theta, X)]\right]$$

↳ also depends on truth

↑ only a function of:
- joint model $(\theta, Y, X|d)$
- decision rule $d(Y)$

(5)

Stat 238 - Decision Theory - 02/11/25

Outline:

(*) • reorient to setting. have loss & risk def. on the board & binary hyp. setting
‖ • minimizing Bayes risk for 1 hyp. via slope condition
↓ what about multiple hyp.? What is this condition more simply } ← transition
• ~~multi~~ multi-optimization vs. multiple optimization (admissibility, mini-max, Bayes's rule)
‖ • multi-hyp. testing example
~~• Bayesian decision making for optimal search~~

(*) $\ell(\theta, d(y)) = \mathbb{E}_{x \mid d(y), \theta}[\ell_0(\theta, x, \dots)]$ is loss

$R_F(\theta, d) = \mathbb{E}_{Y \mid \theta}[\ell(\theta, d(Y))]$ freq.

$R_P(y, d) = \mathbb{E}_{\theta \mid Y=y}[\ell(\theta, d(y))]$ post.

$R_B(d) = \mathbb{E}_{\theta, Y}[\ell(\theta, d(Y))]$ Bayes.

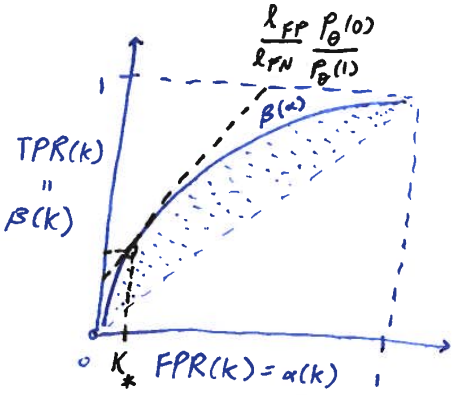
$\mathbb{E}_Y[\cdot]$, $\mathbb{E}_\theta[\dots]$ } risk

Ex:

• Setting: (Binary Simple Hyp. Test)

Lik: $\theta = 0 \Rightarrow H_0: Y \sim P_{Y \mid \theta=0}$, $\theta = 1 \Rightarrow H_1: Y \sim P_{Y \mid \theta=1}$

Prior: $\theta \sim P_\theta$



• decision rule: $d(y) = \begin{cases} 1 & \text{if } LR(y) \geq k \\ 0 & \text{else} \end{cases}$,

• lik ratio: $LR(y) = \frac{Lik(H_1; y)}{Lik(H_2; y)}$

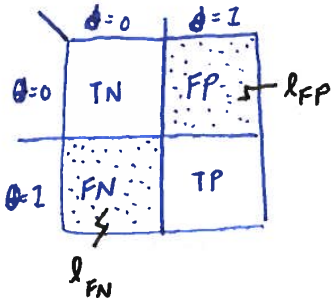
• Error Rates: $FPR(k) = Pr(d(Y) = 1 \mid \theta = 0)$
$FNR(k) = Pr(d(Y) = 0 \mid \theta = 1)$

• loss: $\ell(\theta, d(Y)) = \begin{cases} \ell_{FP} & \text{if } \theta = 0, d = 1 \\ \ell_{FN} & \text{if } \theta = 1, d = 0 \end{cases}$

- **Freq. Risk:** $P_\theta(d(Y)=1 \mid \theta=0)$

$$R_F(\theta, d) = \begin{cases} \ell_{FP} \cdot FPR(k) & \text{if } \theta=0 \\ \ell_{FN} \cdot FNR(k) & \text{if } \theta=1 \end{cases}$$

$P_\theta(d(Y)=0 \mid \theta=1)$



Baye's Risk:

$$R_B(d) = \ell_{FP} \cdot FPR(k) \cdot p_\theta(\theta=0) + \ell_{FN} \cdot FNR(k) \cdot p_\theta(\theta=1)$$

$$= \ell_{FP} \cdot p_\theta(0)\ \alpha(k) + \ell_{FN}\ p_\theta(1)\ (1-\beta(k))$$

$$= \ell_{FP}\, p_\theta(0)\, \alpha + \ell_{FN}\, p_\theta(1)\, (1-\beta(\alpha))$$

optimize w.r.t. α (p-value threshold) $\Leftrightarrow$ w.r.t. k (LR threshold)

$$\alpha_* = \operatorname{argmin}\left\{ \frac{\ell_{FP}}{\ell_{FN}} \frac{p_\theta(0)}{p_\theta(1)} \alpha + (1-\beta(\alpha)) \right\}$$

$$\frac{d}{d\alpha}(\quad \dots \quad) = \frac{\ell_{FP}}{\ell_{FN}} \frac{p_\theta(0)}{p_\theta(1)} - \frac{d}{d\alpha}\beta(\alpha)$$

$\Rightarrow$ α_* and k_* solve: $\frac{d}{d\alpha}\beta(\alpha) = \frac{\ell_{FP}}{\ell_{FN}} \frac{p_\theta(0)}{p_\theta(1)}$

eq.: $\frac{d}{dk}\beta(k) = [\dots]\ \frac{d}{dk}\alpha(k).$

} fixes the desired trade-off between losses

Questions:

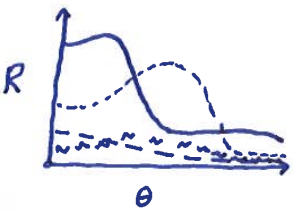
- still unsatisfying... relation to posterior implicit...
- unclear what k_* is
- not clear how to gen. to multiple hypotheses

Criteria for decision rules:

- **Admissibility:** consider freq. risk

  $R(\theta, d)$ ... is a function of θ

  ideally, find a rule d that is:



(i) good enough... $R(\theta, d) \leq r(\theta)$ on region of θ of interest
(often $r(\theta) = r$, uniform risk bound)

(ii) admissible...

- Def: d dominates d' if $R(\theta, d) \leq R(\theta, d')$ $\forall\ \theta$ and $<$ for some θ
- Def: d is admissible if there is no available d' that dominates d

* often many admissible rules often admissible ⇏ sensible (e.g. accurate)...

(iii) never too bad (minimax) ... d minimizes $\max_\theta \{R(\theta, d)\}$ among all θ.

* often far too conservative...

- **Core problems:**
  - treats all θ as equally important or overly stiff (just ignore), only controls domain of θ
  - in principle (i) allows flexible treatment, but we need to choose $r(\theta)$... run into the same issues as choosing a prior

- **Maximize $\mathbb{E}$ utility, Minimize $\mathbb{E}$ risk:**
  - **Bayes's rule:** pick $d_* = \operatorname{argmin}_d \{r(d)\} = \operatorname{argmin}_d \{\mathbb{E}_{\theta,Y}[l(\theta, d(Y))]\}$.
  - often much simpler, gives sensible estimators, prioritizes θ's in a smoother way
  - can get informed prior estimators (empirical), prior info can guide decisions
  - observed Freq. risk averaged over many trials $\xrightarrow[\text{SLLN}]{}$ Bayes's risk.

- **Two Examples:**

  (i) simple hypothesis test... choosing decision thresholds

  (ii) optimal search/experimental design... what to measure next?

(i) Simple Binary Hypothesis Test:

$H_0: Y \sim P_{Y|H_0}$, $H_1: Y \sim P_{Y|H_1}$ $\Rightarrow$ optimal test stat: $\arg\left(\frac{\text{Lik}(H_1|Y=y)}{\text{Lik}(H_0|Y=y)}\right) = LR_1(y)$ — likelihood principle

$\theta = \{0 \text{ or } 1\}$ (indicator for truth)

$P_\theta = [P_0, P_1]$

decision rule:

$$d(y) = \begin{cases} 1 & \text{if } LR_1(y) \geq K \\ 0 & \text{else} \end{cases}$$ is $\text{Lik}(H_1|y) \geq k\,\text{Lik}(H_0)$

Posterior Risk: given $Y=y$

$$R_p(y, d) = \mathbb{E}_{\theta|Y=y}[\ell(\theta, d(Y))] = \begin{cases} P_{\theta=0|y} \cdot \ell_{FP} & \text{if } d(y)=1 \\ P_{\theta=0|y} \cdot \ell_{FN} & \text{if } d(y)=0 \end{cases}$$

how to choose K?

let $FPR(K) = Pr(d(Y)=1 \mid \underbrace{\theta=0}_{H_0})$

$TPR(K) = Pr(d(Y)=1 \mid \underbrace{\theta=1}_{H_1})$

Bayes's Risk:

$$R_B(d) = P_{\theta=0, \text{given } d(y)=1} \cdot \ell_{FP} + P_{\theta=1, d(Y)=0} \cdot \ell_{FN}$$

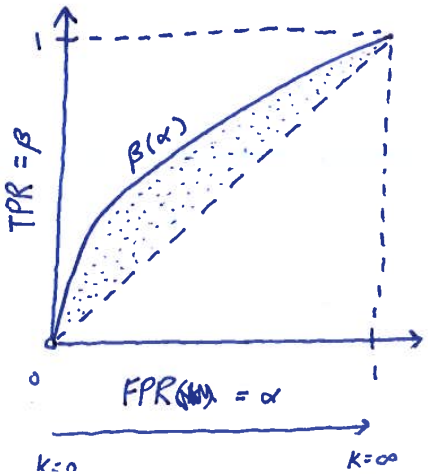
$$= P_{\theta=0} \cdot FPR(K) \cdot \ell_{FP} + P_{\theta=1} \cdot FNR(K) \cdot \ell_{FN}$$

$FPR(K) = \alpha$, $FNR(K) = (1-\beta(\alpha))$

loss: $\ell(\underbrace{d}_{\hat{Y}}, \underbrace{\theta}_{Y}) = \begin{cases} \ell_{FP} & \text{if } d=1, \theta=0 \\ \ell_{FN} & \text{if } d=0, \theta=1 \end{cases}$

Freq. Risk: given $\theta=\theta$

$$R_F(\theta, d) = \begin{cases} FPR(K) \cdot \ell_{FP} & \text{if } \theta=0 \\ FNR(K) \cdot \ell_{FN} & \text{if } \theta=1 \end{cases}$$



$$R_B(\alpha) = (\ell_{FP} P_{\theta=0})\alpha + (\ell_{FN} P_{\theta=1})(1-\beta(\alpha))$$

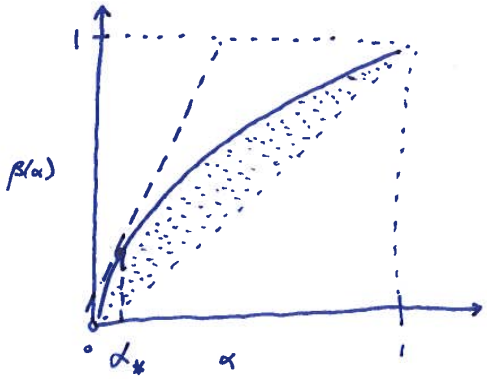
to maximize, can rescale

$$R_B(\alpha) \overset{\alpha}{\propto} \left(\frac{\ell_{FP}}{\ell_{FN}} \frac{P_{\theta=0}}{P_{\theta=1}}\right)\alpha + (1-\beta(\alpha))$$

$$\frac{d}{d\alpha} \cdots \overset{\alpha}{+} \left(\frac{\ell_{FP}}{\ell_{FN}} \frac{P_{\theta=0}}{P_{\theta=1}}\right) - \frac{d}{d\alpha}\beta(\alpha)$$

α_* where $\beta'(\alpha) = \left(\frac{\ell_{FP}}{\ell_{FN}}\right) \cdot \left(\frac{p_{\theta=0}}{p_{\theta=1}}\right)$ a slope condition

loss ratio prior odds



- suppose:

$\ell_{FP} = 1$ $p_{\theta=0} = 1/100$

$\ell_{FN} = 10$ $p_{\theta=1} = 1/10{,}000$

$\beta_*' = \frac{1}{10} \cdot \frac{10{,}000}{100} = \frac{1}{10} \cdot 100 = 10$

notice $\frac{d}{d\alpha}\beta(\alpha) = \frac{d}{d\alpha} TPR(\alpha) = \frac{d}{d\alpha} Pr(d_\alpha(Y) = 1 \mid \theta = 1)$

$= \frac{d}{d\alpha} Pr(LR(Y) \geq K(\alpha) \mid \theta = 1)$

too much implicit differentiation too deep into hyp. testing

$= \frac{d}{d\alpha}(1 - PR(LR(Y) < K(\alpha) \mid \theta = 1))$

$= -\frac{d}{d\alpha} \int_{y=0}^{K(\alpha)\ LR^{-1}(K(\alpha))} p_{Y \mid \theta=1}(y)\, dy$

$= - p_{Y = LR^{-1}(K(\alpha))}(y \mid \theta = 1) \cdot \frac{d}{d\alpha} LR^{-1}(K(\alpha))$

$= - Lik(\theta = 1 ; \underbrace{LR^{-1}(K(\alpha))}_{y_*(\alpha)}) \cdot \frac{1}{\frac{d}{dy} LR(y)\big|_{y_*(\alpha)} \cdot \frac{d}{d\alpha} y_*(\alpha)}$

(ii) optimal search... $\Theta \sim p_\theta$, want to resolve

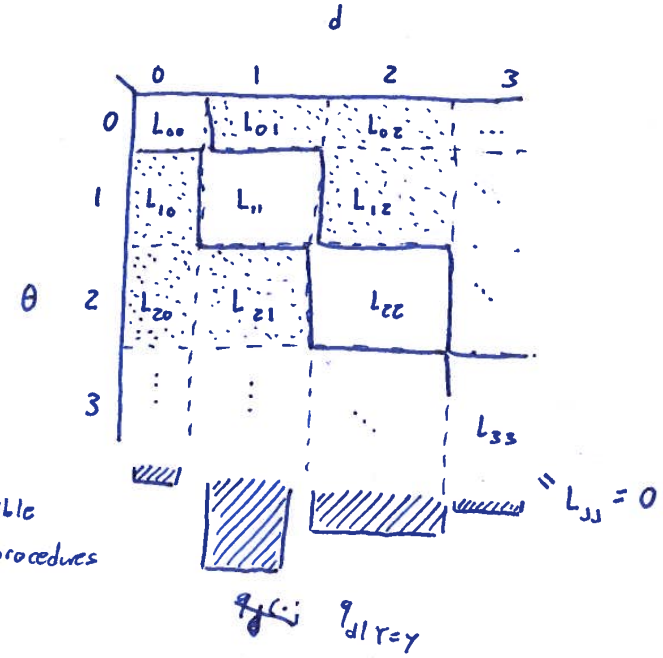
- action set: $d \in \{M = \{m_1, m_2, \dots m_M\}$ available measurements we could make/questions we could ask

- given we choose $d = m \Rightarrow$ ~~temp plan~~ $Y \mid \Theta = \theta \sim p_{Y|\theta}(\cdot\,; m)$

- induces a posterior

$$p_{\theta|Y} \propto p_\theta \cdot p_{Y|\theta;m}$$

- **Example:**
  - **Setting:** Multiple Hypothesis Testing
    - $\theta \in \{1, 2, \dots m\} \Rightarrow \{H_j\}_{j=1}^{m}$
    - $Y|\theta \sim P_{Y|\theta}$
    - **loss:** $\ell(\theta, d(Y)) = L_{\theta, d}$
    - **generic decision rule:** $d(Y) \sim q_{d|Y=y}$



all possible decision procedures

- Freq. approach is messy... try from posterior risk
  ⇓ [why? admissibility...]
- **posterior risk:**

$$R_P(Y,d) \triangleq \mathbb{E}_{\substack{\theta|Y \\ d(Y)}}[\ell(\theta, d(Y))] = \mathbb{E}_{\substack{\theta|Y \\ d(Y)}}[L_{\theta, d(Y)}] = P_{\theta|Y}^{T} L\, q_{d|Y}$$

- **Baye's risk:**

$$R_B(d) = \mathbb{E}_{Y}[R_P(Y,d)] = \mathbb{E}_{Y}[P_{\theta|Y}^{T} L\, q_{d|Y}] \quad \dots \text{ looks messy}$$

notice:

$$R_B(d) = \mathbb{E}_{Y}[R_P(Y,d)]$$

if $d_*(y) = \underset{d}{\operatorname{argmin}}\{R_P(y,d)\}$ then $d_*(y) \le d(y) \ \forall\ d$

$$\therefore \mathbb{E}_{Y}[R_P(Y,d_*)] \le \mathbb{E}[R_P(Y,d)] \ \forall d\,!$$

(the reverse is true almost everywhere in Y)

Bayes's Rule: choose, equivalently

$$d_*(y) = \underset{d \in \mathcal{D}}{\operatorname{argmin}} \{ R_p(y, d) \} \iff d_* = \underset{d \in \mathcal{D}}{\operatorname{argmin}} \{ R_\theta(d) \}$$

... can solve for d_* by min. at each y...

.. not multi-opt., rather multiple, independent opt. problems, 1 for each y...

Application (multiple testing):

$$R_p(y, d) = p_{\theta|Y=y}^T \, L \, q_{\theta d|Y=y}$$

find $q_* = \underset{q \in \mathbb{R}^{m+}}{\operatorname{argmin}} \{ p_{\theta|y}^T \, L \, q \}$ $\quad \sum_j q_j = 1$

linear programming!

solved at a vertex $\Rightarrow$ deterministic rule (or mixture of equally good det. rules)

use $d_*(y) = \underset{d}{\operatorname{argmin}} \{ [p_{\theta|y}^T L]_d \} = \underset{d}{\operatorname{argmin}} \{ \mathbb{E}_{\theta|y} [L_{\theta, d}] \}$

... compute $L^T p_{\theta|y}$, pick the hyp. w/ smallest average loss

- **Ex:** 0,1 loss ... $L_{i,j} = 0$ if $i=j$, 1 else

  then $(p_{\theta|y}^T L)_j = (\sum_{d \neq \theta} p_{\theta|y}(d))_j = Pr(\theta|y \neq j) = 1 - Pr(\theta|y = j)$

  then $d_*(y) = j_* = \underset{j}{\operatorname{argmax}} \{ Pr(\theta|Y=y = j) \} = \hat{\theta}_{MAP}(y)$!

- **Ex:** binary case...

  $$j_* = \underset{}{\operatorname{argmin}} \{ \underset{d=0}{\ell_{FN} p_{\theta=0|y}}, \underset{d=1}{\ell_{FP} p_{\theta=0|y}} \}$$

  $$= \underset{}{\operatorname{argmin}} \{ \ell_{FN} p_{\theta=1} \underbrace{p_{y|\theta=1}}_{\text{lik}}, \ell_{FP} p_{\theta=0} \underbrace{p_{y|\theta=0}}_{\text{lik}} \}$$

  $$\rightarrow d_*(y) = \begin{cases} 1 \text{ if } \ell_{FN} p_{\theta=1} Lik(1,y) \geq \ell_{FP} p_{\theta=0} Lik(0,y) \\ 0 \text{ else} \end{cases}$$

  restate in terms of posterior odds $\longrightarrow$ decision threshold: $\frac{Lik(1,y)}{Lik(0,y)} = K_* = \frac{\ell_{FP} p_0}{\ell_{FN} p_1}$

Stat 238 - Modeling: Conjugate Models (I) - 02/13/25

Outline: 1. Conjugate Models

- utility ... { trade-off between model constraint and tractibility of inference problem } and simplicity of argument "when a simple thing is..." ✓
- desires:
  - (i) as expressive as possible
  - (ii) include un or weakly informative priors
  - (iii) plausible in some settings
  - (iv) interpretable
  - (v) work w/ important likelihoods
  - (vi) scale well
- ... N, N as the best (?) compromise example

2. Settings:

(a) linear regression...

- predictor variables X
- unknown coefficients θ
- observe $Y \sim N(X\theta, C_y)$

$$\begin{bmatrix} | \\ Y \\ | \end{bmatrix} = \begin{bmatrix} -- x_i^T -- \\ X^T \end{bmatrix} \begin{bmatrix} | \\ \theta \\ | \end{bmatrix} + \begin{bmatrix} | \\ \varepsilon \\ | \end{bmatrix}$$

(e.g. $y_i = x_i^T\theta + \varepsilon_i$, $\varepsilon \sim N$)

- usually an abstract modeling story ... θ are descriptive "fictitious unknowns" or "unmeasurable unknowns" ⇒ simplicity, clarity of argument, comprehensible, justifiable, bias, beneficence

(b) linear measurement w/ gaussian noise

$$y_i = M_i^T x + \varepsilon_i \Big|_{i=1}^n, \quad x \in \mathbb{R}^d, \ y \in \mathbb{R}^n, \ M_i \in \mathbb{R}^d,$$

make $\varepsilon \sim N(0, \sigma_{y_i}^2)$

- usually for some unknown physical state ... x are real "measurable unknowns" ⇒ realism of model or precision of inference

3. the N, N model ... normal r.v. w/ unknown mean...

$$x \sim N(\bar{x}, C_x)$$

$$y = M^T x + \varepsilon, \quad \varepsilon \sim N(0, C_y)$$

$x \in \mathbb{R}^d$, $y \in \mathbb{R}^n$, $n \neq d$

- does this contain "uninformative" prior?

take $C_x = \sigma_x^2 I$ and $\lim_{\sigma_x \to 0} \ldots$

prior → uniform ... ill-defined! ⇒ improper prior

- what about C_y?

$C_y \in \{$ $n \times n$ symm. p.s.d. matrices $\}$

$\binom{n}{2}$ dim. cone... how to put a distribution on this?

⇒ r. mat. theory

$C_y = \sigma_y^2 I$ ⇒ just need a prior on $\sigma_y^2 \in \mathbb{R}^+$

or $\sigma_y \in \mathbb{R}^+$

- let's try uninformative...

$s = \sigma_y \sim \text{Unif}([a, b])$

$\Downarrow$

$\sigma_y^2 = (\sigma_y)^2$

$v = s^2 \quad \tilde{} \quad s = v^{1/2}$

$|dv| f_v(v) = f_s(s(v)) |ds|$

$f_v(v) = f_s(s(v)) \frac{|ds|}{|dv|} = f(s(v)) \left| \frac{d}{dv} s(v) \right|$

$= \frac{1}{|b-a|} \left| \frac{d}{dv} v^{1/2} \right| = \frac{1}{2} \frac{1}{|b-a|} v^{-1/2}$

not uniform

Pareto if $b \to \infty$

$v \sim \text{Pareto}(a, 1/2)$

$\mathbb{E}[v] = \infty$!

$v = \sigma_y^2 \sim \text{Unif}([a^2, b^2])$

$\Downarrow$

$s = v^{1/2} \quad \tilde{} \quad v = s^2$

$|ds| f_s(s) = f_v(v(s)) |dv|$

$f_s(s) = f_v(v(s)) \frac{|dv(s)|}{|ds|} = f_v(v(s)) \left| \frac{d}{ds} v(s) \right|$

$= \frac{1}{|b^2 - a^2|} \left| \frac{d}{dv} v^2 \right| = \frac{2}{|b^2 - a^2|} s$

not uniform

clearly improper!

it's not so easy to be agnostic if we aren't sold on our parameterization

(ask about β, binomial)

save for now... see HW 2, BDA

4. posterior under the $\mathcal{N}, \mathcal{N}$ model...

$$p_{x|y}(x) = \propto p_x(x)\, p_{y|x}(y), \propto e^{-\frac{1}{2}\underbrace{\left((x-\bar{x})^T C_x^{-1}(x-\bar{x}) + (y-M^T x) C_y^{-1}(y-M^T x)\right)}_{\text{quadratic in } x}}$$

(y: fixed)

$$\Rightarrow x|y \sim \mathcal{N}(\underbrace{\bar{x}_{|y}, C_{x|y}}_{\text{just need to update moments...}})$$

(·) to simplify, solve for $\bar{x}=0$

at end, shift back ($x \to x-\bar{x}$, $y \to y - M^T\bar{x}$)

(··) want to match densities... enough to match quadratics up to constant

$$x^T C_x^{-1} x + (y-M^T x)^T C_y^{-1}(y-M^T x)$$

$$= x^T(C_x^{-1} + M C_y^{-1} M^T)x$$

$$-2\, y^T C_y^{-1} M^T x$$

$+ \ldots$ ← constant in x

$$(x-\bar{x}_{|y})^T C_{x|y}^{-1}(x-\bar{x}_{|y})$$

$$= x^T C_{x|y}^{-1} x$$

$$-2\, \bar{x}_{|y}^T C_{x|y}^{-1} x$$

$+ \ldots$

must hold for all x...

matching coefficients:

$$\underbrace{(C_x^{-1} + M C_y^{-1} M^T)}_{\text{precisions add}} = C_{x|y}^{-1} \Rightarrow \boxed{C_{x|y} = (C_x^{-1} + M C_y^{-1} M^T)^{-1}}$$

(Cov. update)

$$y^T C_y^{-1} M^T = \bar{x}_{|y}^T C_{x|y}^{-1} \Rightarrow C_{x|y}^{-1}\bar{x}_{|y} = M C_y^{-1} y$$

$$\bar{x}_{|y} = C_{x|y} M C_y^{-1} y$$

shifting back:

$$\bar{x}_{1y} - \bar{x} = C''_{x1y} M C_y^{-1}(y - M^T \bar{x})$$

mean update

$$\boxed{\bar{x}_{1y} = \bar{x} + C_{x1y} M C_y^{-1}(y - M^T \bar{x})}$$

Overview:

prior: $X \sim N(\bar{x}, C_x)$ $\xrightarrow{M, y}$ posterior: $x|y \sim N(\bar{x}_{1y}, C_{x1y})$

measure: $y = M^T x + \underset{\sim N(0, C_y)}{z}$

$\bar{x} \longrightarrow \bar{x}_{1y} = \bar{x} + C_{x1y} M C_y^{-1}(y - M^T \bar{x})$ (post mean; prior mean)

$C_x \longrightarrow C_{x1y}^{-1} = C_x^{-1} + M C_y^{-1} M^T$

- intuition: suppose ~~$M = m$~~ $d=1$, $n=1$, $m=1$

$$C_x = \sigma_x^2, \quad C_y = \sigma_y^2$$

now: $\underbrace{\sigma_{x1y}^{-2} = \sigma_x^{-2} + \sigma_y^{-2}}_{\text{precision adds}} \Rightarrow \sigma_{x1y}^2 = (\sigma_x^{-2} + \sigma_y^{-2})^{-1} = \underbrace{(1 + \underbrace{\sigma_x^2 \sigma_y^{-2}}_{SNR})^{-1}}_{\text{reduction of var}} \sigma_x^2$

$$= \frac{1}{1 + SNR} \sigma_x^2$$

$$\bar{x}_{1y} = \bar{x} + (1 + \underbrace{SNR}_{\sigma_x^2 \sigma_y^{-2}})^{-1} \sigma_x^2 \sigma_y^{-2}(y - \bar{x})$$

$$= \bar{x} + \frac{SNR}{1 + SNR}(y - \bar{x})$$

$$= \underbrace{\left(1 - \frac{SNR}{1+SNR}\right)}_{1 - s(SNR)} \bar{x} + \underbrace{\frac{SNR}{1+SNR}}_{s(SNR)} y$$

post mean = conv. comb. of prior mean & obs.

so: $\boxed{\frac{\sigma_{x1y}}{\sigma_x} = \sqrt{\frac{1}{1 + SNR}}}$ post & red. of var is controlled by SNR

SNR controls info content of an observation

Stat 238 - Lecture 9 - 02/18/25

• (9.1) Intuition in the Normal, Normal Model

• Last class: if

unknowns ($\mathbb{R}^d$): $x \sim N(\hat{x}, C_x)$,

$n \leq d$

observations ($\mathbb{R}^n$): $y = M^T x + z$, $z \sim N(0, C_y)$

$y|x \sim N(M^T x, C_y)$

then: $x|y \sim N(\hat{x}_{1y}, C_{x1y})$ where:

• var. update: $C_{x1y}^{-1} = C_x^{-1} + M C_y^{-1} M^T$

• mean update: $\hat{x}_{1y} = \hat{x} + C_{x1y} M C_y^{-1} (y - M^T \hat{x})$

• today: let's make sense of this result...

• intuition (in 1-D): $n = 1$, $d = 1$, $M = 1$ then: $C_x = \sigma_x^2$, $C_y = \sigma_y^2$

(1) var. update: $\sigma_{x|y}^{-2} = \sigma_x^{-2} + \sigma_y^{-2}$ ← "precisions add"

or... $\sigma_{x|y}^2 = (\sigma_x^{-2} + \sigma_y^{-2})^{-1}$

• information: how does σ_x^2 change after observing y?

$$\sigma_{x|y}^2 = (\sigma_x^{-2} + \sigma_y^{-2})^{-1} = (\sigma_x^{-2}(1 + \sigma_x^2/\sigma_y^2))^{-1}$$
$$= (1 + \underbrace{\sigma_x^2/\sigma_y^2}_{\text{"signal-to-noise ratio" = SNR}})^{-1} \sigma_x^2$$

so: $\sigma_{x|y} = \underbrace{(1 + SNR)^{-1/2}}_{\text{information gain}} \sigma_x$

• regularization: consider $\sigma_{x|y}^2$ for $\sigma_x = 10$ (*) vs. $\sigma_x = 1/10$ (**), $\sigma_y \approx 1$

(*) $\sigma_{x|y}^2 = (\sigma_x^{-2} + \sigma_y^{-2})^{-1} = (1/100 + \sigma_y^{-2})^{-1} \approx (1 - \frac{\sigma_y^2}{10^2}) \sigma_y^2$

if $\sigma_x \to \infty$ then $\sigma_{x|y}^2 \to \sigma_y^2$

(prior does not reduce uncertainty relative to likelihood if flat)

} Weak Regularization

(**) $\sigma_{x|y}^2 = (\sigma_x^{-2} + \sigma_y^{-2})^{-1} = (100 + \sigma_y^{-2})^{-1} \approx (1 - \frac{\sigma_y^2}{100}) \sigma_x^2$

if $\sigma_x \to 0$ then $\sigma_{x|y}^2 \to \sigma_x^2$

(prior strongly reduces uncertainty relative to likelihood when sharp)

} Strong Regularization

(2) the mean update:

$$\hat{x}_{1|y} = \hat{x} + C_{x1y} M C_y^{-1} \underbrace{(y - M^T \hat{x})}_{y - \mathbb{E}[y]} = \hat{x} + \sigma_{x|y}^2 \sigma_y^{-2} (y - \hat{x})$$

$$= \hat{x} + \underbrace{(\sigma_x^{-2} + \sigma_y^{-2})^{-1} \sigma_y^{-2}}_{\text{"NSR"} = SNR^{-1}} (y - \hat{x})$$

$$= (\sigma_y^2 \sigma_x^{-2} + 1)^{-1} = (1 + \sigma_y^2/\sigma_x^2)^{-1}$$

$$= (1 + SNR^{-1})^{-1} = \frac{SNR}{1 + SNR}$$

$$\hat{x}_{1|y} = \hat{x} + \underbrace{\frac{SNR}{1 + SNR}}_{s(SNR)} \underbrace{(y - \hat{x})}_{y - \mathbb{E}[y]} = (1 - s(SNR))\,\hat{x} + s(SNR)\,y$$

So: $\hat{x}_{1|y} = (1 - s(SNR))\,\hat{x} + s(SNR)\,y$ where: $s(SNR) = SNR/(1 + SNR)$

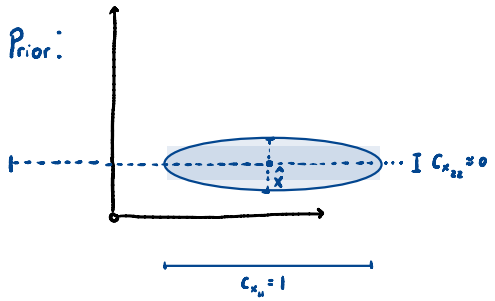
Convex combination of prior mean and observation!

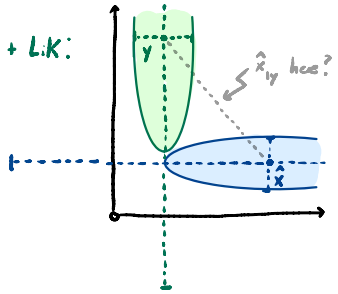
$SNR = \sigma_x^2/\sigma_y^2$

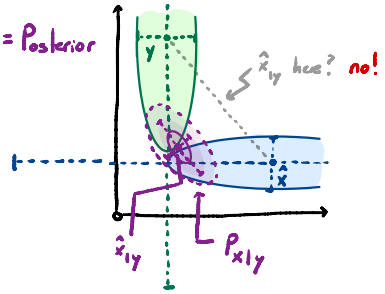
($s(SNR) \to 1$ as $SNR \to \infty$; $\to 0$ as $SNR \to 0$)

- note: posterior means are not, for all models, a convex combination of the prior mean and the MLE...

e.g. $n=2$, $d=2$, $C_x = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}$, $C_y = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$





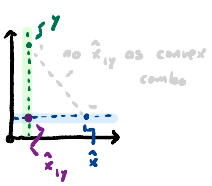


large post density needs large lik & prior density

extreme case:

$C_{x_{22}} \to 0$

$C_{y_{11}} \to 0$



... sensible, necessary behavior attend to shape of distributions

• (9.2) Improper & Uninformative Priors

• We said that we often want our conjugate families to include some uninformative priors...

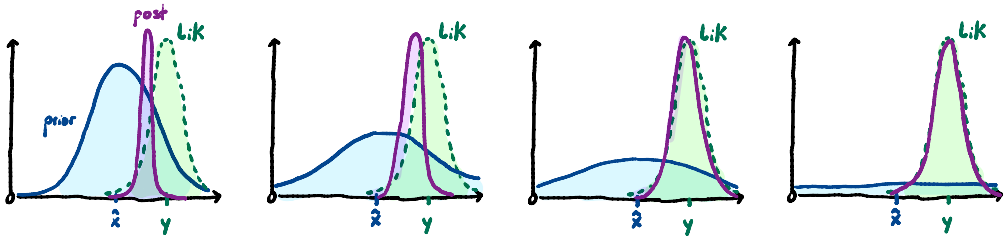
• e.g. $X \sim N(\hat{x}, C_x)$, $y|x \sim N(M^T x, C_y)$ } idea: take $C_x \to \infty$, $M = I$ ("infinite uncertainty in x a priori")

entirely sensible, well-defined posterior in the limit

• then: $C_{x|y} = (C_x^{-1} + C_y^{-1})^{-1} \longrightarrow (0 + C_y^{-1})^{-1} = C_y$

$\hat{x}_{|y} = \hat{x} + C_{x|y} C_y^{-1}(y - \hat{x}) \longrightarrow \hat{x} + C_y C_y^{-1}(y - \hat{x}) = y$

} $x|y \sim N(y, C_y)$ in lim as $C_x \to \infty$



$\sigma_x \approx 1$ $\sigma_x \approx 2$ $\sigma_x \approx 4$ ... $\sigma_x \approx$ large

But, ... what happened to the prior?

if $C_x \to \infty$ then $p_x \to$ uniform, but... uniform on $\mathbb{R}^d$!

un (ill) defined... normalization = ∞ or 0... mathematically

but, the same as having used $p_x(x) = 1$ for any x

- not a well defined distribution

but...

- induces a well defined posterior given an appropriate likelihood function

this is an improper prior

• **Def:** a function p_x s.t. $p_x(x) \geq 0$ $\forall$ unknown x is an improper prior for a likelihood $Lik(x; y) = p(y|x)$ if: (induces a proper posterior)

a) $\int_{\text{all } x} p_x(x)\, dx = \infty$, but... b) $\int_{\text{all } x} p_x(x)\, Lik(x; y)\, dx < \infty$

• most sensible as limits of proper priors...

- So far, we've assumed that C_y is known...
  - What if the covariance in the "measurement" is unknown?
  - Can we put a prior on C_y or σ_y^2?

  ... and, in lieu of information, pose an "uninformative" prior?

- Two challenges:

  1. C_y is a positive semi-definite, real-valued, symmetric $n \times n$ matrix...

     need a distribution on the space of ... matrices...

     messy, this is a "cone" in $\mathbb{R}^{n^2}$

     - simple choice: $C_y = \text{diag}(\vec{\sigma}_y^2)$ where $\sigma_{y_j} \perp\!\!\!\perp \sigma_{y_i}$, put a 1D prior on each variance

     - subtler choice: pose a random matrix model...

       - e.g.: $C_y = U \Sigma_y U^T$, draw $U \perp\!\!\!\perp \Sigma_y$
         (SVD)

         - $U \sim$ uniform on $n \times n$ orthonormal matrices (sample the col's sequentially picking random rotations of e_1, restricted so $\perp$ to everything before)
         - $\Sigma_y = \text{diag}(\vec{\sigma}_y^2)$, draw $\{\sigma_{y_i}\}_{i=1}^n$ i.i.d.
           (principal S.D.'s)

       nice, reduces to picking a prior on 1-D variances/SD's

  (worse approach: $C_y = RR^T$, Cholesky, then draw entries of $R \perp\!\!\!\perp$ w/ different dist. for diagonal and off diagonal entries)

2. want an "uninformative" prior on S.D.'s / variances...

- let $s = \sigma$, $v = \sigma^2$ then $s(v) = v^{1/2}$, $v(s) = s^2$
- should we model v or s?
  - hopefully our answers don't depend on this choice... it is arbitrary

model s: $v(s) = s^2$, $s(v) = v^{1/2}$ **model v**

- $s \in [0, \infty)$, try $s \sim \text{Uniform}([a, b])$ ($a \to 0$, $b \to \infty$, improper)
- fixes $p_s(s)$...
- what is $p_v(v)$?

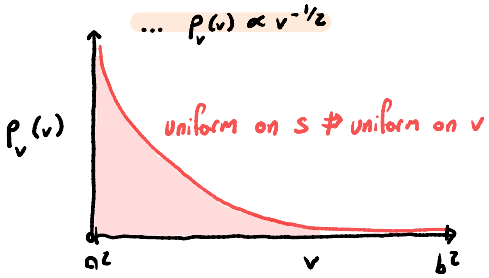
$$p_v(v)|dv| = p_s(s(v))|ds(v)|$$
$$p_v(v) = p_s(s(v))\left|\frac{d}{dv}s(v)\right|$$
$$= \frac{1}{|b-a|}\left|\frac{d}{dv}v^{1/2}\right|$$
$$= \frac{1}{2|b-a|}v^{-1/2} \quad \text{for } v \in [a^2, b^2]$$

not uniform

take $a \to 0$, $b \to \infty$ get:

improper Unif on s $\Rightarrow$ improper Pareto on v

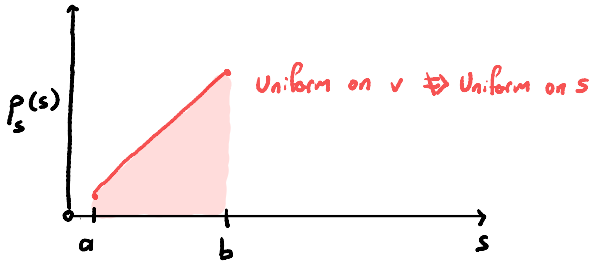
... $p_v(v) \propto v^{-1/2}$



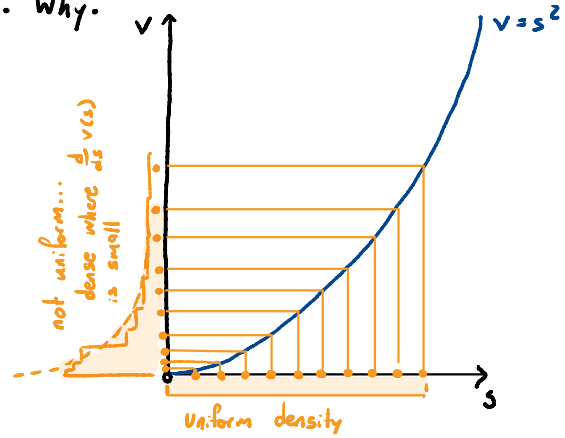
- $v \in [0, \infty)$, try $v \sim \text{Uniform}([a^2, b^2])$
- fixes $p_v(v)$...
- what is $p_s(s)$?

$$p_s(s)|ds| = p_v(v(s))|dv(s)|$$
$$p_s(s) = p_v(v(s))\left|\frac{d}{ds}v(s)\right|$$
$$= \frac{1}{|b^2-a^2|}\left|\frac{d}{ds}s^2\right|$$
$$= \frac{2}{|b^2-a^2|}s \quad \text{for } s \in [a, b]$$

not uniform



... Why?



this is a generic problem...
constructing "uninformative" or "uniform" priors for a continuous unknown requires the user to commit to a specific parameterization/coordinate space for the unknowns

- Outro:
  - want a general framework for constructing informative conjugate priors... given likelihoods.
    - e.g. what is conjugate for σ or σ^2?
    - want conjugacy (for now) for:
      (a) simplicity (interpret results)
      (b) tractability (analysis makes computation easier)

• (9.3) Conjugate Models via Exponential Families

• **Problem:** want a generic mathematical formalism for building informative, conjugate priors...

(e.g. what is a conjugate prior on σ^2 if $y|\sigma^2 \sim \mathcal{N}(\hat{\mu}, \sigma^2)$?)

- informative: because we can't be generically uninformative might as well incorporate information intentionally
- conjugate: for simplicity & tractability
  - cannot aim for realism ... (this aim lives outside the formalism)
  - ... so, hope to be expressive

• **Def:** a family of priors $\mathcal{P} = \{p_\theta\}$ is conjugate for a family of likelihood functions $\mathcal{L} = \{Lik(\cdot; y) = p_{y|\cdot}\}$ if, for any observation y, the posterior: $p_{\theta|y}$ is in $\mathcal{P}$.

prior, lik $\Rightarrow$ post

• **Ex:** $\theta \sim Beta$, $y|\theta \sim Bin \Rightarrow \theta|y \sim Beta$ ... "Beta, Binomial $\Rightarrow$ Beta"

$x \sim \mathcal{N}$, $y|x \sim \mathcal{N} \Rightarrow x|y \sim \mathcal{N}$ ... "Normal, Normal $\Rightarrow$ Normal"

in $\mathcal{P}$, in $\mathcal{L}$ $\Rightarrow$ in $\mathcal{P}$

• **Def.** a random quantity $X \in \mathcal{X}$ is drawn from an exponential family with parameters $\theta \in \Theta$

$$p_x(x; \theta) \propto f_\cdot(x) \cdot e^{\vec{\phi}(\theta)\cdot\vec{u}(x)} = f(x)\, e^{\sum_{j=1}^{m} \phi_j(\theta)\, u_j(x)} = f(x)\, e^{\vec{\phi}\cdot\vec{u}(x)}$$

where:
- $f(\cdot): \mathcal{X} \to \mathbb{R}^+$ is a "base measure"
- $\vec{u}(\cdot): \mathcal{X} \to \mathbb{R}^m$ are a set of "features" $\quad \vec{u}(x) = \begin{bmatrix} u_1(x) \\ u_2(x) \\ \vdots \\ u_m(x) \end{bmatrix}$,
- $\vec{\phi}(\cdot): \Theta \to \mathbb{R}^m$ are the "natural parameters" $\quad \vec{\phi}(\theta) = \begin{bmatrix} \phi_1(\theta) \\ \phi_2(\theta) \\ \vdots \\ \phi_m(\theta) \end{bmatrix}$

and: $Z(\theta) = \int_{x\in\mathcal{X}} f(x)\, e^{\phi(\theta)\cdot u(x)}\, dx$ is the "partition function" or, "normalization function"

then: $p_x(x; \theta) = \frac{f(x)}{Z(\theta)} e^{\vec{\phi}(\theta)\cdot\vec{u}(x)} = e^{\log(f(x)) + \vec{\phi}(\theta)\cdot\vec{u}(x) - \log(Z(\theta))\cdot 1}$

$\log(Z(\theta))$: "log-partition function"

• an alternate perspective: $\mathcal{P} = \{\text{all } p(\cdot;\theta)\} = \{\text{all } p \text{ s.t. } \log(p(\cdot)) \in \text{span}(\vec{u}(\cdot), 1) + \log(f(\cdot))\}$

(span: all $\phi_1 u_1(x) + \phi_2 u_2(x) + \dots \phi_m u_m(x) - \phi_0 \cdot 1$, $\phi_0 = -\log(Z(\theta))$; x for which $Z(\theta) < \infty$)

- **Ex:**
  - $X \sim \exp(\lambda) \Rightarrow p_x(x;\lambda) \propto 1 \cdot e^{-\lambda x}$ for $x \geq 0$
    - $f(x) = \{1 \text{ if } x \geq 0,\ 0 \text{ if } x < 0\}$
    - $m = 1$
    - $\vec{\phi}(\lambda) = \lambda$
    - $\vec{u}(x) = x$
    - $z(\lambda) = \int_0^\infty e^{-\lambda x} = -\frac{1}{\lambda} e^{-\lambda x}\Big|_0^\infty = 1/\lambda \Rightarrow \log(z(\lambda)) = -\log(\lambda)$
  - $X \sim \text{Poiss}(\lambda) \Rightarrow p_x(x;\lambda) \propto \lambda^x/x! = e^{\log(\lambda)x - \log(x!)} = e^{\log(\lambda)x - \log(\Gamma(x+1))}$
    - $f(x) = \mathbb{1}(\text{is } x \in \mathbb{Z})$
    - $m = 2$
    - $\vec{\phi}(\lambda) = [\log(\lambda), -1]$
    - $\vec{u}(x) = [x, \log(\Gamma(x+1))]$
    - $z(\lambda) = \sum_{x=0}^{\infty} \frac{\lambda^x}{x!} = e^\lambda \Rightarrow \log(z(\lambda)) = \lambda$
  - $X \sim \text{Bin}(n,p) \Rightarrow p_x(x;n,p) = \binom{n}{x} p^x (1-p)^{n-x} = \binom{n}{x} e^{\log(p/1-p)x + \log(1-p)\cdot n}$ $e^{\log(p)x + \log(1-p)(n-x)}$
    - $f(x) = \mathbb{1}(x \in \{0,1,\dots n\}) \frac{\Gamma(n+1)}{\Gamma(x+1)\Gamma(n-x+1)}$
    - $m = 1$
    - $\vec{\phi}(p) = \log\left(\frac{p}{1-p}\right)$ = "log-odds"
    - $\vec{u}(x) = x$
    - $z(p) = (1-p)^{-n} \Rightarrow \log(z(p)) = -n\log(1-p)$
  - $X \sim N(\mu, \sigma^2) \Rightarrow p_x(x;\mu,\sigma^2) \propto e^{-\frac{1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2} = e^{-\frac{1}{2}\frac{1}{\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2}\frac{\mu^2}{\sigma^2}1}$
    - $f(x) = 1 \quad \forall\ x \in \mathbb{R}$
    - $m = 3$
    - $\vec{\phi}(\mu,\sigma) = [-\frac{1}{2}\frac{\mu^2}{\sigma^2}, \frac{\mu}{\sigma^2}, -\frac{1}{2}\frac{1}{\sigma^2}]$
    - $\vec{u}(x) = [1, x, x^2]$
    - $z(\phi)$ normalizes ... $\sqrt{2\pi}\sigma\, e^{-\frac{1}{2}\frac{\mu^2}{\sigma^2}} \Rightarrow \log(z(\phi)) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\frac{\mu^2}{\sigma^2}$

... and so on. Essentially all distributions ...

