Title
General Purpose MCMC Sampling for Bayesian Model Averaging

Author
Boyles, Levi B.

Publication Date
2014

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

General Purpose MCMC Sampling for Bayesian Model Averaging

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Levi Beinarauskas Boyles

Dissertation Committee:
Professor Max Welling, Chair
Professor Babak Shahbaba
Professor Padhraic Smyth

2014

DEDICATION

To Mom, Dad, Seth, Keilah, and Cris.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Max Welling for his helpful guidance and support throughout my graduate career. I would also like to thank Jimmy Foulds for many helpful discussions on network analysis, and Dilan Görür for many helpful discussions on Bayesian nonparametrics and Kingman's coalescent.

This material is based on work supported by the National Science Foundation under Grant No. 0914783.

CURRICULUM VITAE

Levi Beinarauskas Boyles

Education

Doctor of Philosophy in Computer Science **2014**
University of California, Irvine *Irvine, California*

Master of Science in Computer Science **2010**
University of California, Irvine *Irvine, California*

Bachelor of Science in Physics and Computer Science **2008**
Carnegie Mellon University *Pittsburgh, Pennsylvania*

Teaching Experience

Teaching Assistant, Introduction to Artificial Intelligence **Fall 2010**
University of California, Irvine

Teaching Assistant, Project to Artificial Intelligence **Winter 2011**
University of California, Irvine

Refereed Conference Publications

J. Foulds, L. Boyles, C. DuBois, P. Smyth and M. Welling *Stochastic Collapsed Variational Bayesian Inference for Latent Dirichlet Allocation* KDD 2013

L. Boyles and M. Welling *The Time-Marginalized Coalescent Prior for Hierarchical Clustering* NIPS 2012

D. Gorur, L. Boyles and M. Welling *Scalable Inference on Kingman's Coalescent using Pair Similarity* AISTATS 2012

L. Boyles, A. Korattikara, D. Ramanan and M. Welling *Statistical Tests for Optimization Efficiency* NIPS 2011

A. Korattikara, L. Boyles, M. Welling, J. Kim and H. Park *Statistical Optimization of Non-Negative Matrix Factorization* AISTATS 2011

Works Under Review

L. Boyles and M. Welling *Retrospective Jump Sampling for Bayesian Model Averaging*

L. Boyles and M. Welling *Refractive Sampling*

ABSTRACT OF THE DISSERTATION

General Purpose MCMC Sampling for Bayesian Model Averaging

By

Levi Beinarauskas Boyles

Doctor of Philosophy in Computer Science

University of California, Irvine, 2014

Professor Max Welling, Chair

In this thesis we explore the problem of inference for Bayesian model averaging. Many popular topics in Bayesian analysis, such as Bayesian nonparametrics, can be cast as model averaging problems. Model averaging problems offer unique difficulties for inference, as the parameter space is not fixed, and may be infinite. As such, there is little existing work on general purpose MCMC algorithms in this area. We introduce a new MCMC sampler, which we call Retrospective Jump sampling, that is suitable for general purpose model averaging. In the development of Retrospective Jump, some practical issues arise in the need for a MCMC sampler for finite dimensions that is suitable for multimodal target densities; we introduce Refractive Sampling as a sampler suitable in this regard. Finally, we evaluate Retrospective Jump on several model averaging and Bayesian nonparametric problems, and develop a novel latent feature model with hierarchical column structure which uses Retrospective Jump for inference.

Chapter 1

Introduction

The body of scientific knowledge is ever growing, and with it grows the size of the scientific community. This growing scientific community is also increasingly Bayesian; as scientists recognize that their particular problem might benefit from the inclusion of prior information, the ability to integrate over uncertainty, and a more natural interpretation of the resulting inference. Bayesian inference gives us the *posterior distribution*, the probability of a parameter given the observed data. Indeed, it is not hard to see why a scientist may be interested in the probability of a parameter given a measurement rather than in the probability of the measurement given the parameter.

The posterior distribution of a parameter is a useful object; with it we can compute arbitrary expectations of functions of interest, including summary statistics such as the mean and variance of the parameter, to more sophisticated quantities such as the expected cost (in time, money, or human lives) involved with continuing an experimental evaluation of a new medical treatment. However, the posterior distribution usually cannot be represented in closed form, and inference must then be done using an approximate inference technique such as Markov Chain Monte Carlo (MCMC). MCMC is general enough to handle most Bayesian inference tasks, however, the design and implementation of such methods typically require a good deal of expertise. It is common in publications involving Bayesian methodologies that a good deal of attention is spent explaining and justifying the MCMC method used; some critics claim that this even occurs to the point of neglecting discussion of the assumptions, strengths, and weaknesses associated with the proposed statistical model. As the Bayesian community grows, this problem is likely to get worse.

There have been some recent developments in "general purpose" MCMC sampling methods that address this problem. These methods, such as slice sampling [44] or Hamiltonian Monte Carlo [10, 45], are general in the sense that only a density function (and perhaps its gradient) are needed to perform inference. One still requires some expertise in order to use such algorithms effectively, however, the availability of such methods as "black boxes" in software packages is beneficial for many reasons. For one, any fresh implementation of a MCMC algorithm will likely contain bugs which are difficult to track down; having a standard implementation that is openly available to the public eliminates a great deal of redundant work. Secondly, publications are more free to discuss the implications of the model at hand without getting bogged down in technical detail. Finally, it allows those who may not be interested implementing a MCMC procedure themselves to make use of such methods[1]. This third reason is likely the most important to the Bayesian statistics advocate; perhaps the largest hurdle to the widespread adoption of Bayesian methodology is its computationally intensive nature. To a scientist who is "on the fence" regarding Bayesian analysis, reliable and readily available software packages for Bayesian inference will increase the attractiveness and improve the practicality of Bayesian analysis.

Such software packages already exist for problems of fixed dimension. However, in a model averaging setting, we wish to infer the model underlying the data, along with the associated

[1] Note we are not advocating the replacement of careful inspection and understanding of an inference method with an automated method whose output is taken without discretion. Of course, convergence diagnosis and chain quality will always be in issue of any MCMC method; a good software package will include tools for assessing the quality of the chain produced by the software.

parameters. For example, we may be interested in inferring the number of components in a Gaussian Mixture Model, along with the parameters of the components themselves. Some probabilistic programming languages, that is, computer languages where a model or generative process is specified by the user and inference is can then be performed automatically, are capable of handling model uncertainty, however these languages still rely on an underlying sampling algorithm. The success of probabilistic programming or any other software package for model averaging is contingent on the inference algorithms it uses. Thus, improvements in the generality and efficiency of such algorithms would be a boon to the utility of such software.

Our main contribution is Retrospective Jump (RTJ) sampling, a sampling algorithm suitable for model averaging tasks. Retrospective Jump is aimed at this gap in general purpose sampling methods for model averaging. RTJ operates by sampling from mixtures of posterior distributions, where the mixture is taken over a finite subset of the potentially infinite set of models under consideration. In this way, RTJ is able to explore higher and lower dimensional spaces *before* deciding to select the associated models, simplifying the problem of initializing parameters which have yet to be represented.

In this dissertation, we will introduce a few novel MCMC algorithms. As it will be important to understand the underpinnings of a valid MCMC method, we review Markov Chain Monte Carlo in Chapter 2. Chapter 3 details model averaging and introduces some fundamental Bayesian nonparametric models that we will see throughout. Chapter 4 discusses existing work for inference in variable dimension and BNP models. Chapter 5 introduces Refractive Sampling, a MCMC sampler for finite target distributions which, for many problems of interest, is capable of finding high probability regions and exiting the transient phase more quickly than existing samplers. Refractive Sampling plays an important role as the black-box sampler for Retrospective Jump Sampling. Chapter 6 introduces Retrospective Jump Sampling (RTJ), a general purpose sampler suitable for model averaging. Chapter 7 demonstrates the application of RTJ to a novel model for hierarchical latent feature modelling, the Infinite Sites Feature Prior. Finally, we conclude in Chapter 8. A list of common notation can be found in Appendix A.

Chapter 2

Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a statistical inference method that is widely applicable. Generally speaking, it is used for computing expectations of interest – for example, we may be interested in the posterior mean and variance of some important model parameter θ. MCMC simulates from the posterior distribution of θ, allowing us to approximate $E[g(\theta)]$ for arbitrary functions g.

2.1 Monte Carlo

Consider the problem of computing the mean of some distribution with density $p(x)$, with x lying in some sample space Ω, for example $\mathbb{R}^d$:

$$E[x] = \int_{x \in \Omega} x p(x) dx \tag{2.1}$$

Computing this integral numerically can be difficult if Ω is a high dimensional space. Any integration technique that divides Ω into segments will need a number of segments exponential in the dimension d, this is the so-called *curse of dimensionality*. Monte Carlo techniques offer an alternative that avoids this cost to high dimensional integrals. If we can sample from $p(x)$, we can approximate $E[x]$ via a Monte Carlo estimate using T samples from $p(x)$.

$$E[x] \approx \frac{1}{T}\sum_{i=1}^{T} x_i \tag{2.2}$$

where $x_i \sim p(x)$. Furthermore, we can compute expectations of functions of x easily,

$$E[g(x)] \approx \bar{g}_T \triangleq \frac{1}{T}\sum_{i=1}^{T} g(x_i) \tag{2.3}$$

We can assess the convergence rate of $\bar{g}_T$ by considering the properties of the variance as T increases:

$$\mathrm{Var}(\bar{g}_T) = \frac{1}{T^2}\sum_{i=1}^{M} \mathrm{Var}(g(x)) \approx V_T \triangleq \frac{1}{T^2}\sum_{i=1}^{T} (g(x_i) - \bar{g}_T)^2 \tag{2.4}$$

Therefore, if g is sufficiently well-behaved,

$$\frac{\bar{g}_T - E[g(x)]}{\sqrt{V_T}} \tag{2.5}$$

is asymptotically standard Normal via the Central Limit Theorem. Furthermore, $TV_T \to \mathrm{Var}(g)$. Therefore, the error of the estimate $|\bar{g}_T - E[g(x)]|$ converges at a rate proportional to $\frac{1}{\sqrt{T}}$. This rate does not depend on the dimension of x, thus Monte Carlo avoids the curse of dimensionality.

Unfortunately, it is not commonly the case that we can draw independent samples from $p(x)$. Furthermore, we typically can only evaluate $p(x)$ up to a proportionality constant. For example, in a Bayesian analysis with data X and parameter θ, we want to sample from $p(\theta|X)$. We have by Bayes' Rule

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{\int p(X|\theta)p(\theta)d\theta} \tag{2.6}$$

Thus we cannot evaluate $p(\theta|X)$ efficiently due to the integral in the denominator. However, this integral is a constant with respect to θ, so we can easily evaluate

$$p(X, \theta) = p(X|\theta)p(\theta) = Zp(\theta|X) \tag{2.7}$$

where $Z = p(X) = \int p(X|\theta)p(\theta)d\theta$ is the normalization constant, also called the *partition function*.

MCMC is an integration technique that forms a Markov Chain whose stationary distribution is equivalent to that of the distribution of interest, with the idea being that the Markov Chain is easier to sample. After drawing many samples from such a Markov Chain, we will have a set of samples that approximates a set of draws from $p(x)$, which can then be used for estimating expectations. In the next section, we review Markov Chains and their relation to MCMC. This section follows Chapters 6 and 7 of [50]; we refer the reader there for more detail.

2.2 Markov Chains

A *Markov Chain* is a sequence of random variables X_t, $t \in \mathbb{N}$, so that $X_t \perp X_s | X_{t-1}$ for all $s < t - 1$. That is, X_t is independent of all preceding variables, conditioned on the previous one. From this it is easy to conclude that X_t is independent of all other variables conditioned on X_{t-1} and X_{t+1}. One simple example of a Markov Chain is

$$X_t \sim \mathcal{N}(X_{t-1}, 1) \tag{2.8}$$

where $X_1 \sim \mathcal{N}(0, 1)$. We are interested in the asymptotic behavior of a Markov Chain. That is, we are interested in the distribution of $X_\infty \triangleq \lim_{t\to\infty} X_t$, if the limit exists. In the case of (2.8), we can view $X_t = \sum_{s=1}^{t} Y_s$, where $Y_s \sim \mathcal{N}(0, 1)$. So, $X_t \sim \mathcal{N}(0, t)$, giving X_∞

distributed as the improper uniform on the real line. A bit more interesting example is to take

$$X_t \sim \mathcal{N}(X_{t-1}, \frac{1}{2^t}) \tag{2.9}$$

In this case, the sum of variances converges, giving $X_\infty \sim \mathcal{N}(0, 2)$. As the variance depends on t, this is an example of a *time heterogeneous* Markov Chain. When the form of $X_t|X_{t-1}$ does not depend on t, the chain is *time homogeneous*.