- **Sufficient Statistics:**
  - given $X^{(n)} = \{X_i\}_{i=1}^n \overset{iid}{\sim} p(x;\theta)$, the set of functions $\{s_j(x^{(n)})\}_{j=1}^m$ are **sufficient** if:
    - Θ are conditionally $\perp\!\!\!\perp$ of $X^{(n)}$ given $\vec{s}(x^{(n)}) = [s_1(x^{(n)}), \dots s_m(x^{(n)})]$
    - equivalently:
      - (i) $\Theta | x^{(n)} \sim \Theta | x^{(n)'}$ if $\vec{s}(x^{(n)}) = \vec{s}(x^{(n)'})$
      - (ii) $Lik(\theta; x^{(n)}) = Lik(\theta; x^{(n)'})$ at all θ if $\vec{s}(x^{(n)}) = \vec{s}(x^{(n)'})$
      - (iii) all of the information $x^{(n)}$ provide about θ is contained in $\vec{s}(x^{(n)})$ ... $I[\theta; x^{(n)}] = I[\Theta; \vec{s}(x^{(n)})]$

  - e.g. the sample mean and var. are sufficient for the population mean μ & variance of a $\mathcal{N}$

- what are the sufficient statistics for an exponential family?

$$Lik(\theta; x^{(n)}) = \prod_{j=1}^n Lik(\theta; x_j) = \prod_{j=1}^n f(x_j)\, e^{\vec{\phi}(\theta)\cdot\vec{u}(x_j) - \log(z(\theta))}$$

$$\propto e^{\sum_{j=1}^n \vec{\phi}(\theta)\cdot\vec{u}(x_j) - n\log(z(\theta))}$$

- so: $\frac{1}{n}\log(Lik(\theta; x^{(n)})) \overset{+c}{=} \frac{1}{n}\sum_{j=1}^n \vec{\phi}(\theta)\cdot\vec{u}(x_j) - \log(z(\theta)) = \vec{\phi}(\theta)\cdot\left[\frac{1}{n}\sum_{j=1}^n \vec{u}(x_j)\right] - \log(z(\theta))$

  ($\overset{+c}{=}$: up to a constant in θ)

  $\vec{s}(x^{(n)}) = \frac{1}{n}\sum_{j=1}^n \vec{u}(x_j)$ are sample averages of $\vec{u}$

- then: $Lik(\theta; x^{(n)}) = Lik(\theta; x^{(n)'})$ if $\vec{s}(x^{(n)}) = \vec{s}(x^{(n)'})$

...

$\vec{s}(x^{(n)}) = \frac{1}{n}\sum_{j=1}^n \vec{u}(x_j)$ are sufficient statistics.

- **Building Conjugate Models for/from Exponential Families:**
  - **Suppose that:**

$$p(y|\theta) \propto f(y)\, e^{\vec{\phi}(\theta)\cdot\vec{u}(y)}$$

(constant in θ)

then: $Lik(\theta; y^{(n)}) = e^{n(\vec{\phi}(\theta)\cdot\vec{s}(y^{(n)}) - \log(z(\theta))) + \sum_{j=1}^n \log(f(y_j))}$

$$\log(Lik(\theta; y^{(n)})) = n\left(\vec{\phi}(\theta)\cdot\vec{s}(y^{(n)}) - \log(z(\theta)) + \frac{1}{n}\sum_{j=1}^n \log(f(y_j))\right)$$

  - **Suppose:** $\theta \sim p_\theta(\theta)$
  - **then:** $p_{\theta|y^{(n)}}(\theta) \propto p_\theta(\theta)\, Lik(\theta; y^{(n)})$

$$\frac{1}{n}\log(p_{\theta|y^{(n)}}(\theta)) \overset{+c}{=} \frac{1}{n}\log(p_\theta(\theta)) + \vec{\phi}(\theta)\cdot\vec{s}(y^{(n)}) - \log(z(\theta)) + \dots$$

(only depends on y... constant)

so:

$$\frac{1}{n}\log(p_{\theta|y^{(n)}}(\theta)) \overset{+c}{=} \frac{1}{n}\log(p_\theta(\theta)) + \vec{\phi}(\theta)\cdot\vec{s}(y^{(n)}) - \log(z(\theta))$$

for conjugacy, need $p_{\theta|y^{(n)}} \in P$ for any $y^{(n)}$

$\Leftrightarrow$ need $\exp\left(n\cdot\frac{1}{n}\log(p_{\theta|y^{(n)}}(\theta))\right) \in P$ for any $y^{(n)}$

$\Leftrightarrow$ need $\frac{1}{n}\log(p_{\theta|y^{(n)}}(\theta)) \in Q$ for any $y^{(n)}$ where $Q = \{\log(p) \text{ for } p \in P\}$

... need Q containing $\frac{1}{n}\log(p_\theta(\theta)) + \vec{\phi}(\theta)\cdot\vec{s}(y^{(n)}) - \log(z(\theta))$ for all $y^{(n)}$

... for all possible $\vec{s}$ ...

$$Q \in \left\{\frac{1}{n}\log(p_\theta(\theta)) + \vec{\phi}(\theta)\cdot\vec{s} - \log(z(\theta)) \mid \vec{s}\right\}$$
$$\in \frac{1}{n}\log(p_\theta(\theta)) + \text{span}(\vec{\phi}(\theta)) - \log(z(\theta))$$

so: Q must contain the linear function space $\text{span}(\vec{\phi}(\theta))$

... idea, given $\mathcal{L} = \{p(y|\theta) \propto f(y)\, e^{\vec{\phi}(\theta)\cdot\vec{u}(y)}\}$

choose $P = \{p(\theta;\alpha,\beta) \propto g(\theta;\beta)\, e^{\vec{\alpha}\cdot\vec{\phi}(\theta) + \alpha_0\log(z_y(\theta))}\} \Rightarrow Q \overset{+c}{=} \text{span}(\vec{\phi}(\cdot), \log(z_y(\cdot))) + \ldots + \log(g(\cdot;\beta))$

then: $\log(p_{\theta|y^{(n)}}(\theta)) = \log(g(\theta;\beta)) + \vec{\alpha}\cdot\vec{\phi}(\theta) + n\,\vec{s}(y^{(n)})\cdot\vec{\phi}(\theta) - (1-\alpha_0)\log(z_y(\theta))$

$$= [\vec{\alpha} + n\vec{s}(\theta), \alpha_0 - 1]\cdot[\vec{\phi}(\theta), \log(z_y(\theta))] + \log(g(\cdot;\beta)) \in Q$$

$\therefore p_{\theta|y^{(n)}}(\cdot) \in P$ so P is conjugate for $\mathcal{L}$.

- **Moral:** if $p(y|\theta) \in \mathcal{L}$ where $\mathcal{L}$ is an exponential family w/ natural parameters $\vec{\phi}(\cdot)$ and sufficient statistics $\vec{s}(y^{(n)}) = \frac{1}{n}\sum_{j=1}^{n}\vec{u}(y_j)$ then, if P contains an exponential family on θ of the form

$$p(\theta) \propto g(\theta;\beta)\, e^{\vec{\alpha}\cdot\vec{\phi}(\theta) + \alpha_0\log(z_y(\theta))} = p(\theta;\alpha,\beta)$$

(an exp. family w/ features $[\vec{\phi}(\cdot), \log(z_y(\cdot))]$)

then, P is conjugate for $\mathcal{L}$, and (general "moment" update)

$$p(\theta|y^{(n)}) = p(\theta;\ \alpha + n\vec{s}(y^{(n)}),\ \alpha_0 - 1,\ \beta)\ .$$ ▨

- note: you may need to rearrange an initial parameterization to get the natural parameters & suff. stats.

e.g. $X \sim N(\mu, \sigma^2) \Rightarrow p_x(x; \mu, \sigma^2) \propto e^{-\frac{1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2} = e^{-\frac{1}{2}\frac{1}{\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2}\frac{\mu^2}{\sigma^2}}$
$= e^{-\frac{1}{2}(x-\mu)^2 \cdot \frac{1}{\sigma^2}}$

then: $\vec{s}(x^{(n)}) = \left[1, \frac{1}{n}\sum_{j=1}^{n} x_j, \frac{1}{n}\sum_{j=1}^{n} x_j^2\right]$, $\vec{\phi}(\mu, \sigma^2) = \left[-\frac{1}{2}\frac{\mu^2}{\sigma^2}, \frac{\mu}{\sigma^2}, -\frac{1}{2}\frac{1}{\sigma^2}\right]$

- if σ^2 known
  - ... $[\mu^2, \mu]$
  - ... P is the exp w/ features μ^2, μ for $\mu \in (-\infty, \infty)$
    includes all normal dist. on μ so
    normal is conjugate for unknown μ
- if μ is known ...
  - ... $[1/\sigma^2]$, $\vec{s}(x^{(n)}) = \frac{1}{n}\sum_{j=1}^{n}(x_j - \mu)^2$ = sample var.
  - ... P is the exp. family w/ features σ^{-2} for $\sigma \in [0, \infty)$
    includes all inverse gamma dist
    $(p(\sigma) \propto \underbrace{(\sigma^2)^{-(a-1)}}_{g(\sigma; a)} e^{-\underbrace{b/\sigma^2}_{b \cdot \phi(\sigma) = \sigma^{-2}}}$
    inverse gamma is conjugate for unknown σ^2. ▨

Stat 238 - Lecture 10 - M-estimation in Exponential Families

- Last time: a recipe for conjugate priors
- Today: practice:
  - (·) forming conjugate priors from exponential families
  - (··) expressive and maximally uninformative conjugate priors
  - (···) fitting exponential families
    (for fitting the prior, performing MLE or MAP)

(·) Forming a conjugate prior:

{ want to parameterize nonnegative functions...
parameterize their log... via the span (linear combo) of a collection of feature functions...

- last time: standard parameterization of distributions $\overset{\circ\circ}{\circ}$ e.g. $\log(p(x;\phi)) = \phi_0 + \phi_1 x + \phi_2 x^2 + \dots$
- Ex: N w/ known mean, unknown scale...

$$Y|\sigma^2 \sim N(\bar{y}, \sigma^2)$$

then:

$$p_{Y|\sigma^2}(y) = (2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2}\frac{(y-\bar{y})^2}{\sigma^2}}$$

$$= \exp\left(-\tfrac{1}{2}(\sigma^2)^{-1}\cdot(y-\bar{y})^2 - \tfrac{1}{2}\log(\sigma^2)\cdot 1 - \tfrac{1}{2}\log(2\pi)\right)$$

$$= \exp\left(\phi_1(\sigma^2)\cdot u_1(y) + \phi_2(\sigma^2)\cdot u_2(y) + C\right)$$

(only σ^2 dependent)

or

$$= \exp\left(\phi_1(\sigma^2)\cdot u_1(y) - \log(z(\sigma^2))\right)$$

= natural parameter
= feature function
= partition function
= constant

- building a conjugate prior for σ^2...
  - last class: pick p_{σ^2} from a family P that includes an exponential family over σ^2 whose features include, at least:
    - $\{\phi_j(\cdot)\}$ ... all natural parameters of the likelihood
    - $\log(z(\cdot))$ ... the log partition
- so, for σ^2: features $= \{\phi_1(\sigma^2) = \sigma^{-2},\ \phi_2(\sigma^2) = \log(\sigma^2)\}$

- so, P should include:

$$\{\text{all } p(\sigma^2; \alpha, \beta) \propto \underbrace{g(\sigma^2; \beta)}_{\text{arbitrary base measure}} e^{\alpha_1 \phi_1(\sigma^2) + \alpha_2 \phi_2(\sigma^2)} \text{ over all } g, \beta, \alpha\}$$

e.g. (i) pick $g(\sigma^2; \beta) = 1$ (uniform base measure)

then:

$$p(\sigma^2; \alpha) \propto e^{-\alpha_1 \sigma^{-2} - \alpha_2 \log(\sigma^2)} = (\sigma^2)^{-\alpha_2} e^{\frac{-\alpha_1}{\sigma^2}}$$

has the form of an inverse gamma density

- the inverse gamma family is a minimal conjugate family only as many free prior parameters as necessary
  - pro: in a sense, minimally informative (up to the coordinate system on the unknowns / base measure)
  - con: minimally expressive

(ii) maybe we want a more expressive/larger prior family... can add flexibility via $g(\sigma^2; \beta)$...

- pick: $g(\sigma^2; \beta)$ to be an gamma distribution

$$g(\sigma^2; \beta_1, \beta_2) \propto (\sigma^2)^{\beta_2} e^{-\beta_1 \sigma^2}$$

then P has 4 free parameters...

$$p(\sigma^2; \alpha, \beta) \propto (\sigma^2)^{\beta_2} e^{-\beta_1 \sigma^2} (\sigma^2)^{-\alpha_2} e^{-\alpha_1/\sigma^2}$$

$$= (\sigma^2)^{\beta_2 - \alpha_2} e^{-\left[\frac{\alpha_1}{\sigma^2} + \beta_1 \sigma^2\right]}$$

< not a standard dist. family

- more general prior family (recover old family w/ $\beta = 0$)
- pros: more expressive, still conjugate (just update α's w/ suff. stats after an observation)
- cons: more expressive ⇒ harder to fit, more specific ... demands more information

(..) expressive and minimally informative priors:

- trade-off: we can always expand P
  - at minimum P contains $\{ g(\theta)\, e^{\alpha \cdot [\phi(\theta),\, \log(z,(\theta))]} \}$
    if so, conjugate
  - larger P are more expressive ... minimize misspecification errors
  - larger P have more d.o.f. ...
    - can encode more specific information
    - ⇕
    - need more information to fit...

- how to balance this tradeoff?

(i) start w/ a minimal conjugate family (choose $g(\theta)$ to encode hard/absolute knowledge)

$$P = \{ g(\theta)\, e^{\vec{\alpha} \cdot [\vec{\phi}(\theta),\, \log(\tilde{z}_y(\theta))]} \text{ over all } \vec{\alpha} \}$$

- leading objective: conjugacy ... i.e. tractability (analytic ⇒ comp.)
  simplicity (interpretable updates)

(ii) identifies a minimal set of sufficient statistics for the prior...

- Sample Averages: $\frac{1}{m} \sum_{j=1}^{m} [\vec{\phi}(\hat{\theta}_j),\, \log(\tilde{z}_y(\hat{\theta}_j))] = \vec{s}_\theta$

  over a withheld or prior dataset; estimates to θ's from prior dataset

(iii) add any additional statistics of θ that:

a) we can estimate from sample averages on past data
b) we think expands P in a helpful direction

$$\vec{t}_\theta = \frac{1}{m} \sum_{j=1}^{m} \vec{u}_\theta(\hat{\theta}_j)$$

- then, select the "least informative" expansion of P s.t.

(i) $p(\theta) = 0$ if $g(\theta) = 0$ (obey hard constraints)
(ii) P includes the minimal conjugate family
(iii) $\mathbb{E}_{\theta \sim p_\theta}[\vec{u}_\theta(\theta)] = \vec{t}_\theta$ (match observed moments)

- this is a constrained optimization problem over the space of distributions
- for an appropriate notion of "information" it admits a simple soln...

- Def: the entropy (or self-information) in a dist p_x is:

$$H[X] = H(p_x) \equiv -\mathbb{E}_{X \sim p_x}[\log(p_x(x))]$$

- measures how uncertain we are about $X \sim p_x$ via the maximal rate at which we could learn about/resolve X under an optimal sequential search/inference procedure
  - the larger $H[X]$ the more effort needed to resolve $X \sim p_x$
  - reducing $H[X]$ corresponds to gaining info about X since we would now need to ask fewer questions to resolve X
- in this sense, maximizing $H(p_x)$ minimizes the information contained in the prior
  (if unconstrained, maximizing H selects the uniform dist)

- Thm: given $P = \{p$ s.t. (i) $p(\theta) = 0$ if $g(\theta) = 0$ (domain/support restriction), (ii) $\mathbb{E}_{\theta \sim p}[\vec{\phi}(\theta)]$ (minimal info from conjugacy demand), $\mathbb{E}_{\theta \sim p}[\log(z_y(\theta))] = \vec{s}_\theta$ and (iii) $\mathbb{E}_{\theta \sim p}[\vec{u}_\theta(\theta)] = \vec{t}_\theta\}$ (extra info)

then:

$p_* = \underset{p \in P}{\operatorname{argmax}}\{H(p)\}$ is a member of the exponential family:

$$p_*(\theta; \vec{\alpha}, \vec{\beta}) \propto g(\theta)\, e^{\vec{\alpha} \cdot [\vec{\phi}(\theta),\, \log(z_y(\theta))] + \vec{\beta} \cdot \vec{u}_\theta(\theta)}$$

w/ parameters $\vec{\alpha}, \vec{\beta}$ chosen to satisfy the moment constraints (ii) & (iii).

(...) ok, (..) gave a principled procedure for picking a conjugate prior family...

- to fit the family we need to know how to choose the prior parameters $\vec{\alpha}, \vec{\beta}$...
  - w/out a prior on $\vec{\alpha}, \vec{\beta}$ this is a frequentist parameter estimation problem... our only guide/tool is the likelihood principle

- how to pick $\vec{\alpha}, \vec{\beta}$ to maximize the likelihood of observed prior info (e.g. samples of past θ)? (or... $\hat{\theta}$'s)

- general problem: given a seq. of i.i.d. samples from an exponential family,

$$\{X_j\}_{j=1}^m \overset{iid}{\sim} p_x(\cdot\,;\vec{\alpha}), \quad p_x(x;\vec{\alpha}) \propto f(x)\exp\left(\vec{\alpha}\cdot\vec{u}_x(x)\right)$$

how can we find $\vec{\alpha}$ which maximize $p_x(x;\vec{\alpha})$?

- this is an MLE question for exponential families...
- use to find MLE when likelihood is exponential family...
- to fit parameters of conjugate priors...
- if we use an exponential family lik $\Rightarrow$ exponential family conjugate prior, $p_\theta \in P$
  $\Rightarrow$ exponential family posterior, $p_{\theta|y} \in P$

  so, also need for MAP estimation...

- an amazing answer:

w/ uniform base measure

- Thm: given a seq. of i.i.d. samples from an exponential family,

$$\{X_j\}_{j=1}^m \overset{iid}{\sim} p_x(\cdot\,;\vec{\alpha}), \quad p_x(x;\vec{\alpha}) \propto \underbrace{f(x)}_{=1}\exp\left(\vec{\alpha}\cdot\vec{u}_x(x)\right)$$

then: $\vec{\alpha}_* = \underset{\vec{\alpha}}{\operatorname{argmax}}\left\{p_x^{(m)}(x^{(m)};\vec{\alpha}) = \prod_{j=1}^m p_x(x_j;\vec{\alpha})\right\}$

must be chosen to satisfy the moment matching constraints:

$$\mathbb{E}_{x\sim p(\cdot;\alpha_*)}\left[\vec{u}_x(x)\right] = \vec{s}_x(x^{(m)}) = \frac{1}{m}\sum_{j=1}^m \vec{u}_x(x_j)$$

and α_* is unique if $\{u_j(\cdot)\}_{j=1}^m$ are linearly $\perp$ w.r.t. f.

MLE and "Method of Moments" are equivalent if we use the sufficient statistics!
(otherwise MoM $\neq$ MLE)

- Why !?
  - proof sketch:

(i) $\log(\tilde{z}(\vec{\alpha}))$ = the cumulant generating function of $p(\cdot;\vec{\alpha})$

⇓

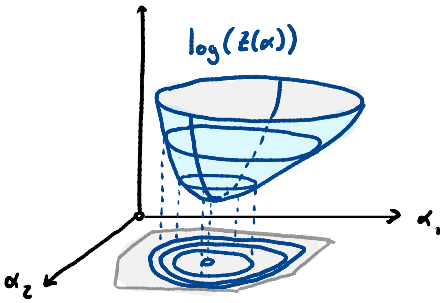
(ii) $\therefore \nabla_{\vec{\alpha}} \log(\tilde{z}(\vec{\alpha})) = \mathbb{E}_{x\sim p(\cdot;\vec{\alpha})}[\vec{u}_x(x)]$, and

⇓ $\nabla^2_{\vec{\alpha}} \log(\tilde{z}(\vec{\alpha})) = \mathbb{C}\mathrm{ov}_{x\sim p(\cdot;\vec{\alpha})}[\vec{u}_x(x)]$

(iii) $\mathbb{C}\mathrm{ov}[\ldots]$ is p.s.d. if $\{\vec{u}\}$ are lin. ⫫

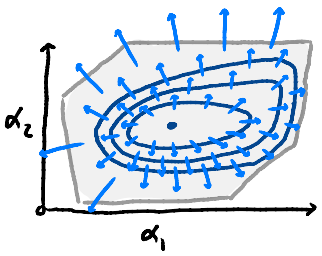
⇓

(iv) $\log(\tilde{z}(\cdot))$ is strictly convex in $\vec{\alpha}$



⇓

(v) the mapping $\vec{\alpha} \longrightarrow \mathbb{E}_{x\sim p(\cdot;\vec{\alpha})}[x] = \nabla_{\vec{\alpha}} \log(\tilde{z}(\vec{\alpha}))$ is invertible



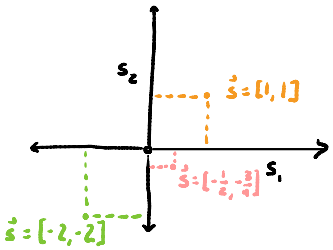
(vi) $\vec{\alpha}_* = \operatorname{argmax}_{\alpha} \{ f(\alpha) \exp(\vec{\alpha}\cdot\vec{s}_x(x^{(m)}) - \log(\tilde{z}(\vec{\alpha}))) \}$) if $f(\alpha) \propto 1$

$= \operatorname{argmax}_{\alpha} \{ \vec{\alpha}\cdot\vec{s}_x(x^{(m)}) - \log(\tilde{z}(\vec{\alpha})) \}$

then: $\nabla_{\vec{\alpha}} (\vec{\alpha}\cdot\vec{s}_x(x^{(m)}) - \log(\tilde{z}(\vec{\alpha})))\big|_{\vec{\alpha}=\vec{\alpha}_*} = 0$

⇓

(vii) $\vec{s}_x(x^{(m)}) = \nabla_{\vec{\alpha}} \log(\tilde{z}(\vec{\alpha})) \underset{\text{(ii)}}{=} \mathbb{E}_{x\sim p(\cdot;\vec{\alpha}_*)}[\vec{u}_x(x)]$



(v) + (vii) $\vec{\alpha}_*$ are the unique soln to the moment matching problem:

$$\mathbb{E}_{x\sim p(\cdot;\vec{\alpha}_*)}[\vec{u}_x(x)] = \vec{s}_x(x^{(m)}) \quad \blacksquare$$

Stat 238 - Weakly Informative Priors - NC - 02/27/25

- Priors as regularizers for regression models
- 2 examples

	regularization aim
1. Logistic/multinomial regression	- avoid implausibly large coeff.
2. Linear regression	- sparse coeff.

1. Logistic/Multinomial Regression:

- predictors $x_1, x_2, \dots x_m$
- n obs. comp.

$$X = \begin{bmatrix} & \overset{j^{th}\text{ predictor}}{\downarrow} & \\ \text{—} & \vec{x}(1)^T & \text{—} \\ & \vdots & \\ \text{—} & \vec{x}(n)^T & \text{—} \end{bmatrix}_{i^{th}\text{ unit}}$$

- observe categorical outcomes/data

$$Y = [Y_1, \dots Y_n], \quad Y_{(i)} \sim p_{Y|x_{(i)},\beta}$$

where $p_{Y|X_{(i)},\beta} = \text{Multinomial}(n_i, \underbrace{q(X_{(i)},\beta)}_{\text{unknown}})$ ← category prob.s

- Could aim to estimate q directly for each unit

but... we want to back up to X, β...

- Conjugate Prior on q's: $\text{Lik}(q; n_i, Y_{(i)}) \propto \prod_{c=1}^{|C|} q_c^{\sum_{j=1}^{n_i} \mathbb{1}(Y_{j(i)}=c)}$

$\to \log(\text{Lik}) = \sum_{c=1}^{|C|} \log(q_c) \cdot \left(\sum_{j=1}^{n_i} \mathbb{1}(Y_{j(i)}=c)\right)$

$\log(q_c) = \vec{\phi}(q)$

- e.g. see 5 of class 1, 7 of class 2, 2 of class 3, 1 of class 4

$\text{Lik}(q; n_i, Y_{(i)}) \propto q_1^5 q_2^7 q_3^2 q_4$

$p_\alpha(q) \propto \exp(\vec{\alpha} \cdot \vec{\phi}(q)) \propto \prod_{c=1}^{|C|} q_c^{\alpha_c}$

- Conjugate Prior on q's is a Dirichlet distribution

$$p(q) = \frac{1}{B(\vec{\alpha})} \prod_{c=1}^{|C|} q_c^{\alpha_c - 1}, \quad B(\vec{\alpha}) = \frac{\prod_{c=1}^{|C|} \Gamma(\alpha_c)}{\Gamma(\sum_{c=1}^{|C|} \alpha_c)}$$

- Uninformative on q's:
  - naive idea: Uniform over q s.t. $q_c \geq 0$, $\sum_{c=1}^{|C|} q_c = 1$
  - invariance principle:

$$\log(\text{Lik}(q; \dots)) = \sum_{c=1}^{|C|} \log(q) \cdot \left(\sum_{j=1}^{n_i} \mathbb{1}(Y_j(i) = c) \right)$$

$$\nabla_q^2 \dots = \nabla_q \left(\frac{1}{q} (\dots) \right) = -\frac{1}{q^2} (\dots)$$

$$- \dots$$

$$= \text{diag}\left(\frac{1}{q_c^2} \left(\sum_{j=1}^{n_i} \mathbb{1}(Y_j(i) = c) \right) \right)$$

$$-\mathbb{E}[\nabla_q^2 \dots] = \text{diag}\left(\frac{q_c \cdot n_i}{q_c^2} \right) = \text{diag}(n_i q_c^{-1})$$

for a single draw $\text{diag}(q_c^{-1}) = I(q)$

~~do $p(q) \propto \sqrt{\det(I(q))}$~~

multi-dimensional idea: $p(q) \propto \det(I(q)^{1/2}) = \prod_{c=1}^{|C|} q_c^{-1/2}$

$= \text{Dirichlet}(\vec{\alpha} = 1/2)$

(e.g. for binomial data $p(q) \propto q_1^{-1/2} q_0^{-1/2} = q_1^{-1/2} (1 - q_1)^{-1/2}$

"Jeffery's prior"

- equivalent to a conjugate prior w/ $1/2$ post obs. of each outcome.

- want to model q's as a function of predictors so we can pool share info between obs units w/ different predictors (usually $n_i = 1$ for all i if any cont. predictors)
- idea: the natural parameters are $\phi(q) = \log(q)$

  try to build a model that establishes a ~~funct~~ $\log(q)$ given X

  e.g.

  $$X(i) \cdot \beta = \log(q(i)), \text{ add a constant offset}$$

  $$X(i) \cdot \beta + \beta_0 = \log(q(i))$$

- then:

  $$q_c(i) = e^{\beta_0} e^{(X(i)\cdot\beta)_c}$$

  β_0 implied by normalization

  $$\sum_c q_c(i) = 1 \Rightarrow e^{\beta_0} = 1 / \sum_c e^{(X(i)\cdot\beta)_c}$$

  $$\beta_0 = -\log\left(\sum_{c=1}^{|C|} \exp(X(i)\cdot\beta(c))\right)$$

then: $q(i) = \underset{q}{\text{softmax}}(X(i)\cdot\beta)$

- Ex: 2 classes: only need q_1 since $q_0 = 1 - q_1$

$$q_1(i) = \frac{e^{X(i)\cdot\beta_1}}{e^{X(i)\cdot\beta_1} + e^{X(i)\cdot\beta_0}} = \left[1 + e^{X(i)\cdot(\overbrace{\beta_0 - \beta_1}^{\text{"}-\beta\text{"}})}\right]^{-1}$$

$$= \left[1 + \exp(-X(i)\cdot\beta)\right]^{-1}$$

$$= \text{logistic}(X(i)\cdot\beta)$$

now $Y(i) \sim \text{Bin}(n(i), q(i))$

where $q(i) = \text{logistic}(\eta(i))$

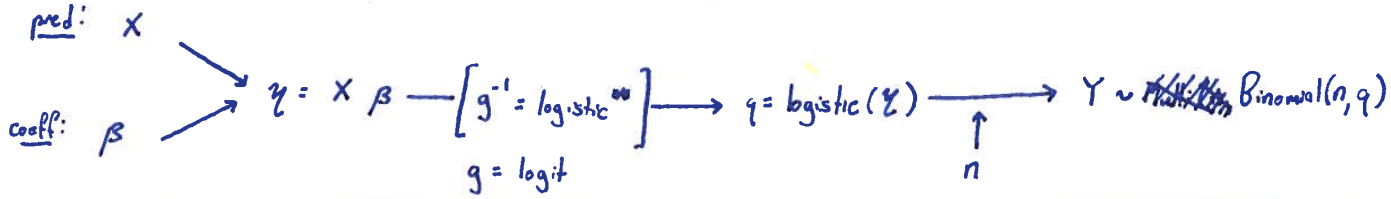
$\eta(i) = X(i)\cdot\beta$

logistic regression

ex of a GLM

Want: a prior on β...

how do we build this? can we back up our knowledge about a prior on q?



- Student's-t pdf density

$$p_\beta(\beta; s, \nu) \propto \left(1 + \frac{(\beta/s)^2}{\nu}\right)^{-\frac{(\nu+1)}{2}} \propto \left(\nu + (\beta/s)^2\right)^{-\frac{(\nu+1)}{2}}$$

Cauchy: $\nu = 1$

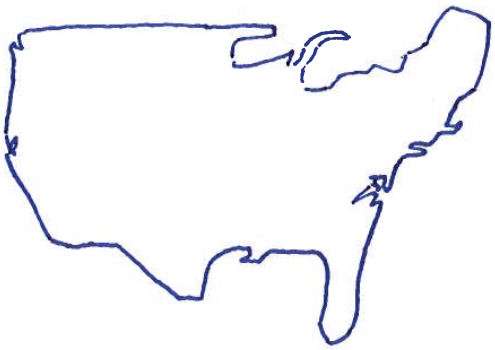
$$p_\beta(\beta; s, 1) \propto \left(1 + (\beta/s)^2\right)^{-1}$$

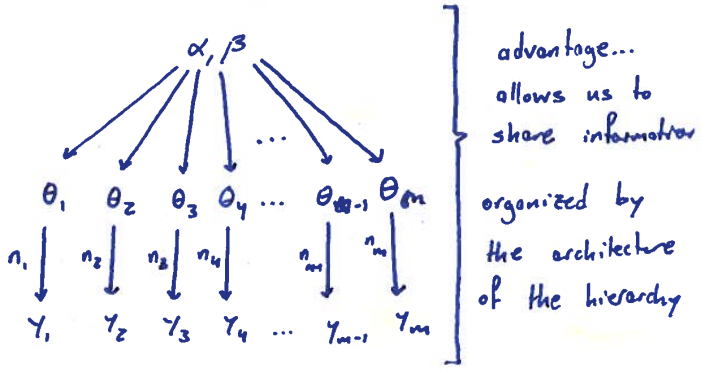
Stat 238 - Lecture 13 - Hierarchical Models - 03/04/25

- **Learning Goals:**
  - What is a hierarchical model?
  - ~~de Finetti's Thm.~~
  - Exchangeability & De Finetti's Thm
  - Example hierarchical models

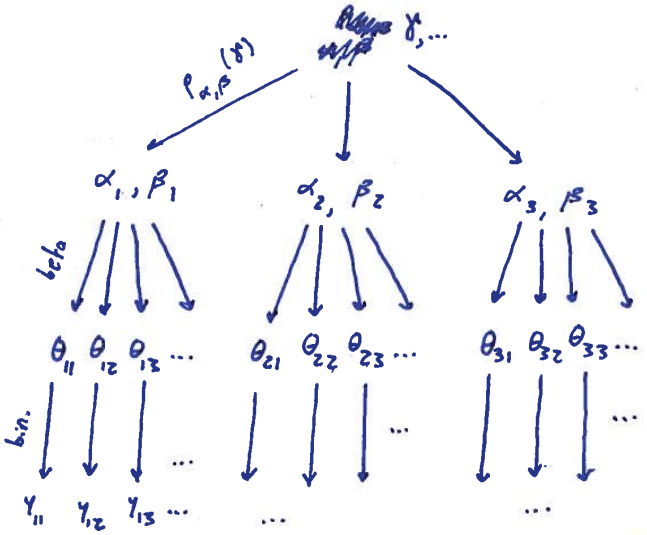
- **Logistics:**
  - Quiz 1, 2, 3 retake form
  - Quiz 4 next Tuesday, practice up by tomorrow night at the latest
  - HW 3 this Thursday
  - Project mini's up by Thursday

- **Hierarchical Models:**
  - **Example:** Kidney Cancer model from lab 2, ~~or cat tumor example in BDA 5~~





- could imagine changing the hierarchy ...
  - e.g. group by some classification of the countries first



- e.g. make "taller" (add a prior on α, β) (hyperprior)



- What assumption are we making when we group ... a set of ~~nodes~~ variables in the hierarchy as children of ~~set~~ set of variables one level up in the hierarchy?

- ~~two standard ideas~~

(•) conditional ⫫ ⇒ define

(••) exchangeable ⇒ ...

- **Definition**: X & Y are conditionally $\perp\!\!\!\perp$ given Z if

$$X \mid Z=z \perp\!\!\!\perp Y \mid Z=z \quad \forall z$$

- **Definition**: a set of r.v.'s $\{X_1, X_2, \dots X_n\}$ are exchangeable

if the joint distribution of $\{X_1, \dots X_n\}$ is invariant to (symmetric under) any permutation of the node labels...

$$p(X_1=x_1, X_2=x_2, \dots X_n=x_n) = p(X_1=x_{\sigma(1)}, X_2=x_{\sigma(2)}, \dots X_n=x_{\sigma(n)})$$

- essentially an interchange/permutation symmetry on the ~~elements~~ indices of the sequence

- Ex. if $X^{(n)} = \{X_i\}_{i=1}^n$, $X_i \overset{iid}{\sim}$ i.i.d. then $X^{(n)}$ are exchangeable

- Ex. if ~~$\theta^{(n)} = \{\theta_i\}_{i=1}^n$, $\theta_i \overset{iid}{\sim} \text{Beta}(\alpha,\beta)$~~ $\theta \sim \text{Beta}(\alpha,\beta)$

Then $Y_i \overset{iid}{\sim} \text{Bin}(n, \theta_i)$ form an exchangeable seq.

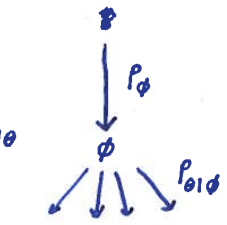
but $Y_i \not\perp\!\!\!\perp Y_j$. since observing $Y_1, \dots Y_{n-1} \xrightarrow{\text{inform}} \theta \xrightarrow{\text{inform}} Y_n$

} exchangeable r.v.'s need not be $\perp\!\!\!\perp$

this is a common construction...

- ~~draw $\phi \sim p_\phi$~~
- ~~then draw~~

- draw $\theta \sim p_\theta$
- then draw $\{Y_i\}_{i=1}^n \overset{iid}{\sim} p_{Y|\theta}$



Then Y are exchangeable w/ joint dist = a mixture (average of dist) of joint dist. of i.i.d seq's

$$p(y^{(n)}) = \mathbb{E}_{\theta \sim p_\theta}\left[p(y^{(n)}|\theta)\right] = \mathbb{E}_{\theta \sim p_\theta}\left[\prod_{j=1}^n p_{Y|\theta}(y_j)\right].$$

~~Common~~ 1929

- **Thm (de Finetti)**:

1. an infinite seq. of r.v.'s $X_1, X_2, X_3, \dots$ is exchangeable if any finite subsequence $\{X_{i_1}, X_{i_2}, \dots X_{i_n}\}$ is exchangeable

~~2. where $X_i \in \{0,1\}$ form Lmax~~ then

7. • $X_1, X_2, \ldots \in \{0,1\}$ are exchangeable if and only if $\exists$ some probability measure ρ_θ on $[0,1]$ s.t. $X_1, \ldots$ are identically distributed to the seq. $Y_1, \ldots$ where $Y_1, \ldots$ are drawn i.i.d. cond. on θ from Bernoulli(θ) for $\theta \sim \rho_\theta$.

any ∞ seq. of binary r.v.

• e.g. all ∞ seq. of exchangeable binary r.v.'s may be represented as ~~draws~~ mixtures of i.i.d. random sequences (Bernoulli) ~~###~~

binary

• all ∞ seq. of exchangeable binary r.v.'s may be represented as draws from a hierarchical model where the r.v.'s are conditionally $\perp\!\!\!\perp$, conditional on a random parameter...

• Generalizations: $X_1, X_2, \ldots \in \mathcal{X}$

• Thm (de Finetti, 1937):

• also applies to ∞ exchangeable seq. of real r.v.'s (any ∞ exch seq. of $X_i \in \mathcal{X} = \mathbb{R}$ may be represented as a mixture of i.i.d. sequences)

• Thm (Hewitt and Savage, 1955):

• also applies to "Baire measurable" r.v.'s $\in \mathcal{X}$ = a compact Hausdorff-space

extends to very general cases

largely need $\mathcal{X}$ to be a complete space

• Thm (Alam, 2020):

• also applies to $\mathcal{X}$ = a Hausdorff space provided the common marginal on X_i is a Radon measure

• Why ∞ sequences?

• Counterexample: if $[X_1, X_2] \in [0,1]^2$, $P_{X^{(2)}} = \begin{matrix} 0 \\ 1 \end{matrix}\begin{bmatrix} 0 & 1 \\ 0 & 1/2 \\ 1/2 & 0 \end{bmatrix}$

can't construct as any mixture of i.i.d. joints $\begin{bmatrix} p_0^2 & p_1 p_0 \\ p_0 p_1 & p_1^2 \end{bmatrix}$

5

- the de-Finetti - Hewett- Savage thm morals:

(i) representation theory justification for prot Bayesian modeling...

- if we cannot distinguish $X_1, X_2, \ldots$
- and believe they are random (no dist. specified)

then we behave, under any prob. model as if we believed $X_1, \ldots$ were drawn i.i.d. from a parametric distribution w/ an unknown parameter drawn from a prior

⇒ if our view of the prob. of a sequence of events is subjective but exchangeable then we behave as if the events were $\perp\!\!\!\perp$ w/ an objective underlying sampling distribution, where our uncertainty about the prob. dist. over the seq. ~~can be~~ is expressed as a subjective prior over the unknown sampling dist.

⇒ "starting from ~~purely subjective~~ a specific, but purely subjective expression of convictions, we should act as if events were driven by objective chances."

(ii) exchangeability generally follows from ignorance (indistinguishability of the units)

- see "objections to exchangeable models"
  - units actually differ
  - but are exchangeable if we can't tell them apart
  - should treat as if drawn from a common distribution
  - "objecting to exchangeability for modeling ignorance is no more reasonable than objecting to i.i.d. models for samples from a common population, or to displaying points in a scatterplot w/out individual labels"

(iii) exchangeability directs architecture of hierarchical models...

- group variables as the children of a parent variable/variables (siblings) if indistinguishable or we believe them to be exchangeable, ≈ exchangeable, or, we choose to discard (not model) their distinctions

Exercise w/ class: go back to kidney cancer example and redesign architecture

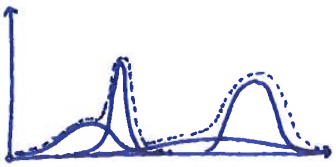
4

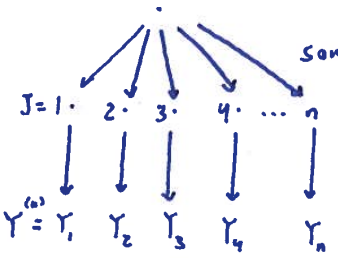
- **Justifications/Motives for Hierarchical Modeling:**

1. representation theory (veracity in structure ~ weak information (up to weak information) model w/ minimal inf.
   rhetorical advantage → structures model according to exchangeability communicable, but... still need a distributional assumption on each branch)

2. compromises expressivity and tractability:
   - builds the prior as a composition/mixture of distributions
   - usually the pieces are individually tractable (e.g. conjugate)
   - but, the architecture is, as a whole expressive

- **Ex:** gaussian mixture w/ n modes





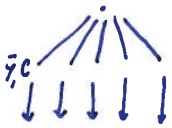
sample mode index $J \sim p_J$

where $Y \mid J=j \sim N(\mu_j, C_j)$

- gaussian mixtures are dense in the space of distributions
  (any p can be approximated to arbitrary accuracy by a finite gaussian mixture for sufficiently large n)

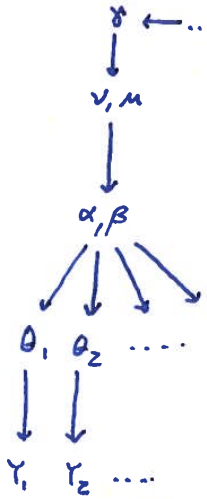
expressive!

- like other structured statistical models (e.g. neural nets) may fit w/ more, even many more, unknown parameters than data points
  - benign overfitting...
    - constraints in info sharing from model architecture
    - marginalize over uncertainty (have, in principle, a full posterior)



⚠ be careful interpreting parameters that are buried too deep in the architecture, the high dimensional inference may overfit even if the marginal does not... e.g. exchange symmetry in posterior over Gaussian mixture)

- **A Key question:**
  - when to stop? can always add a hyperprior



when should this end?

(i) if we don't branch, can marginalize over the nuisance variables to collapse the hierarchy... in principle stop at last branch point ... can still be useful to express as mixtures

(ii) arrive at a prior that we
- (·) know for sure from other data
- (··) choose subjectively (un or weakly informative)
- (···) lose out on trade-off between expressivity, tractability, and simplicity

⇒ switch to MLE of unknown hyperprior parameters

MAP estimator:

- **Definition:** $\hat{x}_{MAP}(y) = \arg\max_{x \in \mathcal{X}} \{ p_{x|Y}(x) \}$

- **Consistency:** if $\mathcal{X}$ is finite, $Y^{(n)} \overset{iid}{\sim} p_{y|x_*}$ then $\hat{x}_{MAP}(Y^{(n)}) \longrightarrow x_*$ w/ prob. 1.
  - if the "mild" regularity conditions hold from unit 1 then $\hat{x}_{MAP}(Y^{(n)}) \longrightarrow x_*$ w/ prob. 1 as $n \to \infty$.

  Q: for class ask what these are

- **Rational:** if $\mathcal{X}$ is finite, Bayes optimal for 0, 1 (right/wrong) loss
  if $\mathcal{X}$ is continuous, optimal $\uparrow$ under a sequence of refining coarse-grainings

- **Observation:** under the regularity conditions, when n is large, the likelihood is sharply peaked, so is the posterior...
  - can approximate for x near $\hat{x}_{MAP}(y)$ via
    - 2nd order Taylor of the log posterior...

$$\log(p_{x|Y=y}(x)) \underset{+C}{=} -\frac{1}{2}(x - \hat{x}_{MAP}(y))^T H(\hat{x}_{MAP})(x - \hat{x}_{MAP}(y))$$

where:

$$H(\hat{x}_{MAP}) = \nabla_x^2 \overset{\log(\ )}{p(x|y)}\Big|_{x=\hat{x}_{MAP}} = \begin{bmatrix} \partial_{x_1}^2 & \partial_{x_1}\partial_{x_2} & \cdots \\ \partial_{x_2}\partial_{x_1} & \partial_{x_2}^2 & \\ \vdots & & \ddots \end{bmatrix} \log(p(x|y))\Big|_{x=\hat{x}_{MAP}}$$

- notice: if the likelihood factors into a product of n i.i.d. draws

$$H(\hat{x}) = \nabla_x^2 \log(p_x(x))\Big|_{x=\hat{x}} + n \underbrace{\left(\frac{1}{n} \mathbb{E}_{J \sim Uni[1,n]} \left[\nabla_x^2 \overset{!\,\hat{x}}{\log(p_{Y_J|x}(Y_J))} \right] \right)}_{\xrightarrow[n\to\infty]{} I[\hat{x}]}$$

$$\frac{1}{n}(\ldots) \to \underbrace{\frac{1}{n} \nabla_x^2 \log(p_x(x))\Big|_{x=\hat{x}(Y^{(n)})_{MAP}}}_{\text{pert. to asymptotic normal due to prior}} + I[\hat{x}_{MAP}(Y^{(n)})]$$

Stat 238 - Lecture 15 - Bayesian Point Estimation - 03/11/25

- logistics: 1. reduced workload plan
  2. HW 3 up
  3. Quiz 4 today, 5 next week

- learning goals:
  - MAP estimation, consistency review
  - Laplace approximation from asymptotic normality
  - convergence theory for Laplace approx
  - standard methods:
    - if log-posterior is convex ⇒ convex optimizers (explicit gradients, numerical, or auto-diff)
      Gradient descent or Newton
    - if data is i.i.d. the objective factors, SGD for large problems (but, only if lik factors)
  - warning: multimodality
    - MAP framework is inappropriate
    - multiple modes may survive in mixture models w/ indistinguishable class assignment or for sparsity promoting models

- initialization tricks:
  - run many opt. initialized from prior if we can sample from prior
  - initialize from MLE if analytically available & use simulated annealing to gradually turn the prior on

- Outline:

(1) logistics (2) non-conjugate review (hit de Finetti)

(3) define MAP estimator
- ask about consistency/decision theoretic justification

(4) Laplace approximation to the posterior
[- convergence theory]

(5) standard numerics
- objective (log post)
- initialization
- optimizer

(6) warning

Then:

$$\left[X \mid Y^{(n)} = y^{(n)} \overset{\approx}{\sim} N\left(\hat{x}_{MAP}(y^{(n)}), \underbrace{-H(\hat{x}_{MAP}(y^{(n)}))^{-1}}_{= -\left(\nabla_x^2 \log\left(p_{X|Y^{(n)}}(x \mid Y^{(n)} = y^{(n)})\right)\Big|_{x=x_*}\right)^{-1}} \right) = \text{"Laplace Approximation"} \right]$$

$$\xrightarrow[n\to\infty]{} n\left(I\left[\hat{x}_{MAP}(y^{(n)})\right] + \underbrace{\mathcal{O}(1/n)}_{\text{prior term.}}\right)$$

↳ useful as simplest post. approx. (as proposal for rejection/importance sampling, initialize for samplers)

- often available analytically when the prior & likelihood are built as a composition of parametric functions.
- need only take derivatives.

- **Consistency of Laplace Approximation:**
  - generally, convergence results in the sense that some div. b/tween post & approx vanishes almost surely at a specified rate in n, or is bounded above almost surely by a decaying function in n.
  - **Ex:** if $Y^{(n)}$ are drawn i.i.d. and the usual "mild conditions" the Laplace approximation $p^{(Lap)}_{X|Y^{(n)}}$ converges, in Hellinger distance, to the true posterior (almost surely w.r.t. to sample sequences $Y^{(n)}$)

$$d_H\left(p_{X|Y^{(n)}}, p^{(Lap)}_{X|Y^{(n)}}\right) = \mathcal{O}(n^{-1/2})$$

  - where:

$$d_H(p,q) = \sqrt{1 - \int_X \sqrt{p(x)q(x)}\,dx}$$

($=0$ iff $p = q$ a.e., $\in [0,1]$ ($=1$ if supports are disjoint & is a bounded metric on prob. space)

$$d_H(p,q) \le d_{TV}(p,q) \le \sqrt{2}\, d_H(p,q)$$

  - Total variation distance:

$$d_{TV}(p,q) = \sup_A \left| \int_{x \in A} (p(x) - q(x))\,dx \right| = \sup_A \left\{ |P(x \in A) - Q(x \in A)| \right\}$$

(the largest absolute difference in prob. applied to any event)

$$= \underbrace{\frac{1}{2} \sum_x |P(x) - Q(x)| \quad \text{or} \quad \frac{1}{2}\int |p(x) - q(x)|\,dx \quad \text{when continuous}}_{\frac{1}{2}\|p - q\|_1}$$

(21)

- Corollary:
  - Bernstein von-Mises Theorem: under given $Y^{(n)} \overset{iid}{\sim} P_{Y|x_*}$ then, under "mild regularity" conditions

$$d_{TV}\left(P_{x|Y^{(n)}}, N(\theta_{x_*}, (n\, I[x_*])^{-1})\right) \longrightarrow 0$$

almost surely w.r.t. sequences $Y^{(n)}$.

  - finite sample bounds also exist... (see references)

- Computation:

[- Find $\hat{x}_{MAP} = \operatorname{argmax}\{P_{x|Y=y}(x)\} = \operatorname{argmin}\{-\log(P_{x|Y=y}(x))\}$
  constrained to x s.t. $P_x(x) > 0$]

  - objective: usually, minimize $-\log(\text{posterior})$

$$f(x|y) = -\tfrac{1}{n}\log(p(x|Y=y)) = \underbrace{-\tfrac{1}{n}\log(\text{prior})}_{\text{"regularization"}} - \underbrace{\tfrac{1}{n}\log(\text{likelihood})}_{\text{"fidelity"}}$$

↑ more stable

note factorization if i.i.d. data

  - domain: if the support of P_x is awkard, reparameterize

(e.g. $x \geq 0 \Rightarrow$ use $x = e^z$, $z = \log(x)$ optimize over z

$x \in [0,1] \Rightarrow$ use $x = \text{logistic}(z)$, $z = \text{logit}(x)$ optimize over z

$X \in$ p.s.d. symm. matrices $\Rightarrow$ $X = RR^T$, optimize over R (rank-r))

- initialization:
  - if we can sample from P_x run many optimizers initialized from $X \sim P_x$
  - if we can compute the MLE analytically then adopt an annealing approach
    - initialize at MLE
    - iteratively resolve for $\hat{x}_{MAP}(y; \hat{\lambda})$ using $f^{(\lambda)}(x|y) = -\tfrac{\lambda}{n}\log(\text{prior}) - \log(\text{lik})$ for $\lambda = 0 \longrightarrow 1$
    - "annealing" is using prior $P_x(x)^\lambda$ for $\lambda \in [0,1]$. "flattens prior"

(3)

- methods:
  - if f convex $\Rightarrow$ convex optimizer
  - often, multiple $\hat{\theta}$ optimizers in Θ (convex)
    - e.g.
      - gradient descent
      - Newton trust-region search
      - Langevin gradient descent
      - if lik factors, SGD, ADAM

      (at least to return only one maximizer)

derivatives:
- often available analytically
  don't need evidence
- if not:
  - use num. diff.
  - or use auto-diff.

- Warning: not appropriate if we expect a multimodal posterior

(e.g. Gaussian mixture model

save for T.

$\gamma \sim \rho_\gamma$

$J \sim \rho_J$

$\mu, C \mid J=j \sim \rho_{\mu,C}(\cdot \mid \gamma_j)$

μ_γ, C_γ

$N(\mu_\gamma, C_\gamma)$

$Y \sim N(\mu, C)$

(i) draw $J \sim \rho_J$

(ii) draw $\mu, C \mid J=j \sim \rho_{\mu,C}(\cdot \mid \gamma_j)$

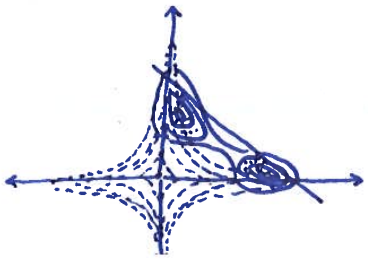
(iii) draw $Y \mid \mu, C \sim N(\mu, C)$

problem: if $\gamma_j = \gamma_i$ for all $i, j \in \mathcal{I}$
then the posterior must be symmetric under all permutations of class labels in $\mathcal{I}$
w/ K classes usually have $K!$ modes, all well separated)

* if the posterior is reflection symm can extend to multimodal estimate via reflection
only need 1 mode
can constrain $\mu_i \leq \mu_{i'}$ if $i \leq i'$

(e.g. sparsity promoting priors

highly nonconvex log prior
if lik is indiscriminate along some direction
then many modes persist



- How sensitive is $\hat{x}_{MAP}$ to the data / prior assumptions?

  - let $f(s, \gamma)$ be the log post. w/ suff. stats s & prior par. γ

  - at optimizer:

$$\nabla_x f(x; s, \gamma)\Big|_{x=\hat{x}(s,\gamma)} = 0$$

  - implicitly differentiate:

let $J_{\hat{x}}(s, \gamma)$ = the Jacobian of $\hat{x}(s, \gamma)$ w.r.t. s, γ at s, γ

$$\left(J_{\hat{x}}(s,\gamma)_{ij} = \partial_{\phi_j} \hat{x}_i(\phi)\right)$$

let $H_f(x; s, \gamma)$ = the Hessian of f at x given s, γ

let $J_{\nabla_x f}(s, \gamma; x)$ = the Jacobian of $\nabla_x f$ w.r.t. s, γ at x

Then:

$$H_f(\hat{x}(s,\gamma); s, \gamma)\, J_{\hat{x}}(s, \gamma) = -J_{\nabla_x f}(s, \gamma; \hat{x})$$

$$\Rightarrow \boxed{J_{\hat{x}}(s, \gamma) = -\underbrace{H_f(\hat{x}(s,\gamma); s, \gamma)^{-1}}_{\text{Cov of Laplace}} J_{\nabla_x f}(s, \gamma; \hat{x})}$$

Consequence: $s \to s + \delta s$, $\gamma \to \gamma + \delta\gamma$

$$\hat{x}_{MAP}(s+\delta s, \gamma+\delta\gamma) \simeq \hat{x}_{MAP}(s, \gamma) - H_f(\hat{x}(s,\gamma); s, \gamma)^{-1} \{\nabla_x \left(\nabla_s f(\hat{x}; s, \gamma) \cdot \delta s + \nabla_\gamma f(\hat{x}; s, \gamma) \cdot \delta\gamma \right)$$

$\hat{x}_{MAP}$ tends to be sensitive in the directions where the Laplace approx predicts large post. var

Stat 238 - Lecture 17 - MAP Estimation in Structured Models

- Logistics:
  - Quiz 5 today
  - Quiz 4 retakes
  - HW 3 posted, due after break
  - Lab 7 due after break

- Learning Goals:
  - Coordinate-Ascent Example (joint MAP)
  - Expectation-Maximization (marginal MAP)

- MAP estimation in structured models:

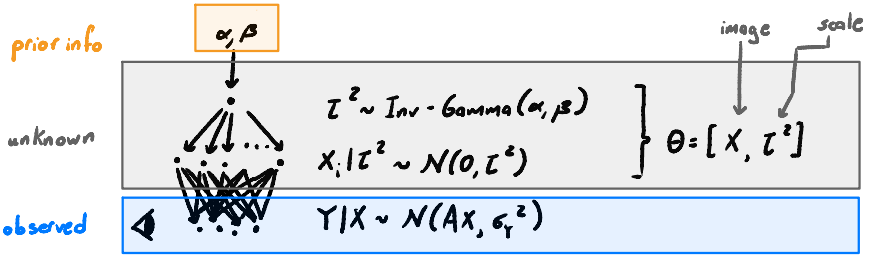
1. Coordinate Ascent:

setting: observe $Y=y$, $Y \sim p_{Y|\theta}$ where $\theta \sim p_\theta$ } want the joint MAP: $\hat{\theta}_{MAP}(y) = \arg\max_\theta \{ p(\theta | Y=y) \}$

- assume that θ partitions into blocks:

$$\theta = [\theta^{(1)}, \theta^{(2)}, \dots \theta^{(m)}] \quad \text{where} \quad \theta^{(b)} = [\theta_1^{(b)}, \theta_2^{(b)}, \dots]$$

e.g. lab 7's model:



- idea: iterate over the blocks optimizing over the unknowns w/in the block while holding the rest fixed

e.g.

(1) optimize over x w/ fixed τ ⇄ (2) optimize over τ w/ fixed x

- general algorithm:

  (·) initialize $\theta^{(0)}$

  (··) fix a schedule $\{s(t)\}^{\cdots}$, $s(t) \subseteq \{1, 2, \dots |\theta|\}$ } usually s is chosen to cycle through the blocks
  s.t., $\forall t \;\; \exists \; t' > t, \;\; t' < \infty \;\; \bigcup_{h \in [t,t']} s(h) = \{1, 2, \dots |\theta|\}$

  (···) for $t = 0$ until convergence:

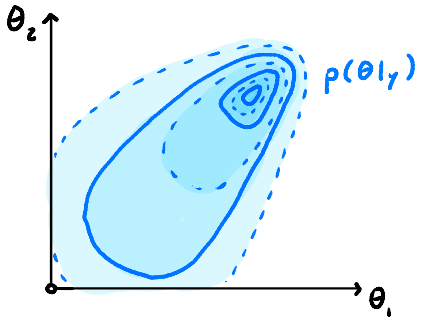
  (a) let $\theta_{s(t)} = \{\theta_j \mid j \in s(t)\}$, $\theta_{-s(t)} = \{\theta_j \mid j \notin s(t)\}$

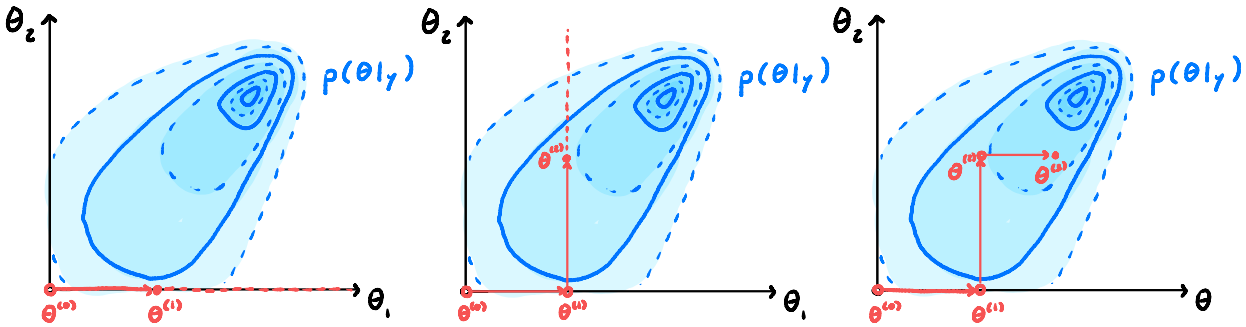
  (b) set:

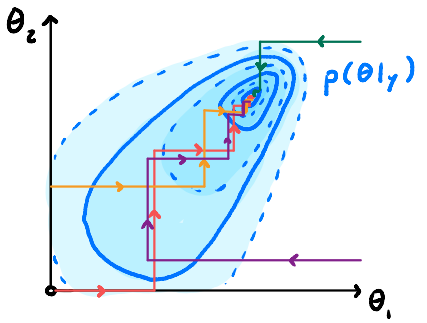
$$\theta^{(t+1)}_{s(t)} = \underset{\theta_{s(t)}}{\operatorname{argmax}} \{p(\theta_{s(t)}, \theta^{(t)}_{-s(t)} \mid y)\}, \quad \theta^{(t+1)}_{-s(t)} = \theta^{(t)}_{-s(t)}$$

  ↑ optimize over; ↳ hold fixed ↗

- Ex: $\theta = [\theta_1, \theta_2]$, $s = \{1, 2, 1, 2, 1, 2, \dots\}$







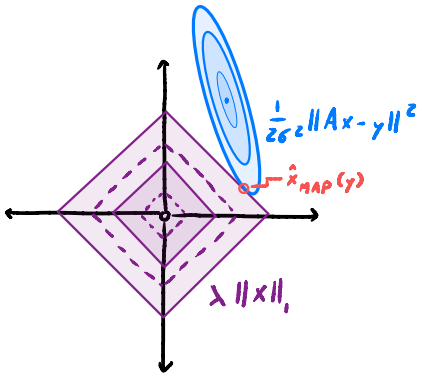
- Convergence: Coordinate-ascent will converge from any initial condition in B if:
  - $\exists$ a unique optimizer $\in B$
  - B is a convex region, $\text{Vol}(B) > 0$
  - the objective is strictly convex/concave on B
  - the objective is 2nd differentiable on B

- Ex: Lasso / L_1 regularization

$$\left[\begin{array}{ll} X_i \overset{iid}{\sim} p_X \quad \text{where} \quad p_X(x_i) \propto e^{-\lambda|x_i|} & \Rightarrow p_X(x) \propto e^{-\lambda \sum_i |x_i|} = e^{-\lambda\|x\|_1} \\ Y = AX + \xi, \ \xi \sim \mathcal{N}(0, \sigma^2 I) & \Rightarrow p_{Y|X}(y) \propto e^{-\frac{1}{2\sigma^2}\|Ax-y\|_2^2} \end{array}\right]$$

so:

$$\hat{x}_{MAP}(y) = \arg\max_x \{p(x|y)\} = \arg\min_x \{-\log(p(x|y))\}$$
$$= \arg\min_x \left\{\frac{1}{2\sigma^2}\|Ax-y\|_2^2 + \lambda\|x\|_1\right\}$$



- no closed form solution for general A
- but... a convergent coordinate ascent algorithm

- idea: try $S = \{1, 2, 3, \ldots n, 1, 2, 3, \ldots n, \ldots\}$
  (just cycle through the unknowns)

- subproblem: optimize w.r.t. x_i, fixing x_{-i}

- subproblem: minimize $\frac{1}{2\sigma^2}\|Ax-y\|_2^2 + \lambda\|x\|_1$
  w.r.t. x_i
  for x_{-i} fixed

- notice: $Ax = \sum_j a_j x_j = a_i x_i + \sum_{j\neq i} a_j x_j = a_i x_i + \begin{bmatrix} & \\ A_{-i} & \\ & \end{bmatrix}\begin{bmatrix} \\ x_{-i} \\ \end{bmatrix}$

$$A = \begin{bmatrix} | & | & & | & & | \\ a_1 & a_2 & \cdots & a_i & \cdots & a_n \\ | & | & & | & & | \end{bmatrix} \Rightarrow \begin{bmatrix} | & | & & | & & | \\ a_1 & a_2 & \cdots & a_{i-1} & & a_n \\ | & | & & | & & | \end{bmatrix}\begin{bmatrix} \\ a_i \\ \\ \end{bmatrix}\begin{bmatrix} | & & | \\ a_{i+1} & \cdots & a_n \\ | & & | \end{bmatrix}$$

$$\Downarrow \qquad \Swarrow$$

$$\begin{bmatrix} | & & | & | & | \\ a_1 & \cdots & a_{i-1} & a_{i+1} & \cdots & a_n \\ | & & | & | & & | \end{bmatrix} \equiv A_{-i}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \Rightarrow \begin{bmatrix} \\ [\,] \\ \\ \\ [\,] \\ \\ \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_{i+1} \\ \vdots \\ x_n \end{bmatrix} \equiv x_{-i}$$

so: $\frac{1}{2\sigma^2}\|Ax-y\|_2^2 = \frac{1}{2\sigma^2}\|a_i x_i - r_i\|_2^2$ where $r_i = y - A_{-i}x_{-i}$

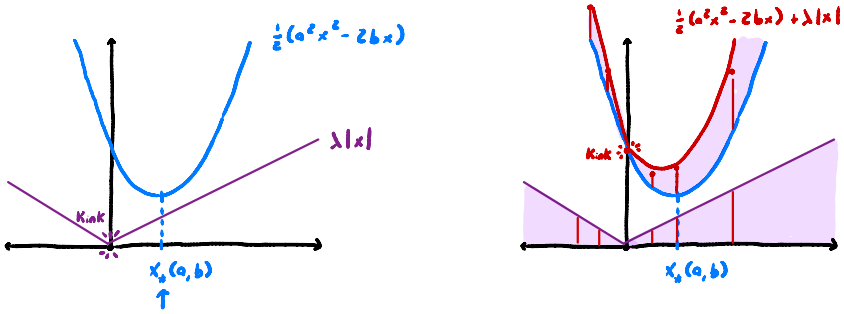
and: $\lambda\|x\|_1 = \lambda\sum_j |x_j| = \lambda|x_i| + \lambda\sum_{j\neq i}|x_j| = \lambda|x_i| + \lambda\|x_{-i}\|_1$ (does not depend on x_i, can ignore when optimizing w.r.t. x_i)

so: we want to find $\hat{x}_i = \arg\min_{x_i}\left\{\frac{1}{2\sigma^2}\|a_i x_i - r_i\|_2^2 + \lambda|x_i|\right\}$

so: we want to find $\hat{x}_i = \arg\min_{x_i} \left\{ \frac{1}{2\sigma^2} \| a_i x_i - r_i \|_2^2 + \lambda |x_i| \right\}$

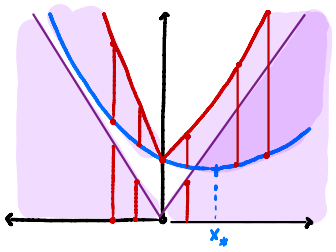
$$\frac{1}{2\sigma^2} \| a_i x_i - r_i(x_{-i}) \|_2^2 + \lambda |x_i| \overset{+c}{=} \frac{1}{2\sigma^2} \left(\|a_i\|^2 x_i^2 - 2(a_i^T r_i) x_i \right) + \lambda |x_i|$$

problem: $\hat{x} = \arg\max_x \left\{ \frac{1}{2}(a^2 x^2 - 2bx) + \lambda |x| \right\}$



$x_*(a,b) = \arg\min_x \left\{ a^2x^2 - 2bx \right\} \Rightarrow \frac{d}{dx} a^2x^2 - 2bx = 2(a^2x - b) \Rightarrow x_*(a,b) = \frac{b}{a^2}$

- the objective is diff. everywhere but zero and is strictly convex
- either: minimized at zero, or where $\frac{d}{dx}$ of obj $= 0$



notice: if $x_*(a,b) > 0$ then $\hat{x} \geq 0$

... < 0 ≤ 0

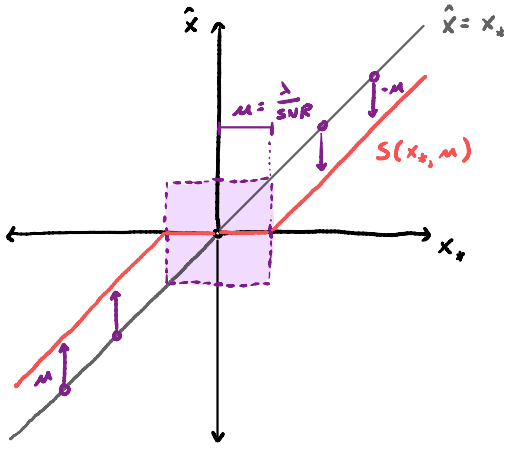
so, WLOG, assume $x_*(a,b) > 0$, then extend soln via symmetry

- if $x_*(a,b) > 0$ then solve: $\hat{x} = \arg\min_{x \geq 0} \left\{ \frac{1}{2\sigma^2}(a^2x^2 - 2bx) + \lambda |x| \right\} = \arg\min_{x \geq 0} \left\{ \ldots + \lambda x \right\}$

$\frac{d}{dx} \frac{1}{2\sigma^2}(a^2x^2 - 2bx) + \lambda x = \frac{1}{\sigma^2}(a^2 x - b) + \lambda$ set $= 0$ $\Rightarrow \hat{x} = \max\left\{0, \frac{b - \sigma^2\lambda}{a^2}\right\} = \max\left\{0, x_* - \frac{\sigma^2}{a^2}\lambda\right\}$

- Soln: $\hat{x}(a, b, \sigma^2, \lambda) = S(x_*(a,b); m)$ where $m = \frac{\lambda}{SNR}$, $SNR = \frac{a^2}{\sigma^2}$

and: $S(x; m) = \begin{cases} x - m & \text{if } x > m \\ 0 & \text{if } |x| \leq m \\ x + m & \text{if } x < -m \end{cases}$



so: $\hat{x}_i(x_{-i}, y) = S\left(\underbrace{\frac{a_i^T r_i}{\|a_i\|^2}}_{a_i^\dagger x_* = \text{Proj}(r_i \text{ onto } a_i)}, \frac{\lambda}{SNR_i} \right)$ for $r_i = y - A_{-i} x_{-i}$, $SNR_i = \frac{\|a_i\|^2}{\sigma^2}$

Coordinate Ascent/Lasso Algorithm:

(·) $x^{(0)} = 0$

(··) for $t = 0$ to convergence

(a) let $i = \text{mod}(t, n) + 1$

(b) compute the remainder: $r = y - A_{:,i} x_{-i}^{(t)}$

(c) compute LS soln: $x_* = \frac{a_i^T r}{\|a_i\|^2}$ and $SNR = \frac{\|a_i\|^2}{\sigma^2}$

(d) shrink/threshold:

$$x_i^{(t+1)} = S(x_*, \lambda/SNR)$$

$$x_{j \neq i}^{(t+1)} = x_{j \neq i}^{(t)}$$

- extremely easy to implement
- converges from any initial condition
- workhorse of sparse inference

2. Expectation-Maximization:

observe $Y = y$, $Y \sim P_{Y|\theta}$, $\theta \sim P_\theta$

$\theta = [x, z]$ want MAP estimator for x marginalizing over z ("nuisance parameters")

$$\hat{x}_{MAP}(y) = \arg\max_x \{p(x|y)\} = \arg\max_x \left\{\int_{\text{all } z} p(x, z|y)\, dz\right\} = \arg\max_x \left\{\mathop{\mathbb{E}}_{z \sim p(z|y)} [p(x|z,y)]\right\}$$

problem! $\int$ inside the objective

- **EM idea:** if we knew z we would set:

$$\hat{x}_{MAP}(y, z) = \arg\max_x \{p(x|z,y)\} = \arg\max_x \{\log(p(x|z,y))\}$$

given x, we can approximate z by sampling $z \sim p(z|x,y)$

try replacing optimization over z (as in coord. ascent) w/ $\mathbb{E}$ over $z \sim p(z|x,y)$...

(1) let $x^{(t)}$ be a current guess at $\hat{x}_{MAP}$

(2) let:

$$Q(x|x^{(t)}) = \mathop{\mathbb{E}}_{z \sim p(z|x^{(t)}, y)} [\log(p(x|z,y))]$$

("E-step" = define expected objective)

(3) set: $x^{(t+1)} = \arg\max_x \{Q(x|x^{(t)})\}$ ("M-step" = maximize)

(4) iterate...

- **Why is this easier?**

(1) log densities are simpler functions than densities (easier to $\int$) (especially true for exponential families)

(2) $\log(p(x|z,y))$ usually factors into tractable terms

(3) if $z \to x \to Y$ so $Y \perp\!\!\!\perp z \,|\, x$ then many terms are irrelevant so may be ignored

- **Ex: Mixture Model:** $z \to x \to y$ then $y \perp\!\!\!\perp z \mid x$ so...

  (.) $p(z|x,y) = p(z|x)$

  (..) $\log(p(x|z,y)) \overset{+c}{=} \log(p(x,z|y)) \overset{+c}{=} \log(p(y,x,z)) = \log(p(y|x,z)\,p(x|z)\,p(z))$

  $= \log(p(y|x)) + \log(p(x|z)) + \log(p(z))$

  ($\perp\!\!\!\perp$ of z; constant in x, does not change maximizer)

  so $Q(x|x^{(t)})$ can be reduced to:

$$Q(x|x^{(t)}) = \log(p(y|x)) + \mathbb{E}_{z\sim p(z|x)}\left[\log(p(x|z))\right]$$

- Suppose, in addition, that $p(x|z)$ is an exponential family...

$$p(x|z) \propto f(x)\, e^{\vec{\phi}(z)\cdot\vec{u}(x)} \quad \text{so} \quad \log(p(x|z)) \overset{+c}{=} \log(f(x)) + \vec{\phi}(z)\cdot\vec{u}(x)$$

now:

$$Q(x|x^{(t)}) = \underbrace{\log(p(y|x))}_{\text{fidelity}} + \overbrace{\log(f(x))}^{\text{base regularization}} + \mathbb{E}_{z\sim p(z|x^{(t)})}\left[\vec{\phi}(z)\right]\cdot\vec{u}(x)$$

all we need are these in closed form

often possible... especially if z conjugate to $x|z$

- **Why does EM work?**

> - **Thm:** each step $x^{(t)} \to x^{(t+1)}$ increases the objective: $p(x|y)$ (each step is an ascent step)

- **Proof:** $p(x|y)\,p(z|x,y) = p(x,z|y) \Rightarrow p(x|y) = \dfrac{p(x,z|y)}{p(z|x,y)}$

$$\log(p(x|y)) = \log(p(x,z|y)) - \log(p(z|x,y))$$

$$= \mathbb{E}_{z\sim q}\left[\log(p(x,z|y))\right] - \mathbb{E}_{z\sim q}\left[\log(p(z|x,y))\right] \quad \forall q \text{ since the LHS is constant in } z$$

$$= \mathbb{E}_{z\sim p(z|x^{(t)},y)}\left[\log(p(x,z|y))\right] - \mathbb{E}_{z\sim p(z|x^{(t)},y)}\left[\log(p(z|x,y))\right]$$

$$\overset{+c}{=} Q(x|x^{(t)}) + D_{KL}(p(z|x^{(t)},y)\,\|\,p(z|x,y))$$

KL divergence: ≥ 0 and $=0$ iff $x = x^{(t)}$ minimized at $x = x^{(t)}$

$x^{(t)} \to x^{(t+1)} = \operatorname{argmax}_x \{Q(x|x^{(t)})\}$ increases $Q(x|x^{(t)})$ and D_{KL} so is an ascent step on $p(x|y)$. ▨

- Ex: $\tau^2 \sim \text{Inv-Gamma}(\alpha, \beta) \longrightarrow x_i|\tau^2 \overset{iid}{\sim} \mathcal{N}(0, \tau^2) \longrightarrow y|x \sim \mathcal{N}(Ax, \sigma^2 I)$

  equivalently, $s = (\tau^2)^{-1}$

  $s \sim \text{Gamma}(\alpha, \beta^{-1}) \longrightarrow x_i|s \overset{iid}{\sim} \mathcal{N}(0, s^{-1})$

  s, τ^2 are conjugate to $x|\tau^2$ or $x|s$ and are members of an exp. family + y is cond. ind. of τ^2 or s given x

(·) E-step: $p(y|x) \propto \exp\left(-\frac{1}{2\sigma^2}\|Ax-y\|_2^2\right) \Rightarrow \log(p(y|x)) \overset{+C}{=} -\frac{1}{2\sigma^2}\|Ax-y\|_2^2$

$p(x|s) = \left(\frac{s}{2\pi}\right)^{n/2} \exp\left(-\frac{s}{2}\|x\|_2^2\right) \Rightarrow \log(p(x|s)) \overset{+C}{=} -\frac{s}{2}\|x\|_2^2$

so: $Q(x|x^{(t)}) \overset{+C}{=} -\frac{1}{2}\left[\frac{1}{\sigma^2}\|Ax-y\|_2^2 + \mathbb{E}_{s\sim p(s|x^{(t)})}[s]\,\|x\|_2^2\right]$

need $s|x$: $p(s|x) \propto p(s,x) = p(x|s)\,p(s)$, $p(s) \propto s^{\alpha-1}e^{-\beta s}$

$p(s|x) \propto s^{n/2}e^{-\frac{s}{2}\|x\|_2^2} s^{\alpha-1}e^{-\beta s} = s^{\alpha+n/2-1}e^{-(\beta+\frac{1}{2}\|x\|_2^2)s}$

so: $s|x \sim \text{Gamma}\left(\alpha + \frac{n}{2}, \left(\beta + \frac{1}{2}\|x\|_2^2\right)^{-1}\right)$

form of $\text{Gamma}(\alpha+n/2, (\beta+\frac{1}{2}\|x\|_2^2)^{-1})$

generally: $\mathbb{E}_{W\sim\text{Gamma}(a,b^{-1})}[W] = a\,b^{-1}$ so: $\mathbb{E}_{s\sim p(s|x)}[s] = \frac{\alpha + n/2}{\beta + \frac{1}{2}\|x\|_2^2}$

compare to $\hat{s}_{MAP}(x) = \frac{\alpha + \frac{n}{2} + 1}{\beta + \frac{1}{2}\|x\|_2^2}$

$\therefore \quad Q(x|x^{(t)}) = -\frac{1}{2}\left[\frac{1}{\sigma^2}\|Ax-y\|_2^2 + \lambda(x^{(t)})\,\|x\|_2^2\right]$ where: $\lambda(x^{(t)}) = \frac{\alpha + \frac{n}{2}}{\beta + \frac{1}{2}\|x\|_2^2}$

(··) M-step: $x^{(t+1)} = \underset{x}{\text{argmax}}\{Q(x|x^{(t)})\} = (A^TA + \sigma^2\lambda(x^{(t)})I)^{-1}A^Ty$

EM algorithm for $\hat{x}_{MAP}$ (marginal over τ^2 or s)...

(·) initialize $x^{(0)} = 0$

(··) iterate $t = 0, \ldots$ to convergence

(a) set: $\lambda = \frac{\alpha + \frac{1}{2}n}{\beta + \frac{1}{2}\|x^{(t)}\|^2}$

(b) set: $x^{(t+1)} = (A^TA + \sigma^2\lambda I)^{-1}A^Ty$

Compare: Coordinate-Ascent for $(\hat{x}, \hat{\tau}^2)_{MAP}$ also iteratively reweighted LS, but... for joint MAP, use: $\lambda = \frac{\alpha + \frac{1}{2}n + 1}{\beta + \frac{1}{2}\|x^{(t)}\|^2}$ then, set $\tau^2_{MAP} = \lambda^{-1}$

- Distribution $\approx$:
  - Current pipeline: ~~model~~ ~~optimize~~

    data + model $\xrightarrow[\text{optimize}]{}$ $\hat{x}_{MAP}(y)$ $\xrightarrow[\text{log post}]{\text{Taylor exp}}$ Laplace $\approx$ ~~$\mathcal{N}(\hat{x}_{MAP}, H^{-1}(\hat{x}))$~~

    $\approx$ posterior
  - Can we improve this approximation?
    - 2 ideas:

      1. fatter tails $\longrightarrow$ use a multivariate t-distribution that matches the normal locally

         $$p(x|y) \underset{\propto}{\approx} \left(\nu + \|x - \hat{x}_{MAP}\|^2_{H(\hat{x})}\right)^{-\frac{\nu+d}{2}}$$

         not guaranteed to be more accurate but looser tail assumptions (don't overestimate evidence)

      2. reweight via:

         a. importance sampling or b. rejection sampling

a: true dist: q (target) proposal: p_r (can sample from) define: weight $w(x) = \frac{q(x)}{p(x)}$

then: $q(x) = w(x)\, p_r(x)$ so $\int_{x\in A} q(x)\,dx = \int_{x\in A} w(x)\, p_r(x)\,dx$

use $f = \mathbb{1}(x \in A)$

$$\int_{x\in A} f(x)\, q(x)\,dx = \int_{x\in A} f(x)\, w(x)\, p_r(x)\,dx$$

just want to watch $\mathbb{E}$'s...

$$\mathop{\mathbb{E}}_{x\sim q}[f(x)] = \int_x f(x)\, w(x)\, p_r(x)\,dx = \mathop{\mathbb{E}}_{x\sim p_r}[f(x)\, w(x)] \approx \frac{1}{n}\sum_{j=1}^{n} f(x_j)\, w(x_j)$$

$x_j \overset{iid}{\sim} p_r$

problem for Bayes... q is only known up to proportionality...

w is only known up to proportionality...

so, estimate the norm. factor for w via $\frac{1}{n}\hat{\sum_{j=1}^{n}} w(x_j)$

(reweight so $w(x_j)$ sum to one)

$\Rightarrow$ use $\mathbb{E}_{x\sim q}[f(x)] \approx \dfrac{\hat{\sum_{j=1}^{n}} f(x_j)\, w(x_j)}{\hat{\sum_{j=1}^{n}} w(x_j)}$ $\xrightarrow{\text{SLLN}}$ desired $\mathbb{E}$ for any f...

(denominator: essentially $\hat{Z}$ needed for evidence)

- Easy method for correcting posterior $\tilde{\approx}$

1. let $q(x) = p(x,y)$ up to norm...

   (usually $q(x) \propto \exp(\log(p(y|x)) + \log(p(x)))$

2. let $pr(x) = \text{post}\ \tilde{\approx}$ e.g. $\text{Lap}(x|y)$ or $\text{Lap}_t(x|y;\nu)$

then, do posterior inference by:

(a) form the proposal (e.g. data + model $\longrightarrow \hat{x}_{MAP} \longrightarrow \text{Lap}_t(x|y;\nu)$

$\{ \Rightarrow \vdots \Longrightarrow$ mixture model $\}$

(b) $\tilde{\approx}$ $\mathbb{E}$ against true posterior by:

(i) sample $\{x_j\}_{j=1}^{N} \overset{iid}{\sim} p_t$

(ii) set: $\mathbb{E}_{x|y}[f(x)] \approx \hat{\sum_{j=1}^{n}} f(x_j)\, \hat{w}(x_j)$

where: $\hat{w}(x_j) = \dfrac{w(x_j)}{\hat{\sum_{i=1}^{n}} w(x_i)}$, $w(x) = \dfrac{q(x)}{pr(x)}$.

(b) rejection sampling is a stochastic approx to importance sampling...

(i) pick proposal s.t. $\exists$ $M \hat{>} 0$ s.t. $M\, pr(x) \geq q(x|y)$ $\forall$ M

(ii) $\approx$ importance weighting w.r.t. $M\, pr \Rightarrow w(x) = \frac{1}{M} \frac{q(x|y)}{pr(x)}$

by:

(·) $\{X_j\}_{j=1}^{N} \overset{iid}{\sim} pr(x)$

(··) set $a(x) = \frac{1}{M} \frac{q(x)}{pr(x)}$

(···) draw $\xi \sim Uni[0,1]$

(*) if $\xi \leq a(x) \Rightarrow$ keep X_j

(**) else $\Rightarrow$ reject X_j (discard it)

- Fact: if $M\, pr(x) \geq q(x)$ $\forall$ x then rejection sampling produces exact sample draws from q.

Pros:

- Imp:
  - ~~unbiased~~ estimator for all $\mathbb{E}$
  - sample efficient
  - works for any pr
  - simple, easy to use/adapt
- Rej:
  - exact post. sampling
  - simple, easy to use/adapt

Cons:

- high $\mathbb{V}$ if pr underestimates target ... inefficient if high $\mathbb{V}$
- needs M... (how to find?)
- wastes samples... (inefficient)

both:
more efficient if $pr \approx q$.
inefficient otherwise

often used to when conditioning on a likely event (e.g. $\mathcal{N}$ w/ $\mathbb{P}[X \notin A]$ s.t. $x \in A$ w/ low prob.... $\mathcal{N}(\bar{x}, \Sigma)$ for $x_j \geq 0$ $\forall_j$ if $\bar{x}_j \gg 0$ $\forall_j$ [σ_j relative

- **rejection sampling:**
  - stochastic % to importance sampling where-in we randomly choose to accept or reject proposed samples...
  - see lecture notes from week before spring break.