2.2.1 Invariant Distributions

Most MCMC algorithms are constructed using time homogeneous chains, so we restrict our attention to this case. If a chain is time homogeneous, we can specify it with a single transition operator, called the *kernel function*:

$$K(x, A) = P(X_t \in A|X_{t-1} = x) \tag{2.10}$$

with an associated density $K(x, y)$. In order for a chain to simulate from a specified target distribution $\pi(x)$, we need π to be *invariant* with respect to K, that is, applying K as an operator to π gives back π:

$$\pi(A) = (K\pi)(A) = \int K(x, A)\pi(dx) \tag{2.11}$$

for all Borel sets A. Chains that allow for invariant distributions are called *positive* chains.

There are some choices for $X_t|X_{t-1}$ such that there is no unique invariant distribution, or that X_t does not converge to the invariant distribution. For example, we may take

$$X_t \sim \text{sgn}(X_{t-1})\text{Exponential}(1) \tag{2.12}$$

In this case, the Markov Chain is highly dependent on the initial state; if the initial state is positive, then X_t is always positive, and similarly if the initial state is negative. Thus, in this case there are two base invariant distributions, the positive and negative Exponential distributions, and *any* mixture of these distributions is also invariant under this kernel. This is an example of chain that is not *irreducible*. As another example, we may take

$$X_t \sim \text{Exponential}(2X_{t-1}) \tag{2.13}$$

This would give a divergent sequence, thus there is no invariant distribution. This is an example of a chain that is not *recurrent*. Finally, taking

$$X_t \sim -\operatorname{sgn}(X_{t-1})\text{Exponential}(1) \tag{2.14}$$

gives a sequence that oscillates between positive and negative exponential distributions. The Laplace distribution is invariant to the kernel of this chain, however X_t does not converge to the Laplace distribution, as at any step the distribution of X_t is either positive or negative Exponential. This is an example of a chain that is not *aperiodic*.

In some sense, aperiodicity is less important than the other conditions, as expectations taken with respect to aperiodic chains may still converge. However, as we will see later, most MCMC algorithms are aperiodic anyway.

These examples illustrate the pitfalls that can occur when constructing a Markov Chain. If we are pursuing a Markov Chain whose equilibrium distribution is the same as some specified target distribution, we need the chain to have a well defined, unique equilibrium distribution. A chain is said to be *ergodic* if any choice for the initial state leads to a unique equilibrium distribution. One formal definition of ergodicity can be made as follows: a chain with kernel

K is ergodic with invariant distribution π if

$$\lim_{n\to\infty} \left\| \int K^n(x,\cdot)\mu(dx) - \pi \right\|_{TV} = 0 \tag{2.15}$$

where $||\cdot||_{TV}$ is the total variation norm, and μ is an arbitrary initial distribution.

Note that a chain that is invariant to distribution π is invariant to any measure[1] of the form $C\pi$ with C a positive constant. This is convenient if we only know π up to a constant of proportionality.

2.2.2 Ergodicity

The three cases in the previous section where the chain had no unique invariant distribution are counterexamples of three conditions that are required for an ergodic chain.

The first condition is *irreducibility.* Loosely speaking, a chain is irreducible if any state can be reached from any other state. More formally, a state is ψ-irreducible if for measure ψ, for all sets A with $\psi(A) > 0$, and for all states x, there exists n such that

$$K^n(x, A) > 0 \tag{2.16}$$

If this condition is not met, there may be multiple invariant distributions.

The second condition is *recurrence.* A chain is recurrent if, when starting from some state we will return to it infinitely often. More formally, a chain is *Harris recurrent*, if the chain

[1] In this chapter we assume a basic familiarity with measures. Uninitiated readers may see Section 3.2 for a brief introduction.

is ψ-irreducible and for every A such that $\psi(A) > 0$ and for every $x \in A$

$$P\left(\sum_{n=1}^{\infty} \mathbb{1}(X_t \in A) = \infty | X_1 = x\right) = 1 \tag{2.17}$$

If this condition is not met, some sets of states (called *transient* states) may only be explored for a finite number of steps. Thus the chain cannot have an invariant distribution with positive probability assigned to these sets. Note this definition includes irreducibility as a prerequisite condition.

Finally, there is *aperiodicity*. A chain is periodic if there exist sets of states such that the waiting times to return have non-unit g.c.d. For a formal definition, we need the concept of a *small set*. A set C is small if for all $x \in C$ and all sets A, there exists integer m and nonzero measure ν_m such that

$$K^m(x, A) \geq \nu_m(A) \tag{2.18}$$

That is, there exists a m such that $x \in C$ can reach any set with positive probability. The intuition is that C is "small" in the sense there is a component of the (compounded) transition probability that is independent of x and shared for all choices of $x \in C$. That is, the kernel is not too sensitive to variations of x, as long as $x \in C$.

A ψ-irreducible chain has a cycle of length d if there exists a small set C with integer M, and distribution ν_M such that the g.c.d. of

$$\{m \geq 1 | C \text{ is small for } \nu_m \geq \delta_m \nu_M, \delta_m > 0\} \tag{2.19}$$

is d. If $d > 1$, then C is only periodically small, meaning repeated compositions of the transition kernel does not converge. This means the chain can be decomposed into sets of states that communicate with each other in a cyclic fashion. As it turns out, d is independent of the small set C, and the *period* of the chain is taken as d. A chain is aperiodic if $d = 1$, and if a chain is not aperiodic, then the chain will not converge to any invariant distribution.

A chain that has these three properties is ergodic:

Theorem 2.2.1 *If a chain (X_T) is has invariant distribution π, is Harris recurrent and aperiodic, then*

$$\lim_{n\to\infty} \left\| \int K^n(x,\cdot)\mu(dx) - \pi \right\|_{TV} = 0$$

for every initial distribution μ.

Furthermore, we can compute expectations of functions of interest by using an ergodic chain (X_T), rather than independent samples in a Monte Carlo estimate:

$$S_T(g) = \frac{1}{T}\sum_{t=1}^{T} g(X_t) \tag{2.20}$$

The error of this estimate will go to zero as T goes to infinity:

Theorem 2.2.2 (The Ergodic Theorem) *If a chain (X_n) has an invariant finite measure π, then the following are equivalent:*

1. *If $f, g \in L^1(\pi)$, and $\int g(x)d\pi(x) \neq 0$, then*

$$\lim_{T\to\infty} \frac{S_T(f)}{S_T(g)} = \frac{\int f(x)d\pi(x)}{g(x)d\pi(x)}$$

2. *(X_T) is Harris recurrent*

Therefore, an average computed with ergodic Markov chains with invariant measure π will converge to the corresponding expectation taken with respect to π. As π only needs to be a finite measure, we only need to specify a target distribution up to a normalization constant.

2.3 Detailed Balance and MCMC

We wish to construct Markov chains where $X_\infty \sim \pi$ for some specified target distribution $\pi(x)$. Luckily, it is uncommon that we need to verify by hand that a kernel K satisfies all the desired properties:

1. Invariant to π
2. Irreducible
3. Recurrent
4. Aperiodic

Instead, we may check that the chain follows *detailed balance*, a sufficient condition for invariance, and, if given ψ-irreducibility, a sufficient condition for recurrence. A kernel follows detailed balance with invariant density π if

$$K(y,x)\pi(y) = K(x,y)\pi(x) \tag{2.21}$$

The intuition here is that the probability flow out of state x into state y is the same as the probability flow out of state y into state x. This is a simple condition to check (or ensure) that a kernel follows. The vast majority of MCMC algorithms used in statistical inference follow detailed balance.

The other two criteria, irreducibility and aperiodicity, must still be verified. Aperiodicity is generally easy to show; for example, any MCMC method that has an accept/reject step will be aperiodic as transitions of the form $X_t = X_{t-1}$ will occur with nonzero probability. Showing irreducibility requires more work, however for many sensible kernels, showing that any state can be reached from any other state in a finite number of steps is not terribly difficult.

2.4 Examples

The following sections provide a few examples of popular MCMC algorithms. To simplify exposition, we present the algorithms without consideration of numerical issues. Typically, a MCMC implementation will take the log-density of the target distribution as input, rather than the density itself. As the density is often the product of many small terms, directly representing it may result in underflow. Thus, the reader should keep in mind that, in a true implementation, $\pi(x)$ should be replaced with $\ln \pi(x)$, $\frac{\pi(x')}{\pi(x)}$ with $\ln \pi(x') - \ln \pi(x)$, and Uniform$(0, \pi(x))$ with $\ln \pi(x)$ − Exponential(1), to name a few examples.

2.4.1 Metropolis Hastings

Metropolis Hastings (MH) [39, 24] is perhaps the simplest MCMC algorithm available. Metropolis Hastings operates by making proposed updates from a *proposal* distribution q that is easy to sample from, and then deciding whether to accept or reject the update.

Algorithm 1 shows one step of the MH sampler. The acceptance probability α ensures that the kernel of MH follows detailed balance. A simple choice for the proposal distribution q is a symmetric Gaussian distribution; this choice is known as *Random Walk Metropolis*.

As MH follows detailed balance, the chain has π as its invariant distribution. If $q(\cdot|x)$ is positive everywhere, then any state can be reached from any other state, and the chain is irreducible, also giving recurrence as the chain follows detailed balance. Finally, the chain is aperiodic due to the reject step; the chain can remain at a specific value of x for a random number of iterations.

Algorithm 1 Metropolis Hastings

Given: Target density π, initial state x, proposal distribution q
Sample $y \sim q(y|x)$
Let

$$\alpha \leftarrow \min\left(1, \frac{\pi(y)}{\pi(x)}\frac{q(x|y)}{q(y|x)}\right)$$

Take

$$x' \leftarrow \begin{cases} y & \text{with probability } \alpha \\ x & \text{otherwise} \end{cases}$$

return x'

2.4.2 Slice Sampling

Slice sampling [44] is a MCMC technique that operates by introducing an auxiliary variable u that, given the parameter x, varies uniformly from 0 to $\pi(x)$, giving the joint for x and u:

$$\pi(x, u) \propto \mathbb{1}\left(u < \pi(x)\right) \tag{2.22}$$

This can be seen as the uniform distribution on (x, u), but constrained so that $u < \pi(x)$. Put another way, this is the uniform distribution over the area under the curve $\pi(x)$. Integrating out u gives back $\pi(x)$, so if we can sample (x, u) jointly and discard u, we will have a valid Markov chain on x.

Sampling (x, u) jointly can be done by sampling each variable in turn from horizontal and vertical cross sections (called slices) of the area under $\pi(x)$. $u|x$ is simply drawn from $\text{Uniform}(0, \pi(x))$, however sampling $x|u$ first requires computing bounds on the slice by a "stepping out" procedure, see Algorithm 2.

Stepping out is not guaranteed to compute the full horizontal slice. Instead, it computes a random interval (x_l, x_r) that contains x. This is done in a way that still follows detailed balance. The idea is that for a fixed choice of u, any state x' reached by x using the interval (x_l, x_r) has the same probability of constructing the same interval (x_l, x_r) when going in reverse.

As long as the target distribution $\pi(x)$ is not comprised of "probability islands" separated by a regions of 0 probability with width greater than the stepping width w, then slice sampling is irreducible and thus ergodic.

2.4.3 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [10, 45] is a MCMC algorithm that makes use of gradient information in order avoid random walk behavior and improve sample efficiency. HMC introduces a momentum variable $p \sim \mathcal{N}(0, M)$, and is invariant to the joint distribution

$$\pi(x, p) = \exp\left(-H(x, p)\right) \tag{2.23}$$

where H is the Hamiltonian

$$H(x, p) = -\ln \pi(x) + \frac{1}{2} p^T M^{-1} p \tag{2.24}$$

Algorithm 2 Slice Sampling

Given: Target density π, initial state x, stepping width w, stepping iterations m
Sample $u \sim \text{Uniform}(0, \pi(x))$
Sample $x_l \sim x - \text{Uniform}(0, w)$
Set $x_r \leftarrow x_l + w$

////// Stepping out //////
$m_l \leftarrow \lfloor \text{Uniform}(0, m) \rfloor$
$m_r \leftarrow m - 1 - m_l$
while $u < \pi(x_l)$ **and** $m_l > 0$ **do**
    $m_l \leftarrow m_l - 1$
    $x_l \leftarrow x_l - w$
end while
while $u < \pi(x_r)$ **and** $m_r > 0$ **do**
    $m_r \leftarrow m_r - 1$
    $x_r \leftarrow x_r + w$
end while

////// Sample and shrink interval //////
$x' \sim \text{Uniform}(x_l, x_r)$
while $\pi(x') > u$ **do**
    if $x' < x$ **then**
        $x_l = x'$
    end if
    if $x' > x$ **then**
        $x_r = x'$
    end if
    $x' \sim \text{Uniform}(x_l, x_r)$
end while
return x'

Sampling from $\pi(x, p)$ and discarding p produces samples from $\pi(x)$. The most common way to produce a Markov chain with invariant distribution $\pi(x, p)$ is the *leapfrog* integrator[2]

$$p \leftarrow p + \frac{\varepsilon}{2} g(x) \tag{2.25}$$

$$x \leftarrow x + \varepsilon M^{-1} p \tag{2.26}$$

$$p \leftarrow p + \frac{\varepsilon}{2} g(x) \tag{2.27}$$

where ε is the stepsize parameter and $g(x) = \nabla_x \ln \pi(x)$ is the gradient of the log-density. These updates are repeated for L leapfrog iterations, and then the resulting state (x', p') is accepted with probability

$$\alpha(x, p \to x', p') = \min\left(1, \frac{\exp(f(x') - p'^T M^{-1} p'/2)}{\exp(f(x) - p^T M^{-1} p/2)}\right) \tag{2.28}$$

This algorithm follows detailed balance: the mapping $(x, p) \to (x', p')$ has unit Jacobian (see [45] for details), and (2.28) is the standard Metropolis Hastings acceptance probability.

As p is independent of x, we may draw $p \sim \mathcal{N}(0, M)$ at the beginning of every iteration. See Algorithm 3.

[2] Integration in this context refers to simulation from a differential equation by discretization. See Section 5.1 or [45] for more background on the motivation of the leapfrog method.

Algorithm 3 Hamiltonian Monte Carlo

Given: target density π, gradient g, initial state x
Given: stepsize ε, leapfrog iterations L, covariance M
Sample $p \sim \mathcal{N}(0, M)$
Set $x' \leftarrow x$
Set $p' \leftarrow p$
for $l = 1 : L$ **do**
  $p' \leftarrow p' + \frac{\varepsilon}{2} g(x')$
  $x' \leftarrow \varepsilon M^{-1} p'$
  $p' \leftarrow p' + \frac{\varepsilon}{2} g(x')$
end for
Take

$$\alpha \leftarrow \min \left(1, \frac{\exp(f(x') - p'^T M^{-1} p'/2)}{\exp(f(x) - p^T M^{-1} p/2)} \right)$$

Take

$$x' \leftarrow \begin{cases} x' & \text{with probability } \alpha \\ x & \text{otherwise} \end{cases}$$

return x'

Chapter 3

Model Determination

In designing a model for an inference task, one must make many important decisions that will affect the outcome of the inference. Should this parameter be constrained to be positive? How should it tie in with other, related parameters? Should this matrix be symmetric? Should there be 10 clusters or 100? What is the loss function? A Bayesian has a further need to specify the priors of the parameters of interest: Should this parameter be heavy-tailed? Should it be skewed? These choices make up the problem of *model selection*. In some cases, there is no obvious choice, and the best choice may depend on the particular data on hand. There has thus been extensive study on performing model selection in an automated way, using the data to inform the models selected.

Consider an inference task in which we have a finite number of models under consideration, indexed by m. Perhaps the most common strategy for model selection is to use one of the many available Information Criteria, for example the Akaike Information Criterion (AIC) [1]. The AIC of a model m is defined as

$$AIC(m) = 2k - 2\ln(L_{\max}) \tag{3.1}$$

where $L_{\max} = \max_\theta p(X|\theta, m)$ is the maximum likelihood of the data X over the parameters θ under model m, and k is the number of free parameters of the model. Models with lower AICs are preferred, as they exhibit a lower information loss in representing the true model as measured using the KL-divergence. The AIC increases with k, so overly complex models are penalized. The AIC is usually used for model selection in maximum-likelihood frameworks.

Another choice, which is suitable for Bayesian inference, is the Bayes Factor [31]:

$$BF(m_1, m_2) = \frac{p(X|m_1)}{p(X|m_2)} \tag{3.2}$$

where $p(X|m)$ is the marginal likelihood of X under model m. As the parameters of each model are integrated out, overly complex models will be naturally penalized. If a prior belief on the available models is available, then one may instead use the posterior odds ratio:

$$PO(m_1, m_2) = \frac{p(X|m_1)P(m_1)}{p(X|m_2)P(m_2)} \tag{3.3}$$

A rather different approach is to treat the model as a random variable M, and infer it along with all other parameters; see, for example, [19], which motivates Reversible Jump MCMC via this Bayesian Model "selection" problem.

Taking M as random gives it the full Bayesian treatment, where we are interested in the posterior distribution of M, and not a single value. Predictions can still be made with this posterior over M, and as this involves computing expectations over a range of models, this is called *model averaging*. A more general classification encompassing both model selection and model averaging is *model determination* [19].

If $\theta^{(i)}$ are the parameters associated with model m_i, then we can take $\theta = \bigcup_i \theta^{(i)}$ and infer

$$p(\theta, M = m_i|X) \propto p(X, \theta, M = m_i) = p(X|\theta^{(i)}, M = m_i)p(\theta)P(M = m_i) \tag{3.4}$$

where $p(M = m_i)$ is the prior probability of model m_i, and $p(\theta)$ is defined agnostic to the value of M. The posterior probability of model m_i is

$$p(M = m_i|X) = \frac{p(X|M = m_i)P(M = m_i)}{\sum_j p(X|M = m_j)p(M = m_j)} = PO(m_i, \bigcup_j m_j) \tag{3.5}$$

Thus the posterior probability of $M = m_i$ is simply the posterior odds ratio of m_i versus all models combined. So, Bayesian inference of M will also guard against overly complex models. This is the approach we consider throughout this thesis.

We will focus our attention on the case where the models m_i are *nested*, that is, if $\Omega^{(i)}$ is the space of possible $\theta^{(i)}$, then $i < j$ implies that $\Omega^{(i)} \subseteq \Omega^{(j)}$. We call such sets of models *variable dimension models*, wherein the model structure is shared across all models m_i, and all that changes is some notion of dimension. Most Bayesian nonparametric (BNP) models are examples of this case. Despite the attention given to nested model determination problems, we note that many of the techniques explored also apply to general model determination problems.

3.1 Model Averaging

Consider a basic model determination task, where we believe the data to be modeled well by a mixture of Gaussians, but we do not know the number of components. We might write a generative model for the data X, with $X_i \in \mathbb{R}^d$:

$$\begin{aligned}
K &\sim \text{Poisson}(\lambda) \\
w &\sim \text{Dirichlet}(\alpha \mathbf{1}^{(K)}) \\
\mu_k &\sim \mathcal{N}(0, \Sigma_\mu) \\
\Sigma_k &\sim \text{InvWishart}(\Psi, \nu) \\
z_i &\sim \text{Categorical}(w) \\
X_i &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})
\end{aligned} \tag{3.6}$$

In a standard setting where K is known, we would be interested in sampling the posterior distribution of w, μ, and Σ (in this case z can be integrated out, so we may restrict our attention to the continuous parameters):

$$p(w, \mu, \Sigma | X, K) \propto p(w, \mu, \Sigma, X, K) = p(X | w, \mu, \Sigma, K) p(w, \mu, \Sigma | K) \tag{3.7}$$

In this case, we can perform inference by constructing an ergodic Markov Chain whose invariant distribution is proportional to $p(w, \mu, \Sigma, X | K)$, and sample w, μ and Σ, holding X fixed. The difficulty of this inference task is that the degrees of freedom in each of w, μ and Σ depends on K, so this problem cannot be framed as sampling from some distribution in a finite dimensional space.

Still, there are inference techniques that can handle this type of problem, for example Reversible Jump MCMC, which we will see in detail later. Reversible Jump infers K along with all other parameters to build a Markov chain on state spaces of varying dimension. The ability to make predictions *averaged* over the posterior distribution, rather than simply choosing a point estimate for the parameters, is one of the main advantages of Bayesian inference. We are free to do this even in the variable dimension context by averaging over the models as well:

$$p(X_{N+1}) = E_{p(K,w,\mu,\Sigma|X)}[p(X_{N+1}|K,w,\mu,\Sigma)] \approx \frac{1}{T}\sum_{t=1}^{T} p(X_{N+1}|K(t),w(t),\mu(t),\Sigma(t)) \quad (3.8)$$

where $\theta(t)$ is the t^{th} MCMC iterate for parameter θ.

Another approach would be to define the mixture with a *Bayesian nonparametric* prior, for which w is infinite dimensional by definition. In this setting, the z_i are not integrated out, and the elements of w are only represented if there is data assigned to the associated component (that is, w_k is represented if $z_i = k$ for some i). Thus the "effective K" for the Bayesian nonparametric model is simply the number of clusters to which data has been assigned.

In either case, the parameter spaces we are dealing with are potentially infinite. Measures are the natural means through which to reason about random variables on infinite parameter spaces, and are fundamental to the development of Bayesian nonparametrics. We briefly review measures and related concepts in the next section; much of the material follows Chapter 2 of [3], we refer the reader there for more detail.

3.2 Measures

Consider a univariate continuous random variable X, which can be represented in two ways, either through the probability density function (pdf) f or the cumulative distribution function (cdf) F. Thus,

$$F(b) = \int_{-\infty}^{b} f(x)dx = P(X \leq b) \quad (3.9)$$

The cdf is often extended to take intervals as its argument, so that

$$F([a, b]) = F(b) - F(a) = P(a \leq X \leq b) \tag{3.10}$$

Measures can be used to generalize this concept to higher dimensions. A *measure* is a nonnegative function on *Borel-measurable*[1] subsets of a space Ω that gives some notion of the "size" of the set. For example, the Lebesgue measure λ, put simply, is the volume of the set in the traditional sense: if $\Omega = \mathbb{R}^n$ and C_n is a n-dimensional cube with edge length 2, then $\lambda(C_n) = 2^n$.

However, other measures are possible that give more or less weight to particular regions of the space. For example, for a given measure μ and the cube C_n described above, we may have $\mu(C_n) = 5$, or $\mu(C_n) = \infty$, or $\mu(C_n) = 0$. If $A \subseteq \Omega$, $\mu(\Omega)$ may either be finite or infinite. A measure μ is σ-finite if Ω is a countable union of sets of finite measure, for example the Lebesgue measure is not finite but is σ-finite.

Finite measures have the benefit that they can be normalized so that $\mu(\Omega) = 1$, and such normalized measures are called *probability measures*. We may associate a random variable X to a probability measure μ on a space Ω, so that $X \in \Omega$, and

$$P(X \in A) = \mu(A) \tag{3.11}$$

Furthermore, there is an associated density function of μ:

[1] A Borel-measurable set is a set that can be constructed via countable union, countable intersection, and complementation of open sets. Thus, the Borel sets capture nearly all sets that may be of interest, and all sets in this dissertation may be assumed to be Borel-measurable unless otherwise specified.