(i) pick a proposal p_r s.t. $\exists\ M > 0$ where

$$M\, p_r(x) \geq p_*(x) \quad \forall\ x$$

(ii) set $\tilde{w}(x) = \frac{1}{M} \frac{p_*(x)}{p_r(x)}$ then $\tilde{w}(x) \in [0, 1]$

(iii) approximate importance weighting by

(a) $x^{(n)} = \{x_j\}_{j=1}^{n} \overset{iid}{\sim} p_r$

(b) draw $\{\xi_j\}_{iid}^{(n)} \sim U_{n:[0,1]}$

(*) if $\xi_j \leq \tilde{w}(x_j) \Rightarrow$ keep x_j

(**) else $\Rightarrow$ discard x_j

} keep x_j w/ prob. $\tilde{w}(x_j)$

then:

$$\mathbb{E}_{x \sim p_*}[f(x)] \approx \frac{1}{n} \sum_{j=1}^{n} f(x_j)\, \mathbb{1}(\xi_j \leq \tilde{w}(x_j))$$

$$\mathbb{E}[\ldots] = \mathbb{E}_{\substack{x \sim p_r \\ \xi \sim U_{n:}}}[f(x)\, \mathbb{1}(\xi \leq \tilde{w}(x))] = \mathbb{E}_{x \sim p_r}\left[f(x)\, \mathbb{E}_{\xi | x}[\mathbb{1}(\xi \leq \tilde{w}(x))]\right]$$

$$= \mathbb{E}_{x \sim p_r}[f(x) \max(\tilde{w}(x), 1)]$$

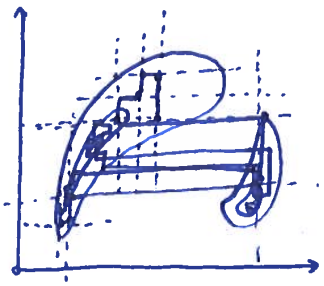
$$= \mathbb{E}_{x \sim p_r}[f(x)\, \tilde{w}(x)] = \mathbb{E} \text{ of importance weighted } \approx \text{ using } \tilde{w}$$

$$= \mathbb{E}_{x \sim p_*}[f(x)]$$

should use $\hat{w} = \frac{\tilde{w}}{\text{sample av. of } \tilde{w}}$ to $\approx w$

- pros & cons table (see last week's notes)

- Gibb's Sampling
  - idea, we often build models from conditionally conjugate components or from distributions whose ~~marginals may be sampled~~ conditionals are easy to sample from...
  - sample by breaking unknowns into blocks: $\theta = [\theta^{(1)}, \theta^{(2)}, \dots \theta^{(B\ m)}]$
    - (i) iterate VIA $t = 1, \dots$ to convergence
    - (ii) set $b = s(t)$ (often $b = \text{mod}(t, m) + 1$)
    - (iii) draw $\theta_{\#b}^{(t+1)} \sim p_{\theta_{\#b} | \theta_{-b}, y}$



pros: works well if the blocks are not highly dependent on each other & the conditionals can be sampled from efficiently; can make large jumps

- Ex:

$$\left.\begin{array}{l} \tau^2 \sim \text{Inv-Gamma}(\alpha, \beta) \longrightarrow X_i | \tau^2 \overset{iid}{\sim} \mathcal{N}(0, \tau^2) \\ \quad \uparrow\ s = (\tau^2)^{-1} \qquad\quad \uparrow \\ s \sim \text{Gamma}(\alpha, \beta^{-1}) \longrightarrow X_i | s \overset{iid}{\sim} \mathcal{N}(0, s^{-1}) \end{array}\right\} Y \sim \mathcal{N}(AX, \sigma_Y^2 I)$$

(i) iterate $t = 1$ to convergence:

(a) $X^{(t+1)} \sim p_{X | s^{(t)}, y}$ = normal posterior of normal, normal $X^{(t+1)} \sim \mathcal{N}(0, \check{s}^{(t)\,-1} I)$

(b) $s^{(t+1)} \sim p_{s | X^{(t+1)}} = \text{Gamma}(\alpha + n/2, (\beta + \tfrac{1}{2}\|\check{X}^{(t+1)}\|_2^2)^{-1})$ ↑ $Y \sim \mathcal{N}(AX^{(t+1)}, \sigma_Y^2 I)$

Stat 238 - Lecture 19 - Intro to Sampling

- Topics:
  1. importance sampling review
  2. rejection sampling
  3. Gibbs sampling
  4. MCMC - intro to Metropolis-Hastings

  } morals: the more accurate the proposal the more efficient the method
  - variance of importance weighted average
  - rejection rate of rejection sampling

  to do:
  1. state algorithms
  2. show convergence/consistency (prove Gibbs via MCMC)

- Sampling:
  - essential goal: draw samples from a target distribution p_* (usually, the posterior)
    - why? b/c the empirical dist. of a large selection of i.i.d. samples can be used to $\approx$ any calculation involving p_*
      - in particular to generate approximations to: $\mathbb{E}_{x \sim p_*}[f(x)]$ for a set of functions $\{f\} = \mathcal{F}$ — use for inference, can also pass samples forward for prediction/sampling/checking

      (e.g. $\mathbb{E}[x]$, $\mathbb{E}[x^2]$, $\mathbb{E}[p_*(x)]$, $\mathbb{E}[\log(p_*(x))]$, $\mathbb{E}[\mathbb{1}(x \in E)]$, ...)

      iff. if $\mathbb{E}_{x \sim p}[f(x)] = \mathbb{E}_{x \sim q}[f(x)]$ $\forall f$ then $p(x)$ and $q(x)$ are identical

  - idea from last Thursday:
    - Importance Sampling:
      (i) proposal dist: p_r s.t. we can draw $x^{(n)} = \{x_j\}_{j=1}^n \overset{iid}{\sim} p_r$ where $\text{Supp}(p_r)$ contains $\text{Supp}(p_*)$

      (ii) $p_*(x) \propto \tilde{p}_*(x)$ where $\tilde{p}$ is unnormalized

      (iii) let $\tilde{w}(x) = \tilde{p}_*(x)/p_r(x)$

      (iv) set:

$$\mathbb{E}_{x \sim p_r}[f(x)] \approx \frac{1}{n}\sum_{j=1}^n f(x_j)\,\hat{w}(x_j) \quad \text{where} \quad \hat{w}(x_j) = \frac{\tilde{w}(x_j)}{\frac{1}{n}\sum_{j=1}^n \tilde{w}(x_j)}$$

then: can use to approximate any $\mathbb{E}_{x\sim p_*}[f(x)]$ given some p_r that covers p_* (essentially, reweight the sample average)

- efficiency depends on how closely p_r approximates p_* ...

- effective sample size...

- $\mathrm{Var}_{x_j\sim p_r}\left[\frac{1}{n}\sum_{j=1}^{n} f(x_j)\right] = \frac{1}{n}\mathrm{Var}_{x\sim p_r}[f(x)]$

for simplicity, approximate $\hat{w}(x_j)$ w/ $w(x_j) = \frac{p_*(x_j)}{p_r(x_j)}$ } to equate...

$\mathrm{Var}_{x_j\sim p_r}\left[\frac{1}{n}\sum_{j=1}^{n} f(x_j)w(x_j)\right] = \frac{1}{n}\mathrm{Var}_{x\sim p_r}[f(x)w(x)]$

$n_* = n_{imp}\cdot \frac{\mathrm{Var}_{x\sim p_*}[f(x)]}{\mathrm{Var}_{x\sim p_r}[f(x)]}$ $\quad \hat{w}(x)$ $\quad w(x)$

$\mathrm{Var}_{x\sim p_r}[f(x)] = \mathbb{E}_{x\sim p_r}\left[(f(x)w(x) - \underbrace{\mathbb{E}_{x\sim p_r}[f(x)w(x)]}_{=\mathbb{E}_{x\sim p_*}[f(x)]=\bar{f}_*})^2\right] = \mathbb{E}_{x\sim p_r}\left[(f(x)w(x) - \bar{f}_*)^2\right]$

$= \underbrace{\mathbb{E}_{x\sim p_r}[f(x)^2 w(x)^2]}_{} - \bar{f}_*^2$

$= \int_{x\sim p_r} f(x)^2 \frac{p_*(x)^2}{p_r(x)}\,dx = \int_x f(x)^2 \underbrace{\left(\frac{p_*(x)}{p_r(x)}\right)}_{w(x)} p_*(x)\,dx = \mathbb{E}_{x\sim p_*}[f(x)^2 w(x)]$

$\mathrm{Var}_{x\sim p_r}[f(x)] = \mathbb{E}_{x\sim p_*}[f(x)^2 w(x)] - \mathbb{E}_{x\sim p_*}[f(x)]^2$

so $n_* = n\left(\frac{\mathbb{E}_{x\sim p_*}[f(x)^2 w(x)] - \mathbb{E}_{x\sim p_*}[f(x)]^2}{\mathbb{E}_{x\sim p_*}[f(x)^2] - \mathbb{E}_{x\sim p_*}[f(x)]^2}\right) n_{imp}$

or write both $\mathbb{E}$ using p_r

$= \left(\frac{\mathbb{E}_{x\sim p_r}[(f(x)w(x) - \bar{f}_*)^2]}{\mathbb{E}_{x\sim p_r}[(f(x) - \bar{f}_*)^2 w(x)]}\right)$

- notation:
  - $p_*(x)$ = target, $\tilde{p}_*(x)$ = un-normalized target (e.g. $\tilde{p}_*(x) = Lik(x;y)\,prior(x)$)

  $p_*(x) = \frac{1}{Z}\tilde{p}_*(x)$, $Z = \int_x \tilde{p}_*(x)$

- $p_1(x)$ = proposal, tractable, can sample from

(0)

• Importance - Sampling:

$$\mathbb{E}_{x\sim p_*}[f(x)] = \mathbb{E}_{x\sim p_r}[f(x)\,w(x)] \quad \text{if} \quad w(x) = \frac{p_*(x)}{p_r(x)}$$

$$\approx \frac{1}{n}\sum_{j=1}^{n} f(x_j)\,w(x_j) \quad \text{if } x_j \overset{iid}{\sim} p_r$$

• problem: $p_* \propto \tilde{p}_*$, $w \propto \tilde{w}(x) = \frac{\tilde{p}_*(x)}{p_r(x)}$, $\hat{w}(x)$

• notice: $\mathbb{E}_{x\sim p_r}[\tilde{w}(x)] = \int_x \frac{p_*(x)}{p_r(x)}\,p_r(x)\,dx = \int_x p_*(x)\,dx = 1$

$\therefore \tilde{w} \propto w$ so $w(x) = \frac{\tilde{w}(x)}{\mathbb{E}_{x\sim p_r}[\tilde{w}(x)]}$

$$\mathbb{E}_{x\sim p_*}[f(x)] \approx \frac{\sum_{j=1}^{n} f(x_j)\,\tilde{w}(x_j)}{\sum_{j=1}^{n} \tilde{w}(x_j)} = \sum_{j=1}^{n} f(x_j)\,\hat{w}^{(n)}(x_j)$$

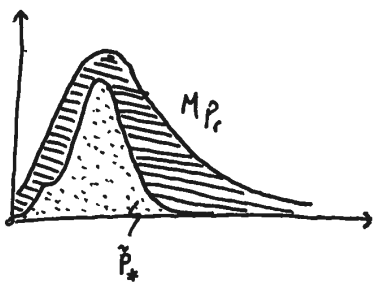
$$\hat{w}^{(n)}(x) = \frac{\tilde{w}(x)}{\frac{1}{n}\sum_{j=1}^{n}\tilde{w}(x_j)}$$

then $w(x) \approx \hat{w}^{(n)}(x)$

Rejection:

• Suppose $\exists\ M$ s.t. $M\,p_r(x) \geq \tilde{p}_*(x)\ \ \forall\ x$

• let: $a(x) = \frac{\tilde{p}_*(x)}{M\,p_r(x)}$



• to sample $X \sim p_*$:

(i) draw $Y \sim p_r$

(ii) compute $a(Y)$

(iii) draw $U \sim \text{uni}[0,1]$

(iv) if $U \leq a(Y)$

$X = Y$

else

return to (i)

two questions:

1. how often will we reject/accept? (efficiency)
2. what is the dist. of X? (accuracy)

1. $P_r(U \leq a(Y)) = \int_x P_r(U \leq a(x) \text{ and } Y = x)\,dx$

$$= \int_x p_r(x)\,a(x)\,dx = \int_x p_r(x)\frac{1}{M}\frac{\tilde{p}_*(x)}{p_r(x)}\,dx = \frac{1}{M}\int_x \tilde{p}_*(x)\,dx$$

$$= \frac{1}{M}\,Z \quad \text{where} \quad Z = \tilde{p}_*(x)/p_*(x), \quad p_*(x) = \frac{1}{Z}\,\tilde{p}_*(x)$$

$$\max_M \{ P_r(\text{accept}) \} = \max_M \left\{ \frac{Z}{M} \right\} = \frac{Z}{\min(M)}$$

$$M \geq \tilde{P}_*(x) / P_r(x) \Rightarrow \min(M) = \max_x \left(\tilde{P}_*(x) / P_r(x) \right)$$

$$= Z \cdot \min_x \left\{ P_r(x) / \tilde{P}_*(x) \right\} = \min_x \left\{ \frac{P_r(x)}{P_*(x)} \right\} = \left[\max_x \{ w(x) \} \right]^{-1}$$

2. $P_r(X=x) = P_r(Y=x \mid U \leq a(Y)) = \dfrac{P_r(Y=x \;\&\; U \leq a(Y))}{P_r(U \leq a(Y))} = \dfrac{P_r(Y=x)\, P_r(U \leq a(Y) \mid Y=x)}{\frac{1}{M} Z}$

$$= \frac{M}{Z} P_r(x)\, a(x) = \frac{M}{Z} P_r(x) \frac{\tilde{P}_*(x)}{M P_r(x)} = \frac{1}{Z} \tilde{P}_*(x) = P_*(x) .$$

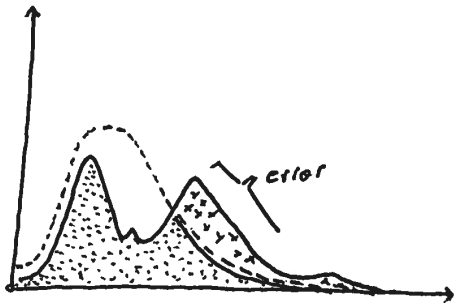
$a(x) = \hat{\max}(\,\cdot\,, 1)$

$X \sim P_*(x)$. ▨

if $a(x)$ exceeds 1 for some x then have:

$$P_r(X=x) = \frac{M}{Z} P_r(x) \max \left(\frac{\overset{Z}{\overset{\vee}{P_*(x)}}}{M P_r(x)}, 1 \right) = \max \left(P_*(x), \frac{M}{Z} P_r(x) \right)$$

$$\propto \max \left(\tilde{P}_*(x), M P_r(x) \right)$$



- Both rejection & importance sampling are more efficient when $P_r \simeq P_*$
  - if $P_*(x) \gg P_r(x)$ in tails (under-cover), then:
    - rejection rate is high ⇒ sample inefficient
    - the weight dist. is heavy tailed ⇒ imp. weighted averages are highly variable

• Metropolis-Hastings & Markov Chain Monte Carlo:

• **Idea:** let $\{X_t\}_{t=1}^{\infty}$ be a Markov chain (discrete time) that is irreducible and aperiodic (usually not a problem in continuous spaces)

if the chain is ergodic then:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} f(X_j) \xrightarrow{\text{in prob}} \mathbb{E}_{X\sim p_*}[f(X)] \quad \forall\, f$$

where:

p_* is the unique stationary dist. of the chain.

• let $T(x \to x') = \Pr(X_{t+1} = x' \mid X_t = x)$

then, if $X_t \sim p_t$, $\Pr(X_{t+1} = x') = \sum_{\text{all } x} T(x \to x')\, p_t(x) = T\, p_t$

if $T_{x',x} = T(x \to x')$

• p_* is stationary if

$T\, p_* = p_*$ (if $p_t = p_*$ for some t then $p_{t+s} = p_*$ $\forall\, s \geq 0$)

that is:

$$\forall\, x' \quad p_*(x') = \sum_x T(x \to x')\, p_*(x') \Rightarrow \underbrace{(T(x' \to x') - 1)}_{1 - \sum_{x \neq x'} T(x' \to x)} p_*(x') + \sum_{x \neq x'} T(x \to x')\, p_*(x')$$

$$\Rightarrow \underbrace{\sum_{x \neq x'} \left(T(x \to x')\, p_*(x) - T(x' \to x)\, p_*(x') \right)}_{\left[\left[\forall\, x' : \sum_{x \neq x'} J_{x,x'}[p_*] = 0 \right]\right]} = 0$$

"complex balance" = no net flux in or out

(3)

- "detailed" balance: $J_{x,x'}[\rho_*] = 0 \quad \forall\ x, x'$

  requires: $T(x \to x')\,\rho_*(x) = T(x' \to x)\,\rho_*(x') \quad \forall\ x, x'$

  can rearrange: $\dfrac{T(x \to x')}{T(x' \to x)} = \dfrac{\rho_*(x')}{\rho_*(x)}$ ⇒ we want: transition ratio = odds (available without normalizing)

  for all x, x' pairs, connected ~~follow~~

- can use to design chains s.t. $\rho_{*ss} = \rho_* \quad \forall\ \rho_*$ known up to normalization

- Metropolis-Hastings:

  (i) draw $x_0 \sim \rho_0$ (initialize)

  (ii) for $t = 0$, to convergence

  (a) propose $x' \sim J(x \to \dots)$ (jump kernel) where $J(x \to x') \neq 0$ if $J(x' \to x) \neq 0$ and $\rho_*(x') \neq 0$

  (b) calculate: $a = \dfrac{\tilde{\rho}_*(x')}{\tilde{\rho}_*(x)} \cdot \dfrac{J(x' \to x)}{J(x \to x')}$

  $\max(\ a\ ,\ 1)$

  (c) draw $\xi \sim \mathrm{Uni}[0, 1]$

  (*) if $\xi \leq a$, set $x_{t+1} = x'$

  (**) else $x_{t+1} = x_t$

- Metropolis: pick J symmetric

(say this is ↘ on $\tilde{\rho}_*$)

$T(x \to x') = J(x \to x') \dfrac{J(x' \to x)}{J(x \to x')} \dfrac{\tilde{\rho}_*(x')}{\tilde{\rho}_*(x)}$

$T(x' \to x) = J(x' \to x) \cdot 1$

ratio $= \dfrac{\tilde{\rho}_*(x')}{\tilde{\rho}_*(x)} = \dfrac{\rho_*(x')}{\rho_*(x)}$ ✓

then $\rho_{ss} = \rho_*$ ∎

(notice: if $a \neq 1$ for some x' for s.t. $J(x \to x') \neq 0$ for any x & connected then aperiodic)

(4)

Lecture 20 - MCMC

- Outline:

1. Markov Chain Basics

2. Derive MH as class
   - give detailed balance & proposal/jump construction, ask for acceptance rate protocol
   - lab 6 example:
     a) derive $\tilde{p}_*$ ⇒ how to compute acceptance ratios
     b) show problems w/ inappropriate proposal

3. Gibbs Sampling:
   - state
   - example

Lecture 21: advanced MCMC
- Sim. Temp.
- HMC

Lecture 22: dist ∞
- KNN · GMM via EM · CAVI

Projects:
- Sim. Temp. & Teleporting Walkers in Ising
- Reading on phylogenetic trees & MH
- advanced Bayesian Computation (ABC, €, SVGD)

1. Markov Chains * on board at start

- Stochastic process $\{X(t)\}_{t=0}^{\cdots}$ s.t. $X(t+h)$ is cond. ⊥⊥ of $X(t-s)$ given $X(t)$
- Transition Rate: $T(x \to x') = P_r(X(t)=x' \mid X(t+1)=x)$

- Distribution: $p_{\text{state}}(x) = P_r(X(t)=x)$ (t)

- Stationary Distribution: p_s is a dist. s.t. if $p(\cdot, t) = p_s(\cdot)$ then $p(\cdot, t+h) = p_s \quad \forall \ h \geq 0$.

- if irr. ap. then attracting
  $p(\cdot, t) \to p_s(\cdot)$ in d_{TV} as $t \to \infty$

- if $X(0) \sim p_s$ then $\{X(t)\}_{t=0}^{\cdots}$ is a stationary & ergodic process

~~Stationarity: $\forall x'$, $p_s(x') = \cdots$~~

then: $p(x', t+1) = \sum_{\text{all } x} p(x, t)\, T(x \to x')$

if stationary: $p_s(x') = \sum_{\text{all } x} p_s(x)\, T(x \to x') = p_s(x')\, T(x' \to x') + \sum_{\text{all } x \neq x'} p_s(x)\, T(x \to x')$

$$\underbrace{(T(x' \to x') - 1)}_{1 - \sum_{x \neq x'} T(x' \to x)}\, p_s(x') + \sum_{x \neq x'} p_s(x)\, T(x \to x') = 0$$

$$\sum_{x \neq x'} \underbrace{p_s(x)\, T(x \to x') - p_s(x')\, T(x' \to x)}_{J_{x \to x'}[p_s]} = 0$$

$$\underbrace{\sum_{x \neq x'} J_{x \to x'}[p_s] = 0}_{\text{net flux out of any state must } = 0 \ \ldots \text{ complex balance}} \qquad \forall \ x$$

- **Def:** detailed balance iff $J_{x \to x'}[p_s] = 0 \quad \forall \ (x, x')$

$$p_s(x)\, T(x \to x') = p_s(x')\, T(x' \to x) \qquad \forall \ (x, x')$$

$$\underbrace{\frac{T(x \to x')}{T(x' \to x)}}_{\text{transition ratio}} = \underbrace{\frac{p_s(x')}{p_s(x)}}_{\text{stationary odds}}$$

Fact: if $\{X(t)\}_{t=0}^{\infty}$ is irr., aperiodic, and obeys detailed balance then:

(i) p_s uniquely det the set of all transition ratios (for possible transitions)

(ii) the collection of all transition ratios determines p_s

idea: if we want $p_s = p_*$, it is sufficient to pick an aperiodic, irreducible chain s.t.

$$(*) \quad \frac{T(x \to x')}{T(x' \to x)} = \frac{p_*(x')}{p_*(x)} \quad \text{for all possible transitions}$$

$$= \frac{\tilde{p}_*(x')}{\tilde{p}_*(x)} \quad \dots$$

2. Metropolis-Hastings:

- idea: step $X(t) \longrightarrow X(t+1)$ in 2 stages

(i) $X(t) \longrightarrow$ candidate: $Y \sim p_r(Y \mid X(t) = x)$

(ii) randomly decide to accept Y ($X(t+1) = Y$ or $x(t)$) w/ an acceptance prob. that is chosen s.t. the process satisfies (*).

(ii) $U \sim \text{Unif}([0,1])$, compare U to $a(Y, x(t))$

if $U \le a(Y, x(t))$

accept: $X(t+1) = Y$

else

reject: $X(t+1) = x(t)$

then:

$$T(x \to x') = p_r(Y = x' \ \& \ \text{accept} \mid X = x)$$

$$= p_r(Y = x' \mid X = x)\, p_r(\text{accept} \mid Y = x', X = x)$$

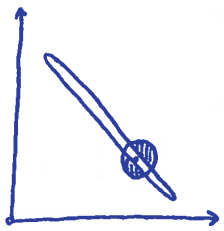
$$= p_r(x' \mid x)\, a(x', x)$$

$$(*) \quad \frac{T(x \to x')}{T(x' \to x)} = \frac{p_r(x \to x')\, a(x \to x')}{p_r(x' \to x)\, a(x' \to x)} = \frac{\tilde{p}_*(x')}{\tilde{p}_*(x)} \Rightarrow \left[\frac{a(x \to x')}{a(x' \to x)} = \frac{\tilde{p}_*(x')\, p_r(x' \to x)}{\tilde{p}_*(x)\, p_r(x \to x')} \right] \quad (**)$$

13

- any: $a(x \to x'),\ a(x' \to x) \in [0,1]$ satisfying

$$\frac{a(x \to x')}{a(x' \to x)} = \frac{\tilde{p}_*(x')\, p_t(x' \to x)}{\tilde{p}_*(x)\, p_t(x \to x')}$$ will suffice



- MH: pick as asymmetrically as possible (most directed version of the chain)

$$q(x \to x') = \min\left(\underbrace{\frac{\tilde{p}_*(x')}{\tilde{p}_*(x)} \frac{p_t(x' \to x)}{p_t(x \to x')}}_{r(x \to x')},\ 1 \right) = \begin{cases} 1 & \text{if } \tilde{p}_*(x') \geq \frac{p_t(x \to x')}{p_t(x' \to x)} \tilde{p}_*(x) \\ r(x \to x') & \text{else} \end{cases}$$

- Metropolis: special case when p_t is symm

$$= \begin{cases} 1 & \text{if } \tilde{p}_*(x') \geq \tilde{p}_*(x) \\ \tilde{p}_*(x') / \tilde{p}_*(x) & \text{else} \end{cases}$$

Example:

- lab 6: $\eta(x,\beta) = \beta_0 + \beta_1 \log(x) \longrightarrow q(x,\beta) = \text{logistic}(\eta(x,\beta)) = [1 + e^{-\eta(x,\beta)}]^{-1}$

$\to Y \mid x, \beta \sim \text{Bin}(n, q(x,\beta))$ given $\{(x_i, y_i)\}_{i=1}^{4}$

$$\text{Lik}(\beta) \propto \prod_{i=1}^{4} (1 + e^{-(\beta_0 + \beta_1 \log(x_i))})^{-y_i} (1 + e^{\beta_0 + \beta_1 \log(x_i)})^{-(n_i - y_i)}$$

~~$\to \log(\text{Lik}(\beta)) \propto \sum_{i=1}^{4} \log(\cdots)$~~

for priors, see lab 6 demo

- Experiments: start w/ symm. proposal $ too few samples
  (~ 2,000 → 4,000 → 10,000 → 40,000 → 80,000)
  ask about stride: (def. 100 does going to 10 help?)

- Comment/show what happens for an inappropriate proposal.

- What is the ideal proposal?? ($\rho_t = \rho_*$, converge in 1 step... intractable
  but... ρ_* often has tractable conditionals)

Gibbs Sampling:

- idea: $X = [X^{(1)}, X^{(2)}, \dots X^{(m)}]$

  pick a schedule $\{s(t)\}_{t=0}^{\dots}$, ~~$t \to X$~~ < as usual, could be randomized or cyclic

  iterate: for $t=0,$ to convergence

  (·) ~~sample $X^{s(t)}$ from $\rho(X^{s(t)} \mid X^{(-s(t))})$~~

  sample $X^{(s(t))}(t+1) \sim \rho(X^{(s(t))} \mid X^{(-s(t))}(t))$

  where $X \sim \rho_*$ ~~for all t~~

- Ex: lab 7

$$\tau^2 \sim \text{Inv-Gamma}(\alpha, \beta) \longrightarrow X_i \overset{iid}{\sim} \mathcal{N}(0, \tau^2) \longrightarrow Y \sim \mathcal{N}(AX, \sigma^2 I)$$

$$\updownarrow\ S = (\tau^2)^{-1}$$

$$S \sim \text{Gamma}(\alpha, \beta^{-1}) \longrightarrow X_i \overset{iid}{\sim} \mathcal{N}(0, S^{-1})$$

$S \to X \to Y$, so $S \perp\!\!\!\perp_{1X} Y$

~~joint-posterior~~

- conditional-posteriors: $S \mid X, Y \sim S \mid X$

$$p(S=s \mid X=x) \propto s^{\alpha-1} e^{-\beta s}\, s^{n/2} e^{-\frac{1}{2}s\|x\|^2} = s^{(\alpha+n/2)-1} e^{-(\beta+\frac{1}{2}\|x\|^2)}$$

$$= \text{Gamma}(\alpha + n/2,\ \beta + \tfrac{1}{2}\|x\|_2^2)$$

cc

$X \mid S, Y$: $p(X=x \mid S=s, Y=y) \propto e^{-\frac{1}{2\sigma^2}\|Ax-y\|^2} e^{-\frac{1}{2}s\|x\|^2}$

$e^{-\frac{1}{2\sigma^2}\|Ax-y\|^2 - \frac{1}{2}s\|x\|^2} = e^{-\frac{1}{2}\left(\frac{1}{\sigma^2}\|Ax-y\|^2 + s\|x\|^2\right)}$

normal. (update mean & cov using normal-normal)

Lecture 21 - Stat 238 - Towards Efficient MCMC - 04/08

Outline:

(i) main MCMC challenge → slides

(ii) Gibbs sampling → slides
- algorithm → slides
- as ~~MCMC~~ MH w/ acceptance of 1 → board ✓
- lob 7 example → slides
- limitations → slides

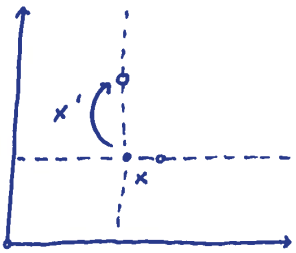
(iii) ∇ based methods
- ∇ of log → slides? slides/board.
- Langevin
- Metropolis-adjusted Langevin

- Gibbs is MH w/ acceptance ratio 1:

- Proof: $X = [X^{(1)}, X^{(2)}, \dots X^{(m)}]$

$X(t) \to X(t+1)$ via m intermediate steps, let's use a randomized schedule...

~~$T(x \to x') = \frac{1}{m} \sum_{s=1}^{m} P_s(x'_s \mid x'_{-s} = x_{-s})$~~



Then $T(x \to x') = \begin{cases} 0 & \text{if } x' \neq x \text{ on more than one block} \\ \frac{1}{m} P_*(x'_s \mid x'_{-s} = x_{-s}) & \text{if } x' \text{ and } x \text{ only differ in block } s \end{cases}$

So $\frac{T(x \to x')}{T(x' \to x)} = \frac{\frac{1}{m} P_*(x'_s \mid x'_{-s} = x_{-s})}{\frac{1}{m} P_*(x_s \mid x_{-s} = x'_{-s})} = \frac{P_*(x'_s \text{ and } x'_{-s} = x_{-s})}{P_*(x_s \text{ and } x_s = x'_{-s})} \times \frac{P_*(x'_{-s} = x_{-s})}{P_*(x_{-s} = x'_{-s})} = \frac{P_*(x')}{P_*(x)}$

so:

$$a(x \to x') = \min\left\{ \frac{\tilde{p}_*(x')}{\tilde{p}_*(x)} \frac{T(x' \to x)}{T(x \to x')},\ 1 \right\} = \min\left\{ \frac{p_*(x')}{p_*(x)} \frac{p_*(x)}{p_*'(x')},\ 1 \right\} = 1$$

so MH w/ proposal:

(i) draw $s \sim \text{Unif}([0, m])$

(ii) draw $x_s' \sim p_*(x_s' \mid x_{-s}' = x_{-s})$

(iii) accept candidate w/ prob = 1

} = Gibbs sampling.

$$dX(t) = \vec{u}(X(t))\,dt + \underbrace{\sigma(X(t))}_{=\sigma}\,dW(t) \Rightarrow \partial_t p(x,t) = -\nabla_x\cdot(\vec{u}(x)\,p(x,t)) + \tfrac{1}{2}\sigma^2\nabla_x\cdot[\nabla_x p(x,t)]$$

$$= -\nabla_x\cdot\left(\vec{u}(x)\,p(x,t) - \tfrac{1}{2}\sigma^2\nabla_x p(x,t)\right)$$

$$= -\nabla_x\cdot\left(\underbrace{J(x,t)}_{=0 \text{ in det. balance.}}\right)$$

$$J(x,t) = \vec{u}(x,t)\,p(x,t) - \tfrac{1}{2}\sigma^2\nabla_x p(x,t)$$

$$p(x,t) = e^{S(x,t)}$$

$$\Big\} \left(\vec{u}(x,t) - \tfrac{1}{2}\sigma^2\nabla_x S(x,t)\right)p(x,t) = 0 \quad \forall x$$

$$\nabla_x S(x,t) = +\tfrac{1}{2}\sigma^2\vec{u}(x,t)$$

if $\vec{u}(x,t) = \nabla_x \log(\tilde{p}_*(x))$

$$\nabla_x S(x) = \tfrac{2}{\sigma^2}\nabla_x \log(\tilde{p}_*(x))$$

$$\Rightarrow S(x) = \tfrac{2}{\sigma^2}\log(\tilde{p}_*(x))$$

$$p_s(x) \propto e^{\frac{2}{\sigma^2}\log(\tilde{p}_*(x))} = \tilde{p}_*(x)^{(2/\sigma^2)} = \underbrace{\tilde{p}_*^{1/\tau}}_{\text{annealed target}} = \tilde{p}_* \text{ if } \tau = 1 \quad (\sigma = \sqrt{2})$$

* could move temperature onto the drift term...

- two algorithms:

(i) Euler-Maruyama (or more advanced simulation of) the SDE itself

(ii) Metropolis-Adjusted Langevin:

(a) $Y \sim \mathcal{N}(X_t + \nabla_x \log(\tilde{p}_*(x))\big|_{X_t}\Delta t,\ \sigma^2\Delta t\, I)$

(b) accept $X_{t+1} = Y$ w/ prob. according to MH rule. else reject $X_{\ldots} = X_t$

• acceptance probability:

$$\min\left\{ \frac{\tilde{p}_t(x')}{\tilde{p}_t(x)} \frac{p_1(x' \to x^*)}{p_1(x \to x')}, 1 \right\}$$

$$p_1(x' \to x) = \frac{1}{(2\pi)^{d/2}(\sigma^2\Delta t)^{d/2}} \exp\left(-\frac{1}{2\sigma^2\Delta t}\|x - (x' + \nabla_x \log(\tilde{p}(x))|_{x=x'}\Delta t)\|^2\right)$$

$$p_1(x \to x') = \frac{1}{\dots} \dots \exp\left(-\frac{1}{2\sigma^2\Delta t}\|x' - (x + \nabla_x \log(\tilde{p}(x))|_{x=x}\Delta t)\|^2\right)$$

$$= e^{-\frac{1}{2\sigma^2\Delta t}\left(\|\underbrace{(x-x')}_{-\Delta x} - \nabla_x \log(\tilde{p}(x))|_{x=x'}\Delta t\|^2 - \|\underbrace{(x'-x)}_{+\Delta x} - \nabla_x \log(\tilde{p}(x))|_{x=x}\Delta t\|^2\right)}$$

$$= e^{-\frac{1}{2\sigma^2\Delta t}\left(\|\Delta x\|^2 + 2(\nabla_x \log(\tilde{p}(x))|_{x=x'}\Delta t)\cdot\Delta x + \|\nabla_x \log(\tilde{p}(x))|_{x=x'}\Delta t\|^2 - \|\Delta x\|^2 + 2(\nabla_x \log(\tilde{p}(x))|_{x=x}\Delta t)\cdot\Delta x - \|\nabla_x \log(\tilde{p}(x))|_{x=x}\Delta t\|^2\right)}$$

$$= e^{-\frac{1}{2\sigma^2\Delta t}\left(2\underbrace{(\nabla_x \log(\tilde{p}(x))|_{x=x'}\Delta t + \nabla_x \log(\tilde{p}(x))|_{x=x}\Delta t)}_{2\times \text{average gradient}}\cdot\Delta x + \underbrace{\|\nabla_x \log(\tilde{p}(x))|_{x=x'}\Delta t\|^2 - \|\nabla_x \log(\tilde{p}(x))|_{x=x}\Delta t\|^2}_{\mathcal{O}(\Delta x^2)}\right)}$$

for small Δx $\approx e^{-\frac{2}{2\sigma^2}\left(\frac{1}{2}(\nabla_x \log(\tilde{p}(x))|_{x=x'} + \nabla_x \log(\tilde{p}(x))|_{x=x})\cdot\Delta x\right)}$

• doesn't simplify cleanly... skip.

~~Convergence of Gibbs sampling follows since Gibbs is an instance of HH~~

LECTURE 21

Towards Efficient MCMC

Proposals and Challenging Posteriors

Stat 238, Spring 2025 @ UC Berkeley
Alex Strang

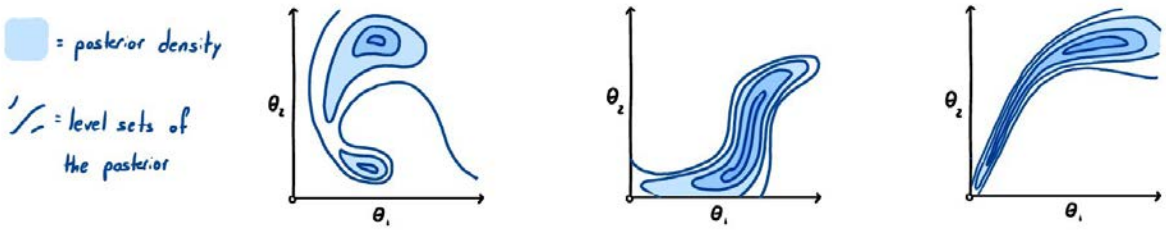
1

MCMC
(MH and Proposal Design)

2

Approximate Inference: Sampling

- Computing exact posteriors can be hard (and often impossible)
- Idea: use samples to approximate the posterior
  - Getting samples for arbitrary posteriors is hard (but possible)
  - Have to choose a sampling method
  - In most cases, use an iterative scheme (Markov Chain Monte Carlo)
  - Can usually sample efficiently *if* we have a "good" proposal



MCMC Methods

Metropolis Hastings:

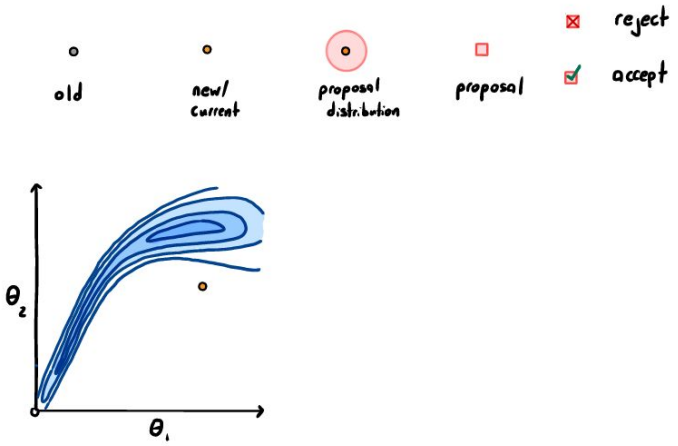
- Propose a small change to your current sample
- Compare the posterior density at the proposal to the current sample
- Accept/reject the proposal accordingly

Most the design freedom enters through the choice of the proposal…

- What are the design constraints on the proposal?
- When will a proposal produce an efficient MH scheme?
- When will a proposal produce an inefficient MH scheme?