Theorem 3.2.1 (The Radon-Nikodym Theorem) *If μ is an absolutely continuous σ-finite measure with respect to a σ-finite measure λ, then there exists a function f such that*

$$\mu(A) = \int_A f d\lambda = \int_A f(x)\lambda(dx) \tag{3.12}$$

f is called the Radon-Nikodym derivative of μ, and can be denoted as $\frac{d\mu}{d\lambda}$. When μ is a probability measure, and λ is the Lebesgue measure, f is the usual probability density function.

The integral shown in Theorem 3.2.1 is a Lebesgue integral; recall that the Lebesgue integral can be constructed by taking horizontal slices of a function, rather than vertical as in the Riemann integral; if we define

$$f^*(t) = \mu(\{x|f(x) > t\}) \tag{3.13}$$

then we can construct the Lebesgue integral in terms of a Riemann integral:

$$\int_A f d\mu = \int_0^\infty f^*(t)dt \tag{3.14}$$

Note that the Lebesgue integral has several advantages; for one, if we have a σ-finite measure on an infinite dimensional space, we can construct integrals over sets with infinite dimension that may still be finite. Thus it is possible to reason about random variables that live in infinite dimensional spaces, if we use a *prior* that is a probability measure on the infinite space as our measure of integration.

3.2.1 Infinite Product Spaces

A *product space* is the space formed by taking the Cartesian product of the elements of multiple spaces, for example if we have spaces Ω_1 and Ω_2, we can define the product space $\Omega = \Omega_1 \times \Omega_2$, and ω is an element of Ω if we can write $\omega = (\omega_1, \omega_2)$, and $\omega_j \in \Omega_j$. If we have d spaces, we can similarly define a product space of dimension d. If we have measures μ_j for each of the corresponding spaces, then we can define the measure on the set $A_1 \times \ldots \times A_d$ as

$$\mu(A_1 \times \ldots \times A_d) = \prod \mu_j(A_j) \tag{3.15}$$

We next extend this definition to the limit $d \to \infty$. Let $\Omega = \prod_{j=1}^{\infty} \Omega_j$, and define C_d to be a *cylinder* with base B^d if

$$C_d = \{\omega \in \Omega | (\omega_1, \ldots \omega_d) \in B^d\} \tag{3.16}$$

We can construct a series of measures on the bases of dimension d given a "conditional" probability measure $\mu_d(\omega_1, \omega_2, \ldots \omega_{d-1}, d\omega_d)$, where we have defined a measure on Ω_d conditioned on an element in $\prod_{j=1}^{d-1} \Omega_j$. We can construct the marginal probability of the base B^d

$$P_d(B^d) = \int_{\Omega_1} \mu_1(d\omega_1) \int_{\Omega_2} \mu_2(\omega_1, d\omega_2) \ldots \int_{\Omega_d} \mathbb{1}(\omega \in B^d) \mu(\omega_1, \ldots, \omega_{d-1}, d\omega_d) \tag{3.17}$$

P_d can be seen as the probability of the event B^d, with all variables not in $\prod_{j=1}^{d} \Omega_j$ marginalized out. By Theorem 2.7.2 of [3], there is a unique probability distribution P that agrees with P_d on all d-dimensional cylinders.

It is thus reasonably straightforward to define probability measures on infinite product spaces[2]. However, in this dissertation, we are primarily interested in MCMC inference algorithms in infinite parameter spaces. Measures are not suitable for representation on a

[2]Using Kolmogorov's Extension Theorem, it is even possible to define measures over a *continuum* of spaces.

computer, which would generally involve an expensive integration. We would rather work with density functions; however we cannot represent a density function on an infinite space either. We may, however, consider the Radon-Nikodym derivative taken with respect to the Lebesgue measure on the base B^d of a cylinder, giving a d dimensional marginal density function.

If we take λ_d to be the Lebesgue measure on $\prod_{j=1}^{d} \Omega_d$, and given a measure P for Ω

$$\frac{dP}{d\lambda_d} = \frac{dP_d}{d\lambda_d} \tag{3.18}$$

That is, we may take the density function of P_d as only the parameters in the base of a d-dimensional cylinder affect λ_d.

3.3 Bayesian Nonparametrics

One way to define a model with potentially infinitely many parameters is to define a prior such as that in (3.6): first define a prior on the dimension of the parameter space, and then define the parameters conditioned on this dimension. Note that this prior puts probability 1 on models with finite dimension, that is

$$P\left(\left|\{w_i | w_i > 0\}\right| < \infty\right) = 1 \tag{3.19}$$

An alternative is to allow the parameters to live in an infinite dimensional space and defining a suitable prior in that space. For example w in (3.6) may be defined according to a stick-breaking procedure such as (3.23). This prior puts probability 1 on models with infinite dimension. However, in this GMM example, a finite dataset will only be associated with finitely many components, and thus only finitely many parameters need be represented.

The distinction between these two priors is important for considering the *consistency* properties of Bayesian models involving them. In this context, consistency is the property that the data eventually overwhelms the prior, so that eventually the model with find the "true" parameter. [15] showed that in infinite dimensional settings, priors are not necessarily consistent even when they put positive probability density on the truth. [15] also defines a class of priors, called *tail-free* priors, which are consistent in this sense. This work gave rise to the popularity of the Dirichlet Process.

The question so far has been consistency in distribution: does the model's posterior predictive density converge to the true density generating the data? For models using the Dirichlet Process, the answer is yes. Consistency in these types of problems is an ongoing area of research, see [16, 17, 36, 34]. However, some recent work has made an important point: consistency in distribution does not imply consistency in parameters. In fact, a Dirichlet Process Mixture Model applied to data from a finite mixture will not estimate the number of components correctly, even in the limit of infinite data [40]. Therefore, it is advisable to carefully consider the end goal of the inference task. Infinite priors such as a Dirichlet Process would be well suited for prediction tasks wherein the interpretation of the parameters is not important. If interpretability is important, then perhaps a model such as (3.6) would be more appropriate.

Priors that put probability 1 on models with infinite dimension are classified as *Bayesian nonparametric* models. Bayesian nonparametric (BNP) modelling is growing in popularity among scientists and statisticians; Bayesian models whose complexity adapts to the complexity of the data is a strikingly attractive property. The prior complexity of a BNP model typically grows with N; for example, the expected number of clusters in N data generated from the Dirichlet Process is $O(\ln N)$.[3] This is in accordance with the intuition that more data affords more model complexity. Many practitioners use these models to infer the model

[3] As the introduction of the i^{th} point introduces a new cluster with probability $\frac{\alpha}{i+\alpha-1}$, the expected number of clusters in the prior is $\alpha \sum_{i=1}^{N} \frac{1}{i+\alpha-1} \approx \alpha H_N \approx \alpha \ln N$, where H_N is the N^{th} Harmonic number.

complexity[4] from the data along with the rest of the model, though this is perhaps ill-advised as stated above. With fixed N, many BNP models can be viewed as mixtures of finite models, fitting into the model averaging framework outlined in Chapter 1. Here, we outline two important BNP priors, the Dirichlet Process and the Indian Buffet Process.

3.3.1 Dirichlet Process

The Dirichlet Process (DP) [13] is a distribution over measures. Define an event space Ω, a base probability measure G, a constant $\alpha > 0$, and a finite partition $\{A_i\}_{i=1}^K$ of Ω. Then a draw from the Dirichlet Process $Y \sim \mathrm{DP}(\alpha G)$ follows

$$Y(A_1), Y(A_2), \dots Y(A_K) \sim \mathrm{Dirichlet}(\alpha G(A_1), \alpha G(A_2), \dots \alpha G(A_K)) \tag{3.20}$$

That is, Y is a *random measure*, and the distribution of the measures of a partition are Dirichlet. Thus, Y is probability measure: $Y(\Omega) = 1$. Interestingly, Y can be represented as a countable sum of degenerate measures. Sethuramen [52] showed that Y follows the following recursive distributional equation:

$$Y \stackrel{d}{=} v_1 \delta_{\theta_1} + (1 - v_1) Y \tag{3.21}$$

where $v_1 \sim \mathrm{Beta}(1, \alpha)$, $\delta_\theta(A) = 1$ if $\theta \in A$ and 0 otherwise, and θ_1 is drawn from G. This leads to the following constructive definition of Y:

$$Y(A) = \sum_{i=1}^{\infty} w_i \delta_{\theta_i}(A) \tag{3.22}$$

[4]that is, the dimension of the parameter space.

The θ_i are drawn from G and the w_i determined by *stick-breaking*:

$$\begin{aligned} v_i &\sim \text{Beta}(1, \alpha) \\ w_i &= v_i \prod_{j=1}^{i-1} (1 - v_j) \end{aligned} \tag{3.23}$$

Stick-breaking provides a convenient way to (partially) represent a draw from the DP, and is frequently used when performing inference on models using the DP prior.

Dirichlet Process Mixture Model

A common use for the DP is to use the sticks w_i as mixture weights and the atoms θ_i as cluster parameters for a mixture model. For example, we may define a Dirichlet Process Mixture of Gaussians:

$$\begin{aligned} \theta_k &\sim G \\ v_k &\sim \text{Beta}(1, \alpha) \\ w_k &= v_k \prod_{m=1}^{k-1} (1 - v_m) \\ z_i &= \text{Categorical}(w_k) \\ X_i &\sim F(\theta_{z_i}) \end{aligned} \tag{3.24}$$

For a Dirichlet Process Mixture of Guassians, $\theta_k = (\mu_k, \Sigma_k)$ and $F(\theta_k) = \mathcal{N}(\mu_k, \Sigma_k)$. Typically, the difficulty in performing inference in a DPMM lies in introducing new θ_k that have yet to be explicitly represented, inferring w_k as it has infinite length, and inferring z_i as it is a discrete variable with infinite support.

3.3.2 Chinese Restaurant Process

When the w_i of the DP are used as mixing weights in a mixture model, they can be integrated out and the distribution of assignment variables z_i remains:

$$P(z) = \frac{\Gamma(\alpha)\alpha^K}{\Gamma(N+\alpha)} \prod_{k=1}^{K} \Gamma(n_k) \tag{3.25}$$

where n_k is the number of i such that $z_i = k$. This distribution is the Exchangeable Partition Probability Function (EPPF) [48] for the Chinese Restaurant Process (CRP). The CRP can also be expressed as a conditional prior giving the probability of z_i given all other elements of z (denoted $z_{\neg i}$). Let the counts N and n_k be determined from $z_{\neg i}$ (that is, the counts do not include z_i in their sums), and if there are a total of K clusters represented in $z_{\neg i}$, then

$$P(z_i = k | z_{\neg i}) = \begin{cases} \frac{n_k}{N+\alpha} & \text{if } k \leq K \\ \frac{\alpha}{N+\alpha} & \text{if } k = K+1 \end{cases} \tag{3.26}$$

If G is a conjugate prior of F, then the parameters θ_k can be marginalized and sampling the z_i via Gibbs sampling allows the creation and destruction of clusters. If F and G are not conjugate, then the instantiation of new parameters θ_k when sampling $k = K + 1$ becomes a tricky issue, and Gibbs sampling is no longer applicable.

3.3.3 Indian Buffet Process

Frequently flat clustering models are too restrictive. For example, in a social network setting, an individual may belong to several latent groups; say Alice's hobbies include running and board games, Bob enjoys soccer and running, and Carol enjoys reading. As Alice and Bob share similar interests, a reasonable model may predict an increased probability that Alice and Bob are friends. No individual is restricted to having only one interest, so we may wish to model these latent properties with a "clustering" model where an individual can belong to multiple clusters.

The Indian Buffet Process (IBP) [22] is a prior over binary matrices with unbounded width. A draw Z from the IBP represents the assignments of datapoints to latent "features" or "groups;" $Z_{ik} = 1$ if datapoint i belongs to feature k. As this assignment is not exclusive or constrained in any way, a datapoint may be assigned to multiple features. Although the draw Z has unbounded width, when the height of Z (the number of datapoints observed) is finite, the set of columns with nonzero entries is finite.

The IBP can be characterized by its Exchangeable Feature Probability Function (EFPF) [6]. The prior probability of a matrix Z with N rows and K columns is:

$$P(Z) = \frac{\alpha^K \exp(-\alpha H_N)}{\prod_{h=1}^{2^N-1} K_h!} \prod_{k=1}^{K} \frac{(N-n_k)!(n_k-1)!}{N!} \tag{3.27}$$

where H_t is the t^{th} Harmonic number and $n_k = \sum_{i=1}^{N} Z_{i,k}$. h indexes all possible nonzero binary vectors of length N, and K_h is the number of columns of Z that are equivalent to the h^{th} binary vector.