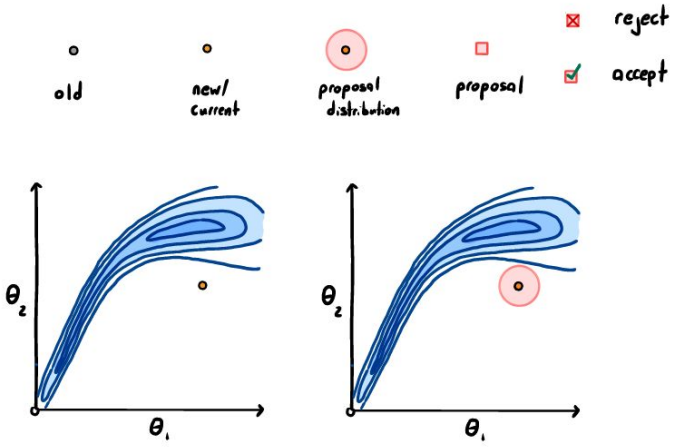
4

Metropolis-Hastings: A Cartoon



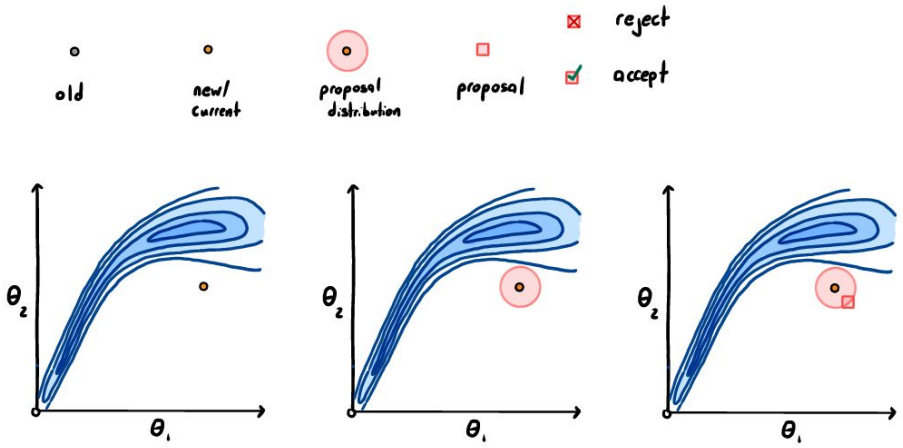
5

Metropolis-Hastings: A Cartoon



6

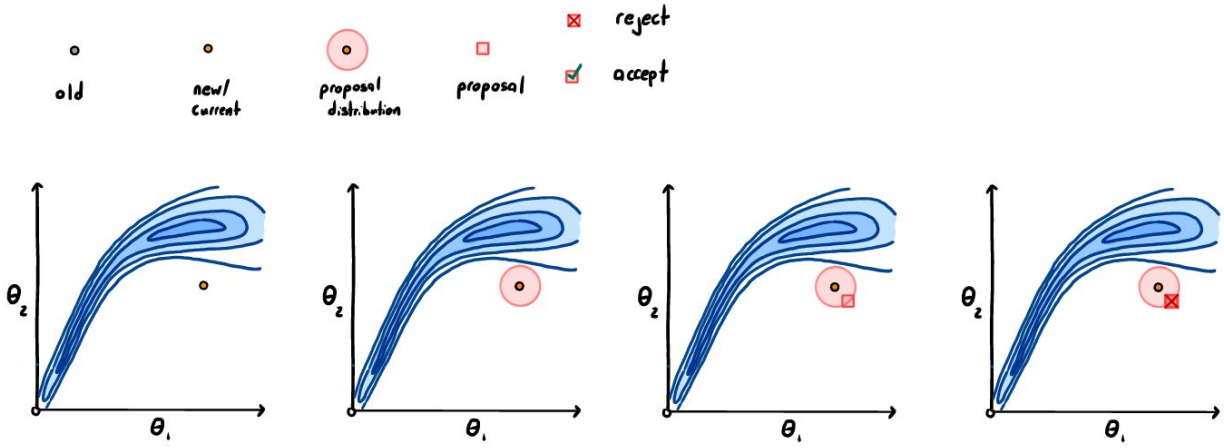
Metropolis-Hastings: A Cartoon



7

CC BY NC SA

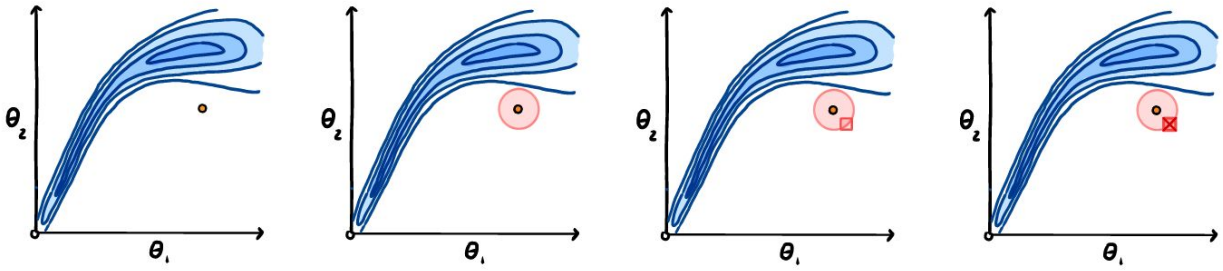
Metropolis-Hastings: A Cartoon



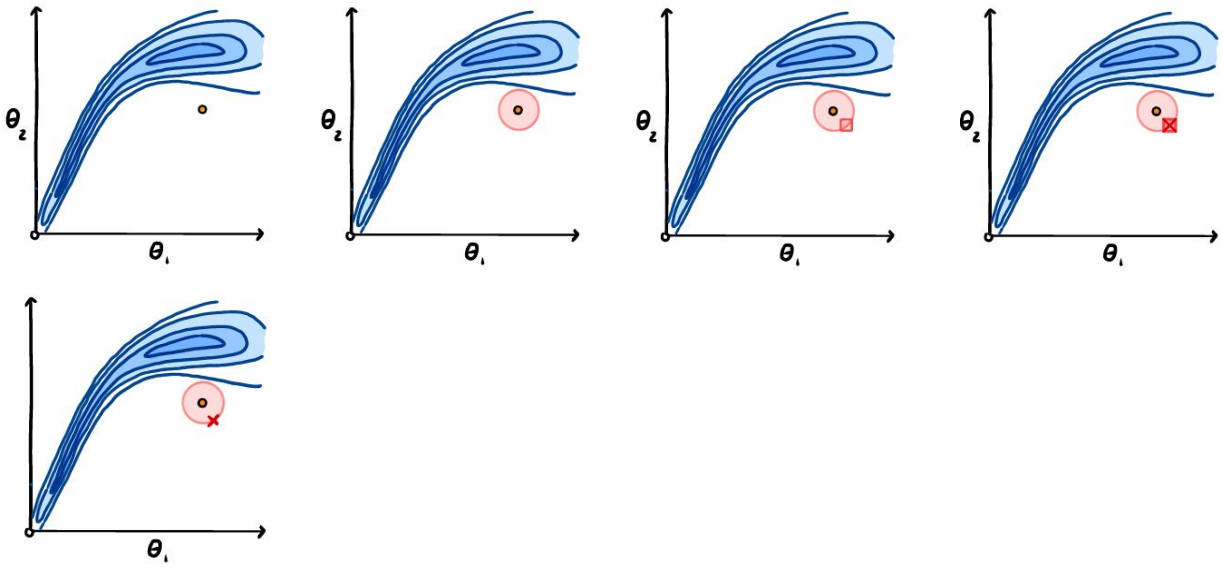
8

CC BY NC SA

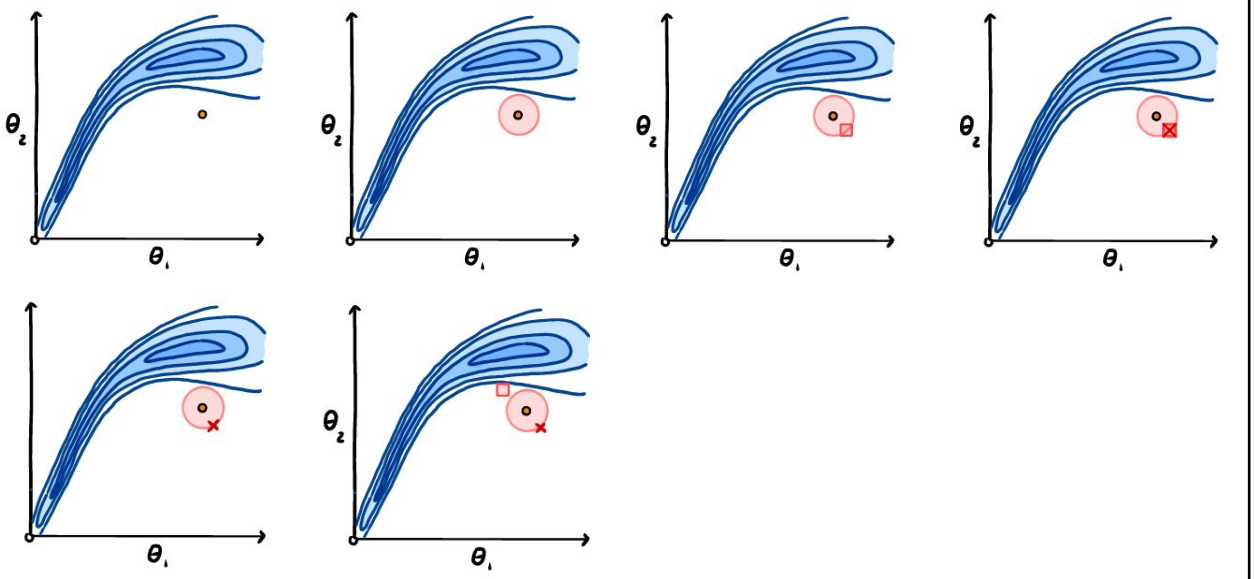
Metropolis-Hastings: A Cartoon



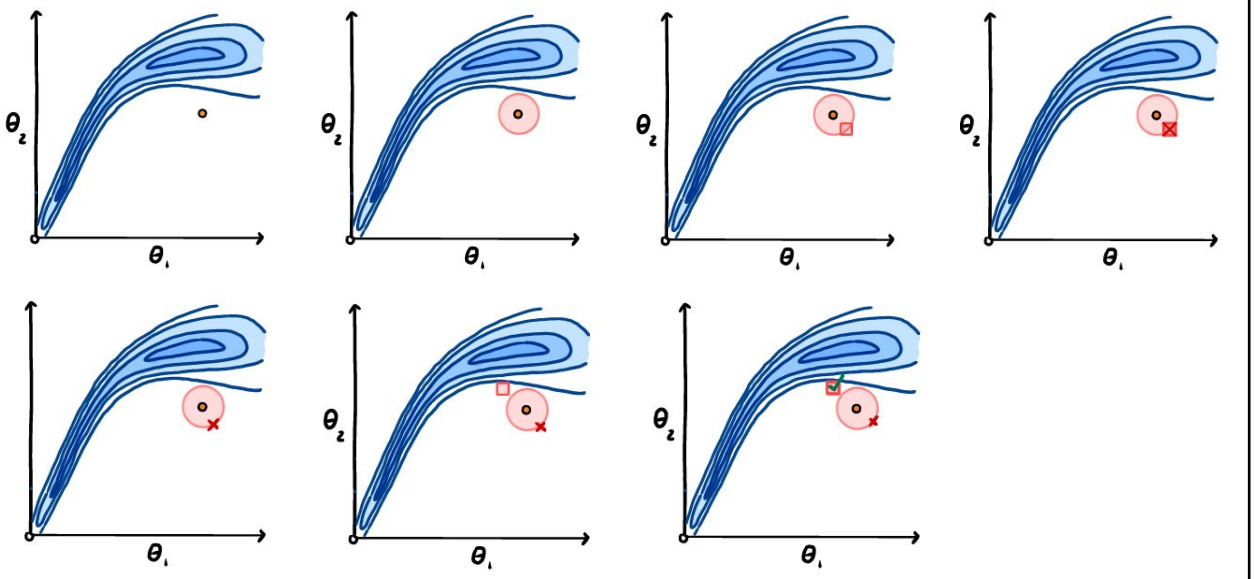
Metropolis-Hastings: A Cartoon



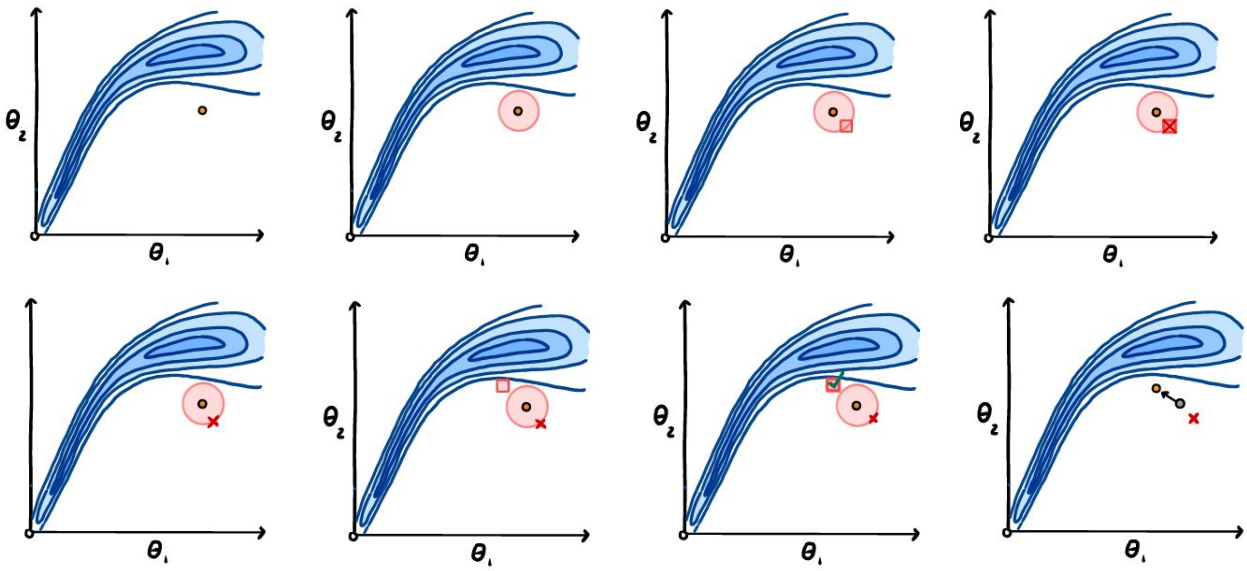
Metropolis-Hastings: A Cartoon



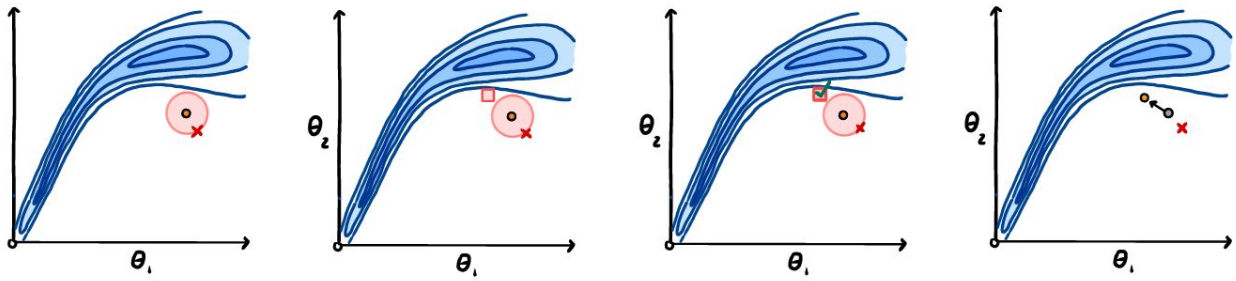
Metropolis-Hastings: A Cartoon



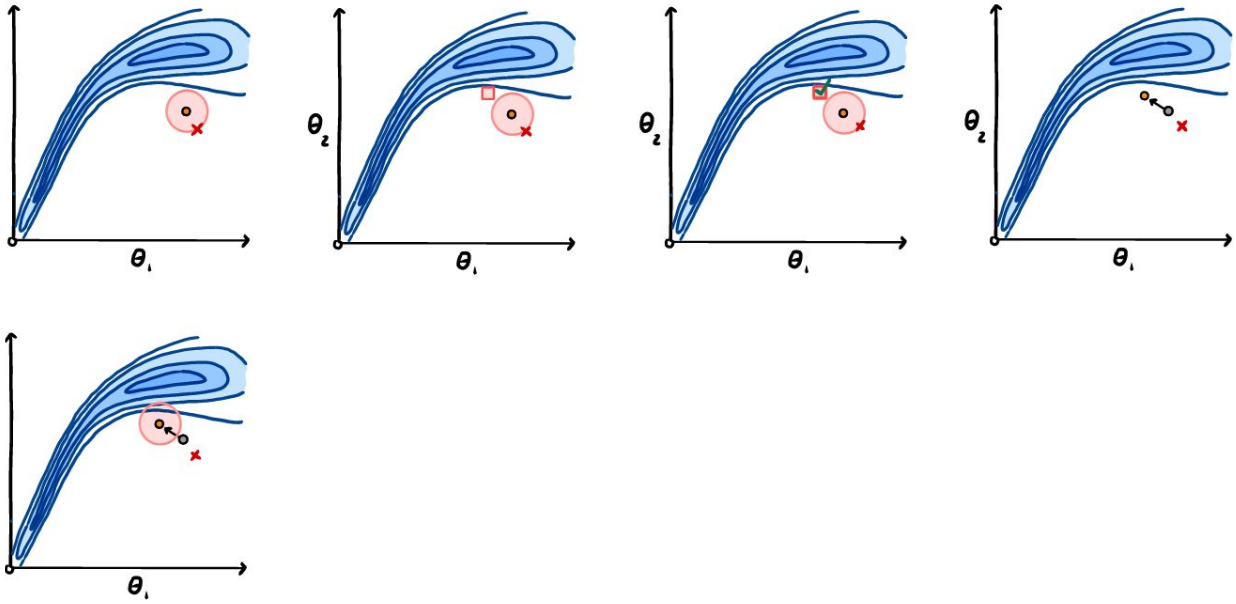
Metropolis-Hastings: A Cartoon



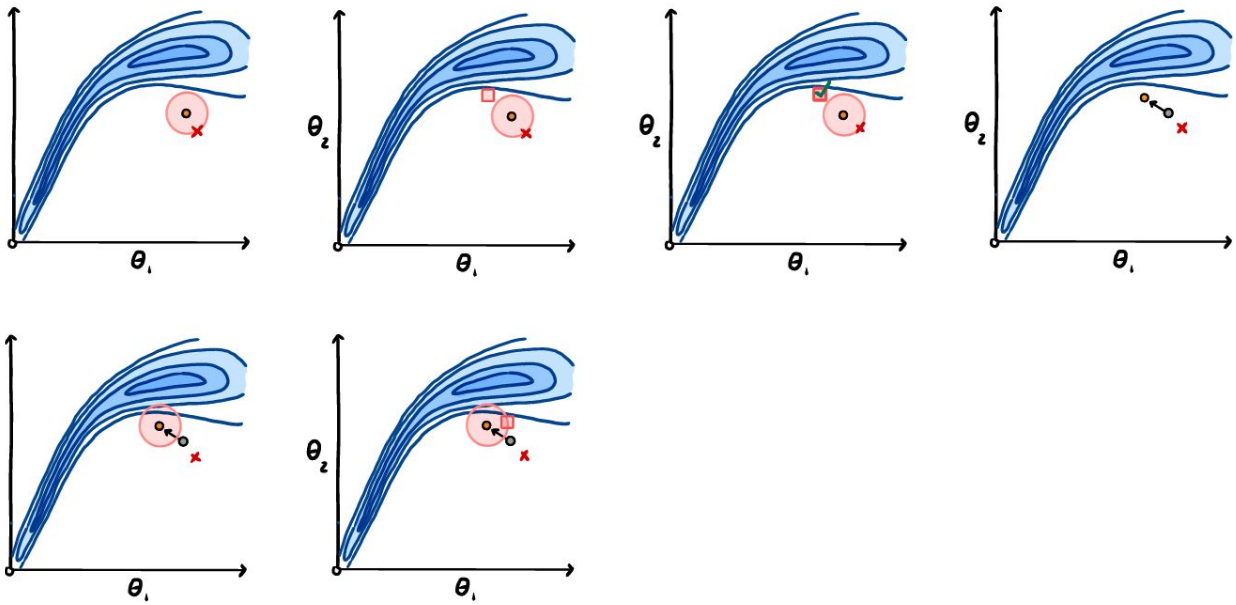
Metropolis-Hastings: A Cartoon



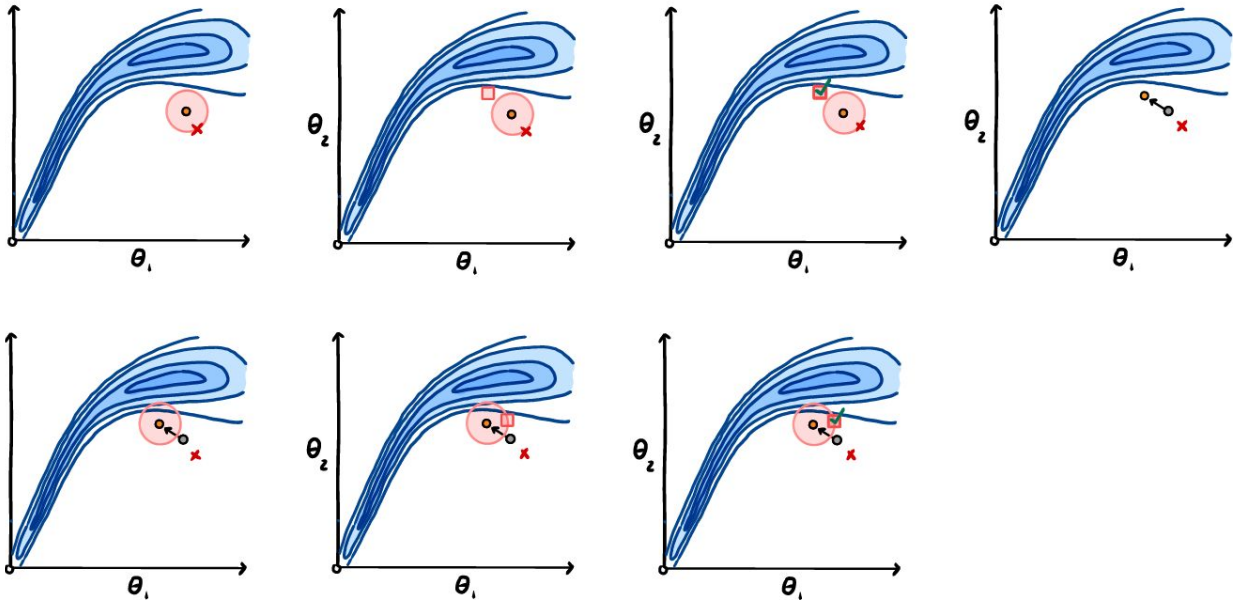
Metropolis-Hastings: A Cartoon



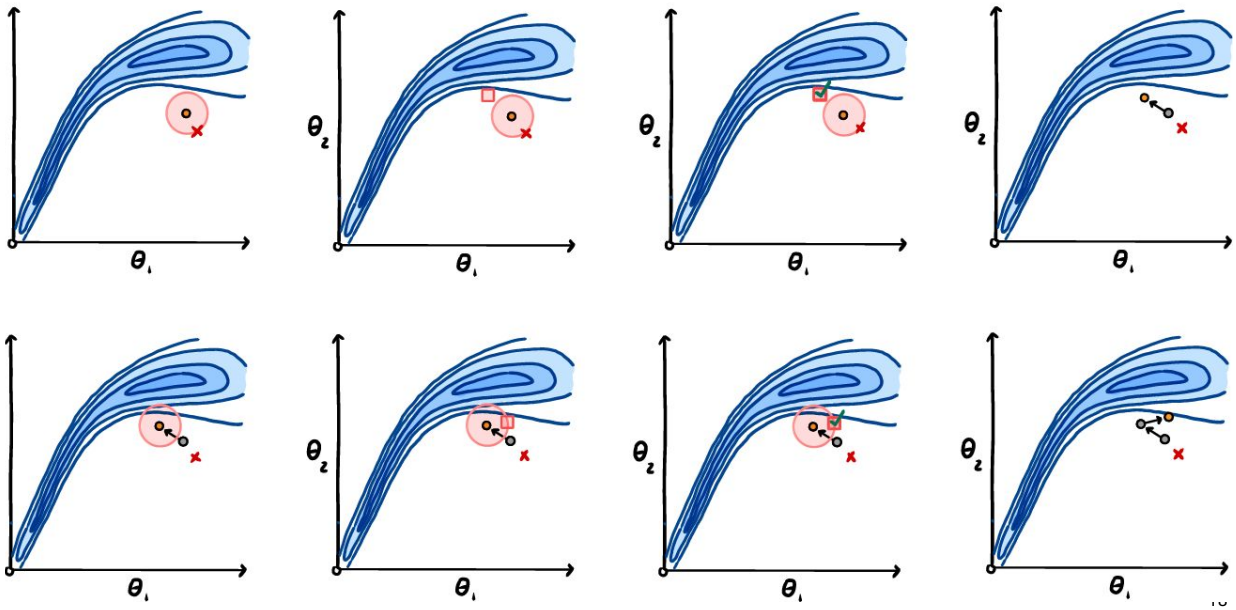
Metropolis-Hastings: A Cartoon



Metropolis-Hastings: A Cartoon



Metropolis-Hastings: A Cartoon



Practical Considerations: Good Proposals Matter

On Proposals:

- Want to take large steps so that samples mix efficiently
- Reduces burn-in time, increases sample efficiency by reducing autocorrelation, may allow jumps between distant modes

But: large steps tend to produce less relevant proposals

- Large steps lead to many rejections

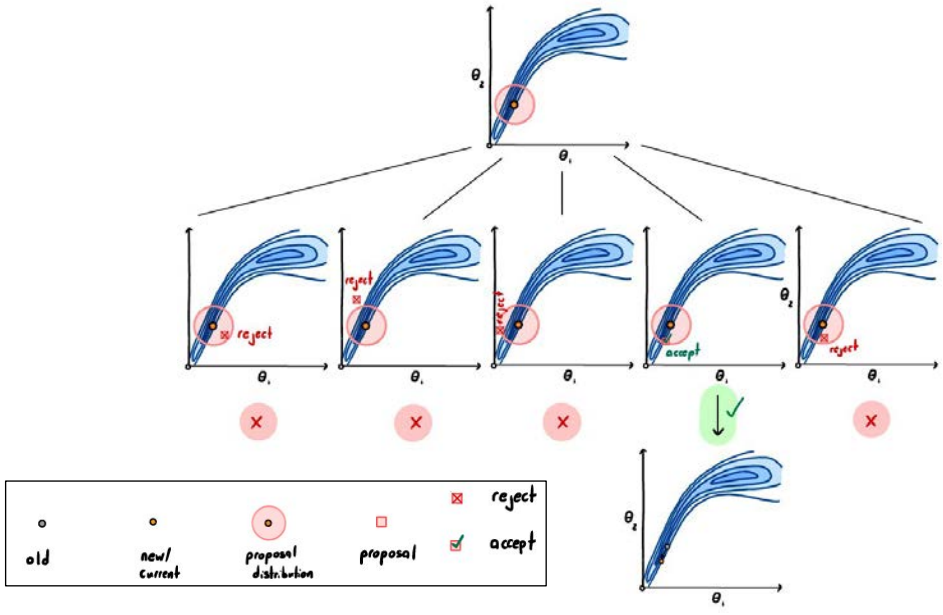
Trade-off:

- Large steps mix more efficiently per accepted step but are usually rejected. Waste most of our time proposing irrelevant candidates.
- Small steps mix inefficiently, but are accepted more often. Waste most of our time mixing slowly.

19

CC BY NC SA

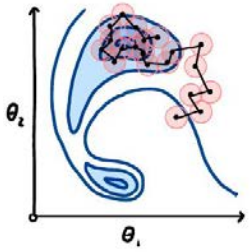
Practical Considerations: Good Proposals Matter



20

CC BY NC SA

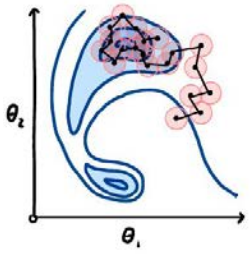
Practical Considerations: Multiple Modes Don't Mix

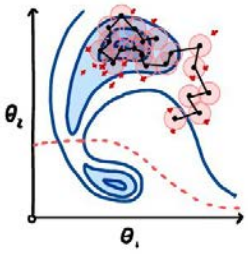


This is a *multimodal distribution.*

It has two peaks separated by a region of low density.

Practical Considerations: Multiple Modes Don't Mix

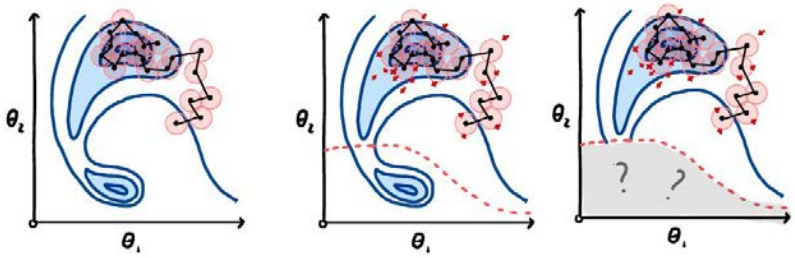




Reject proposals that wander towards the region of low density…

Rarely cross the valley! Rarely find the second mode.

Practical Considerations: Multiple Modes Don't Mix

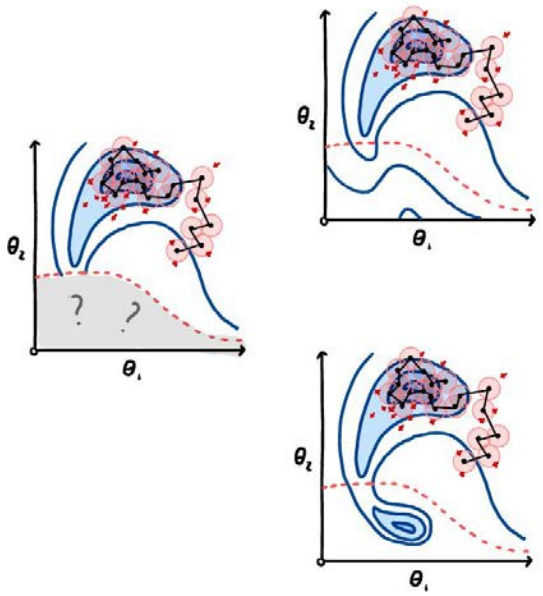


This is sensible, as far as the sampler knows, there is no other mode.

We should rarely wander far into low density regions.

CC BY NC SA

Practical Considerations: Multiple Modes Don't Mix



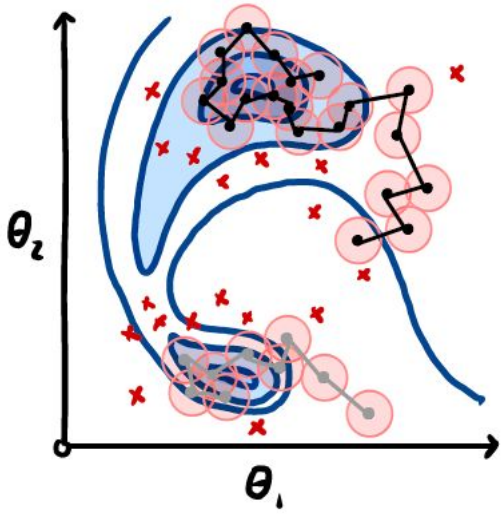
In this case we've done great! Didn't waste time exploring a region of low density.

In this case *we missed a mode!*

CC BY NC SA

24

Practical Considerations: Multiple Modes Don't Mix



This is a hard problem to fix…

Often just try different initializations and hope.

25

CC BY NC SA

Towards Efficient Proposals

Ideal Proposal: Sample each candidate independently from the target…

- Then, acceptance ratio = 1 for all candidates, and mix in a single step
- Intractable (otherwise, we don't need MH)...
- But, can use for inspiration. Can we use more information from the target to inform the proposal?

Ideas:

- The conditionals are often tractable…
- Derivatives of the log target are often tractable…

26

CC BY NC SA

Gibbs Sampling
(Exploit Tractable Conditionals)

27



Gibbs Sampling

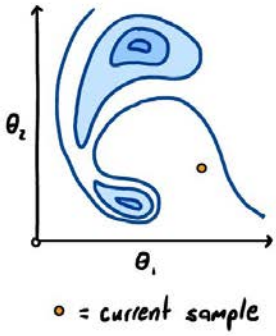
Gibbs Sampling:

- **Iteratively sample one new coordinate at a time while holding the rest fixed** (sample from the exact conditional for the new coordinate given the rest)
- **Advantage:** 1D sampling is easy - can be done in general, can generate *large steps*
  - Either derive the exact conditional (tractable for most hierarchical models)
  - May be able to update many unknowns at once if the unknowns can be partitioned into conditionally independent blocks
  - Or sample via a numerical approximation to the inverse CDF
- **Convergence and Efficiency:** Gibbs Sampling is a special instance of MH with acceptance ratio = 1. Therefore, it converges provided it specifies an irreducible, aperiodic chain (and the target is proper)

28



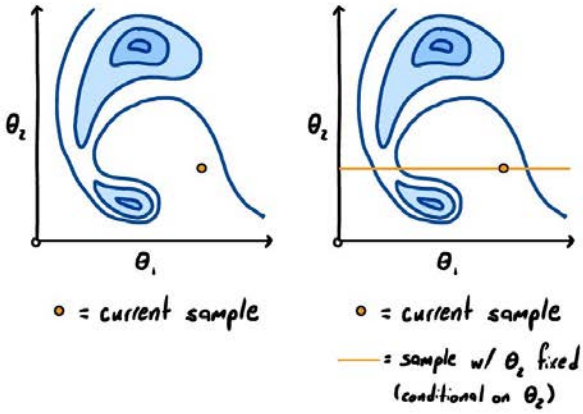
Gibbs Sampling: A Schematic



29

CC BY NC SA

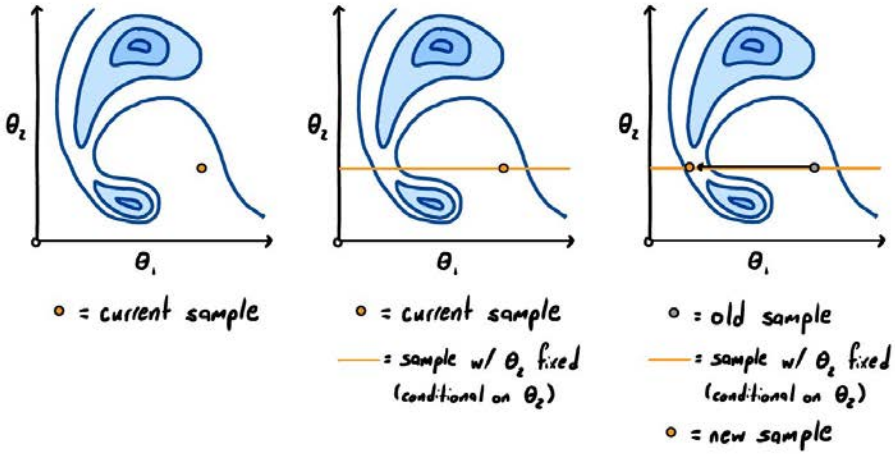
Gibbs Sampling: A Schematic



30

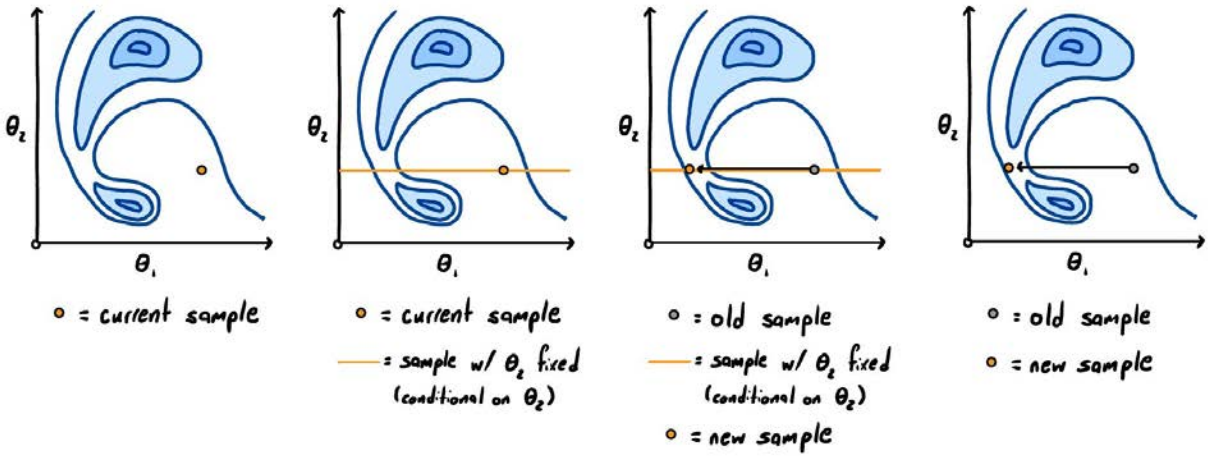
CC BY NC SA

Gibbs Sampling: A Schematic



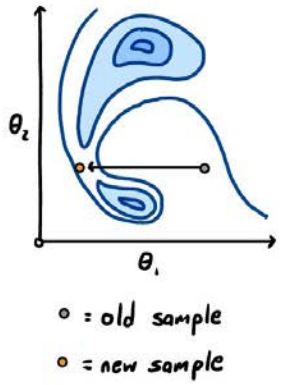
31

Gibbs Sampling: A Schematic



32

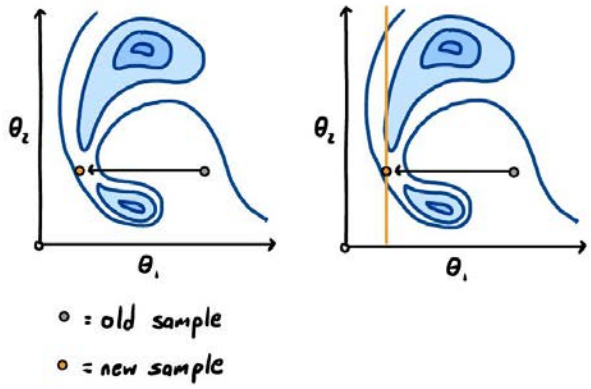
Gibbs Sampling: A Schematic



33

CC BY NC SA

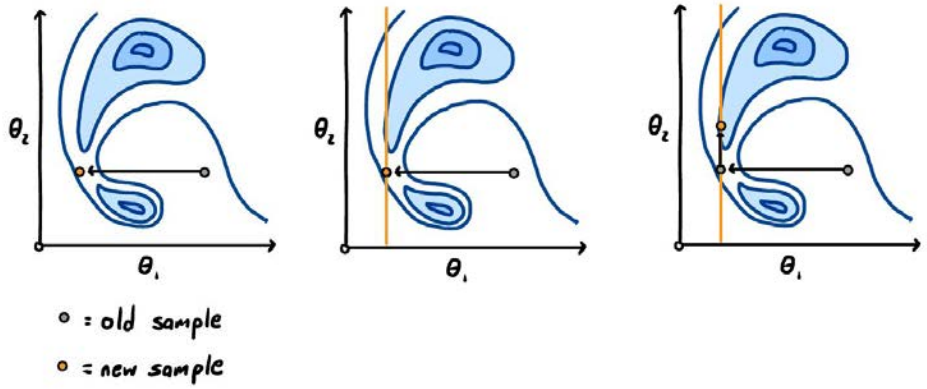
Gibbs Sampling: A Schematic



34

CC BY NC SA

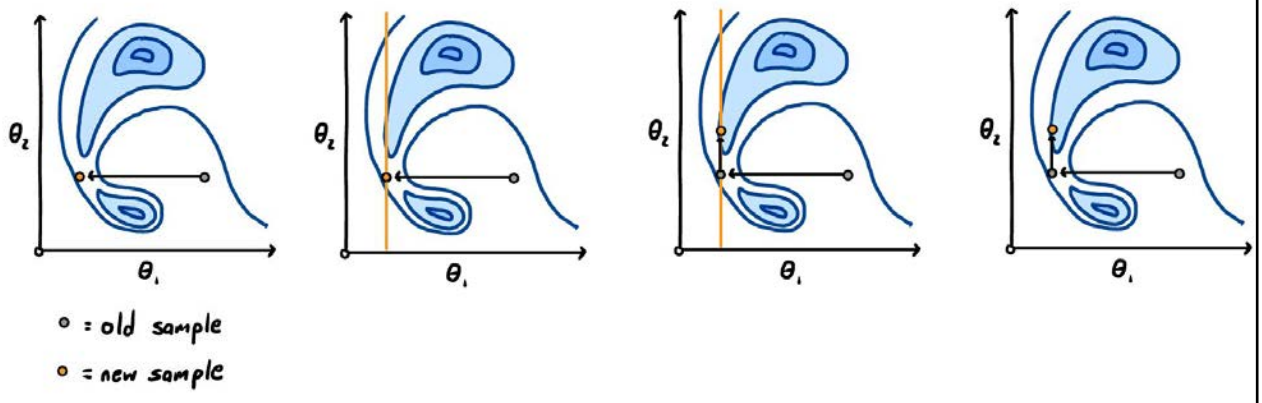
Gibbs Sampling: A Schematic



35

CC BY NC SA

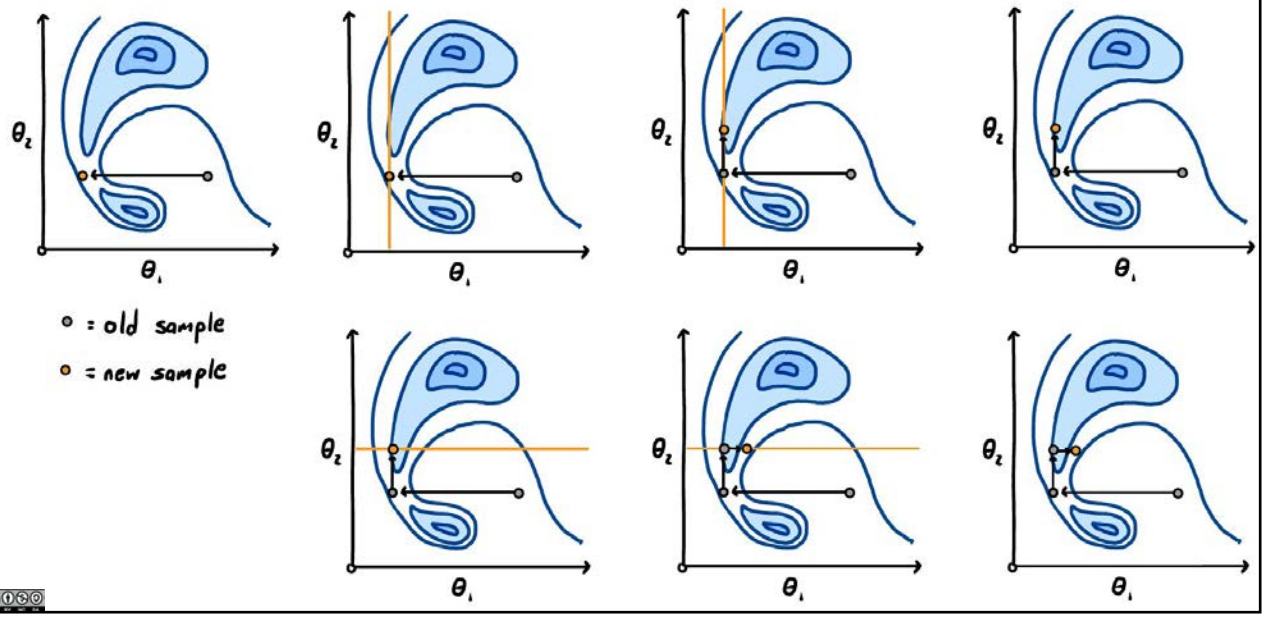
Gibbs Sampling: A Schematic



36

CC BY NC SA

Gibbs Sampling: A Schematic



Lab 7 Example

Model: Student's T prior via a hierarchical scale mixture model…

$$\tau^2 \sim \text{InvGamma}(\alpha, \beta) \quad \rightarrow \quad X|\tau^2 \sim \mathcal{N}(0, \tau^2 I_{n\times n}) \quad \rightarrow \quad Y|X = x \sim \mathcal{N}(Ax, \sigma^2 I)$$

$$S \sim \text{Gamma}(\alpha, \beta^{-1}) \quad \rightarrow \quad X|S = s \sim \mathcal{N}(0, s^{-1} I_{n\times n}) \quad \rightarrow \quad Y|X = x \sim \mathcal{N}(Ax, \sigma^2 I)$$

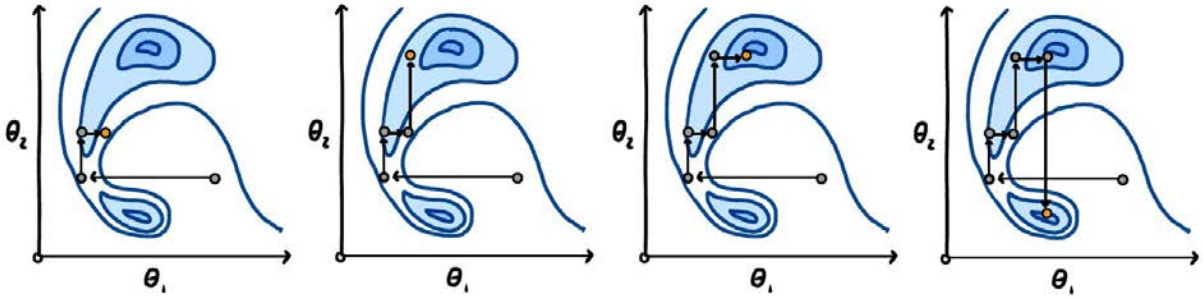
Design Opportunities:

1. The precision *S* is conditionally independent of the data *Y* given the image *X*
2. The model is conditionally conjugate (*S* is conjugate to *X*, *X* is conjugate to *Y* given *S*)

Gibbs Sampler: Alternate between sampling *S* conditional on *X* and *X* conditional on *S* and *Y*.

1. Sample *S* given *X*: $S|X = x, Y = y \sim S|X = x \sim \text{Gamma}\left(\alpha + \frac{n}{2}, \left(\beta + \frac{1}{2}\|x\|^2\right)^{-1}\right)$
2. Sample *X* given *S* and *Y*: $X|S = s, Y = y \sim \mathcal{N}(\ldots, \ldots)$ where …, … follow from a normal, normal update

Gibbs Sampling: Challenges



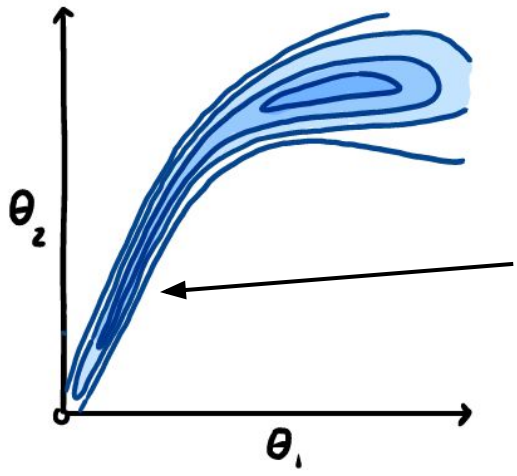
Exercise: *What is a downside of Gibb's sampling? Can you think of an example posterior for which this process would be painfully slow?*

Disadvantage: Only update *one* coordinate at a time!

- Slow if the unknowns are highly coupled and the distribution is concentrated.
- Then we need to update multiple coordinates at once to take large steps.

39

Gibbs Sampling: A Difficult Posterior

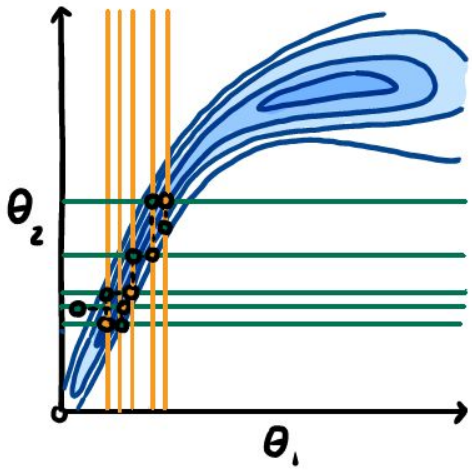


Gibbs sampling only updates *one* coordinate at a time!

- Slow if the unknowns are highly coupled and the distribution is concentrated.
- Then we *need to update multiple coordinates at once to take large steps*.

40

Gibbs Sampling: A Difficult Posterior

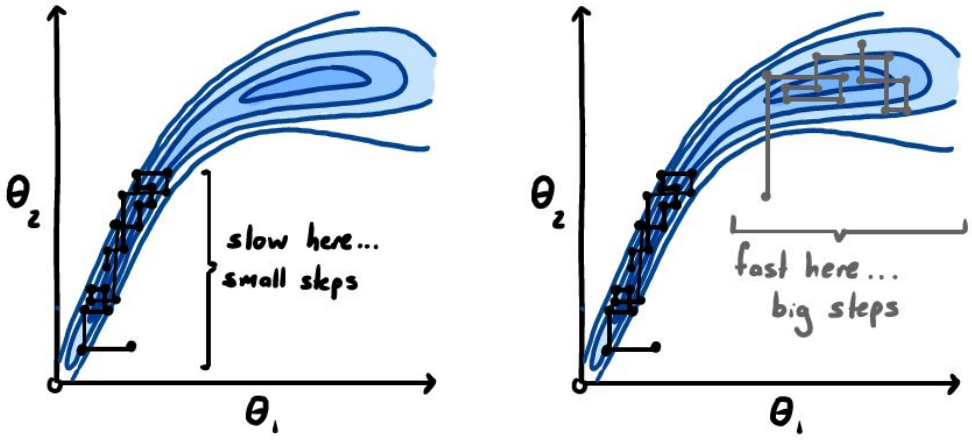


Gibbs sampling only updates *one* coordinate at a time!

- Slow if the unknowns are highly coupled and the distribution is concentrated.
- Then we *need to update multiple coordinates at once to take large steps*.

41

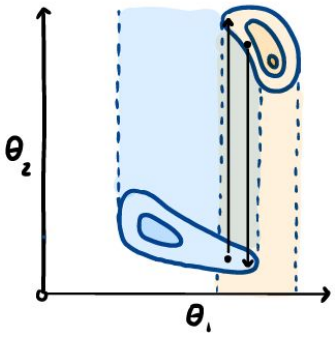
Gibbs Sampling: A Difficult Posterior



42

Gibbs Sampling: Multimodal Posteriors

Gibbs may or may not be able to jump between modes depending on their arrangement.



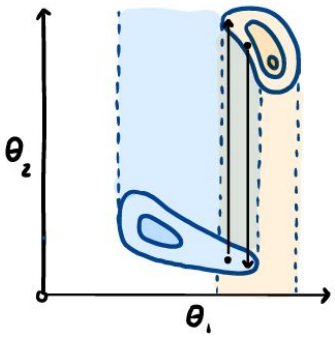
Irreducible! Modes communicate.

43

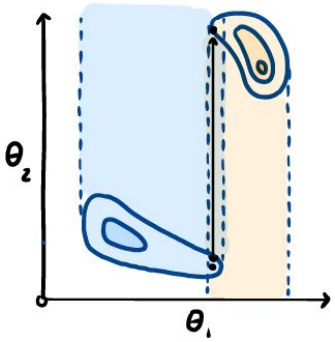
CC BY NC SA

Gibbs Sampling: Multimodal Posteriors

Gibbs may or may not be able to jump between modes depending on their arrangement.



Irreducible! Modes communicate.



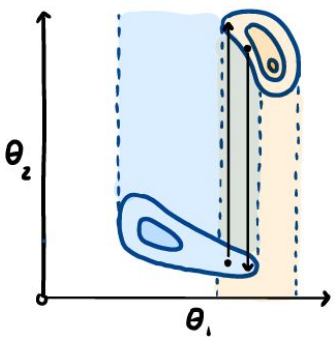
Irreducible, but… modes mix slowly

44

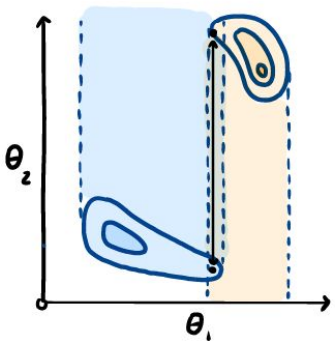
CC BY NC SA

Gibbs Sampling: Multimodal Posteriors

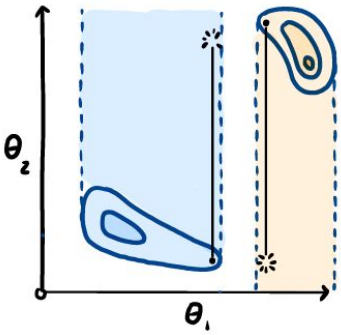
Gibbs may or may not be able to jump between modes depending on their arrangement.



Irreducible! Modes communicate.



Irreducible, but… modes mix slowly



Reducible! Modes do not communicate.

45

Gradient Based Samplers

(Exploit Tractable Log-Derivatives)

46

Gradient Based Samplers

Idea:

- Local information about the target is computable (e.g. Laplace approximation)
- In particular, we can compute derivatives of the *log* target even if we don't know how to normalize the target

$$p_* \propto \tilde{p}_*$$

$$\partial_x^\alpha \log(\tilde{p}_*(x)) = \partial_x^\alpha \log(Z p_*(x)) = \partial_x^\alpha \log(p_*(x)) + \partial_x^\alpha \log(Z) = \partial_x^\alpha \log(p_*(x))$$

$$\nabla_x \log(\tilde{p}_*(x)) = \nabla_x \log(p_*(x))$$

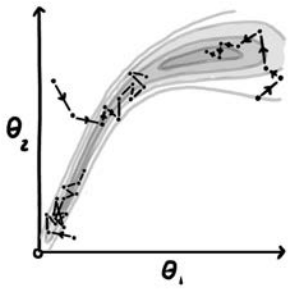
- Can we use these derivatives to inform our proposal/our candidates for an MH procedure?

47

Gradient Based Samplers

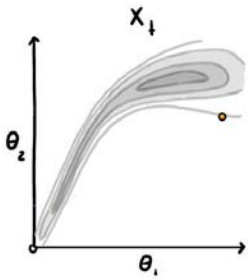
Two Ideas:

1. **Noisy Optimization:** run gradient ascent on the log target + noise
   a. Maximize acceptance ratio by proposing ascent steps
   b. Noise introduces exploration
2. **Noisy Orbits:** simulate a Hamiltonian dynamic that orbits the level sets of an (augmented) log target + noise
   a. Maintain acceptable acceptance ratios by mixing on level sets
   b. Mix efficiently by taking large steps around the target



48

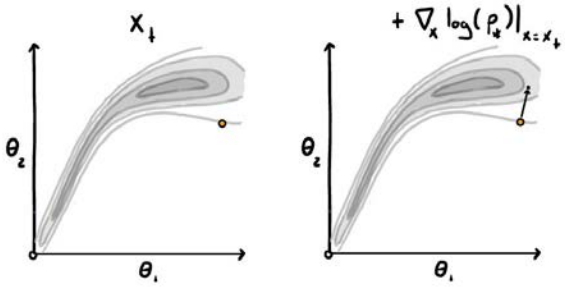
Noisy Optimization



49

CC BY NC SA

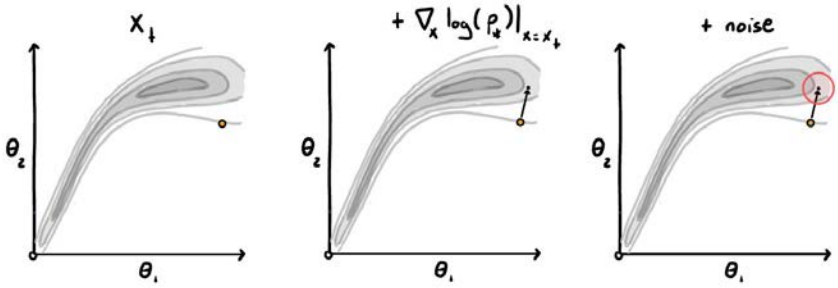
Noisy Optimization



50

CC BY NC SA

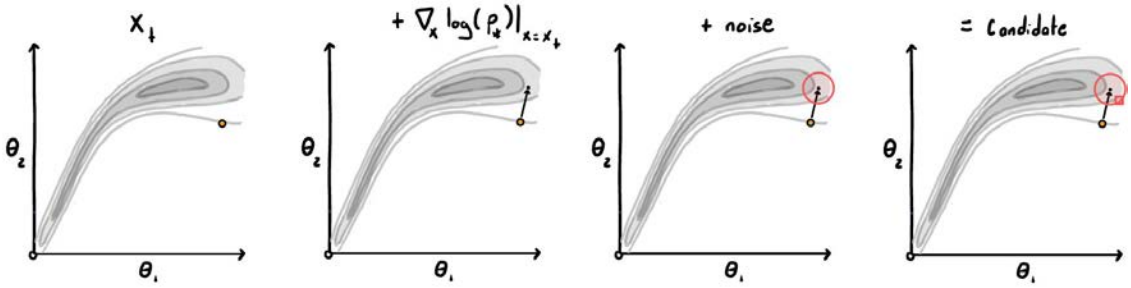
Noisy Optimization



51

CC BY NC SA

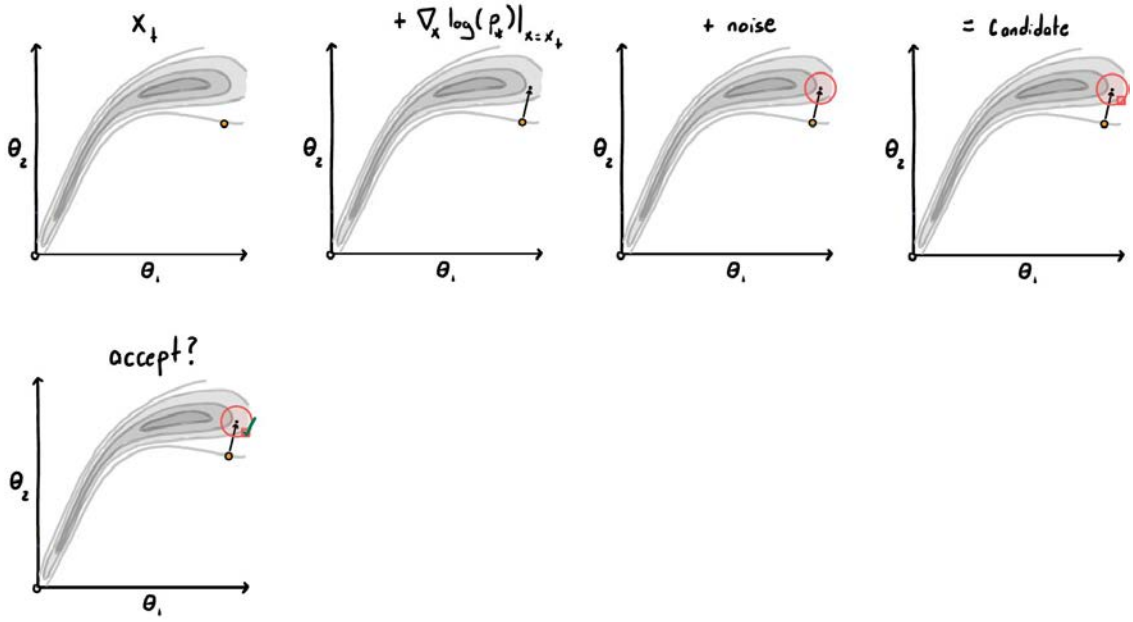
Noisy Optimization



52

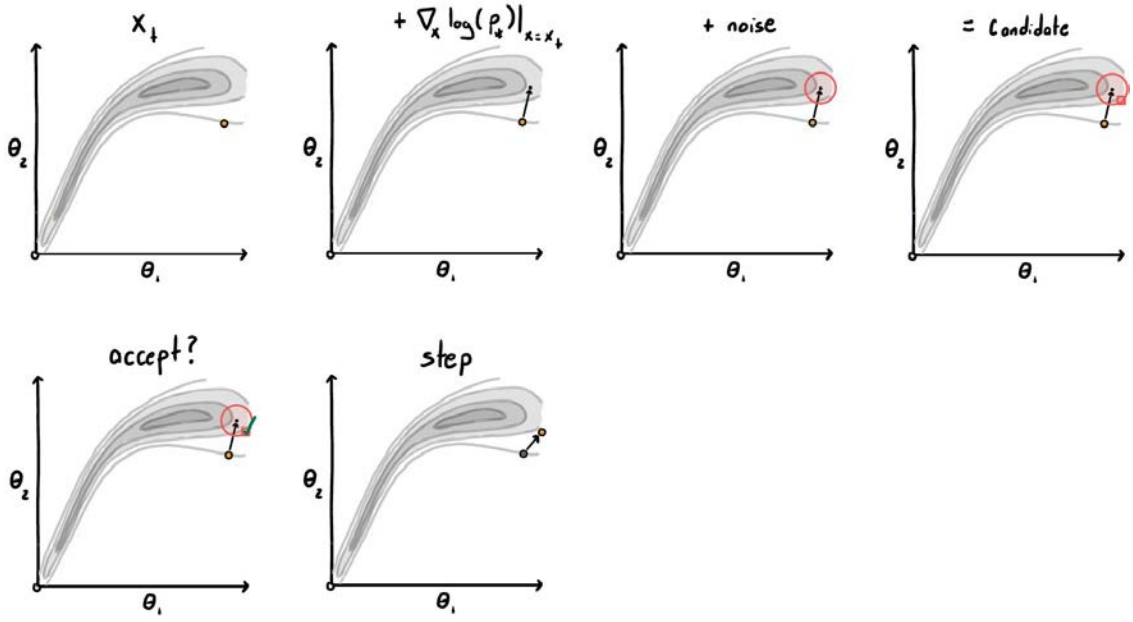
CC BY NC SA

Noisy Optimization



53

Noisy Optimization



54

Stat 238 - Lecture 22 - ∇ based samplers - 04/10/25

Outline:

(i) $\nabla \log(\tilde{p}_\star) = \nabla \log(p_\star)$

(ii) noisy optimization

(a) Langevin (b) Fokker-Plank and annealing (c) Metr. adj. Langevin.

(iii) noisy orbits

(a) ∇ dyn. vs. Hamiltonian dynamics $\Rightarrow$ conserve mechanical energy

(b) HMC idea: propose by randomizing initial momentum

(c) algorithm

(d) joint Monte Carlo interpretation

Stat 238 - Lecture 22 - ∇ based sampling - 04/10/25

Objectives:

- ∇ log target is computable
- ∇ informed proposals:
  (i) noisy optimization ⇒ Metropolis Adjusted Langevin
  (ii) noisy orbits ⇒ Hamiltonian Monte Carlo.

1. ∇ log target is computable:

- **Fact:** given a target ρ_*, known only up to normalization $\tilde{\rho}_*$ we can compute:

$$\partial_x^{\alpha} \log(\rho_*(x)) = \partial_{x_1}^{\alpha_1} \partial_{x_2}^{\alpha_2} \cdots \partial_{x_d}^{\alpha_d} \log(\rho_*(x))$$

for any $\alpha = [\alpha_1, \alpha_2, \ldots \alpha_d] \in \mathbb{Z}^d$ since:

$$\partial_x^{\alpha} \log(\rho_*(x)) = \partial_x^{\alpha} \log(\tilde{\rho}_*(x))$$

for any x where the derivative exists.

Corollary: if $\nabla_x \log(\rho_*(x))$ exists at x then:
$\nabla_x \log(\rho_*(x)) = \nabla_x \log(\tilde{\rho}_*(x))$

- **Proof:** $\tilde{\rho}_*(x) \propto \rho_*(x)$ so $\exists\ z \geq 0, < \infty$ s.t.

$$\tilde{\rho}_*(x) = z\, \rho_*(x) \Rightarrow \log(\tilde{\rho}_*(x)) = \log(\rho_*(x)) + \underbrace{\log(z)}_{\text{constant in } x}$$

$$\Rightarrow \partial_x^{\alpha} \log(\tilde{\rho}_*(x)) = \partial_x^{\alpha} \log(\rho_*(x)) + \underbrace{\partial_x^{\alpha} \log(z)}_{=0} = \partial_x^{\alpha} \log(\rho_*(x))$$

$$\therefore \partial_x^{\alpha} \log(\tilde{\rho}_*(x)) = \partial_x^{\alpha} \log(\rho_*(x)). \quad \blacksquare$$

- **Consequence:** we can work w/ $\nabla_x \log(\rho_*(x))$ knowing only $\tilde{\rho}_*(x)$.

  - in practice, compute $\nabla_x \log(\tilde{\rho}_*(x))$: (1) analytically, (2) auto-diff, (3) numerical differencing.

2. ∇ informed proposals:

- **Key Question:** Can we use $\nabla_x \log(p_*(x))$ to inform $p_r(x \to \dots)$ used to produce candidates for MH?
- **Two Ideas:**

(1) **noisy optimization:**

- choose the proposal to usually produce candidates that are "uphill" on the target density
- in essence, try to choose the proposal to keep the acceptance probability high
- then, the MC specified by the proposal alone

$$T(x \to x') = p_r(x \to x')$$

should have a steady state $\approx p_*$.

Thought Experiment: try to set acceptance to 1 on each step

(2) **noisy orbits:**

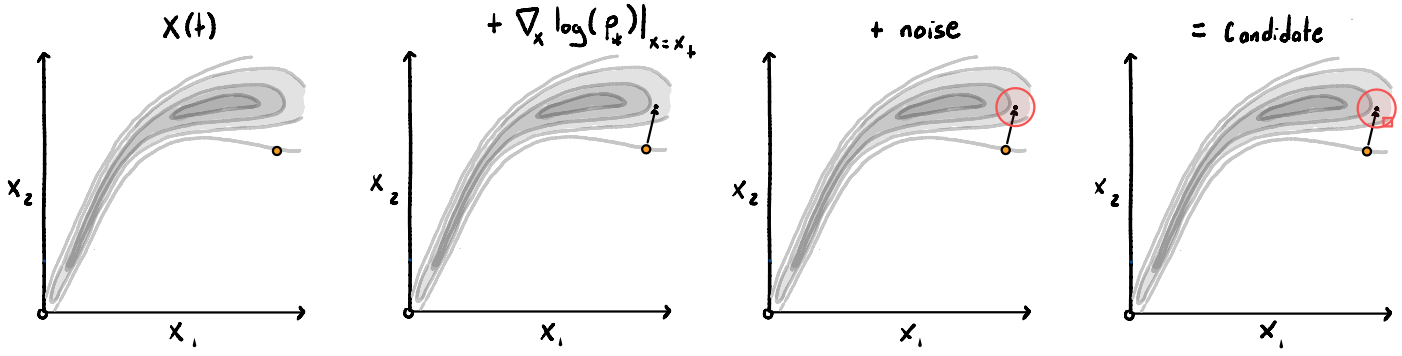
- choose the proposal based on an orbital dynamic that conserves an (augmented) target
- Keep acceptance ratios reasonable
- take large steps

Thought Experiment: can we pose a deterministic dynamic that would mix ergodically over the target?

(1) Noisy Optimization (∇ ascent + noise):

- Consider a proposal of the kind:

$$Y = X(t) + \alpha \nabla_x \log(\tilde{p}_*(x))\Big|_{x = x_t} + \xi, \quad \xi \sim N(0, \beta^2 I)$$



- how to pick α, β?
  - want the proposal dynamic itself to converge to $\rho_s = \rho_*$
    - then, if the dynamic w/out accept/reject obeys detailed balance

$$\rho_r(x \to x') / \rho_r(x' \to x) \approx \rho_*(x') / \rho_*(x)$$ } usually propose uphill steps

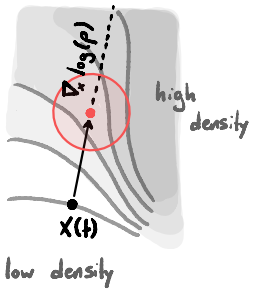
so:

$$a(x \to x') = \frac{\rho_*(x')\, \rho_r(x' \to x)}{\rho_*(x)\, \rho_r(x \to x')} \approx \frac{\rho_*(x')}{\rho_*(x)} \frac{\rho_*(x)}{\rho_*(x')} = 1$$

so the acceptance probability in MH will be close to 1.

- Can be achieved $\approx$ for almost all proposed $x \to x'$ steps if we choose α, β to approximate a Langevin Process ...

- Langevin Process: is a stochastic differential equation (SDE) of the form:



$$dX(t) = \underbrace{\left(\nabla_x \log(\tilde{\rho}_*(x))\big|_{x = X(t)}\right) dt}_{\text{gradient ascent dynamic}} + \underbrace{\sigma\, dW(t)}_{\text{white noise}}$$

$$\frac{d}{dt} x = \nabla_x \log(\tilde{\rho}_*(x))$$

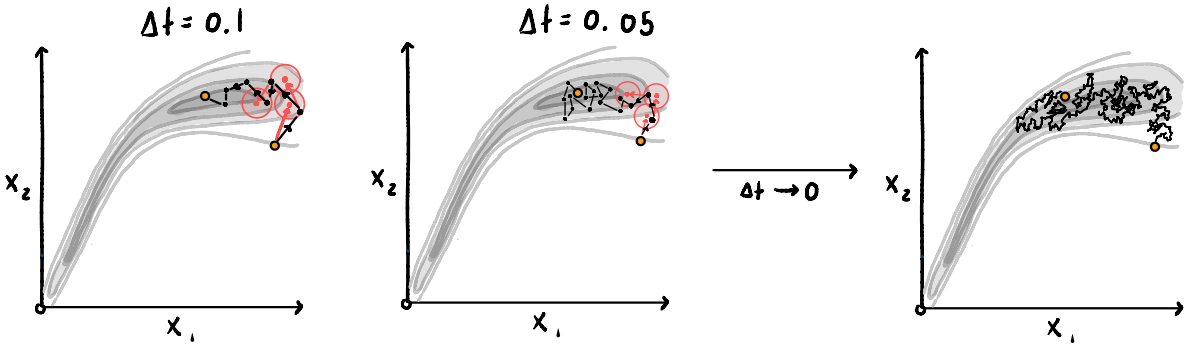
($W(t)$ = Weiner process, or Brownian motion)

formally: the stochastic process formed in the limit as $\Delta t \to 0$ of the discrete-time recursion:

$$X(t+\Delta t) = X(t) + \left(\nabla_x \log(\tilde{\rho}_*(x))\big|_{x = x(t)}\right) \underset{\alpha}{\Delta t} + \underset{\beta = \mathcal{O}(\alpha^{1/2})}{\sqrt{\Delta t}\,\sigma}\, \xi(t)$$

$$\xi(t) \sim \mathcal{N}(0, 1).$$

for finite Δt this is the "Euler-Maruyama" $\approx$ to the SDE



- **Fact**: the steady state density of the Langevin dynamic, ρ_s, is an *annealed* version of the target...

$$\rho_s(x) \propto \rho_*(x)^{1/\tau} \quad \text{for some } \tau > 0 \text{ that depends on } \hat{\sigma} \text{ (noise level)}.$$

  - if $\tau = 1$ then $\rho_s = \rho_*$
  - if $\tau > 1$ then ρ_s is broader than ρ_* ... a useful proposal for importance or rejection sampling

- **Proof**: via the *Fokker-Planck eqn*: let $\rho(x,t) = p(X(t) = x)$

if $dX(t) = \underbrace{\vec{u}(X(t))\,dt}_{\text{det. drift}} + \underbrace{\sigma\, dW(t)}_{\text{noise}}$

then: $\partial_t \rho(x,t) = -\nabla_x \cdot \underbrace{(\vec{u}(x)\,\rho(x,t))}_{\text{advection/transport}} + \underbrace{\tfrac{1}{2}\sigma^2 \nabla_x^2 \rho(x,t)}_{\text{diffusion}}$

$$= \nabla_x \cdot \underbrace{\left(-\vec{u}(x)\rho(x,t) + \tfrac{1}{2}\sigma^2 \nabla_x \rho(x,t)\right)}_{\text{flux } \vec{J}_x[\rho]} = \nabla_x \cdot \vec{J}_x[\rho].$$

net flux out of x

stationarity: $\partial_t \rho(x,t) = 0 \iff$ *Complex balance*: $\nabla_x \cdot \vec{J}_x[\rho] = 0 \;\; \forall\, x$

detailed balance: $J_x[\rho_s] = 0 \Rightarrow -\vec{u}(x)\,\rho_s(x) + \tfrac{1}{2}\sigma^2 \nabla_x \rho_s(x) = 0$

$$\left(\nabla_x \rho_s(x)\right)/\rho_s(x) = \frac{2}{\sigma^2}\,\vec{u}(x)$$

requires $\vec{u}$ to be a conservative vector field $\Leftrightarrow$ det. bal.

$$\longrightarrow \nabla_x \log(\rho_s(x)) = \frac{1}{\tau}\,\vec{u}(x) \quad \text{where } \tau = \tfrac{1}{2}\sigma^2$$

$$= \tfrac{1}{\tau}\nabla_x \log(\tilde{\rho}_*(x)) = \tfrac{1}{\tau}\nabla_x \log(\rho_*(x))$$

need $\nabla_x \log(\rho_s(x)) = \frac{1}{\tau}\nabla_x \log(\rho_*(x)) \;\; \forall\, x$ s.t. $\rho_*(x) \neq 0$

$$\Rightarrow \log(\rho_s(x)) \overset{\text{n.c.}}{=} \frac{1}{\tau}\log(\rho_*(x)) \Rightarrow \rho_s(x) \propto e^{\frac{1}{\tau}\log(\rho_*(x))}$$

$$\Rightarrow \rho_s(x) \propto \rho_*(x)^{1/\tau}. \;\blacksquare$$

- Suggests proposals of the form:

(*) $Y = X(t) + \left(\nabla_x \log(\tilde{p}_*(x))\big|_{x=X(t)}\right)\Delta t + \sigma\sqrt{\Delta t}\,\xi$

$\xi \sim \mathcal{N}(0, I)$

and $\sigma \geq \sqrt{2}$ } $\tau \geq 1$.

} Langevin Proposal

- Metropolis-Adjusted Langevin: given $X(t) = x$

(i) draw candidate Y from Langevin proposal at x

(ii) accept/reject the candidate using the MH criteria

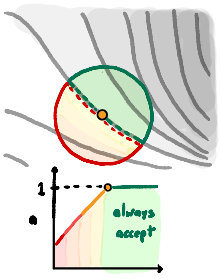
"good" proposals (high acceptance rate)

asymptotically exact

- Pros: • efficient per candidate... acceptance rate ≈ 1
  - rapid/short burn-in (ascends quickly)

- Cons: • based on a small "Δt" argument $\Rightarrow$ short steps
  - still mixes slowly in some cases

- Can we design a method that uses less greedy proposals (is less concerned about the acceptance rate), but takes long steps?
... notice we don't have to go uphill to keep $a = 1$...



(2) noisy orbits: go around instead of up

- physical analogy: imagine a ball rolling on a surface (potential)

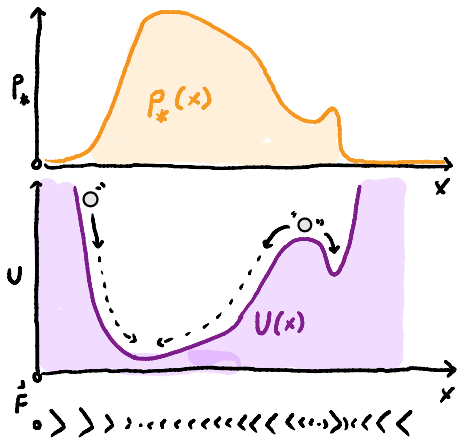
- let: $U(x) = -\log(\tilde{p}_*(x))$ be the "potential"

- Langevin uses overdamped dynamics:

$$\frac{d}{dt}X(t) = \nabla_x \log(\tilde{p}_*(x)) = \underbrace{-\nabla_x U(x)}_{\text{force: } \vec{F}(x) = -\nabla_x U(x)}$$

velocity $\propto$ force...

no momentum, movement in a viscous fluid.

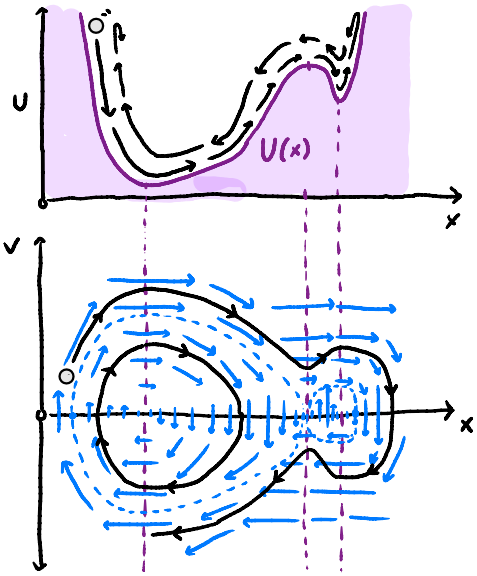


- if we give the ball momentum then get Newtonian mechanics:

acceleration $\propto$ force / mass

- now, the ball rolls back and forth w/out ever stopping in the basin
- constantly trading potential and Kinetic energy, conserving total energy
- now a 2nd order system, need to track position and velocity $\Leftrightarrow$ momentum

"augmenting" the state space



left to run from an initial $x(0), v(0)$ follows a orbit

idea: generate long proposal steps by setting $x(0) = X$, randomly choosing $v(0)$, then simulating forward

now: $M \cdot a = F$ (M: mass, $a = \frac{d}{dt} v$: acceleration, v: velocity)

$M \frac{d}{dt} v = F \Rightarrow \frac{d}{dt}(Mv) = F$

momentum: ϕ

$\Rightarrow \frac{d}{dt}\phi(t) = -\nabla_x U(x)$

$\frac{d}{dt} x(t) = v(t) = M^{-1}\phi(t)$

conserve the total mechanical energy:

$E(x, v) = \underbrace{U(x)}_{\text{potential}} + \underbrace{\frac{1}{2} v^T M v}_{\text{Kinetic}}$

$= U(x) + \frac{1}{2}\phi^T M^{-1}\phi = E(x, \phi)$

- notice: $\nabla_x E(x, \phi) = \nabla_x U(x)$, $\nabla_\phi E(x, \phi) = M^{-1}\phi$

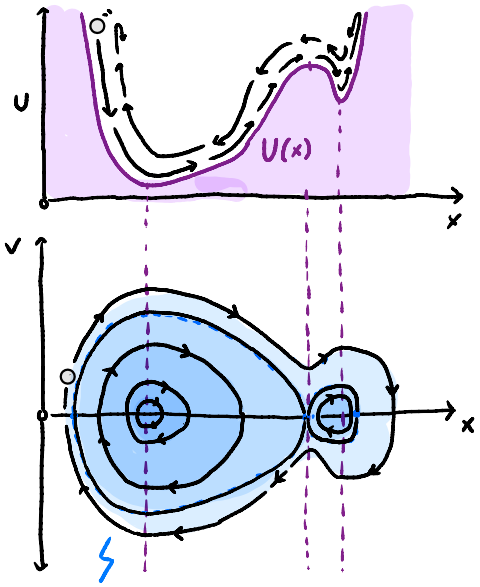
Hamiltonian Dynamics: a coupled dyn. system governed by a Hamiltonian $E : \mathcal{X} \times \Phi \to \mathbb{R}$

$\frac{d}{dt} x(t) = \nabla_\phi E(x, \phi)$, $\frac{d}{dt}\phi(t) = -\nabla_x E(x, \phi)$

- Fact: all Hamiltonian dynamics preserve their Hamiltonian: $E(x(t), \phi(t)) = E(x(0), \phi(0))$

$\frac{d}{dt} E(x, \phi) = \nabla_x E(x, \phi) \cdot \frac{d}{dt} x + \nabla_\phi E(x, \phi) \cdot \frac{d}{dt}\phi$

$= \nabla_x E(x, \phi) \cdot \nabla_\phi E(x, \phi) + \nabla_\phi E(x, \phi) \cdot (-\nabla_x E(x, \phi)) = 0.$ ▨



⊥ intuition to Langevin go around the level sets of the (joint) target instead of uphill.

so: $x(t), \phi(t)$ orbit on level sets of the energy function

$$E(x,\phi) = U(x) + \tfrac{1}{2}\phi^T M^{-1}\phi$$

$$= -\log(p_*(x)) - \left(-\tfrac{1}{2}\phi^T M^{-1}\phi\right)$$

$$= -\log\left(p_*(x)\, \underbrace{e^{-\frac{1}{2}\phi^T M^{-1}\phi}}_{\mathcal{N} \text{ in form}}\right)$$

so $E(x,\phi) \overset{+c}{=} -\log(p_*(x,\phi))$ if we <u>augment the target</u>:

under p_*: $X \sim p_*$, $\phi \sim \mathcal{N}(0, M)$ $\perp\!\!\!\perp$ X.

- <u>Fact</u>: the Hamiltonian dynamic conserves the augmented target
  $p_*(x(t), \phi(t)) = p_*(x(0), \phi(0))$

- <u>Two paths to an algorithm</u>: 1. run MCMC on (X, ϕ) jointly) or

2. given $X_t = x$, initialize a Hamiltonian dynamic from x, ϕ_0 where ϕ_0 is drawn at random

$$\phi_0 \sim p_*(\phi \mid x) = p_*(\phi) \sim \mathcal{N}(0, M) \quad \text{since} \quad X \perp\!\!\!\perp \phi \text{ under joint target}$$

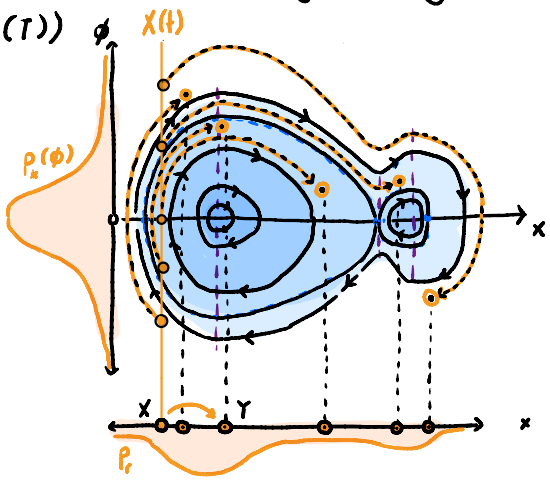
$(x, \phi_0) \longrightarrow (x(s), \phi(s)) \longrightarrow (x(T), \phi(T))$

Hamiltonian Dynamic

↑ long T = large steps

set candidate $Y = x(T)$.

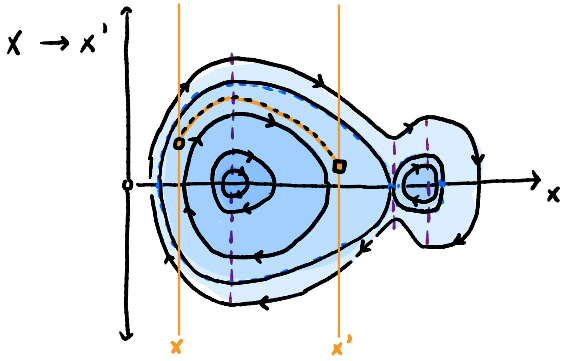
- essentially randomize initial momentum to produce a distribution of candidates



- then, use MH to accept or reject...

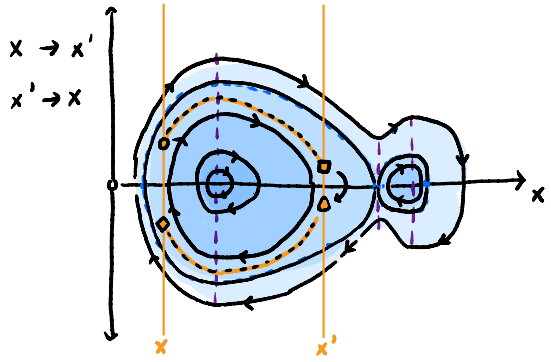
need: $a(x \to x') = \min\left\{ \frac{p_*(x')}{p_*(x)} \frac{p_r(x' \to x)}{p_r(x \to x')}, 1 \right\}$

what is $p_r(x' \to x) / p_r(x \to x')$ under this proposal?



- to go $x \to x'$ need x, ϕ_0 s.t. $x(T) = x'$
- for each such ϕ_0, arrive at $x', \phi(T) = \phi'$
- only one ϕ_0 s.t. $x \to x', \phi'$
- we can match each such path to a reverse path:

$$(x, \phi_0) \longrightarrow (x', \phi')$$
$$(x, -\phi_0) \longleftarrow (x', -\phi')$$



- so, the reverse path is the forward path w/ negated momenta

transition in joint space

$$p_r(x \to x', \phi') = p_t(\phi_0), \quad \phi_0 \sim \mathcal{N}(0, M)$$
$$p_r(x' \to x_0, \phi_0) = p_*(-\phi') = p_*(\phi'), \quad \phi' \sim \mathcal{N}(0, M)$$

- so, working in the joint space:

if simulated exactly, = 1 since preserve p_*

$$a(x_t, \phi_t \to x_{t+1}, \phi_{t+1}) = \min\left\{ \frac{p_*(x_{t+1})\, p_*(\phi_{t+1})}{p_*(x_t)\, p_*(\phi_t)}, 1 \right\} = \min\left\{ \frac{p_*(x_{t+1}, \phi_{t+1})}{p_*(x_t, \phi_t)}, 1 \right\}$$

just the Metropolis rule on the joint target!

- Hamiltonian Monte Carlo: given $x_t = x$ (free parameter: "mass matrix")

(i) draw $\phi_0 \sim \mathcal{N}(0, M)$

(ii) simulate: $\frac{d}{ds} x(s) = \nabla_\phi E(x, \phi) = M^{-1}\phi(s)$, often: leapfrog method

$$\frac{d}{ds} \phi(s) = -\nabla_x E(x, \phi) = \nabla_x \log(\tilde{p}_*(x))\Big|_{x = x(s)}$$

easy to adapt to an annealed target... multiply by $1/Z$ here

from $x(0) = x$, $\phi(0) = \phi_0$ to $x(T), \phi(T)$ (free parameter)

(iii) propose $Y = x(T)$

(iv) accept w/ prob $\min\left\{ \tilde{p}_t(Y)/\tilde{p}_*(x_t) \cdot p_*(\phi(T))/p_*(\phi_0), 1 \right\}$

Stat 238 - Lecture 23 - Density ≈

- Outline:

1. Density Estimation: (from a set of samples)
   - (i) KDE → (iii) GMM via EM
   - (ii) KNN

   } this is a good lecture by itself
   uses: (1) prior modeling, (2) posterior summary and ≈ from samples

[2. Density ≈] (from a function itself)
2 really a different topic

(i) CAVI (ii) SVGD

} Could move to a capstone lecture

---

1. Pipeline thus far

(i) pose model
↓ Lik, prior
(ii) sample for post. inf. ← notice: most samplers don't need a parametric prior or likelihood, only, unnormalized density functions
opens the door to nonparametric approaches to prior modeling that prioritize veracity

- Ex:

(i) K-Nearest Neighbor Density Estimator:

given $\{\theta_i\}_{i=1}^n$ observed, let $R_K(\theta) = d_K(\theta, \{\theta_i\})$ distance to k^{th} nearest neighbor in $\{\theta_i\}$

[≈ w/ density $p(\theta)$ w/

$$\hat{p}(\theta) = \frac{K}{n}\left(\underbrace{V_d \, R_K(\theta)^d}_{\text{volume of } d\text{-dimensional ball of radius } R_K(\theta)}\right)^{-1} \approx p(\theta)$$

]

Stat 238 - Lecture 23 - Sampling & Distribution ≈ - 04/15/25

- Outline:
  1. Sampling Wrap-Up:
     - HMC details
     - Simulated Tempering
  2. Distribution ≈:
     - Role in BDA cycle
     - Ex. Methods: KDE, KNN, GMM (on Thursday)

1. Sampling Wrap Up:

a. HMC details/loose ends:

- Hamiltonian Monte Carlo: given $x_t = x$ (free parameter: "mass matrix")
  - (i) draw $\phi_0 \sim \mathcal{N}(0, M)$
  - (ii) simulate: $\frac{d}{ds} x(s) = \nabla_\phi E(x, \phi) = M^{-1}\phi(s)$, $\frac{d}{ds}\phi(s) = -\nabla_x E(x,\phi) = \nabla_x \log(\tilde{p}_*(x))\big|_{x=x(s)}$ (often: leapfrog method)
    from $x(0) = x$, $\phi(0) = \phi_0$ to $x(T), \phi(T)$ (free parameter)
  - (iii) propose $Y = x(T)$
  - (iv) accept w/ prob $\min\left\{\tilde{p}_*(Y)/\tilde{p}_*(x_t) \cdot p_*(\phi(T))/p_*(\phi_0),\ 1\right\}$

- Questions: (1) How to choose the mass matrix M?
  - Usually, want $M \approx \propto \left(\underset{X \sim p_*}{\mathrm{Var}}(X)\right)^{-1}$ ... improves efficiency
  - to avoid overly singular/ill conditioned M, try $M \approx \propto \left(\underset{X \sim p_*}{\mathrm{Var}}(X) + \varepsilon I\right)^{-1}$
  - in practice, update adaptively by estimating $\mathrm{Var}(X)$ before fixing M for a final MCMC run (adapting MCMC parameters mid-run changes p_s)
  - alternately, set M adaptively using the Hessian of the log target (no guarantee $p_s = p_*$)

(2) How to simulate the dynamic?

$$\frac{d}{ds}x(s) = \nabla_\phi E(x,\phi) = M^{-1}\phi(s),$$
$$\frac{d}{ds}\phi(s) = -\nabla_x E(x,\phi) = \nabla_x \log(\tilde{p}_*(x))\big|_{x=x(s)}$$

so, ideally:

$$x(s+\Delta s) = x(s) + \int_{s'=s}^{s+\Delta s} M^{-1}\phi(s')\,ds'$$
$$\phi(s+\Delta s) = \phi(s) + \int_{s'=s}^{s+\Delta s} \nabla_x \log(\tilde{p}_*(x(s')))\,ds'$$

usually, use a leap frog method:

1. use $x(s), \phi(s)$ to compute $\frac{d}{ds}\phi(s)$ then forward Euler to $\approx \phi(s+\frac{1}{2}\Delta s)$
2. use $\phi(s+\frac{1}{2}\Delta s)$ to $\approx \frac{d}{ds}x(s+\frac{1}{2}\Delta s)$ then "midpoint" Euler to $\approx x(s+\Delta s)$
3. use $x(s+\Delta s)$ to $\approx \frac{d}{ds}\phi(s+\Delta s)$ then "backward" Euler to $\approx \phi(s+\Delta s)$

$\approx \frac{d}{ds}\phi(s)$ as constant over $[s, s+\frac{1}{2}\Delta s]$ and $[s+\frac{1}{2}\Delta s, s+\Delta s]$

$\approx \frac{d}{ds}x(s)$ as constant over $[s, s+\Delta s]$

- $\phi(s+\frac{1}{2}\Delta s) \approx \phi(s) + \frac{1}{2}\frac{d}{ds}\phi(s)\,\Delta s = \phi(s) + \frac{1}{2}\nabla_x \log(\tilde{p}_*(x(s)))\,\Delta s$
- $x(s+\Delta s) \approx x(s) + \frac{d}{ds}x(s+\frac{1}{2}\Delta s)\,\Delta s = x(s) + M^{-1}\phi(s+\frac{1}{2}\Delta s)\,\Delta s$
- $\phi(s+\Delta s) \approx \phi(s+\frac{1}{2}\Delta s) + \frac{1}{2}\frac{d}{ds}\phi(s)\,\Delta s = \phi(s+\frac{1}{2}\Delta s) + \nabla_x \log(\tilde{p}_*(x(s+\Delta s)))\,\Delta s$

↳ LE in the book

(3) How to choose T?

- different suggestions...
  - (·) vary randomly (··) vary adaptively until acceptance is $\approx 65\%$

often handled automatically for you via adaptive methods (e.g. no-U-turn sampling "NUTS")

(4) How to anneal/temper?

- what if we want samples $X \sim$ an annealed target $\tilde{p}_{*,T}(x) \propto \tilde{p}_*(x)^{1/T}$?
- easy, replace $\tilde{p}_*$ w/ $\tilde{p}_{*,T}$ in the algorithm...

$$\frac{d}{ds}x(s) = M^{-1}\phi(s),$$
$$\frac{d}{ds}\phi(s) = \nabla_x \log(\tilde{p}_{*,T}(x))\big|_{x=x(s)} = \nabla_x \log(\tilde{p}_*(x)^{1/T}) = \frac{1}{T}\nabla_x \log(\tilde{p}_*(x))$$

broaden potential U ⇝ $s = T\tilde{s}$ ⇝

$$\frac{d}{d\tilde{s}}x(\tilde{s}) = T M^{-1}\phi(\tilde{s})$$
$$\frac{d}{d\tilde{s}}\phi(\tilde{s}) = \nabla_x \log(\tilde{p}_*(x))$$

- equivalent to: (a) $\phi_0 \sim N(0, TM)$ (initialize w/ excess momentum), (b) run $\frac{d}{ds}x(s) = M^{-1}\phi(s)$, $\frac{d}{ds}\phi(s) = \nabla_x \log(\tilde{p}_*(x))$

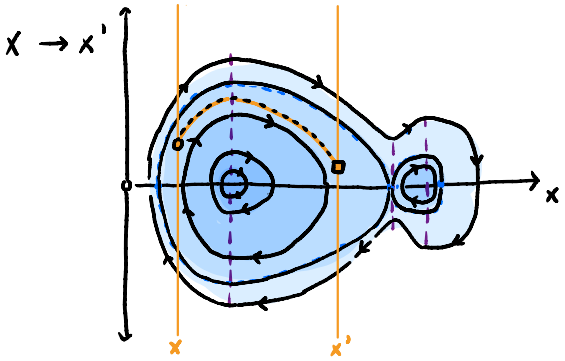
(5) Why a Metropolis acceptance rate on the joint?

- accept w/ prob $\min\left\{ \frac{\tilde{p}_*(Y)}{\tilde{p}_*(x_t)} \cdot \frac{p_*(\phi(T))}{p_*(\phi_0)}, 1 \right\}$

Metropolis on the joint target...

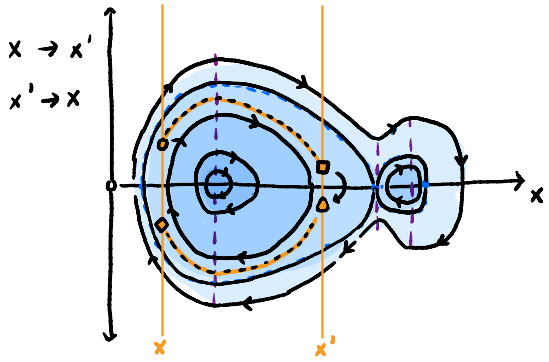
need: $a(x \to x') = \min\left\{ \frac{p_*(x')}{p_*(x)} \frac{p_r(x' \to x)}{p_r(x \to x')}, 1 \right\}$

what is $p_r(x' \to x)/p_r(x \to x')$ under this proposal?