The IBP can also be expressed as a conditional prior. Let $Z_{\neg i,\cdot}$ denote the matrix Z but with row i removed, and define the sums n_k in terms of $Z_{\neg i,\cdot}$. For columns k with $n_k > 0$,

$$Z_{i,k}|Z_{\neg i,\cdot} \sim \text{Bernoulli}(\frac{n_k}{N}) \tag{3.28}$$

For Poisson(α/N) columns with $n_k = 0$, $Z_{i,k}$ is set to 1, forming a number of new features. This perspective is useful for Gibbs sampling in conjugate models, in a similar manner to Gibbs sampling for the CRP.

3.3.4 Stick Breaking for the Indian Buffet Process

There is also a stick-breaking representation for the IBP, related to its connection to the Beta Process [25, 56]. This construction is given in [55] as:

$$\nu_i \sim \text{Beta}(\alpha, 1) \tag{3.29}$$

$$\mu_i = \mu_{i-1}\nu_i = \prod_{j=1}^{i} \nu_j \tag{3.30}$$

Note that in this stick breaking construction, $\mu_i \leq \mu_{i-1}$ for all i, and $\sum_i \mu_i \neq 1$, unlike stick breaking for the DP. The conditional distribution of Z is simply a Bernoulli draw for each entry of Z:

$$Z_{ik}|\mu \sim \text{Bernoulli}(\mu_k) \tag{3.31}$$

Chapter 4

Related Work

One of the primary difficulties in performing inference in model averaging is the problem of *parameter instantiation* – how should a new parameter that has not been explicitly represented be set initially when it is first needed? Furthermore, how should existing parameters adjust to this newly introduced parameter? These choices dramatically affect the ability of the sampler to mix across models. In this chapter we review several MCMC methods for model averaging. The methods outlined here range from generally applicable to prior-specific MCMC algorithms. Each method also has its own means for instantiating parameters; these means are also summarized at the end of this chapter.

4.1 Reversible Jump

Inference for model averaging can be performed using Reversible Jump MCMC (RJMCMC) [19], in which a random walk along M is performed by proposals to higher or lower dimensional representations. This method is generally applicable but requires careful construction of a proposal distribution in order to be effective.

Let the target posterior for models m_k with parameters $\theta^{(k)}$ be $\pi(\theta^{(k)}, m_k)$. Reversible Jump operates by drawing auxiliary random variables so as to match the dimensions of the current and proposed models. Specifically, we propose to jump to model l from k with probability q_{kl}. A proposal is constructed by taking a draw $u_{kl} \sim \rho_{kl}$ and applying an invertible transformation T so that $(\theta^{(l)}, v_{lk}) = T_{kl}(\theta^{(k)}, u_{kl})$, where $T_{lk} = T_{kl}^{-1}$. u and v are chosen so that the dimensions of the augmented spaces are "matched." We then accept $\theta^{(l)}$ with probability

$$\alpha = \min\left(1, \frac{\pi(\theta^{(l)}, m_l)}{\pi(\theta^{(k)}, m_k)} \frac{q_{kl}}{q_{lk}} \frac{\rho_{kl}(u_{kl})}{\rho_{lk}(v_{kl})} \left| \frac{\partial T_{kl}(\theta^{(k)}, u_{kl})}{\partial(\theta^{(l)}, v_{lk})} \right| \right) \tag{4.1}$$

This provides a general framework for sampling in model averaging, but the proposal ρ_{kl} and the transformation T must be tailored for the problem at hand. For example, in split-merge RJMCMC for variable dimension Gaussian Mixture Model settings, T is a transformation that takes one cluster's parameters, and splits it into two clusters randomly depending on the value of u [20, 29, 30]. The reverse move corresponds to merging two clusters randomly.

4.2 Gibbs Inference for BNP Models

Bayesian nonparametric models frequently allow for inference schemes in which each data point is visited in sequence and assigned to some object (say a cluster), conditioned on all other variables. For conjugate models, this assignment step allows for the creation and destruction of "active" objects, giving a random walk on finite representations.

4.2.1 Example: Dirichlet Process Mixture Model

Consider the Dirichlet Process Mixture Model (DPMM), where we have marginalized out the mixing weights to obtain the following model defined in terms of a CRP:

$$\theta_k \sim H \tag{4.2}$$

$$z_i \sim CRP(\alpha) \tag{4.3}$$

$$x_i \sim F(\theta_{z_i}) \tag{4.4}$$

Thus we have $p(x, z, \theta) = p(\theta)p(z)\prod_i p(x_i|\theta_{z_i})$. If H is conjugate to F, then we can marginalize θ to get $p(x, z) = p(z)\prod_k \prod_{i|z_i=k} p(x_i|z_i = k)$. Thus we can evaluate $p(z_i = k, x, z_{\neg i}) \propto p(z_i = k|x, z_{\neg i})$ in order to sample z_i.

If the assignment step for z creates a new partition, then there is an implicit instantiation of a new parameter which is integrated out. However, if this parameter is needed explicitly, we may sample it from the posterior $p(\theta|x, z)$. As the model is assumed to be conjugate in this case, the posterior on θ takes on a closed form which may have readily available sampling algorithms.

4.3 Retrospective Sampling

In [47], the active dimension is sampled by first sampling a uniform variate, and second (retrospectively) sampling the dimension using the inverse CDF over all dimensions – because the chosen uniform variate will always correspond to a finite representation, this can be done tractably.

Consider the Dirichlet Process once again, where we generate the stick lengths w_i according to stick-breaking:

$$v_i \sim \text{Beta}(1, \alpha) \tag{4.5}$$

$$w_i = v_i \prod_{j=1}^{i-1}(1 - v_j) \tag{4.6}$$

If we wish to make a draw $z \sim \text{Categorical}(w)$, we can do this with the following scheme, even though the length of w is infinite:

1. Draw $u \sim \text{Uniform}(0, 1)$
2. Iterate through the sums $S_k = \sum_{i=1}^{k} w_i$, checking for the first k such that $S_k > u$
3. Take this k as the draw from Categorical(w)

This can be done in finite time almost surely. This method can be extended to posterior simulation as well. If we have a likelihood $p(X_j|\theta_i)$, then $p(z_j = i) \propto q_i = w_i p(X_j|\theta_i)$, then we need to sample from Categorical(q_i/c), where $c = \sum_{i=1}^{\infty} q_i$. The normalization constant c adds a complication to the simulation scheme used above, as we generally cannot compute $S_k = \sum_{i=1}^{k} q_i/c$ tractably.

The simplest way to handle this is to construct sequences $c_l(k) \uparrow c$ and $c_u(k) \downarrow c$, so that $c_l(k)$ and $c_u(k)$ depend only on w_i and $p(X_j|\theta_i)$ for $i \leq k$. For example,

$$c_l(k) = \sum_{i=1}^{k} w_i p(X_j|\theta_i) \tag{4.7}$$

$$c_u(k) = c_l(k) + M(1 - \sum_{i=1}^{k} w_i) \tag{4.8}$$

where $M > p(X_j|\theta_i)$ for all θ_i. Given these bounds, we can construct sums $L_{k,l} = \sum_{i=1}^{l} q_i/c_l(k)$

and $U_{k,l} = \sum_{i=1}^{l} q_i/c_u(k)$. After drawing $u \sim \text{Uniform}(0,1)$, we can take $z_j = l$ if for $l \leq k$

$$L_{k,l-1} \leq u \leq U_{k,l} \tag{4.9}$$

This works because $L_{k,l-1} \geq \sum_{i=1}^{l-1} q_i/c$ and $U_{k,l} \leq \sum_{i=1}^{l} q_i/c$, thus we are guaranteed that $S_{l-1} \leq u \leq S_l$.

4.4 Slice Sampling for the Dirichlet Process

Slice sampling for the Dirichlet Process [57] introduces auxiliary uniform variates that allows for efficient inference of BNP models with stick-breaking representations. These uniform variates are inspired from [9], where N auxiliary variables are introduced in the context of a generic Bayesian model, so that:

$$p(\theta, u_1, \ldots u_N) \propto p(\theta) \prod_{i=1}^{N} \mathbb{1}\left(u_i < p(X|\theta)\right) \tag{4.10}$$

where $\mathbb{1}$ is the indicator function. This is distinct from the slice sampling in [44], where

$$p(\theta, u) \propto \mathbb{1}\left(u < p(\theta) \prod_{i=1}^{N} p(X|\theta)\right) \tag{4.11}$$

Consider again the DPMM (3.24). Slice sampling for the DP introduces uniform variates so that:

$$p(\theta, u_1, \ldots u_N) \propto \sum_{k=1}^{\infty} \mathbb{1}(u_i < w_k) \prod_{i=1}^{N} p(X_i|\theta_k) \tag{4.12}$$

Giving the conditional distributions of u_i:

$$u_i|w_k \sim \sum_{k=1}^{\infty} w_k \text{Uniform}(0, w_k) \tag{4.13}$$

$$u_i|z_i, w_k \sim \text{Uniform}(0, w_{z_i}) \tag{4.14}$$

Conditioning on u allows for tractable inference for both the weights w_k and the allocations variables z_i. The conditional for the sticks v_i is a truncated Beta:

$$p(v_k|v_{\neg k}, -) = \text{Beta}(1, \alpha)\mathbb{1}(a_k < v_k < b_k) \tag{4.15}$$

$$a_k = \max_{i|z_i=k} \left\{ \frac{u_k}{\prod_{l<i}(1-v_l)} \right\} \tag{4.16}$$

$$b_k = 1 - \max_{i|z_i>k} \left\{ \frac{u_k}{v_{z_i} \prod_{l<z_i, l\neq i}(1-v_l)} \right\} \tag{4.17}$$

The conditional for the allocations z_i is the truncated likelihood of X_i:

$$p(z_i = k|-) \propto \mathbb{1}(u_i > w_k) p(X_i|\theta_k) \tag{4.18}$$

Because we are conditioning on the slice variable u_i, the set of k for which $p(z_i = k|-)$ is nonzero is almost surely finite. In this step, we need to draw w_k for clusters that have no data until $\mathbb{1}(u_i > w_k)$ can no longer be satisfied. For these empty clusters, the associated θ_k are drawn from the prior, which then allows an update to z_i using (4.18).

4.5 Slice Sampling for the Indian Buffet Process

Slice sampling for the IBP [55] introduces the variable u that determines a set of μ (in the stick-breaking representation (3.30)) that are "active" – u is a lower bound to the μ_k that are explicitly represented. The joint distribution with u is given by

$$p(Z, X, \mu, u) = p(Z, X, \mu)\frac{1}{\mu^*}\mathbb{1}(0 \leq u \leq \mu^*) \tag{4.19}$$

where

$$\mu^* = \min\left(1, \min_{k|\exists i, Z_{ik}=1} \mu_k\right) \tag{4.20}$$

is the minimum stick length across all features for which at least one datapoint is assigned. The conditional distributions of u and Z are then

$$u \sim \text{Uniform}(0, \mu^*) \tag{4.21}$$

$$p(Z|X, \mu, u) \propto p(Z|X, \mu)\frac{1}{\mu^*}\mathbb{1}(0 \leq u \leq \mu^*) \tag{4.22}$$

The conditional for Z is thus simply the conditional on X and μ, but with all columns with $\mu_k < u$ forced to zero, so only a finite number of columns need to be considered for updating. There is a complication to the update to u, as any columns with $\mu_k > u$ can have nonzero entries. Thus, we need to draw μ_k for empty columns of Z until it is guaranteed that further columns would have $\mu_k < u$. Thus the conditional distribution of $\mu_k|\mu_{k-1}, Z_{\cdot,k:\infty} = 0$ is needed:

$$p(\mu_k|\mu_{k-1}, Z_{\cdot,k:\infty} = 0) \propto \exp\left(\alpha\sum_{i=1}^{N}\frac{1}{i}(1-\mu_k)^i\right)\mu_k^{\alpha-1}(1-\mu_k)^N\mathbb{1}(0 \leq \mu_k \leq \mu_{k-1}) \tag{4.23}$$

[55] updates μ_k using adaptive rejection sampling, as $p(\mu_k|-)$ is log-concave in $\log\mu_k$. As for updating μ during its turn in the Gibbs sequence, the implicit ordering of the active μ can be dropped and μ_k drawn conditioned on $Z_{:,k}$

$$\mu_k | Z \sim \text{Beta}(n_k, 1 + N - n_k) \tag{4.24}$$

where $n_k = \sum_{i=1}^{N} Z_{i,k}$. Inactive μ_k can still be updated as in (4.23).

4.6 Approximate Methods

There are also methods that allow for approximate sampling. In some BNP settings, infinite objects are simply truncated to give approximate sampling methods [28], for example we may approximate the DP with Dirichlet$(\alpha\mathbf{1}^{(K)}/K)$, where K is large and $\mathbf{1}^{(K)}$ is a vector of ones of length K. Alternatively, sequential Monte Carlo may be used to approximately sample from the infinite posterior [21].

4.7 Summary

There are many available options for performing inference for model averaging, each with its merits and drawbacks. Reversible Jump MCMC is extremely general, but it can require considerable work in constructing a suitable proposal distribution. Gibbs inference, on the other hand, is straightforward to implement, but it requires that the prior and likelihood are conjugate, limiting its applicability. Retrospective Sampling and the Slice Sampling variants strike a balance between these extremes, being fairly easy to implement and applicable to BNP models having a stick-breaking representation.

Each of these methods deals with parameter instantiation in a different way. Reversible Jump handles this problem directly; the proposal distribution is responsible for handling such issues. In the conjugate case, the parameters can be integrated out, the model sampled, and the parameter drawn directly from the posterior if the newly sampled model requires it. The samplers making use of stick-breaking representations draw some set of new parameters from the prior, without data assigned to the corresponding components. If data are eventually assigned to these new components, the associated parameters can then be updated.

However, none of these algorithms leverage the impressive set of algorithms available for finite dimensional inference to the problem of parameter instantiation. Retrospective Jump Sampling does just that; newly instantiated parameters are given a chance to update conditioned on the data *before* a model is taken for the next MCMC iterate[1].

[1]The split-merge sampler also benefits from this type of "exploration step." However, this is done through a RJMCMC proposal that is typically specified on a per-model basis.

Chapter 5

Refractive Sampling

As mentioned in the introduction, Retrospective Jump Sampling depends on a "black-box" sampler suitable for sampling from finite dimensional distributions, such as Hamiltonian Monte Carlo (HMC). However, in practice HMC's hyperparameters are sensitive to the dimension of the target distribution (and sensitive in general), and so complicates its use as a black box sampler for Retrospective Jump. Here we develop Refractive Sampling, a more robust gradient-informed black-box sampler for finite dimensional problems.

5.1 Introduction

Markov Chain Monte Carlo (MCMC) is an effective tool for performing Bayesian inference. Frequently, a practitioner must design and implement a MCMC algorithm that is suited to his or her problem, which can be time consuming and error-prone. Tools that allow the general application of MCMC to wide varieties of models are thus attractive. State-of-the-art black-box samplers such as slice sampling [44] and Hamiltonian Monte Carlo (HMC) [10, 45] that require only a log-density (and perhaps its gradient) as input are hence popular tools for

Bayesian modelling. Even so, there are some drawbacks: slice sampling does not generalize well to problems in high dimensions, and HMC has some associated hyperparameters which can be difficult to tune. In this chapter we propose refractive sampling, a new black-box sampler that makes effective use of gradient information while remaining easy to tune in complex settings.

Many black-box MCMC algorithms, such as HMC and Reflective Slice Sampling [44] introduce an auxiliary variable $p \sim \mathcal{N}(0, I)$, and propose updates to the state x with target log-density $f(x)$ as follows:

$$p' = T(x, p) \tag{5.1}$$

$$x' = x + wp' \tag{5.2}$$

with the step-size w a parameter of the inference algorithm, and T some transformation on x and p. In Reflective Slice Sampling, we sample a slice variable $s \sim \text{Uniform}(0, f(x))$, and take T as:

$$T(x, p) = \begin{cases} p & \text{if } f(x^{(1/2)}) > s \\ p - 2g(x)\frac{p^T g(x)}{||g(x)||^2} & \text{otherwise} \end{cases} \tag{5.3}$$

where $x^{(1/2)} = x + wp$ and where $g(x) = \nabla_x f(x)$. Proposals from reflective slice sampling are always accepted, provided the reverse reflection would have occurred as well. Reflective slice sampling can be inefficient relative to other algorithms making use of the gradient; the gradient is used only to reflect p near the slice boundary, so it does not strongly influence the chain to find (or escape) regions of high probability.

HMC algorithms operate by performing moves that leave approximately invariant the Hamiltonian:

$$H = -f(x) + \frac{1}{2}p^T M^{-1} p \tag{5.4}$$

Hamilton's equations define a differential equation on x and p:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial H}{\partial p} = M^{-1}p \tag{5.5}$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial H}{\partial x} = g(x) \tag{5.6}$$

Thus, in order to preserve H, an update to p proportional to $g(x)$ should be met with an update to x proportional to $M^{-1}p$ – the leapfrog integrator does this in a reversible way. For HMC using the leapfrog integrator:

$$T(x, p) = p + \frac{\varepsilon}{2}g(x) \tag{5.7}$$

In this case, an additional update to p is made after updating x in order to obtain a reversible procedure. Updates to the state x and momentum $p \sim N(0, M)$ are then:

$$p \leftarrow p + \frac{\varepsilon}{2}g(x) \tag{5.8}$$

$$x \leftarrow x + \varepsilon M^{-1}p \tag{5.9}$$

$$p \leftarrow p + \frac{\varepsilon}{2}g(x) \tag{5.10}$$

This update corresponds to one leapfrog step; multiple steps may be chained together before the accept/reject step with acceptance probability:

$$\alpha(x, p \to x', p') = \min\left(1, \frac{\exp(f(x') - p'^T M^{-1}p'/2)}{\exp(f(x) - p^T M^{-1}p/2)}\right) \tag{5.11}$$

In order to preserve H well, the step size ε must be small enough that the error of discretizing (5.5) and (5.6) is not too large. Typically, there is a narrow range of ε that will produce reasonable acceptance rates while producing large steps; intuitively, we must have ε inversely proportional to $||g(x)||$. If ε is not chosen carefully, HMC will exhibit either low acceptance rates or very small updates per iteration. This problem becomes evident when the gradients become large, particularly when the curvature of f is also extreme.

In many applications, updates are performed in a Gibbs sampling style, where different parameter sets are updated in turn, often because each set requires different hyperparameter settings, or one set has closed form updates. In cases where one set of parameters has more degrees of freedom or is more flexible than another, the more flexible parameters can update *too quickly* and take the chain into a mode that fits the data poorly. For example, if one were sampling the means and covariances of a Gaussian Mixture Model using a black-box sampler, allowing the covariances to update too quickly can result in chains where a few large components (poorly) explain all the data. Because HMC has a narrow range of hyperparameter settings that allow for efficient sampling, it can be difficult to tune multiple HMC algorithms so that some update more slowly than others while still giving efficient sampling.

Finally, HMC can perform poorly in multimodal settings. As the momentum update (5.7) is proportional to the gradient, it is unlikely that updates to x that do not follow large gradients will be proposed. Even though HMC is a valid MCMC sampler, it can be myopic in that it tends to focus on the mode in which the current state happens to reside, resulting in poor mixing.

Techniques such as updates in alternate geometries [18], allowing an intelligent or automatically tuned number of steps [27, 58], and automatically tuning ε [49, 27, 58] are thus popular for their ability to improve acceptance rates and allow larger steps. Even so, these extensions still rely on an underlying sampler that can be sensitive to markedly fluctuating gradients.

Thus, MCMC algorithms that instead use only the gradient direction, and not its magnitude, may be able to escape these problems.

5.2 Refractive Sampling

We wish to construct a MCMC proposal scheme that makes stronger use of the normalized gradient than reflective slice sampling. We would also like it to be easy to tune and not sensitive to the peculiarities of the target distribution – therefore we still desire a proposal that preserves the norm of p rather than allowing the size of steps to grow or shrink with each step taken. One obvious choice is then to add the normalized gradient $\frac{g(x)}{||g(x)||}$ to p, and then normalize to keep the norm preserved.

Consider again the update of the form (5.1). The Jacobian determinant of the joint transformation is:

$$\begin{vmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial p} \\ \frac{\partial p'}{\partial x} & \frac{\partial p'}{\partial p} \end{vmatrix} = \begin{vmatrix} I + w\frac{\partial p'}{\partial x} & w\frac{\partial p'}{\partial p} \\ \frac{\partial p'}{\partial x} & \frac{\partial p'}{\partial p} \end{vmatrix} = \left|\frac{\partial p'}{\partial p}\right| \tag{5.12}$$

where we have used the following determinant identity for block matrices: $\begin{vmatrix} A & B \\ C & D \end{vmatrix} = |D| \cdot |A - BD^{-1}C|$ for invertible D. In Reflective Slice Sampling and HMC, $\left|\frac{\partial p'}{\partial p}\right| = 1$. This is not a required feature of a MCMC sampler, however, as any proposal with nonzero Jacobian can be corrected with an accept/reject step.

Consider the following update for p:

$$p' = T(x,p) = \frac{w_1 p + w_2 v}{||w_1 p + w_2 v||}||p|| \tag{5.13}$$

where $v = \frac{g(x)}{||g(x)||}$ and w_1 and w_2 are positive. This add-then-normalize proposal might be used for gradient informed MCMC algorithms that are less sensitive to gradient magnitudes. In order to construct valid MCMC moves using this proposal, however, it must be reversible:

$$p = T(x, -p') = \frac{-w_1 p' + w_2 v}{|| - w_1 p' + w_2 v||}||p'|| \tag{5.14}$$

(5.13) and (5.14) cannot both be satisfied with constant w_1 and w_2. Instead, we may let them depend on p and x. An interesting class of solutions for (5.13) and (5.14) are those in which w_i behaves differently depending on the sign of $p^T v$. If we can choose $w_1(p^T v)$ and $w_2(p^T v)$ so as to ensure that

$$\operatorname{sgn}(p^T v) = -\operatorname{sgn}(-{p'}^T v) \tag{5.15}$$

while satisfying (5.13) and (5.14), then the proposal is reversible. One solution is to define a positive constants r_l and r_h such that

$$r_l = \begin{cases} w_1||p|| & \text{if } p^T v > 0 \\ ||w_1 p + w_2 v|| & \text{otherwise} \end{cases} \tag{5.16}$$

$$r_h = \begin{cases} ||w_1 p + w_2 v|| & \text{if } p^T v > 0 \\ w_1||p|| & \text{otherwise} \end{cases} \tag{5.17}$$

There are multiple solutions for w_2, the one that satisfies (5.15) gives

$$p' = \frac{r_1}{r_2} p - ||p|| \left[\frac{r_1}{r_2}\cos\theta_1 - \cos\theta_2\right] u \tag{5.18}$$

where

$$r_1 = \begin{cases} r_l & \text{if } p^T v > 0 \\ r_h & \text{otherwise} \end{cases} \tag{5.19}$$

$$r_2 = \begin{cases} r_h & \text{if } p^T v > 0 \\ r_l & \text{otherwise} \end{cases} \tag{5.20}$$

$$u = \mathrm{sgn}(p^T v) v \tag{5.21}$$

$$\cos\theta_1 = \frac{p^T u}{||p||} \tag{5.22}$$

$$\cos\theta_2 = \frac{p'^T u}{||p'||} = \left[1 - \frac{r_1^2}{r_2^2}\left(1 - \cos^2\theta_1\right)\right]^{\frac{1}{2}} \tag{5.23}$$

It can easily confirmed that (5.18) satisfies (5.13) and (5.14).