- to go $x \to x'$ need x, ϕ_0 s.t. $x(T) = x'$
- for each such ϕ_0, arrive at $x', \phi(T) = \phi'$
- only one ϕ_0 s.t. $x \to x', \phi'$
- we can match each such path to a reverse path:

$$(x, \phi_0) \longrightarrow (x', \phi')$$
$$(x, -\phi_0) \longleftarrow (x', -\phi')$$



- so, the reverse path is the forward path w/ negated momenta

transition in joint space

$$p_r(x \to x', \phi') = p_*(\phi_0), \quad \phi_0 \sim N(0, M)$$
$$p_r(x' \to x_0, \phi_0) = p_*(-\phi') = p_*(\phi'), \quad \phi' \sim N(0, M)$$

- so, working in the joint space:

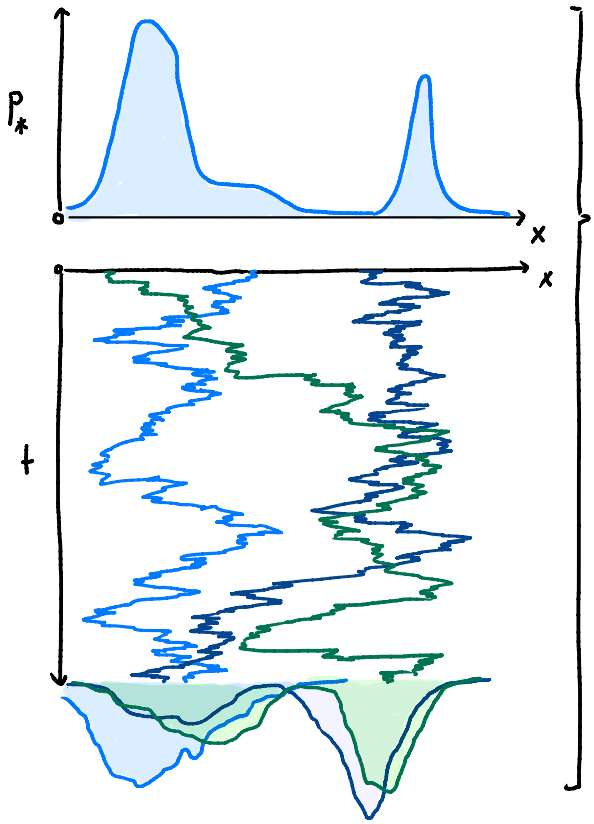
if simulated exactly, = 1 since preserve p_*

$$\left[a(x_t, \phi_t \to x_{t+1}, \phi_{t+1}) = \min\left\{ \frac{p_*(x_{t+1})\, p_*(\phi_{t+1})}{p_*(x_t)\, p_*(\phi_t)}, 1 \right\} = \min\left\{ \frac{p_*(x_{t+1}, \phi_{t+1})}{p_*(x_t, \phi_t)}, 1 \right\} \right]$$

just the Metropolis rule on the joint target!

b. Composite Sampling Methods:

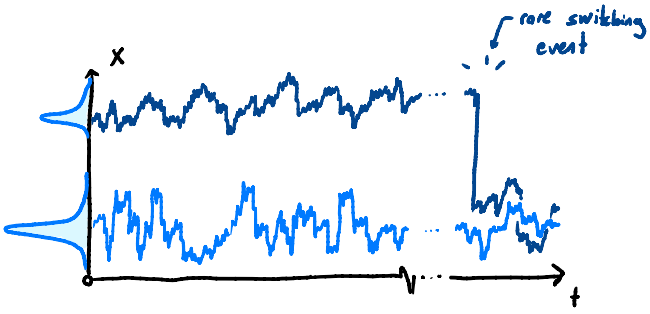
- idea: we can mix multiple chains/targets to speed mixing
- Ex: Simulated Annealing:
  - Suppose p_* is multi-modal w/ well separated modes
  - then, MCMC on p_* directly will mix slowly...



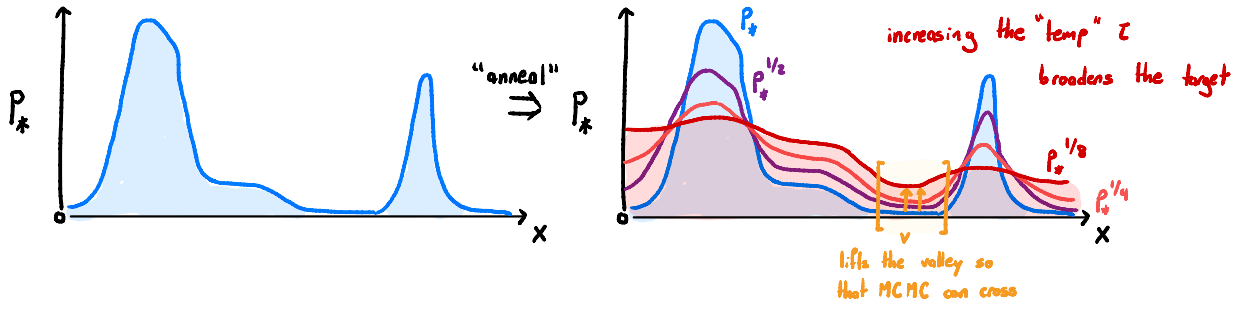
mixes on the time scale of rare crossing events

waiting for transitions between modes to balance

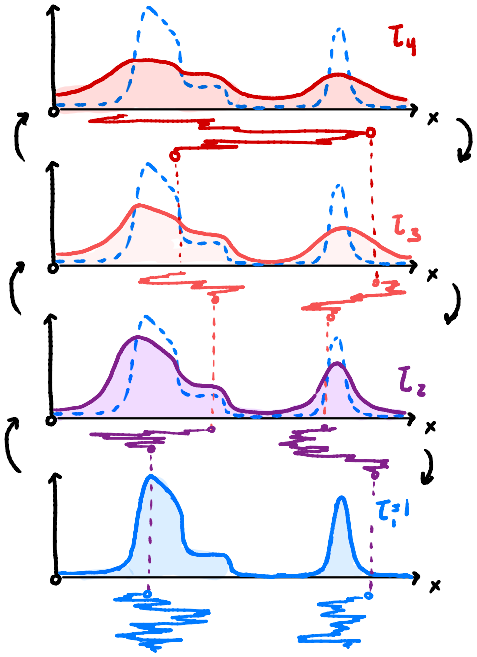
often worse than drawn here... separated modes rarely communicate/share walkers



- idea: the annealed target is easier to sample from and is a relevant proposal...



- **simple idea**: run MCMC on $\tilde{p}_*^{1/\tau}$ for τ large enough then importance weight
  - reasonable... oversampling the tails can reduce variance in sample estimates
- **more sophisticated**: run MCMC on $\tilde{p}_*^{1/\tau}$ for many different τ at once...
- **Simulated Tempering**: let $p_{*,\tau}(x) \propto p_*(x)^{1/\tau} p_0(x)^{1-1/\tau}$ for some base measure p_0 (often uniform, or a very broad dist chosen to cover p_*)



- pick a "**temperature ladder**": $\{\tau_k\}_{k=1}^m$, $\tau_k < \tau_{k+1}$, $\tau_1 = 1$
- pick a target over K, $p_*(K)$ ← set marginal for τ (uniform, or prioritize τ_1)
- run MCMC on $\{x, K\}$ jointly, only store samples when $\tau = 1$ (the unannealed / tempered target)
- **algorithm**: given x_t, K_t

(i) run MCMC on $x_t \to x_{t+1}$ for K_t fixed (MCMC on $p_{*,\tau_{K_t}}$) ↳ e.g. run HMC on $p_{*,\tau_{K_t}}$

(ii) propose a jump $K_t \to K'$ from a proposal $p_r(K \to K')$

(iii) accept according to MH on joint target:

$$p_*(x, K) = p_{*,\tau_K}(x)\, p_*(K) \dots$$

accept w/ prob.:

$$\min\left\{ \frac{p_{*,\tau_{K'}}(x_{t+1})}{p_{*,\tau_K}(x_{t+1})} \cdot \frac{p_r(K' \to K)}{p_r(K \to K')},\ 1 \right\}$$

if $p_0 = \text{Uni}$:

$$= \frac{p_*(x_{t+1})^{1/\tau_{K'}}}{p_*(x_{t+1})^{1/\tau_K}} = \frac{\left(\frac{1}{Z_{K'}}\right)\tilde{p}_*(x_{t+1})^{1/\tau_{K'}}}{\left(\frac{1}{Z_K}\right)\tilde{p}_*(x_{t+1})^{1/\tau_K}} = \frac{Z_K}{Z_{K'}}\,\tilde{p}_*(x_{t+1})^{\frac{1}{\tau_{K'}}-\frac{1}{\tau_K}}$$

need to estimate norm. for each τ_K

Challenge Question: if wrong, what is $p_s(x \mid K=1)$?

2. Distribution Approximation:

- **Task:** try to $\approx$ a density function.

- **Categorization:** (1) Using what information?
  - a. from samples $\Rightarrow$ Density Estimation
  - b. from a functional form $\Rightarrow$ Variational Approximation

  (2) Restricted to what class of densities?
  - a. parametric
  - b. non-parametric

- **Role in Analysis Cycle:**
  - focus on computation (post. inference stage)

input:
- unnormalized posterior function: $\tilde{p}_*(\theta) = Lik(\theta|y)\,\tilde{p}_\theta(\theta)$ * $\rightarrow$

Optimizer / Sampler $\rightarrow$

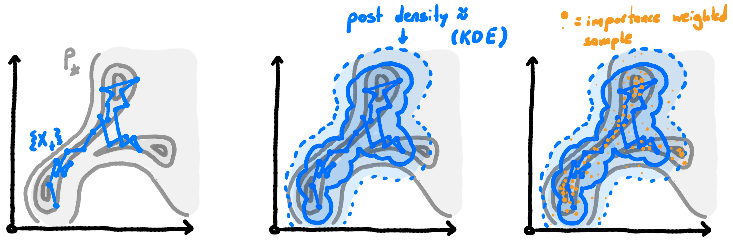
Inferential Summary:
- modal estimator/$\approx$
- $\approx$ posterior samples **

notice: we only need an unnormalized density function for the prior... it need not be parametric

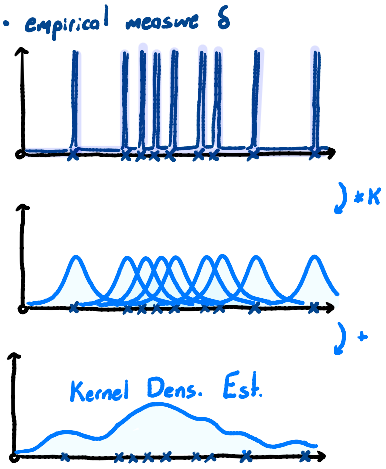
* **Role 1:** given prior data $\{\theta_j\}_{j=1}^N$ use a density estimator to $\approx \tilde{p}_\theta$

Prioritize veracity in modeling ...
- if the estimator is simple then:
  - (a) retain tractability from generality of MCMC tools
  - (b) retain simplicity of regularizing biases... "bias inference towards past samples."

** **Role 2:** given posterior samples approximate posterior density
- often to initialize an importance sampler ... produce dense samples from sparser initial sample... "fill in" MCMC trajectory



Example Density Estimators: for $\theta \in \mathbb{R}^d$



1. Kernel Density Estimator (KDE):

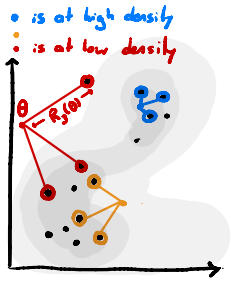
- choose a Kernel function: $K: \Theta \times \Theta \to \mathbb{R}$, $K(\theta, \theta') = K(\theta', \theta)$
  - often, translationally symmetric: $K(\theta, \theta') = K(\theta - \theta')$
    where $K(\theta - \theta') \to 0$ as $\|\theta - \theta'\| \to \infty$
  - often, a function of $\|\theta - \theta'\|$ for some norm
  - e.g.
$$K(\theta, \theta'; \Sigma) = \frac{1}{(2\pi)^{d/2} \det(\Sigma)^{1/2}} e^{-\frac{1}{2}(\theta - \theta')^T \Sigma^{-1} (\theta - \theta')}$$

$$\left[\text{• estimate density: } \hat{p}(\theta; \{\theta_j\}_{j=1}^n) = \left[K * \left(\frac{1}{n}\sum_{j=1}^n \delta(\cdot - \theta_j)\right)\right](\theta) = \frac{1}{n}\sum_{j=1}^n K(\theta, \theta_j)\right]$$

- Key choices: (1) Kernel function (2) the norm used in K
  (often, the "bandwidth" of the Kernel)

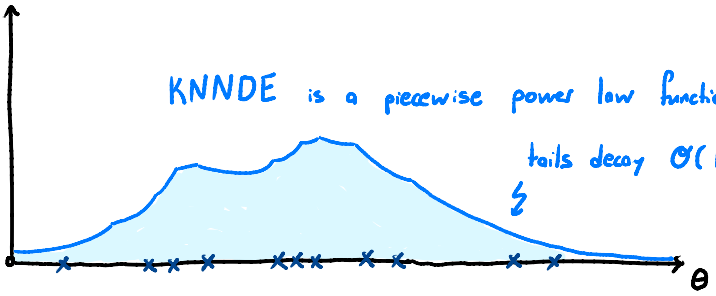
- select to optimize bias-variance trade-off:
  - broader Kernels produce more bias but less variance in estimator
  - optimal bandwidth choices exist asymptotically
    (e.g. minimize MSE or MISE as number of samples $n \to \infty$
    pick bandwidth to decay at some rate in n)
    results depend on regularity of target density, dimension d
  - optimal choices exist for finite n and particular densities (e.g. $\mathcal{N}$ or $\approx \mathcal{N}$)
- consistent (KDE $\hat{p} \to p$ as $n \to \infty$) for appropriate choice of Kernel sequence given n.

2. K-Nearest Neighbor Density Estimator (KNNDE):



- idea: $\hat{p}(\theta)$ is probably large if θ is close to many samples
  ... ... it's nearest samples
  - choose: $K \in \mathbb{Z}$, $K > 1$
  - let:
    - $R_K(\theta)$ = distance from θ to it's K^{th} nearest neighbor in $\{\theta_j\}_{j=1}^n$

$$\left[\hat{p}(\theta; \{\theta_j\}_{j=1}^n, K) = \frac{K}{n}\underbrace{\left(V_d\, R_K(\theta)^d\right)^{-1}}_{\text{volume of } d\text{-dim ball of radius } R_K(\theta)} \approx p(\theta)\right]$$



sensible as a prior:
- bias towards past samples
- but do so weakly (heavy tails)

- Key choice: the # K of neighbors to look out

- select to optimize bias-variance trade-off:
  - larger K produce more bias but less variance in estimator
  - optimal bandwidth choices exist asymptotically
    (e.g. minimize MSE or MISE as number of samples $n \to \infty$
    pick bandwidth to decay at some rate in n)
    results depend on regularity of target density, dimension d

- e.g. $\text{bias}(\hat{p}(x); n, K) = \mathcal{O}((K/n)^{2/d}) + \mathcal{O}(K^{-1})$ w/ constants bounded by roughness of p ($p''(x)/p^2(x)$)
  $\text{Var}(\hat{p}(x); n, K) = \mathcal{O}(K^{-1})$ ↑ in KDE, also depends on d

  $\text{MSE} = \text{bias}^2 + \text{var}$ so

  $$\text{MSE}(\hat{p}(x); n, K) = \mathcal{O}((K/n)^{4/d}) + \mathcal{O}(K^{-1})$$

  ↑ need $K/n \to 0$ as $n \to \infty$ to eliminate bias
  ↑ need $K \to \infty$ to eliminate variance

  choose $K(n)$ to match bias & variance contributions to MSE
  e.g., for $d=1$ pick:

  $$K(n) = \mathcal{O}(n^{4/5}), \quad \text{then} \quad \text{MSE}(\hat{p}(x); K(n), n) = \mathcal{O}(n^{-4/5})$$

- for appropriate $K(n)$ the KNNDE is consistent and minimax optimal (smallest worst case error) in ℓ_p error on density (for some p, depending on knowledge of support)

3. Gaussian Mixture Model: parametric, approximate $\hat{p}(\theta) = \sum_{j=1}^{m} w_j \, N(\theta; \mu_j, C_j)$ ↑ dense in the space of dist's (universal density ☺)
   optimize over $\{w, \mu, C\}$ using Expectation-Maximization.
   ↑ next class...

Stat 238 - Lecture 24 - Density & (intro to) Model Checking - 04/17/25

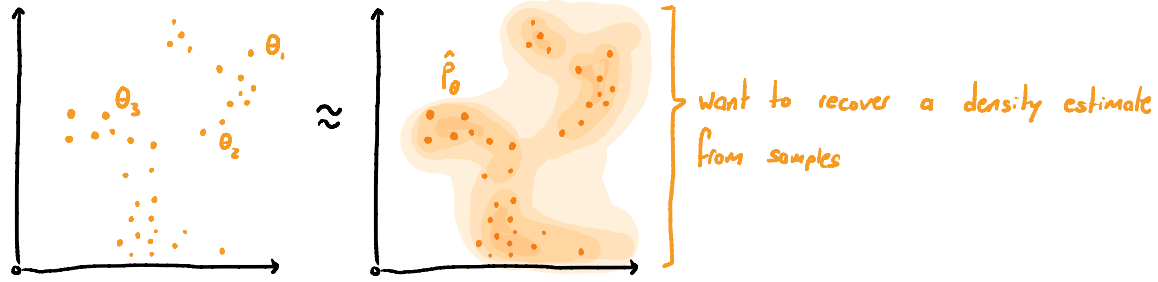
Outline:

1. Density Estimation through GMM → notes } 30 min
2. Model Checking via Sampling...
   (i) Does my computational procedure for past inference work?
   (ii) Is my model plausible?
   } 30 min

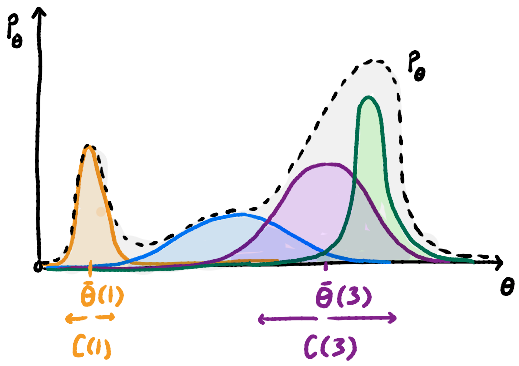
= quiz ← needs 20 min

Stat 238 - Lecture 24 - Density ≈ and Intro to Model Checking - 04/17/25

- **Role:** 1. fit prior model to observed $\{\theta_j\}_{j=1}^n$ via density estimation
  2. "fill in" posterior samples to build posterior density estimates



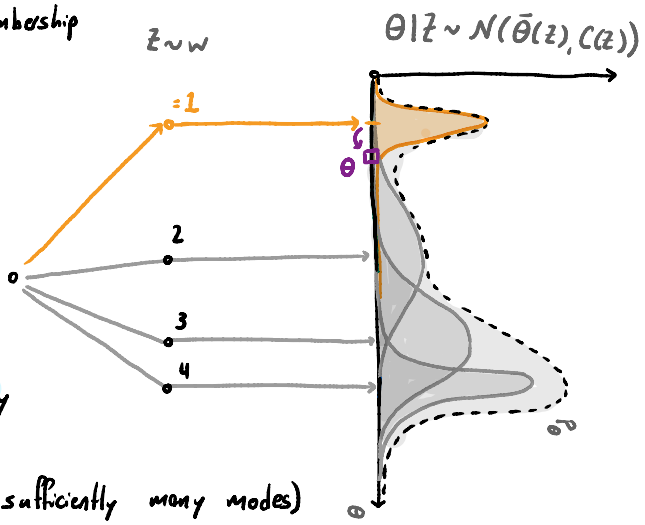
- **Gaussian Mixture Model (GMM):** a linear combination of finitely many normal dist's



- $p(\theta; w, \bar{\theta}, C) = \sum_{j=1}^{m} w_j \, N(\theta; \bar{\theta}(j), C(j))$
  - w: mixture weights, $w_j \geq 0$, $\sum_j w_j = 1$
- relation to KDE:
  (i) $m < n$ (fewer components, group samples into classes)
  (ii) non-uniform mixture weights
  (iii) Covariances $C(j)$ can vary by class

latent variable = class membership

- Sampling model:
  1. Draw class membership $z \sim w$
  2. Draw $\theta | z \sim N(\bar{\theta}(z), C(z))$



- **Expressivity:** GMM's are "universal" density estimators...
  - ∀ density ∃ a GMM (w/ sufficiently many modes) s.t. the GMM is arbitrarily close to the target w.r.t. the weak topology ($\mathbb{E}_p[f] \approx \mathbb{E}_{GMM}[f]$ ∀ f)

- suggests using GMM's for density estimation...
  - given $\{\theta_j\}_{j=1}^n$ find $(m, w, \vec{\theta}, C)$ that maximize the likelihood of sampling Σ from the GMM Σ ... use MLE on $w, \vec{\theta}, C \mid m$
  - Likelihood:
    - "total" $Lik(z, w, \vec{\theta}, C) = \prod_{j=1}^n N(\theta_j; \vec{\theta}(z_j), C(z_j))$
    - "marginal" $Lik(w, \vec{\theta}, C) = \sum_{\text{all } z} \prod_{j=1}^n w(z_j)\, N(\theta_j; \vec{\theta}(z_j), C(z_j))$ ↙ marginalize over latent var.'s
    - so... $\log(Lik(w, \vec{\theta}, C)) = \log\left(\sum_{\text{all } z} \ldots\right)$
      ↑ $\log(\Sigma \ldots)$ intractable
  - this is a marginal MLE problem, marginal over the class assignments
    - marginal MLE ~ Marginal MAP w/ uniform prior...
    - fit using Expectation Maximization.
- GMM density estimation via EM:
  - E: compute marginal objective on $w, \vec{\theta}, C$ marginalizing over latent class assignment z
  - M: set $w, \vec{\theta}, C$ to maximize the marginal objective.

$$p(w, \vec{\theta}, C \mid z, \{\theta_j\}_{j=1}^n) \propto \overbrace{p(w, \vec{\theta}, C \mid z)}^{\text{Uni. for MLE}}\, p(\{\theta_j\}_{j=1}^n \mid w, \vec{\theta}, C, z)$$

- E-step:

$$Q(w, \vec{\theta}, C \mid w_t, \vec{\theta}_t, C_t) = \mathbb{E}_{z \mid w_t, \vec{\theta}_t, C_t, \{\theta_j\}_{j=1}^n}\left[\log\left(\overbrace{Lik(w, \vec{\theta}, C, z)}\right)\right]$$

$$= \mathbb{E}_{z \mid \ldots}\left[\log\left(\prod_{j=1}^n w(z_j)\, N(\theta_j; \vec{\theta}(z_j), C(z_j))\right)\right]$$

$$= \sum_{j=1}^n \underbrace{\mathbb{E}_{z \mid \ldots}\left[\log(w(z_j))\right]}_{\text{controls fit of weights}} + \underbrace{\mathbb{E}_{z \mid \ldots}\left[-\tfrac{1}{2}\|\theta_j - \vec{\theta}(z_j)\|^2_{C(z_j)^{-1}}\right] - \mathbb{E}_{z \mid \ldots}\left[\tfrac{1}{2}\log(\det(C(z_j)))\right]}_{\text{controls fit of means } \vec{\theta} \text{ \& covariances } C}$$

2

- need: $z \mid w, \vec{\theta}, C, \{\theta_j\}_{j=1}^n$ ... notice, $z_i \perp\!\!\!\perp z_j$ if $i \neq j$, can marginalize $\perp\!\!\!\perp$'ly

$$Pr(z_j = z \mid w, \vec{\theta}, C, \theta_j) \propto Pr(z_j = z \mid w) \times Pr(\theta = \theta_j \mid z_j = z, w, \vec{\theta}, C)$$

$$= w(z)\ \mathcal{N}(\theta_j ; \vec{\theta}(z), C(z))$$

normalizing: (cond. prob. j^{th} particle of class z)

$$Pr(z_j = z \mid w, \vec{\theta}, C, \theta_j) = \frac{w(z)\ \mathcal{N}(\theta_j ; \vec{\theta}(z), C(z))}{\sum_{z'} w(z')\ \mathcal{N}(\theta_j ; \vec{\theta}(z'), C(z'))} \equiv \rho_j(z)$$

now:

$$Q(w, \vec{\theta}, C) = \sum_{z,j} \rho_j(z) \left[\log(w(z)) - \frac{1}{2} \|\theta_j - \vec{\theta}(z)\|^2_{C(z)^{-1}} - \frac{1}{2} \log(\det(C(z))) \right]$$

- **M-step:** notice $Q(w, \vec{\theta}, C) = Q_w(w) + Q_{\vec{\theta}, C}(\vec{\theta}, C)$
  optimize separately...

  - update mixture weights: for $\sum_z w(z) = 1, w(z) \geq 0$ ... use Lagrange, check all non-neg. soln.

$$\nabla_w \left[Q_w(w) - \lambda \sum_z w(z) \right] \Big|_{w = w_t} = 0$$

$$\partial_{w_{z'}} [\ \ldots\ ] = \sum_j \rho_j(z') \partial_{w_{z'}} \log(w(z')) - \lambda \Big|_{w = w_t} = 0$$

$$\left[\sum_j \rho_j(z') \right] 1/w_t(z') = \lambda$$

$$w_t(z') = \lambda^{-1} \left[\sum_j \rho_j(z') \right] \propto \left[\sum_j \rho_j(z') \right]$$

(λ^{-1}: normalization)

so: ($Pr(z_j = z \mid w_t, \vec{\theta}_t, C_t, \theta_j)$)

$$\left[w_*(z) = \frac{\sum_{j=1}^n \hat{\rho}_j(z)}{\sum_{z'=1}^m \sum_{j=1}^n \hat{\rho}_j(z')} = Pr\left(z_J = z \mid w_t, \vec{\theta}_t, C_t, \{\theta_j\}_{j=1}^n \right) \right]$$

[if $J \sim Unif(1, \ldots n)$]

weight of class z = conditional prob. a randomly chosen observation belonged to class z

- update mixture means:

$$\nabla_{\bar\theta} Q_{\bar\theta, C}(\bar\theta, C \mid w_t, \bar\theta_t, C_t) = -\nabla_{\bar\theta} \sum_{z,j} p_j(z)\ \frac{1}{2}\ \|\bar\theta(z) - \theta_j\|^2_{C(z)^{-1}}$$

$$\partial_{\bar\theta(z')} \ldots = -\partial_{\bar\theta(z')} \sum_j p_j(z')\ \frac{1}{2}\ \|\bar\theta(z') - \theta_j\|^2_{C(z')^{-1}}$$

$$= -\sum_j p_j(z')\ C(z')^{-1}(\bar\theta(z') - \theta_j)$$

$$= -C(z')^{-1}\left(\sum_j p_j(z')\ (\bar\theta(z') - \theta_j)\right) = 0 \quad \text{at } \bar\theta_*$$

requires: $\left(\sum_j p_j(z')\right)\bar\theta_*(z') = \sum_j p_j(z')\,\theta_j$

so: $$\boxed{\bar\theta_*(z) = \frac{\sum_j \theta_j\, p_j(z)}{\sum_j p_j(z)} = \mathbb{E}_{J|z}[\theta_J]}$$

↑ conditional distr. of sample index given in class z, mixture $w_t, \bar\theta_t, C_t$

- update covariences: follow similar procedure... set $\bar\theta = \bar\theta_*$, look for unconstrained soln first

$$\partial_{\substack{C(z') \\ K\ell}} \left[Q_{\bar\theta, C}(\bar\theta_*, C \mid w_t, \bar\theta_t, C_t)\right] =$$

$$\partial_{\substack{C(z') \\ K,\ell}} \sum_j p_j(z)\left[-\frac{1}{2}\|\theta_j - \bar\theta_*(z)\|^2_{C^{-1}} + \frac{1}{2}\log\left(\det\left(C^{-1}(z')\right)\right)\right]_{C=C_*} = 0$$

$$\partial_{C_{K,\ell}(z')} \sum_j p_j(z')\left[-\sum_{h,i}(\theta_j - \bar\theta_*(z))_h \left[C_*^{-1}\right]_{h,i}(\theta_j - \bar\theta_*(z))_i + \ldots\right] = 0$$

$$\sum_j p_j(z')\,(\theta_j - \bar\theta_*(z))_K (\theta_j - \bar\theta_*(z))_\ell = \partial_{C_{K,\ell}(z')} \log\left(\det\left(C^{-1}(z')\right)\right) \sum_j p_j(z')$$

$$= C_{K,\ell}(z')\left[\sum_j p_j(z')\right]$$

so: $$\boxed{C_*(z) = \frac{\sum_j p_j(z)\,(\theta_j - \bar\theta_*(z))(\theta_j - \bar\theta_*(z))^T}{\sum_j p_j(z)} = \mathbb{E}_{J|z}\left[(\theta_J - \bar\theta_*(z))(\theta_J - \bar\theta_*(z))^T\right]}$$

4

- **Exp. Max. for GMM:**

given $\{\theta_j\}_{j=1}^n$, m modes ← try a couple, $m=1, 2, 4, \ldots$

recurse, from many initial conditions, * EM may converge to local minima

(i) $p_j(z) = \dfrac{w_t(z)\, \mathcal{N}(\theta_j ; \bar{\theta}_t(z), C_t(z))}{\sum_{z'} w_t(z)\, \mathcal{N}(\theta_j ; \bar{\theta}_t(z), C_t(z))}$, $\hat{p}_j(z) = \dfrac{p_j(z)}{\sum_j p_j(z)}$ } E

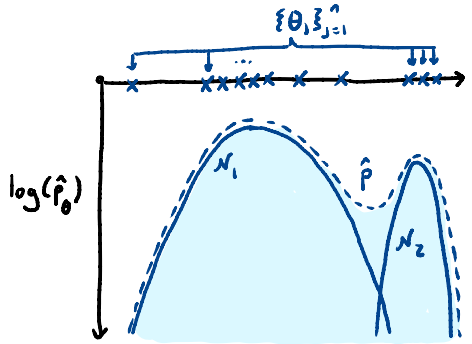
(ii) $w_{t+1}(z) = \dfrac{\sum_j p_j(z)}{\sum_{z'}\sum_j p_j(z')}$ ← update mixture weights.

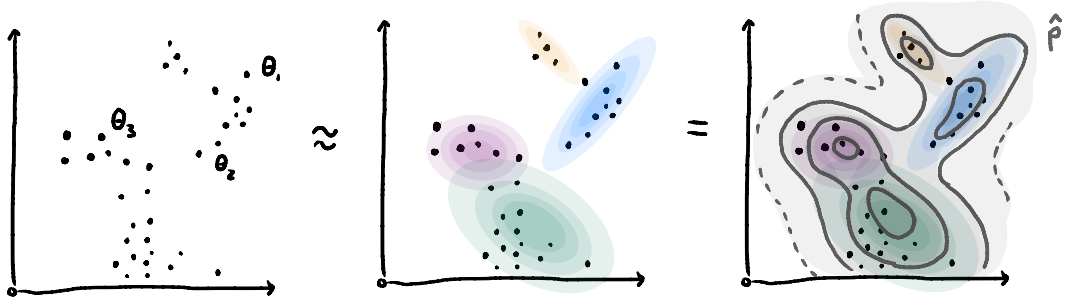
(iii) $\bar{\theta}_{t+1}(z) = \sum_j \hat{p}_j(z)\, \theta_j$ ← update means.

(iv) $C_{t+1}(z) = \sum_j \hat{p}_j(z)\, (\theta_j - \bar{\theta}_{t+1}(z))(\theta_j - \bar{\theta}_{t+1}(z))^T$ ← update Cov's.

} M

- **for prior modeling:**
  - interpretable
  - regularize towards past data
  - easy to sample from
  - smoother priors than KDE or KNN
- bias-variance tradeoff in m
- faster decaying tails than KNN (stronger regularization)
  (can weaken to Student-t mixture w/ large ν)

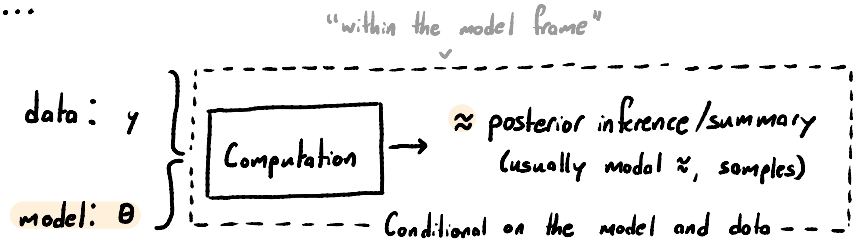




5

• Model & Procedure Checking:

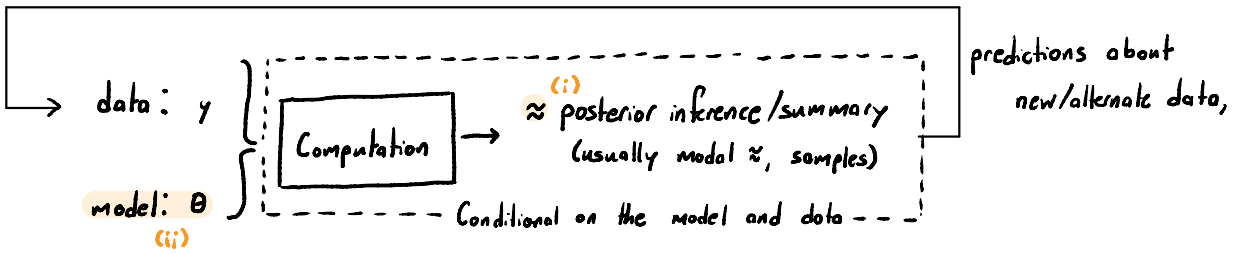
stepping back...



Key questions:

(i) if we used ≈ posterior inference (e.g. MCMC), how can we know if our posterior summaries are exact? (e.g. true to the model/posterior)
... is the procedure consistent w/in the model frame?

(ii) can we validate/check if the model is plausible? } next week/start reading
... "outside" the frame



(i) Coverage Checks to Validate Computational Procedure:

- $\theta \sim p_\theta$ unknown, observe $Y = y$
- let S be any quantity we can compute given θ ... $S = s(\theta)$
- Posterior inferences on S are produced by:

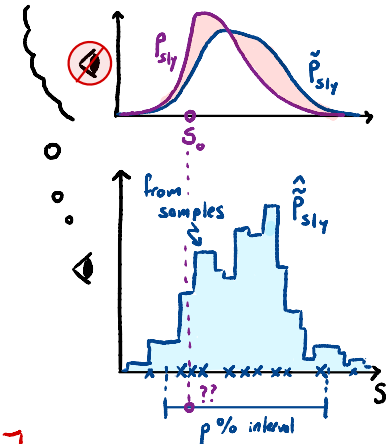
(i) generate $\{\theta_j\}_{j=1}^n \sim \tilde{p}_{\theta|y} \approx p_{\theta|y}$ ⟶ (ii) generate $\{S_j = s(\theta_j)\}_{j=1}^n$

then $S_{j,k} \sim \tilde{p}_{s|y} \approx p_{s|y}$

- Summary quantity, usually use a couple summaries.
- will play the role of a test statistic.

- θ_i for some i's
- $m \cdot \theta$ or $M^T\theta$ (projections)
- $\max(\theta)$, $\min(\theta)$, order statistics
- $\log(p_\theta(\theta))$, $\log(\text{Lik}(\theta))$, $\log(p(\theta, y))$

- to test if $\tilde{p}_{\theta|y} \approx p_{\theta|y}$, test if $\tilde{p}_{s|y} \approx p_{s|y}$.



E.g. p% CI's using $\tilde{p}_{s|y}$ (≈ from samples) should contain about p% of posterior prob mass under $p_{s|y}$

- Ex: does my 95% interval built from $\{s_j\}_{j=1}^n$ contain the true value $s_0 = s(\theta_0)$? (w/ prob. 95%)

- essentially impossible conditioned on a single y if we've exhausted convergence tests (run MCMC longer), importance corrections, method comparison (i.e. internal computational checks)
- would still check conditional on one data set y... no guarantee that the pipeline will be accurate for other data

and we don't know the true $\theta = \theta_0$ thus can't check against the true $S = s_0 = s(\theta_0)$

- want $\tilde{p}_{\theta|y}$ to be a consistently good approximator to $p_{\theta|y}$ for most y...
  - Ambitions: minimax/robust guarantees... $\| \tilde{p}_{\theta|y} - p_{\theta|y} \| < \epsilon$ $\forall$ y or $\forall$ y ∈ some region ← guarantee validity of pipeline even if misspecified
  - Achievable/Testable: $\| \tilde{p}_{s|Y} - p_{s|Y} \| < \epsilon$ (in a TV sense, i.e. $\tilde{p}_{s|y}$ assigns ≈ the same prob. to all sets as $p_{s|y}$) for almost all Y (e.g. w/ high probability) given $Y \sim$ the model.
    ↑ checks validity w/in model. can simulate draws of Y by drawing θ provides true θ to check against inference

- Observation: If the model is well-specified, and $Y \sim p_Y = \int_\theta p_{Y|\theta}\, p_\theta\, d\theta$ ("prior predictive dist"), sample by

(i) $\theta_0 \sim p_\theta \longrightarrow$ (ii) $Y|\theta_0 \sim p_{Y|\theta=\theta_0}$

then $\Pr(S_0 = s(\theta_0) \in p\%$ posterior credible interval$) = p$ ← true (prior) marginal p_s should = ≈ marginal $\tilde{p}_s$ after marginalizing over possible data... $\int_Y \tilde{p}_{s|Y}\, p_Y\, dy = \int_Y p_{s|Y}\, p_Y\, dy$

- and, if we draw n i.i.d. repititions, then the total # of trials where $S_0 \in$ p% interval is Binomial(n, p)

7

- Provides a hypothesis we can test...

Coverage Test

(i) sample truth: $\{\theta_{0,j}\}_{j=1}^{n} \overset{\text{i.i.d.}}{\sim} p_\theta \longrightarrow \{s_{0,j} = s(\theta_{0,j})\}_{j=1}^{n}$ } generate data

(ii) sample data | truth: $\{Y_j \mid \theta_{0,j}\} \sim p_{Y|\theta=\theta_{0,j}}\}_{j=1}^{n}$

(iii) for $j = 1, \dots, n$

perform inference:

(a) perform $\approx$ post. inference given $y = Y_j \longrightarrow \{\theta_{j,k}\}_{k=1}^{m} \sim \tilde{p}_{\theta|Y_j} \longrightarrow \{s_{j,k}\}_{k=1}^{m} \sim \tilde{p}_{s|Y_j}$

(b) build $\approx p\%$ interval $\hat{I}_p(\{s_{j,k}\}_{k=1}^{m})$ ← want to contain truth $p\%$ often.

(c) check "coverage" } did my interval contain the truth?

$$s_{0,j} \in \hat{I}_p(\{s_{j,k}\}_{k=1}^{m})$$

(iv) let $\hat{\tilde{p}} = \frac{1}{n}\sum_{j=1}^{n} \mathbb{1}(s_{0,j} \in \hat{I}_p(\{s_{j,k}\}_{k=1}^{m}))$ ← observed coverage proportion

(v) then, if $\tilde{p}_{\theta|Y} = p_{\theta|Y}$ (posterior inference procedure is exact)

$$H_0: \hat{\tilde{p}} = \frac{X}{n}, \quad X \sim \text{Binomial}(n, p)$$

hypothetical coverage proportion; known design parameters; observed coverage proportion

so, run a hypothesis test on $|\hat{\tilde{p}} - p|$

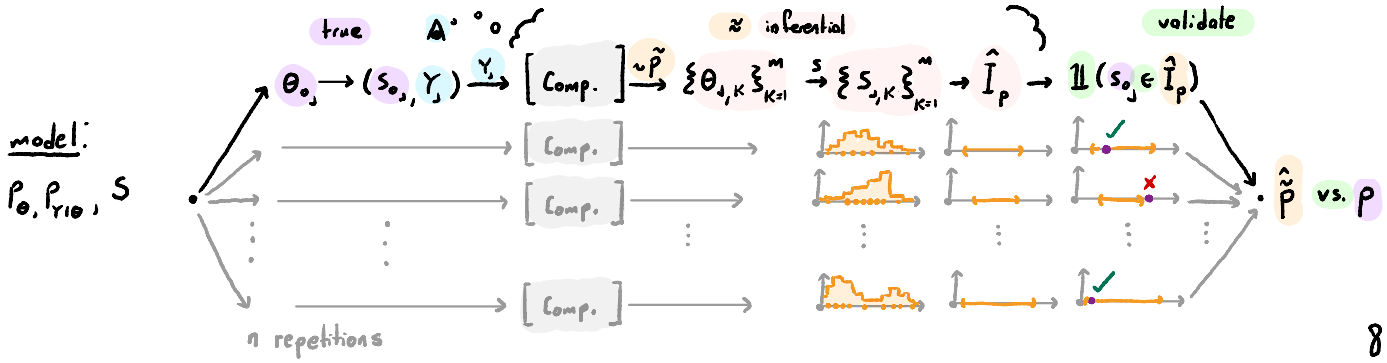
reject if

$$\Pr\left(\left|\frac{X}{n} - p\right| \geq |\hat{\tilde{p}} - p| \text{ if } X \sim \text{Bin}(n, p)\right) \leq \alpha$$

usually $\alpha = 0.05$

} check/validate

- repeat for a series of:
  - (i) p's: [20%, 40%, 60%, 75%, 90%, 95%, 99%]
  - (ii) s's
- report: (i) $\tilde{p}$'s for each p & s } degree of mismatch
  - (ii) associated p-values
  - (iii) pass rate across tests (highlight tests that fail) } stat significance of mismatch



8

- **Example:** reporting coverage rates for a high-D inference problem ($\theta \in \mathbb{R}^d$, $d \approx 10^2$)

$S =$ arbitrary marginals θ_{20} θ_{40} θ_{60} θ_{80} ...

	20	40	60	80	100	120
95%	0.945	0.958	0.942	0.940	0.937	0.953
90%	0.911	0.903	0.903	0.911	0.903	0.911
80%	0.816	0.822	0.798	0.827	0.814	0.837
60%	0.604	0.617	0.614	0.614	0.643	0.638
40%	0.438	0.423	0.430	0.415	0.433	0.420
20%	0.231	0.218	0.210	0.207	0.244	0.215

p ↓

order stat's

$S = \theta_{10\%}$
$S = \theta_{90\%}$

p →

	95%	90%	80%	60%	40%	20%
10%	0.938	0.865	0.833	0.604	0.365	0.198
90%	0.927	0.906	0.740	0.594	0.375	0.156

$S = \log(p_{\theta,Y})$

p →

	95%	90%	80%	60%	40%	20%
coverage	0.938	0.906	0.812	0.688	0.458	0.198

Using $n = 10^3$ rep.
Expect (under H_0)
$SD[\hat{p}] = n^{-1/2}\sqrt{p(1-p)}$
$\leq n^{-1/2}\sqrt{(1/4)}$
$\leq \frac{1}{2} n^{-1/2}$
≈ 0.015
$Bin(1{,}000, p) \approx N$
95% int $\approx \pm 2$ SD
$\leq \pm 0.03$

= true = estimated/≈ (observed coverage)

= significant (statistically) discrepancy

(≈ 12/54 above, $\tilde{p} \neq p$ (we knew this anyways)
but, no systematic errors,
largest discr. in coverage is 9% over
54 summaries and p's, centered ≈ 3%
error in coverage... probably insignificant
given posterior variance in θ
... sufficiently accurate, it would (did)
take many repeated applications to show $\tilde{p} \neq p$.)

Stat 238 - Lecture 25 - Posterior Summary & Validation - 04/22/25

Outline:

1. Coverage Checking
2. Posterior Predictive Checks
3. Cross-Validation

---

1. Coverage Checking: How can we check whether our computational pipeline returns ~~a~~ accurate posterior approximations?

- Key question: if θ_0 = true unknown, $s_0 = s(\theta_0)$, generate ~~##~~ a $\approx 90\%$ (p%) posterior credible interval, what is the probability s_0 is contained in the interval?
- if $\theta_0 \sim p_\theta$ then it should be p. If repeated n times, then the observed # of times $\sim Bin(n, p)$
- Hypothesis Test: draw $\theta_0 \sim p_\theta$ n times, $\{\theta_{0,j}\}_{j=1}^n \to \{s_{0,j} = s(\theta_{0,j})\}_{j=1}^n$ $\to \{\gamma_j \sim p_{\gamma|\theta_{0,j}}\}_{j=1}^n$

  for each, $s_{0,j}$ check: is $s_{0,j} \in \hat{I}_p$

  then $\sum_{j=1}^n \mathbb{1}(s_{0,j} \in \hat{I}_p) \sim Bin(n, p)$ if $\tilde{p}_{\theta|\gamma} = p_{\theta|\gamma}$.

use slide for this?

- Coverage Check:
  1. fix model: p_θ, $p_{\gamma|\theta}$
  2. fix summary: s, coverage %: p
  3. draw $\{\theta_{0,j}\}_{j=1}^n \overset{iid}{\sim} p_\theta \xrightarrow{s} \{s_{0,j} = s(\theta_{0,j})\}_{j=1}^n$
  4. draw $\{\gamma_j \sim p_{\gamma|\theta_{0,j}}\}_{j=1}^n$
  5. run comp. pipeline $\Rightarrow$ return $\approx p$% posterior interval for s (e.g. return $\{\theta_{j,k}\}_{k=1}^m \sim \tilde{p}_{\theta|\gamma_j} \to \{s_{j,k}\}_{k=1}^m$)
  6. let $\hat{\tilde{p}} = \frac{1}{n}\sum_{j=1}^n \mathbb{1}(s_{0,j} \in \hat{I}_p(\{s_{j,k}\}_{k=1}^m))$
  7. compare $\hat{\tilde{p}}$ to p

- show example.

Posterior Predictive Check:

- **Key Q:** does my model $(p_\theta, p_{Y|\theta} \Leftrightarrow p_{\theta,Y})$ appear plausible given the available data?
- idea, consider predictive distributions...

- **Prior Predictive Dist:** $p_Y(y) = \int_\theta p_{Y,\theta}(y,\theta)\, d\theta = \int_\theta p_\theta(\theta)\, p_{Y|\theta}(y|\theta)\, d\theta$

  - **Evidence:** if we observed $Y = y_{obs}$

    evidence $= p_Y(y_{obs})$ = prior predic. prob. of the data we did observe.

    Maximize for MLE of prior parameters

- **Posterior Predictive Dist:** $p_Y(\underbrace{y^{rep}}_{\text{a "replicate" dataset}} | Y = y^{obs}) = \int_\theta p_{Y|\theta}(y^{rep}|\theta)\, p(\theta|y^{obs})\, d\theta$

  - **Concept Q:** What is the significance of $p_Y(Y^{rep} = y^{obs} | Y = y^{obs})$?

    vs. $p_Y(Y^{rep} = y^{rep} | Y = y^{obs})$

- If we withold a dataset (test set) we can use it for external validation.
  - e.g. check if predictions using the PPC hold

- **Post. Pred. Check:** "If the model fits, then replicated data generated under the model should look similar to observed data"

  ... "observed data should not be abnormal under the model."

  - **Test Quantity/Summary:** $s: \mathcal{Y} \to \mathbb{R}$, $s(y)$ = a summary of some aspect of y
    - "Test Stat": $s(y)$
    - "discrepancy measure": $s(y,\theta)$

  - **Post. Pred. p-value:** $\Pr(s(y^{rep},\theta) \geq s(Y,\theta) | y^{obs}) = \iint_{Y,\theta} \mathbb{1}(s(y^{rep},\theta) \geq s(y,\theta))\, \overbrace{p(\theta|y)\, p(y^{rep}|\theta)}\, p(y^{rep},\theta|y^{obs})$

- Compute prior pred. p-values (up to $\approx$) via simulation:

$$y^{obs} \longrightarrow \{\theta_j\}_{j=1}^{n} \sim \tilde{p}(\theta|y^{obs}) \xrightarrow{\text{optional here: may condition on other aspects of the exp model}} \{y^{rep}_{j,k} \sim p(y|\theta_j)\}_{k=1}^{m} \longrightarrow \{s(y^{rep}_{j,k}, \theta_j)\}_{j,k=1}^{n,m}$$

then $$p\text{-value} \approx \frac{1}{n \cdot m} \sum_{j,k=1}^{n,m} \mathbb{1}\left(s(y^{rep}_{j,k}, \theta_j) \geq s(y^{obs}, \theta_j)\right).$$

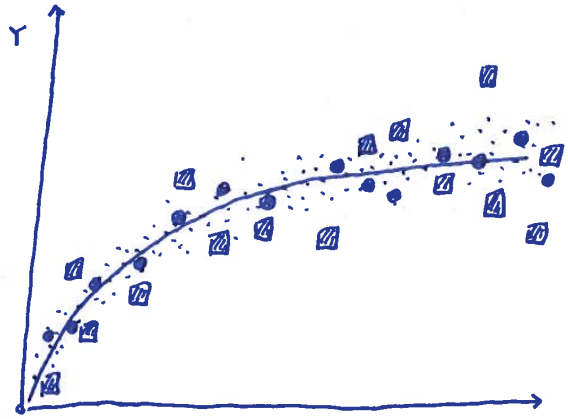
- Set a threshold on the p-value for testing statistical significance.

- Advice For Checking: ~~to perform~~

1. Choosing a summary quantity/test stat/discrepancy measure: "sensitive"
   (i) should be interpretable
   (ii) of scientific/practical relevance
   (iii) chosen to detect model failures (e.g. aspects of the data we didn't build explicitly into the model)
   and it can help to start by graphing (use visuals to identify discrepancies of possible import)
2. Think beyond the likelihood... w/in frame the only way the data ~~influences~~ the relates to the unknown is via the likelihood. We're checking the frame itself (the model used to compute the likelihood). (best measure of predictive accuracy is log likelihood)
   - W/in frame: Likelihood Principle: inferences should depend on only the likelihood (and prior if Bayesian)
   - W/out frame: more to evaluate than the likelihood, can check other aspects of the data's fit to the model.
3. Consider Statistical & Practical Significance: What does an extreme p-value mean??
   - Stat significance: is the discrepancy implausibly large under the proposed model? (i.e. could have arisen due to chance)
   - Practical significance: do errors induced by these discrepancies matter?
     - e.g. are they much smaller than other sources of variance?
       are they sufficient for intended use?
       do they address important aspects of the data?
     - usually the relevant question is not "is my model true?" but "is it close/good enough?"

• reject or refine models w/ discrepancies of statistical and practical importance.

• Ex:



▨ = y_{obs} ⌒ = trend line θ
⁘ = replicate data dist | y_{obs}
● = rep data set

S = discrepancy between Y & trend line

move to start of PPC section)

3. Cross-Validation: idea, want to estimate how well a model will perform on new data

• given $y_{obs} \longrightarrow p_{\theta|y_{obs}} \longrightarrow p_{Y|y_{obs}}$

want $\mathbb{E}_{Y\sim f}[\log(p_{Y|y_{obs}}(Y))] \approx \mathbb{E}_{Y\sim f}[\log(\frac{1}{m}\sum_{k=1}^{m} p(Y|\theta_k))]$

f: true dist for Y

how to $\approx$ this?

$\{\theta_k\}_{k=1}^m \overset{iid}{\sim} \tilde{p}_{\theta|y_{obs}} \approx p_{\theta|y_{obs}}$

• idea: cross-validate... split $y_{obs} = \{[y_{train_j}, y_{test_j}]\}_{j=1}^{K}$ $y_{obs} = \{y_{obs}^{(-j)}\}_{j=1}^{K}$ ← "K-fold"

• e.g.: Leave-One Out: $\{y_{obs}^{(-j)}\}_{j=1}^{K}$, K = # data points, $y_{obs}^{(-j)} = y_{obs_{-j}}$

then use: $\sum_{j=1}^{n} \log(\frac{1}{m}\sum_{k=1}^{m} p(y_{obs_j}|\theta_k))$

$\{\theta_k\}_{k=1}^m \overset{iid}{\sim} \tilde{p}_{\theta|y_{obs_{-j}}} \approx p_{\theta|y_{obs_{-j}}}$

Stat 238 - Lecture 26 - 04/24/25

- **Cross-Validation:**
  - **Key question:** how well would my model perform on *new* data?
    - given $Y \sim f$, observe $Y_{obs} = y_{obs}$ → $p_{\theta | Y_{obs} = y_{obs}}$ (posterior inference on θ) → $p_{Y | Y_{obs} = y_{obs}} = \int_\theta p_{Y|\theta} \, p_{\theta | Y_{rep}} \, d\theta$
      - [model: $(\theta, Y) \sim p_{\theta, Y}$ $\sim (p_\theta, p_{Y|\theta})$]
      - if well-specified $= f$ for some θ_0
    - then imagine $Y_{rep} \sim f$.
    - ask, would $p_{Y|Y_{obs}}$ usually assign high prob. to Y_{rep}?
  - **ideal:** measure fit of a model for f, g via:

$$\underbrace{D_{KL}(f \,\|\, g)}_{\text{ability to distinguish that } g \neq f \text{ using samples from } f} = \mathbb{E}_{Y \sim f}\left[\log(f(Y)/g(Y))\right] = \underbrace{\mathbb{E}_{Y \sim f}\left[\log(f(Y))\right]}_{-H(f)} - \underbrace{\mathbb{E}_{Y \sim f}\left[\log(g(Y))\right]}_{\text{cross-entropy } H(f,g)}$$

$-H(f)$: constant in g

aim is to choose g to minimize

$D_{KL}(f \| g)$ ⇒ minimize cross-entropy loss $H(f, g)$

⇒ maximize $\mathbb{E}_{Y \sim f}\left[\log(g(Y))\right]$ $\overset{N \to \infty}{\approx} \frac{1}{N} \sum_{i=1}^{N} \log(g(Y_i^{rep}))$ (provides a standard for comparing models g)

if $Y_i^{rep} \perp\!\!\!\perp Y_j^{rep}$

$= \log\left(\left(\prod_{i=1}^{N} g(Y_i^{rep})\right)\right)$

$Lik(g; \{Y_i^{rep}\}_{i=1}^N)$

⇒ maximize $Lik(g; \{Y_i^{rep}\}_{i=1}^N)$ in limit as $N \to \infty$

so, try to choose the model $(p_{\theta,Y} \sim p_\theta \cdot p_{Y|\theta})$ that maximizes:

$$\mathbb{E}_{Y^{rep} \sim f}\left[\log\left(p_{Y|Y_{obs}}(Y^{rep})\right)\right] = \mathbb{E}_{Y^{rep} \sim f}\left[\log\left(\int_\theta p_{Y|\theta}(Y^{rep}|\theta) \, p_{\theta|Y}(\theta|Y=Y_{obs}) \, d\theta\right)\right]$$

$$\approx \mathbb{E}_{Y^{rep} \sim f}\left[\log\left(\frac{1}{m}\sum_{i=1}^{m} p_{Y|\theta}(Y^{rep}|\theta_i)\right)\right] \quad \theta_i \overset{iid}{\sim} p_{\theta|Y=Y_{obs}}$$

how to make this $\approx$?

given one dataset Y, split it into $[Y_{train}, Y_{test}]$

"obs" "replicate"

- "Leave-One Out": $Y_{test} = Y_i$, $Y_{train} = Y_{-i}$ $\Rightarrow$ very crude:

$$\approx \log\left(\frac{1}{m}\sum_{j=1}^{m} p_{Y|\theta}(Y_i|\theta_j)\right) \quad \theta_j \overset{iid}{\sim} p_{\theta|Y=Y_{-i}}$$

- average over all possible samples left out

$$\approx \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{1}{m}\sum_{j=1}^{m} p_{Y|\theta}(Y_i|\theta_j)\right) \quad \theta_j \overset{iid}{\sim} p_{\theta|Y=Y_{-i}}$$

Choose the model that maximizes this quantity (success of model to predict an excluded entry, trained on the rest, averaged over the entry left out)

- "K-fold": split into K subsets, repeat leaving out a subset, train on the rest, average over each possible subset to leave out.

- **Sensitivity Analysis:**
  - how sensitive are my inferential conclusions to:
    (i) plausible perturbations to my data?
    (ii) plausible changes in my assumptions?
    → for large perturbations, repeat the process many times for perturbed data (e.g. bootstrap) or assumptions
  - if the perturbations are small, can analyze directly via sensitivity analysis

- Ex: $\hat{\theta}_{MAP}(\underbrace{s(y)}_{\text{suff. stats}}; \underbrace{\alpha}_{\text{prior parameters}}) = \hat{\theta}_{MAP}(s; \alpha)$

  - if the posterior admits a unique, point, global maximizer for all s, α
    in an open neighborhood of s, α, $N(s, \alpha)$
    and is second differentiable (continuously) on a neighborhood of $\hat{\theta}_{MAP}(s', \alpha')$ $\forall\ s', \alpha' \in N(s, \alpha)$ in θ
    and is continuously differentiable in s, α then the implicit function thm applies, we can differentiate implicitly...

$$\nabla_\theta \rho_{\theta|s}(\hat{\theta}_{MAP}(\overbrace{s,\alpha}^{\phi})|\overbrace{s;\alpha}^{\phi}) = 0 \quad \forall\ s, \alpha$$

$$\Updownarrow$$

$$\nabla_\theta \log(\rho_{\theta|s}(\dots)) = 0 \quad \forall\ s, \alpha$$

$$\Updownarrow \quad \partial_{\theta_j}$$

$$\partial_{\phi_i} \underbrace{\log(\rho_{\theta|s}(\hat{\theta}_{MAP}(s,\alpha)|s;\alpha))}_{\text{comp. of functions}} = \sum_k \underbrace{\left[\partial^2_{\theta_j,\theta_k} \log(\rho_{\theta|s}(\theta|s;\alpha))\Big|_{\theta=\hat{\theta}_{MAP}(s;\alpha)}\right]}_{\sum_K H[\hat{\theta}_{MAP}(\phi)]_{JK}} \partial_{\phi_i} \hat{\theta}_{MAP_k}(\phi)$$

$$+ \left[\partial^2_{\phi_i,\theta_j} \log(\rho_{\theta|s}(\theta|\phi))\Big|_{\theta=\hat{\theta}_{MAP}(\phi)}\right] = 0$$

(grad. / score function)

~~so, if very ϕ along direction~~ $\Rightarrow \sum_k \left(-H[\hat{\theta}_{MAP}(\phi)]\right)_{JK} \partial_{\phi_i} \hat{\theta}_{MAP_k}(\phi) = \partial^*_{\phi_i} g_j[\hat{\theta}_{MAP}(\phi), \phi]$

$\forall\ j$

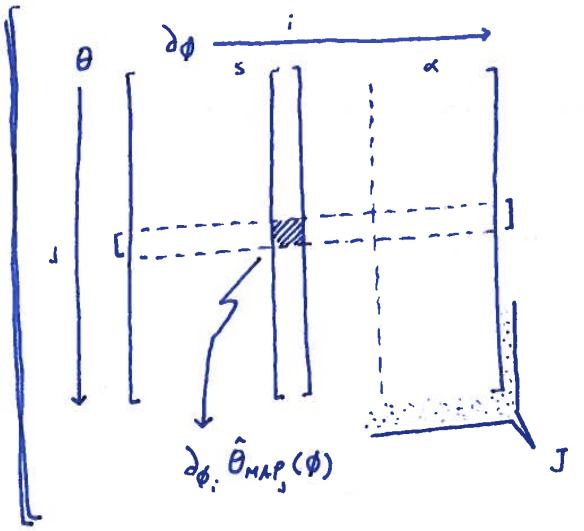
so:

$$-H[\hat{\theta}_{MAP}(\phi), \phi] \cdot \partial_{\phi_i} \hat{\theta}_{MAP}(\phi) = \partial_{\phi_i} g[\hat{\theta}_{MAP}(\phi), \phi]$$

$$\partial_{\phi_i} \hat{\theta}_{MAP}(\phi) = -H[\hat{\theta}_{MAP}(\phi), \phi]^{-1} \partial_{\phi_i} g[\hat{\theta}_{MAP}(\phi), \phi]$$

Jacobian

$$J_\alpha^\beta = D_{\phi_\alpha} \hat{\theta}_{MAP}(\phi)^\beta = -\left[D_\theta^2 \log(p_{\theta|Y}(\theta|\phi)) \right]^{-1}_{\theta=\hat{\theta}_{MAP}(\phi)} \; \beta \; \gamma \; D_{\phi_\alpha} D_{\theta_\gamma} \log(p_{\theta|Y}(\theta|\phi))_{\theta=\hat{\theta}_n}$$



$$= -\left[\underbrace{H[\hat{\theta}_{MAP}(\phi), \phi]^{-1}}_{\text{Hessian used in Laplace} \approx \text{this is the Cov in the Laplace} \approx \text{Cov in posterior}} \partial_{\phi_i} \underbrace{g[\hat{\theta}_{MAP}(\phi), \phi]}_{\text{"score function"}} \right]_j$$

~~if well specified, and a large data set, then $\hat{\theta}_{MAP} \approx \theta_*$~~

$$\Delta\hat{\theta} = J \cdot \Delta s \longrightarrow \mathbb{V}_s[\hat{\theta}] = J \, \mathbb{V}_s[s] \, J^T$$

$$= (-H^{-1}) \, J_{score} \, \mathbb{V}_s[s] \, J_{score}^T \, (-H^{-1})^T$$

$\mathbb{V}_s[score]$

~~if $\hat{\theta} = \theta_* \in$ true par. (assumes well specified)~~

$\approx I[\hat{\theta}] \approx H[\hat{\theta}]$

[• to first order analysis, $\mathbb{V}_s[\hat{\theta}(s,\alpha)] \approx \mathbb{V}_{\theta|s}[\theta]$. (valid in limit of large data set)] $\approx H^{-1} \approx$ Var in posterior

used to form conf. intervals; used to form cred. intervals

4

Stat 238 - Lecture 27 - Capstone Topics: Ensemble MCMC

1. Teleporting Walkers
2. Rare Event via Splitting Algorithms
   - define a rare event
   - variance of estimated prob. is high, slow to discover
   - committors/"reaction coordinate"
   - splitting
   - multilevel · sequential MC · adaptive multilevel

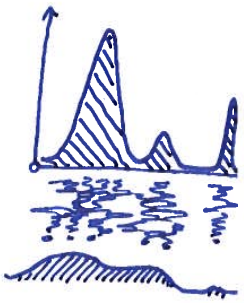
- **Ensemble MCMC:**
  - run multiple chains simultaneously, allowing interaction
  - **Ex:**

1. run N chains in parallel $X_0^i \to X_t^i$, $i=1,\dots N$ (choose any MH you like)
2. resample from a KDE on $\{X_t^i\}_{i=1}^N \to \{\text{~~}Y^{i,k}\}_{k=1,\,i=1}^{M,\,N}$, kernel K (M children per parent)
3. Compute importance weights $\hat{\tilde{w}}_{i,k}$
4. run a single EM type update to ~~run~~ update mixture weights assigned to the parent particles:

$$q(j) \propto \sum_{i,k} \hat{\tilde{w}}(Y^{i,k})\, K(Y^{i,k}, X_t^j)$$

5. resample initial conditions $\underline{\tilde{X}}_0^i \overset{iid}{\sim} q$ for $i=1:N$

$$X_0^i \sim K(\cdot, \underline{\tilde{X}}^i)$$



intuition: reseed KDE on MCMC to avoid large importance weights

- **Ex: Teleporting Walkers** as above, but replace KDE w/ the proposal/transition kernel, only evaluate at MCMC samples

- let p_* be the target, N = # of walkers

product/joint target

$$X^{(N)} = \{X_1, \dots X_N\} \quad \text{want} \quad X^{(N)} \sim p_*^{(N)} = \prod_{i=1}^{N} p_*$$

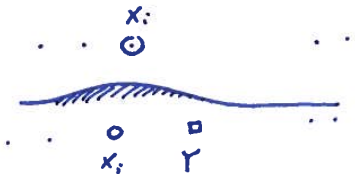
$(X_i \overset{iid}{\sim} p_*)$

idea, run MH on $X^{(N)}$ jointly.

Proposal: $p_r(x \to x')$ for an individual x

(i) choose $i \sim \text{Unif}[1, \dots N]$

(ii) draw candidate $Y \sim p_r(X_i \to \cdot)$



(iii) build KDE using p_r as the kernel

leaving out each possible j and including Y

compute importance weights

$$\tilde{w}(X_i) = \frac{p_*(X_j)}{p_r(Y \to X_j) + \sum_{i \neq j} p_r(X_i \to X_j)}$$

(iv) sample

$$J \sim \tilde{w}(X_i)^{-1} \Rightarrow p_*(J=j) = \frac{1}{Z(X^{(N)}, Y)} \frac{p_r(Y \to X_j) + \sum_{i \neq j} p_r(X_i \to X_j)}{p_*(X_j)}$$

$$Z(X^{(N)}, Y) = \sum_{j=1}^{N} \frac{p_r(Y \to X_j) + \sum_{i \neq j} p_r(X_i \to X_j)}{p_*(X_j)}$$

(v) set $Y_k^{(N)} = \begin{cases} X_i^{(N)} & \text{if } i \neq j \\ Y & \text{if } i = j \end{cases}$

(vi) accept $Y^{(N)}$ using MH on $p_*^{(N)}$ from $X^{(N)}$

(acceptance prob $= \min\left\{ \frac{Z(X^{(N)}, Y)}{Z(Y^{(N)}, X_j)} \frac{q}{q}, 1 \right\}$

instinct:

- nearby walkers are likely to be selected to jump
- if a walker has a low importance weight under the proposal then they are in an oversampled region, good candidate to move

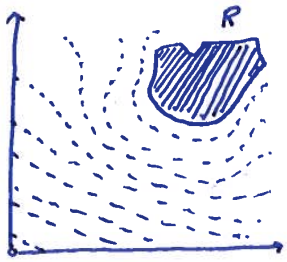
- **Rare Event Sampling:**
  - Let $E = \{X \in R\}$ when $X \sim p_*$
  - rare if $Pr(X \in R \mid X \sim p_*) = \rho$ is small
  - hard to estimate reliably...
    - for ergodic chains, expect to wait at least $\mathcal{O}(\rho^{-1})$ time before seeing $X_t \in R$
    - if $\hat{\rho} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(X_i \in R)$, $X_i \overset{iid}{\sim} p_*$

      then $\mathbb{E}\left[\frac{(\hat{\rho}-\rho)^2}{\rho^2}\right] = \mathbb{V}\left[\frac{\hat{\rho}}{\rho}\right] = \frac{1}{\rho}\mathbb{V}[\hat{\rho}] = \frac{1}{\rho}\frac{\rho(1-\rho)}{N} = \frac{1-\rho}{\rho N} = \mathcal{O}((\rho N)^{-1})$

      need $N = \mathcal{O}(\rho^{-1})$ for reliable estimation (ouch)

- **splitting idea:**
  - run MCMC but branch the simulation when it is "close" to R
  - keep track of probability of each branch by sampling uniformly at each branch point in a tree of simulations

- **ingredients:**
  - let $\phi(x): \mathcal{X} \to \mathbb{R}$ s.t. $\phi(x) \geq L$ iff $x \in R$ for some L



    - ideal: $\phi(x) = \underline{\text{committor}}(x) = Pr(X_\tau \in R \mid X_0 = x)$ for some MCMC scheme run for time τ

      usually unknown, choose heuristically (e.g. $-\min_{x' \in R} \{\|x - x'\|_2^2\}$)

    - idea: fix a sequence: $\{L_j\}_{j=1}^{j_{max}}$ s.t. $L_j < L_{j+1}$, $L_{j_{max}} = L$

      split into r_j copies each time we cross level L_j

      usually want $r_j \approx \left\lceil \frac{P(\max_{t \leq \tau} X_t \geq L_{j-1})}{P(\max_{t \leq \tau} X_t \geq L_j)} \right\rceil$

- **Ex:** Multi-Level Splitting

(0) initial dist η_0, ϕ, levels $\{L_j\}_{j=1}^{m_{max}}$, cloning rates/branching factors $\{r_j\}_{j=1}^{\cdots}$, initial sample size N_0, length τ

(i) draw $\{X_0^i\}_{i=1}^{N_0} \overset{iid}{\sim} \eta_0$

(ii) for $j=1$ to m_{max}

(a) for $i=1$ to N_{j-1}

run trajectory i until $\phi(X_s^i) > L_j$ or until stopping at τ (first reached)

end

discard trajectories that did not reach L_j

clone trajectories that remain r_j times each

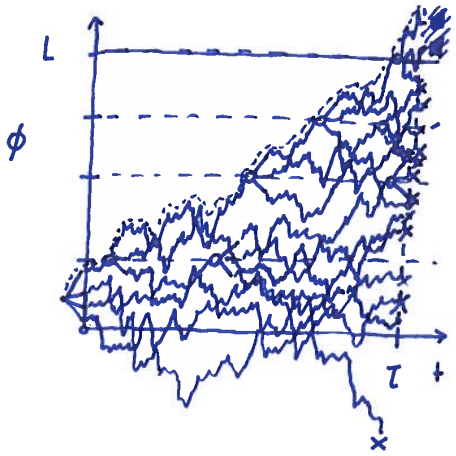
reorder from 1 to N_j

end

for $i=1$ to N_{max}

run trajectory i until time τ

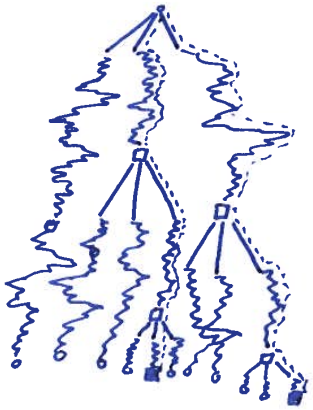
end mark all trajectories that end in R



Estimate $\hat{p}$ as fraction of trajectories ending in R

if we sample from the tree uniformly at each branch

$$\hat{p} = \frac{1}{N_0} \frac{1}{\prod_{j=1}^{m_{max}} r_j} \sum_{i=1}^{N_j} \mathbb{1}(X_\tau^i \in R).$$



- Ex: Adaptive Multi-Level Splitting

  - idea: 1. Keep the # of new walkers initialized from each level constant...

    $r_j = r \approx 1/q$ where q is the fraction that achieve the next level, if $r \notin \mathbb{Z}$, $\in [1, 2]$, reassign failed trajectories at rand.

    2. adaptive: choose desired ρ then pick L based on quantiles of $\max_{0 \le s \le \tau} \{\phi(X_s^i)\}$ so that ρ% cross level L between 0 and τ

  - ~~choose~~

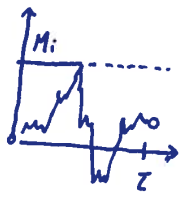
(0) as before except don't pick levels, instead choose N_0 and K (# to discard at each step, $K = (1-q)N$, $q = \frac{N-K}{N}$, $r = \frac{N}{N-K}$

(i) draw $\{X_0^i\}_{i=1}^{N_0} \overset{iid}{\sim} \eta_0$, $m = 0$

(ii) for $i = 1$ to N

(a) run $X_0^i \to X_\tau^i$

(b) set $M_i = \max_{s \in [0,\tau]} \{X_s^i\}$, S_j = crossing times



end

(iii) sort M's ~~high to low~~ low to high

(iv) set $L_m = M_{(K)}$ (k^{th} largest M)

while $L_m < L_{max}$

(a) $m = m + 1$

(b) discard all trajectories i s.t. $M_i \le L_m$

(c) if $M_i = L_m$ for multiple i, discard at random from i s.t. $M_i = L_m$ until discarded K

(d) let I_m = set of remaining trajectories

for $i \in \{1, \dots N\} / I_m$ (discarded trajectories)

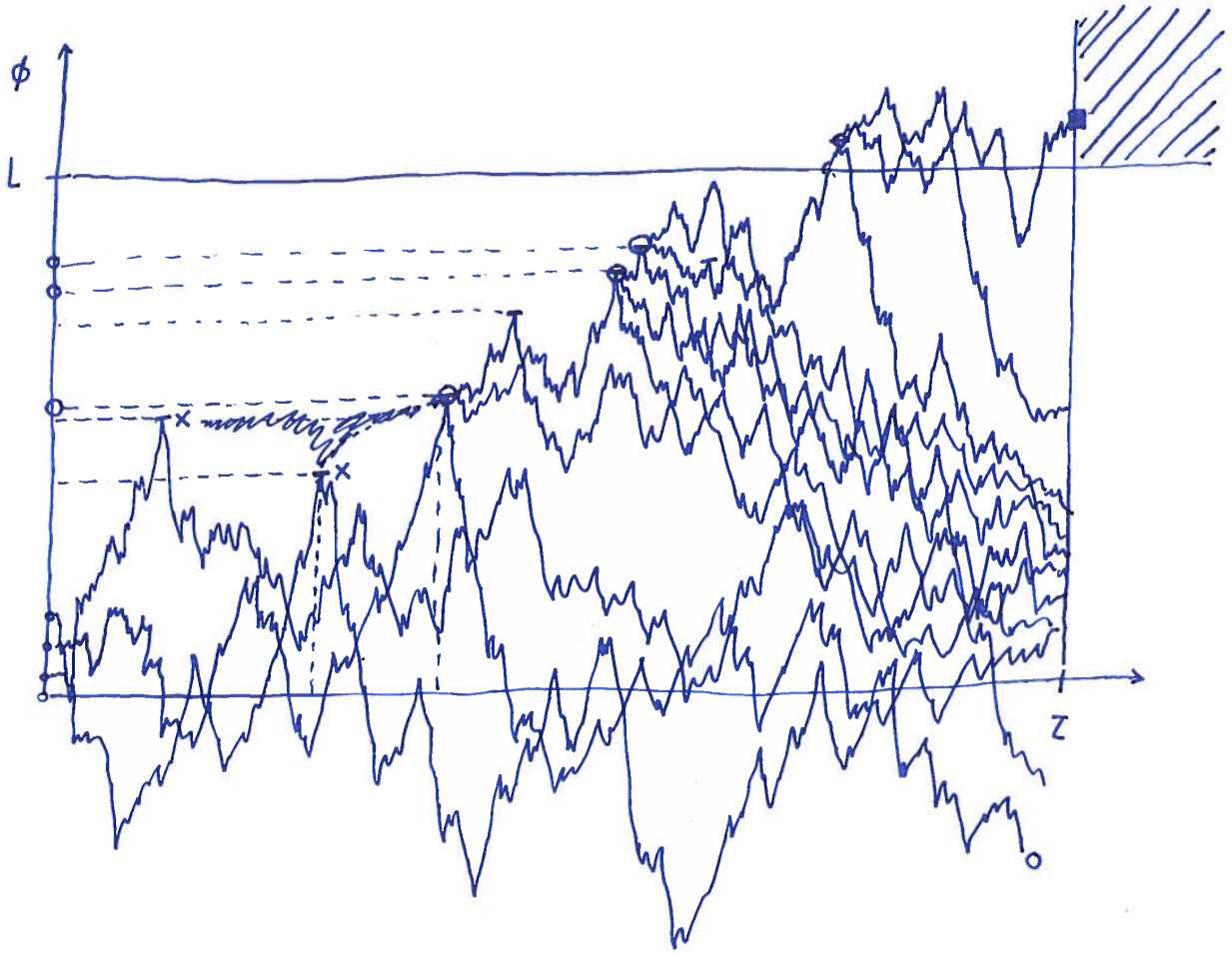
(a) then draw $J \sim I_m$ uniformly,

(b) run $\{X_{S_j}^i \to X_\tau^i\}$ from $X_{S_j}^i = X_{S_j}^J$

(c) set $M_m = \max \phi(X_s^i)$

→ (e) resort and update level repeat

now: $\hat{p} = \{\prod_{m=1}^{end} \frac{N-K}{N}\} \cdot \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(X_\tau^i \in R)$


ϕ
L
z

Stat 238 - VI - 05/01/25

1. Formulation:

- Prior: $\theta \sim p_\theta$ - Likelihood: $Y \sim p_{Y|\theta}$ - post: $p_{\theta|Y} \propto p_{\theta,Y}$ (target: $p_* \propto \tilde{p}_*$)

$$= \frac{p_{Y|\theta}\, p_\theta}{p_Y} \leftarrow \text{evidence}$$

aim: find $q \approx p_*$

- variational class: $q \in \mathcal{Q}$, tractable
- metric or divergence: $D[\cdot,\cdot] : \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$, $D[p,p] = 0 \ \forall p$

find: $q_* = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \{ D[q, p_*] \}$ (*)

- Q's: how to pick: $\mathcal{Q}$? D? algorithm to solve (*)?

2. Divergence:

- standard choice: use $D_{KL}(q \,\|\, p_*)$... why?
- Def: $D_{KL}(q \,\|\, p_*) = \mathbb{E}_{\theta \sim q}\left[\log\left(q(\theta)/p_*(\theta)\right) \right]$
- why?
  1. $D_{KL}(q \,\|\, p_*) \geq 0$, $= 0$ iff $q = p$ & is globally convex
  2. ~~changes in θ~~ we assumed $\mathcal{Q}$ is tractable so $\mathbb{E}_{\theta \sim q}$ is tractable...
     (notice: $\mathbb{E}_{\theta \sim p_*}$ is not, ideally would use $D_{KL}(p_* \,\|\, q)$ since $\min\{D_{KL}(p_* \,\|\, q)\} \Leftrightarrow$ MLE)
  3. changes in D_{KL} w.r.t. Δq can be calculated w/out knowing

$$D_{KL}(q \| p_*) - D_{KL}(q' \| p_*) = \mathbb{E}_{\theta \sim q}[\log(q(\theta)/p_*(\theta))] - \mathbb{E}_{\theta \sim q'}[\log(q'(\theta)/p_*(\theta))]$$

$$= \cancel{\mathbb{E}_{\theta \sim q}[\log(q(\theta))]} - \mathbb{E}_{\theta \sim q'}[\log(q'(\theta))] + \mathbb{E}_{\theta \sim q'}[\log(p_*(\theta))] - \cancel{\mathbb{E}_{\theta \sim q}[\log(p_*(\theta))]}$$

$$= (H(q') - H(q)) + \int_\theta (q'(\theta) - q(\theta))\left(\log(\tilde{p}_*(\theta)) + \log(Z)\right) d\theta$$

$$= (H(q') - H(q)) + \underbrace{\int_\theta (q'(\theta) - q(\theta)) \log(\tilde{p}_*(\theta)) d\theta}_{\text{II of } Z \ldots} - \cancel{\log(Z)} \underbrace{\int_\theta (q'(\theta) - q(\theta)) d\theta}_{1 - 1 = 0}$$

~~$\Delta_{q,q'} D_{KL}(\cdot \| p_*) = - \Delta_{q,q'} H(\cdot)$~~

sensible,

4. Simple, computable functional derivatives:

$$\nabla_q D_{KL}(q \| p_*) = \log(q(\cdot)/p_*(\cdot)) \overset{+c}{=} \log(q(\cdot)/\tilde{p}_*(\cdot))$$

so $$\partial_h D_{KL}(q \| p_*) = \lim_{\varepsilon \to 0} \frac{D_{KL}(q + \varepsilon h \| p_*) - D(q \| p_*)}{\varepsilon} = \langle \log(q(\cdot)/p_*(\cdot)), h(\cdot) \rangle$$

$$D_{KL}(q + \varepsilon h \| p_*) \simeq D_{KL}(q \| p_*) + \varepsilon \int_\theta h(\theta) \log\left(\frac{q(\theta)}{p_*(\theta)}\right) d\theta + \mathcal{O}(\varepsilon^2)$$

5. allows efficient algorithms & ~~intrac~~ analytic soln's to (*) for some $\mathcal{Q}$

6. Distinguishability interpretation via Chernoff-Stein

3. Choosing a variational family:

- trade-off: tractability vs. expressivity
- Standard Ex: assume $\theta_i \perp\!\!\!\perp \theta_j$ if $i \neq j$ so

$$q(\theta) = \prod_{i=1}^{m} q_i(\theta_i)$$ "mean-field family" → all θ's are $\perp\!\!\!\perp$ in the posterior !! ⚠ ... allows ...

- Coordinate Ascent Variational Inference:
  - let $\theta_{-j} = \{\theta_i\}_{i \neq j}$, $q_{-j} = \dots$
  - optimize w.r.t. q_j holding q_{-j} fixed

$$q_j^* \mid q_{-j} = \max_{q_j} \{ ELBO(q_j \prod_{i \neq j} q_i \,\|\, p_{\theta,Y}) \}$$

$$= \max_{q_j} \mathbb{E}_{\theta \sim q}[\dots]$$

$$= \max_{q_j} \{ \log(p_{\theta,Y}(\theta, y)) - \log(q_j(\theta_j) \prod_{i \neq j} q_i(\theta_i)) \}$$

$$- \log(\prod_{i \neq j} q_i(\theta_i)) - \log(q_j(\theta_j))$$

$$- \log(q_{-j}(\theta_{-j}))$$

$\perp\!\!\!\perp$ of q_j, ignore

$$= \max_{q_j} \{ \mathbb{E}_{\theta_{-j} \sim q_{-j}}[\log(p_{\theta,Y}(\theta, y))] - \mathbb{E}_{\theta_j \sim q_j}[\log(q_j(\theta_j))] \}$$

$\theta_j \sim q_j$

$$\mathbb{E}_{\theta_j \sim q_j}[\underbrace{\mathbb{E}_{\theta_{-j} \sim q_{-j}}[\log(p_{\theta,Y}(\theta, y))]}_{f(\theta_j)} - \log(q_j(\theta_j))]$$

- To optimize often work w/ ELBO instead of KL...

- Evidence Lower Bound:

$$D_{KL}(q \| p_{\theta|Y}) = \mathbb{E}_{\theta\sim q}[\log(q(\theta))] - \mathbb{E}_{\theta\sim q}[\log(p_{\theta,Y}(\theta,y)/p_Y(y))]$$

$$= -H(q) - \underbrace{\mathbb{E}_{\theta\sim q}[\log(p_{\theta,Y}(\theta,y))]}_{-ELBO(q\|p_{\theta,y})} + \underbrace{\mathbb{E}_{\theta\sim q}[\log(p_Y(y))]}_{\log(p_Y(y)) \text{ II of } \theta}$$

so $\min_q \{D_{KL}(q\|p_{\theta|Y})\} \Leftrightarrow \underbrace{\max_q \{ELBO(q\|p_{\theta,Y})\}}_{\text{doesn't need the evidence...}}$

- $ELBO(q\|p_{\theta,y}) + \underbrace{D_{KL}(q\|p_{\theta|Y})}_{\geq 0} = \log(\text{evidence})$

so $\log(\text{evidence}) \geq ELBO(q\|p_{\theta,y})$ "evidence lower bound" can guide model selection.

- notice: $ELBO(q\|p_{\theta,Y}) = H(q) + \mathbb{E}_{\theta\sim q}[\log(p_{\theta,Y}(\theta,Y))]$

$$= H(q) + \mathbb{E}_{\theta\sim q}[\log(p_{Y|\theta}(Y=y|\theta))] + \mathbb{E}_{\theta\sim q}[\log(p_\theta(\theta))]$$

$$= \mathbb{E}_{\theta\sim q}[\log(p_{Y|\theta}(y|\theta))] + \mathbb{E}_{\theta\sim q}[\log(p_\theta(\theta)/q(\theta))]$$

$$= \underbrace{\ldots}_{\substack{\text{1. maximize marginal/}\mathbb{E} \\ \text{log lik}}} - \underbrace{D_{KL}(q\|p_\theta)}_{\text{2. min. div. from prior}}$$

$$\max_{q_j} \{ \underbrace{\mathbb{E}_{\theta_j \sim q_j} [f(\theta_j) - \log(q_j(\theta_j))]}_{\mathbb{E}_{\theta_j \sim q_j}[f(\theta_j)] + \underbrace{H[q_j]}_{\text{concave}}} \}$$

... concave, unique optimum...

$$\nabla_{q_j} \underbrace{\int_{\theta_j} (f(\theta_j) - \log(q_j(\theta_j))) q_j(\theta_j) \, d\theta_j}_{= f(\cdot) - \log(q_j(\cdot)) - 1} \Big|_{q_j = q_j^*} = 0$$

$$q_{*j}(\cdot) \propto e^{f(\cdot)}$$

$$\Rightarrow \boxed{q_j^*(\theta_j) = \frac{1}{Z} \exp\left(\underbrace{\mathbb{E}_{\theta_{-j} \sim q_{-j}} \left[\log\left(p_{\theta,\gamma}(\theta_j, \theta_{-j}, \gamma) \right) \right]}_{\text{E step}} \right)}$$

- Iterate: until stopping
  - (i) for $k = 1$ to stopping

$$q^*_{s(k)}(\cdot) = e^{\mathbb{E}_{\theta_{-s(k)} \sim q_{-s(k)}} [\log(p_{\theta,\gamma}(\cdot, \theta_{-s(k)}, \gamma))]}$$

- Special case ... exponential families (for each conditional) (as in cond. conj. hier. models)

$$p(\theta_j \mid \theta_{-j}, y) = f_j(\theta_j) \exp\left(\phi_j(\theta_{-j}, y) \cdot u_j(\theta_j) - \log(Z_j(\theta_{-j}, y))\right)$$

$$q_j^*(\theta_j) \propto \exp\left(\mathbb{E}_{\theta_{-j} \sim q_{-j}}[\log(\dots)]\right)$$

$$= \exp\left(\log(f_j(\theta_j)) + \mathbb{E}_{\theta_{-j} \sim q_{-j}}[\phi_j(\theta_{-j}, y)] \cdot u_j(\theta_j) - \underbrace{\mathbb{E}_{\theta_{-j} \sim q_{-j}}[\log(Z_j(\theta_{-j}, y))]}_{\text{constant in } \theta_j \dots}\right)$$

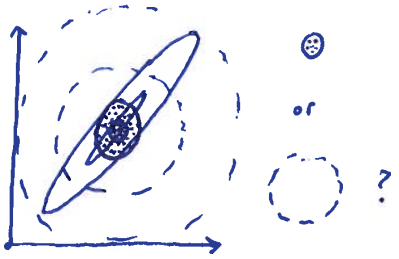
$$\propto f_j(\theta_j)\, e^{\mathbb{E}_{\theta_{-j} \sim q_{-j}}[\phi_j(\theta_{-j}, y)] \cdot u_j(\theta_j)}$$

Each q_j^* stays in the same exp. family as its complete conditional...

CAVI reduces to updating natural parameters of each ↗

- $q_j^t(\theta_j) \overset{\alpha}{=} f_j(\theta_j) \exp(\nu_j^t \cdot u_j(\theta_j))$
- Iterate:
  - for $k = 1$ to stopping:
    - $j = s(k)$
    - $\nu_j^{t+1} = \mathbb{E}_{\theta_{-j} \sim q_{-j}(\cdot\,; \nu_{-j}^t)}[\phi_j(\theta_{-j}, y)]$
    - $\nu_{-j}^{t+1} = \nu_{-j}^t$

← just a parameter update, like other coordinate algorithms often available in closed form...



References

[BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe, *Variational inference: A review for statisticians*, Journal of the American statistical Association **112** (2017), no. 518, 859–877.

[BV09] S.P. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2009.

[Dom12] Pedro Domingos, *A few useful things to know about machine learning*, Communications of the ACM **55** (2012), no. 10, 78–87.

[GCS+13] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin, *Bayesian data analysis, third edition*, Chapman & Hall/CRC Texts in Statistical Science, Taylor & Francis, 2013.

[LLJ16] Qiang Liu, Jason Lee, and Michael Jordan, *A kernelized stein discrepancy for goodness-of-fit tests*, International conference on machine learning, PMLR, 2016, pp. 276–284.

[LW16] Qiang Liu and Dilin Wang, *Stein variational gradient descent: A general purpose bayesian inference algorithm*, Advances in neural information processing systems **29** (2016), 1–9.

[LWZ22] Michael Lindsey, Jonathan Weare, and Anna Zhang, *Ensemble markov chain monte carlo with teleporting walkers*, SIAM/ASA Journal on Uncertainty Quantification **10** (2022), no. 3, 860–885.

[PNR+21] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, Journal of Machine Learning Research **22** (2021), no. 57, 1–64.

[WCLK21] Yiwei Wang, Jiuhai Chen, Chun Liu, and Lulu Kang, *Particle-based energetic variational inference*, Statistics and Computing **31** (2021), 1–17.