The above transformation to p is *refraction*. Refraction occurs according to Snell's Law as follows: if a ray of light p travelling in a medium with index of refraction r_1 passes through a boundary to another medium with index of refraction r_2, then the ray refracts to a ray p'. If the boundary has surface unit normal u (defined so that $p^T u > 0$), then the angle of incidence θ_1 and angle of refraction θ_2 are determined by (5.22) and (5.23), respectively. The refracted ray p' can then be constructed as in (5.18).

Refractive Sampling makes use of this update for $T(x, p)$. We let $r_h > r_l$ so that p is refracted into a higher index of refraction from a lower one if $p^T g(x) > 0$, and vice versa if $p^T g(x) < 0$. That is, the gradient will always be pointing into the side with higher index of refraction, so that p will always be rotated towards the gradient.

Thus we can define $T(x,p)$ as follows:

$$
\begin{aligned}
(u, r_1, r_2) &= \begin{cases} (\frac{g(x)}{||g(x)||}, 1, r) & \text{if } p^T g(x) > 0 \\ (-\frac{g(x)}{||g(x)||}, r, 1) & \text{otherwise} \end{cases} \\
\cos\theta_1 &= \frac{p^T u}{||p||} \\
\cos\theta_2 &= \left[1 - \frac{r_1^2}{r_2^2}\left(1 - \cos^2\theta_1\right)\right]^{\frac{1}{2}} \\
T(x,p) &= \frac{r_1}{r_2} p - ||p|| \left[\frac{r_1}{r_2}\cos\theta_1 - \cos\theta_2\right] u
\end{aligned}
\tag{5.24}
$$

where r is a parameter of the procedure defining the ratio between the indices of refraction r_1 and r_2. This transformation is illustrated in Figure 5.1. The Jacobian of this transformation is

$$
\begin{aligned}
&\left|\frac{\partial T(x,p)}{\partial p}\right| = \\
&\quad \det\left(\frac{r_1}{r_2} I + \cos\theta_2 \left[1 - \left(\frac{r_1\cos\theta_1}{r_2\cos\theta_2}\right)^2\right] \frac{pu^T}{||p||} - \right. \\
&\qquad \left. \frac{r_1}{r_2}\left[1 - \frac{r_1\cos\theta_1}{r_2\cos\theta_2}\right] uu^T\right) = \\
&\qquad \left(\frac{r_1}{r_2}\right)^{d-1} \frac{\cos\theta_1}{\cos\theta_2}
\end{aligned}
\tag{5.25}
$$

where d is the dimension of p.

Note that $\cos^2\theta_2$ can be negative[1]; this occurs when moving from a medium of higher index of refraction to lower, and the angle of incidence is too shallow. In this case, the reverse

[1] Giving a complex $\cos\theta_2$

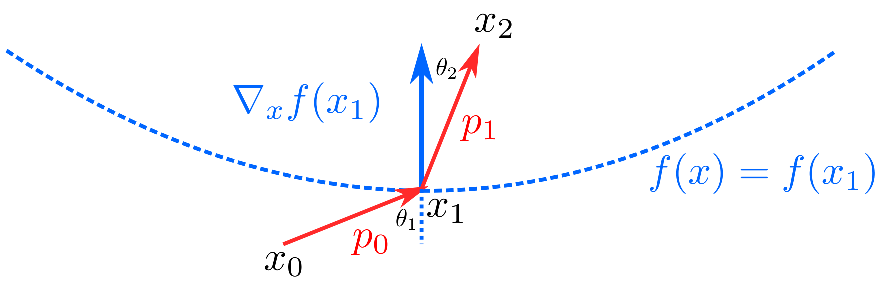

Figure 5.1: One step of a refractive sampling proposal. From state (x_0, p_0) we arrive to x_1. p_0 is then refracted through a surface with normal $\nabla_x f(x_1)$ to produce a p_1 that has been rotated in towards the gradient. Going in reverse with $p = -p_1$ would result in $p' = -p_0$.

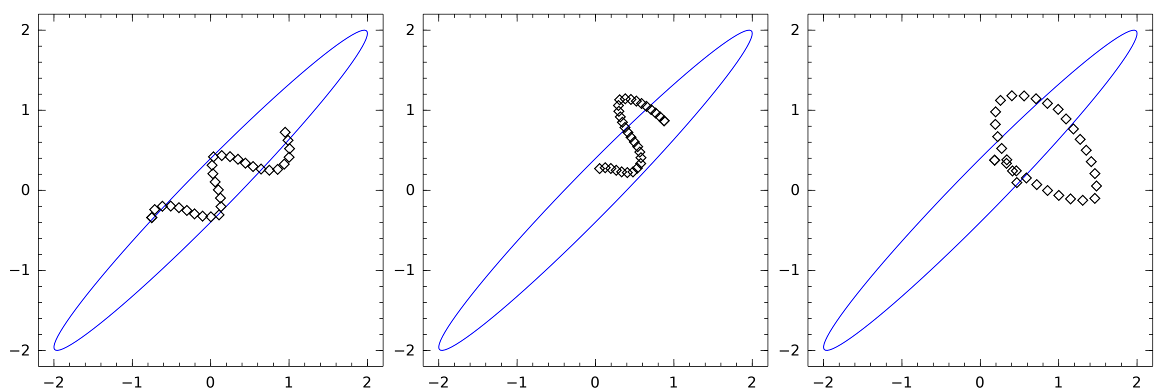

Figure 5.2: Example trajectories of refractive sampling whose final states were accepted. When the sampler repeatedly encounters gradients pointing in the opposite direction to p, it changes course until the angle of incidence to the gradient tangent plane is too shallow and p is reflected, giving a serpentine behavior reminiscent of HMC.

move is impossible, and so we instead reflect[2]. Thus

$$T(x,p) = \begin{cases} \frac{r_1}{r_2}p - ||p|| \left[\frac{r_1}{r_2}\cos\theta_1 - \cos\theta_2\right] u & \text{if } \cos^2\theta_2 > 0 \\ p - 2(p^T u)u & \text{otherwise} \end{cases} \tag{5.26}$$

It is necessary to use a "leapfrog" style algorithm for a reversible proposal, so that an update to p is always done at both the initial state x_0 and the final state x_1:

$$\begin{aligned} p_1 &= T(x_0, p_0) \\ x_1 &= x_0 + wp_1 \\ p_2 &= T(x_1, p_1) \end{aligned} \tag{5.27}$$

This can be repeated m times, where only one intermittent update to p is needed between updates to x.

After performing the above proposal, we determine whether to accept or reject. Let $y = (x, p)$ and let $S(y)$ be the full transformation in (5.27), with m updates to x. We need to choose acceptance probabilities α to satisfy detailed balance:

$$\pi(y)\alpha(y \to S(y)) = \pi(S(y)) \left|\frac{\partial S(y)}{\partial y}\right| \alpha(S(y) \to y) \tag{5.28}$$

Where π is the target density. Taking $y' = S(y)$, we have

$$\alpha(y \to y') = \min\left[1, \frac{\pi(x')}{\pi(x)} \prod_{i=0}^{m} \alpha_i\right] \tag{5.29}$$

[2]Incidentally, this is what occurs in nature as "total internal reflection"

where

$$\alpha_i = \left| \frac{\partial T(x,p)}{\partial p} \right|_{(x,p)=(x_i,p_i)} \tag{5.30}$$

If, for update i, $\cos^2\theta_2 > 0$ and thus refraction was performed, then α_i is of the form (5.25). Otherwise the transformation was reflection and $\alpha_i = 1$. There is no dependence on $\pi(p)$ as we still take $p \sim \mathcal{N}(0, I)$ or some other symmetric distribution. $||p||$ is preserved throughout all updates, thus the terms involving $\pi(p)$ and $\pi(p')$ cancel.

Example trajectories of this sampler are given in Figure 5.2. One issue that might alert the reader is the Jacobian (5.25) depends on $(r_1/r_2)^{d-1}$, and thus may suffer from the curse of dimensionality. We have found that this is not so dire an issue; as the dimension d increases we expect the ratio $\frac{\pi(x')}{\pi(x)}$ to grow/shrink with exponent d as well. See also the experimental evaluation regarding dimensionality, Section 5.4.2.

It is worth noting that the notion of a medium with a static index of refraction does not apply here: the indices of refraction used to refract p are determined entirely by the local gradient and its inner product with p. We are not attaching indices of refraction to various regions of parameter space, and using such a scaffold to propose updates. Rather, the index of refraction associated with one region may be different from iteration to iteration. See Algorithm 4 for pseudocode.

5.2.1 Ergodicity

We have already shown that Refractive Sampling follows detailed balance, thus any specified target distribution will be invariant to the Markov chain produced by Refractive Sampling. It remains to be shown that this invariant distribution is unique and the Markov chain will converge to it; that is, we need to show that Refractive Sampling is ergodic.

Algorithm 4 Pseudocode for Refractive Sampling

Input: x_0, $f(x)$, $g(x)$, m, w, r
$x \leftarrow x_0$
$p \sim \mathcal{N}(0, I)$
$\alpha \leftarrow 1$
for $i = 1 : m + 1$ **do**
  if $p^T g(x) > 0$ **then**
    $u \leftarrow \frac{g(x)}{||g(x)||}$
    $(r_1, r_2) \leftarrow (1, r)$
  else
    $u \leftarrow -\frac{g(x)}{||g(x)||}$
    $(r_1, r_2) \leftarrow (r, 1)$
  end if
  $\cos\theta_1 \leftarrow \frac{p^T u}{||p||}$
  $\cos^2\theta_2 \leftarrow 1 - \frac{r_1^2}{r_2^2}(1 - \cos^2\theta_1)$
  if $\cos^2\theta_2 < 0$ **then**
    $p \leftarrow p - 2(p^T u)u$
  else
    $p \leftarrow \frac{r_1}{r_2} p - ||p|| \left[\frac{r_1}{r_2}\cos\theta_1 - \cos\theta_2\right] u$
    $\alpha \leftarrow \left(\frac{r_1}{r_2}\right)^{d-1} \frac{\cos\theta_1}{\cos\theta_2}\alpha$
  end if
  if $i \leq m$ **then**
    $x \leftarrow x + wp$
  end if
end for
$\alpha \leftarrow \frac{f(x)}{f(x_0)}\alpha$
$z \sim \text{Uniform}()$
if $z < \alpha$ **then**
  return x
else
  return x_0
end if

We begin with irreducibility. Recall a Markov chain is ψ-irreducible if for all sets A with $\psi(A) > 0$, and for all states x, there exists n such that

$$K^n(x, A) > 0 \tag{5.31}$$

As the momentum p is drawn independently every iteration, we need only show irreducibility for the state x.

The transition kernel is composed of two parts that depend on p and the normalized gradient $v = \frac{g(x)}{||g(x)||}$, the refractive case and the reflective case. If $p^T v > 0$, then we are in the refractive case, moving "upwards." Let θ^* be the resulting angle of refraction when the angle of incidence $\theta_1 = \frac{\pi}{2}$; θ^* is the largest possible angle of refraction. The update maps p to a p' with $p'^T v \geq \cos\theta^*$. If $p^T v < 0$, and $-p^T v \geq \cos\theta^*$, then we are in the refractive case moving "downwards." Otherwise, we reflect. As $||p'|| = ||p||$ and every initial p has a unique corresponding reverse move $-p'$, these operations together map the sphere with fixed norm $||p||$ to itself. Thus there is positive density for any orientation of p'. As $p \sim \mathcal{N}(0, I)$ and $x' = x + wp'$, it is easy to see that $K(x, A) > 0$ for $\pi(A) > 0$, $x' \in A$.

As the kernel follows detailed balance with invariant distribution π and is irreducible, it is a positive chain, and thus recurrent (see Proposition 6.36 of [50]). As the kernel includes an accept/reject step, it is aperiodic. Thus Refractive Sampling is ergodic.

5.3 Setting r

For high-dimension problems, it may be difficult to find parameter settings that give large acceptance rates. HMC has the property that as $\varepsilon \to 0$, the acceptance rate goes to 1; however this is not true for refractive sampling when $w \to 0$. Generally speaking, larger d will warrant a smaller r. By roughly matching the equilibrium term $\frac{\pi(x')}{\pi(x)}$ with the Jacobian terms in the acceptance probability, we can choose a setting of r that depends on the gradient that will give high acceptance rates when w is small.

For a high acceptance rate, we want the change in log-density of the posterior to be approximately the same as the log-Jacobian for each step. Thus, when w is small, we want to choose r such that

$$||g||w \approx \ln \alpha_i \tag{5.32}$$

$$||g||w \approx (d-1)\log r + \log\cos\theta_1 - \log\cos\theta_2 \tag{5.33}$$

Letting $r = 1 + x$ and taking first order Taylor expansions of $\log r$ and $\log\cos\theta_2$ around $x = 0$ gives

$$r = 1 + \frac{4||g||w\cos\theta_1^2}{1 + (4d-5)\cos\theta_1^2} \tag{5.34}$$

Allowing r to depend on x in this way does not violate reversibility, nor does it affect the functional form of the Jacobian, as it only directly affects the update to p'. This choice of r allows for a larger number of steps, which can be important for particularly difficult posteriors.

5.4 Evaluation

We compare Refractive Sampling, HMC and the No U-Turn Sampler (NUTS) [27] over several measures of sampler performance. In preliminary experiments on a Gaussian target distribution, we found Reflective Slice Slice sampling was about ten times slower per effective sample than the other algorithms; thus we do not compare to it in the following.

5.4.1 Bimodal Distribution

We begin with a simple demonstration on a two dimensional bimodal target distribution. We define the target distribution as an equal mixture of two Gaussians, with $\mu_1 = [1, 1]^T$ and $\mu_2 = [-1, -1]^T$. We take both covariance parameters to be Σ, with unit diagonal entries, and with the offdiagonal entries $\Sigma_{12} = \Sigma_{21} \leq 0$. As we decrease Σ_{12}, the target distribution becomes a pair of parallel elliptical Gaussians. In order to measure how well a sampler mixes between the two modes, we count the number of times the sampler crosses the line $x_1 = -x_2$, a larger number of crossings indicating better mixing between the two modes. For Refractive Sampling we set $w = 0.5$, $m = 4$, and $r = 1.3$, and for HMC we set $\varepsilon = 0.5$ and $L = 4$. We also tried HMC with a preconditioning matrices $M = \Sigma$ and $M = 10I$.[3] The stepsizes w and ε and numbers of steps m and L were chosen so that the sampler would not be likely to propose from one mode to the other in one step of size w (or ε), but it may be likely in m (or L) steps. We performed four trials for each setting of Σ_{12}, reporting the mean and standard deviation estimates of the number of crossings and acceptance rates.

As seen in Table 5.1, HMC and NUTS cross more often than Refractive Sampling for $\Sigma_{12} \in \{0.0, -0.5\}$, largely due to the higher acceptance rates. However, as Σ_{12} decreases, the number of crossings for the HMC based algorithms degrades quickly, so that Refractive Sampling crosses ten times more often when $\Sigma_{12} = -0.8$. We also tuned HMC to give an acceptance rate of about 60% for the $\Sigma_{12} = -0.8$ case, giving $\varepsilon = 0.8$ and a mild improvement. Using HMC with a preconditioner did not help. HMC using $M = \Sigma$, the natural choice for sampling from either of the modes independently of the other, still degrades quickly when Σ becomes elliptical. We found that increasing L to 20 or 50 did not improve performance commensurate with the increased computational cost.

[3] We found that for $\Sigma_{12} = -0.8$, the stepsize $\varepsilon = 0.5$ was too large for HMC with $M = \Sigma$, and for this experiment we used $\varepsilon = 0.25$.

This demonstrates the fundamental difference between Refractive Sampling and HMC-related samplers: as the target distribution becomes more peaked, the gradient becomes steeper, and HMC has a more difficult time leaving the mode it is currently exploring. Refractive Sampling, on the other hand, is able to jump between peaked modes more freely.

5.4.2 Sample Efficiency

There is an inherent trade off between a MCMC sampler's ability to explore a mode quickly and its ability to escape that mode; a sampler that spends too much time attempting to find alternative modes will have inferior sample efficiency. As such, we should not expect Refractive Sampling to outperform HMC or NUTS on metrics such as Effective Sample Size (ESS) for unimodal posteriors.

We compare sample efficiencies on Bayesian Logistic Regression applied to three benchmark datasets[4]. We mean-centered and whitened all datasets for evaluation. We compute ESS as estimated in [27]: a 50,000 iteration run of NUTS is used to estimate the posterior mean and variance for each parameter, and for each algorithm being evaluated, the ESS for estimators of the mean and central second moment for each parameter is estimated, and the minimum is reported. We tuned Refractive Sampling and HMC manually, trying m and L in $\{1, 2, 4, 8\}$ with various stepsizes in order to maximize ESS per second in preliminary runs. We found that $r = 1.3$ worked well across all datasets. We ran each algorithm for 10,000 iterations, discarding the first 5,000 as burn-in.

We repeated each evaluation for 8 trials and report the mean and standard deviations of the ESS and ESS per second in Table 5.2. On these problems Refractive Sampling is roughly 3-7 times less sample efficient than HMC and NUTS. This is not a prohibitively large difference.

[4]German Credit ($N = 1000$, $d = 24$), Pima Indians ($N = 768$, $d = 8$), and Statlog Heart ($N = 270$, $d = 7$) datasets, all available at the UCI Machine Learning Repository [4]

Table 5.1: Refractive Sampling Bimodal Mixing Example

Σ_{12}	0.0		-0.5		-0.8	
	Num. Cross	Accept Rate	Num. Cross	Accept Rate	Num. Cross	Accept Rate
Refractive	1002.5 ± 54.3	0.449 ± 0.003	760.5 ± 49.4	0.405 ± 0.006	527.0 ± 17.9	0.354 ± 0.003
HMC	2308.0 ± 48.1	0.977 ± 0.002	1163.5 ± 7.7	0.972 ± 0.003	64.3 ± 7.4	0.880 ± 0.004
HMC ($\varepsilon = 0.8$)	3550.5 ± 16.6	0.963 ± 0.001	1626.8 ± 37.1	0.882 ± 0.002	93.3 ± 2.9	0.652 ± 0.004
HMC ($M = \Sigma$)	2288.3 ± 10.2	0.976 ± 0.001	1469.2 ± 12.8	0.917 ± 0.004	78.3 ± 8.1	0.875 ± 0.002
NUTS	2229.0 ± 54.4	0.788 ± 0.008	804.5 ± 12.5	0.808 ± 0.008	44.5 ± 4.1	0.774 ± 0.003

Table 5.2: Refractive Sampling Sample Efficiency

	German Credit		Pima		Heart	
	ESS	ESS/sec.	ESS	ESS/sec.	ESS	ESS/sec.
Refractive	175.8 ± 51.4	4.7 ± 2.0	445.3 ± 44.0	31.4 ± 2.0	92.6 ± 38.7	10.2 ± 4.5
HMC	1140.8 ± 167.8	34.1 ± 8.1	1603.4 ± 155.6	116.7 ± 13.6	359.4 ± 93.9	42.5 ± 10.9
NUTS	623.5 ± 85.8	13.6 ± 4.3	1474.4 ± 207.8	99.6 ± 14.0	912.8 ± 244.0	61.0 ± 16.5

Table 5.3: Refractive Sampling Sample Efficiency – Synthetic Data

	d=20		d=100		d=400		d=1000	
	ESS	ESS/sec.	ESS	ESS/sec.	ESS	ESS/sec.	ESS	ESS/sec.
Refractive	142.0 ± 38.4	6.5 ± 2.0	419.3 ± 59.2	13.6 ± 5.6	119.8 ± 79.3	1.8 ± 1.2	1.5 ± 0.4	0.06 ± 0.01
HMC	503.6 ± 37.2	19.7 ± 1.8	1175.3 ± 112.1	32.7 ± 8.5	1068.2 ± 80.1	17.2 ± 7.8	2928.0 ± 614.8	16.6 ± 5.0
NUTS	1154.1 ± 132.4	112.9 ± 6.5	1517.2 ± 79.0	36.2 ± 4.1	1732.4 ± 102.5	13.1 ± 2.2	1219.5 ± 78.3	5.5 ± 1.1

Dependence on Dimension

To evaluate the behavior of Refractive Sampling in higher dimensions, we compare it against HMC and NUTS on a synthetic logistic regression problem. The data are generated from a mixture of two spherical Gaussians in $d \in \{20, 100, 400, 1000\}$ dimensions, which each component representing one class. We tuned HMC and Refractive Sampling manually, trying m and L in $\{1, 2, 3, 8, 16, 32\}$ with various stepsizes. We found that for these problems, setting r automatically as in Section 5.3 was beneficial as it allowed for larger choices of m.

Table 5.3 summarizes the results of this comparison. For $d = 400$, Refractive Sampling is 10 times less sample efficient than HMC or NUTS, which is not a prohibitively large difference. However, for $d = 1000$, the automatic setting for r requires a small w, and the sample efficiency for Refractive Sampling becomes quite poor[5].

It is unfortunate, yet unsurprising, that Refractive Sampling is not as sample efficient as HMC or NUTS on these simple problems. We reiterate that Refractive Sampling is designed for use in problems where HMC-based algorithms have difficulty; particularly those with pathologically steep gradients or multiple modes. However, sample efficiency remains important. Perhaps the easiest way to combine the sample efficiency of HMC-based sampling with the robustness of Refractive Sampling is to simply mix two such samplers together.

[5]Setting r manually does not improve performance much for this case.

5.4.3 Convergence Comparison

We compare refractive sampling to HMC and NUTS on how well they manage to find high-probability regions of parameter space while sampling. We compare Refractive Sampling, HMC, NUTS with a mixed sampler that randomly chooses either Refractive Sampling or HMC at each iteration – we call this Ref-HMC. We used HMC's initial ε for the initialization scheme in NUTS, all other tuning hyperparameters for NUTS are as in [27].

Gaussian Mixture Model

Gaussian Mixture Models (GMM), despite their apparent simplicity, can be difficult models for black box samplers. The covariance parameters are more flexible than the means, which manifests as modes in the posterior where a few components with large covariances explain the majority of the data, and the remaining clusters explain few points, if any. Specifically, we define our Bayesian GMM as:

$$
\begin{aligned}
w &\sim \text{Dirichlet}(\alpha) \\
\mu_k &\sim \mathcal{N}(0, \Sigma_\mu) \\
\Sigma_k &\sim \text{InvWishart}(\Sigma_0, \nu) \\
z_i &\sim \text{Categorical}(w) \\
X_i &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})
\end{aligned}
$$

We set $K = 5$, $\alpha = \mathbf{2}^{(K)}$ (that is, a K-vector of 2s), $\nu = 2d + 2$, and $\Sigma_\mu = \Sigma_0 = I$. We represent the precision matrix as a modified Cholesky decomposition $\Sigma^{-1} = AD^{-1}A$, where A is lower unitriangular and D is diagonal, with priors as in [8] so that $\Sigma_k^{-1} \sim \text{Wishart}(\Sigma_0, \nu)$. We marginalize out the z variables, leaving a representation for w, μ, L and D.

We compare refractive sampling to HMC and NUTS on inference of the means and variances on the Yeast dataset [4], with dimensions of small variance removed giving $d = 6$. As mixture models have modes with high posterior probability, but poor predictive performance (for example, a single cluster with small variance explaining a single datapoint), we split the data into train/test splits and also evaluate on the held-out data.

We report test log-likelihood in addition to model posterior probability. The test log-likelihood and posterior probabilities we report are not averaged over the chain to highlight the behavior of the chains themselves. Thus, for MCMC iterate θ_t, train set X and test set $X_{(\text{test})}$ we report $p(X_{(\text{test})}|\theta_t)$ and $p(X|\theta_t)p(\theta_t)$.

All algorithms began with the same train/test splits and initial states for a given trial. We ran all samplers for 1000 iterations, with NUTS taking about six times as long as refractive sampling and HMC.

Additionally, we show the result of MAP inference performed by optimization via L-BFGS [37]. The model above is not conjugate, hindering the application of a traditional Expectation Maximization (EM) algorithm. However, we marginalize out the z variables as above, thus this optimization procedure can be viewed as a generalized EM algorithm in which the E step is performed at every update to the parameters. We performed 100 random restarts and report the trial with the highest posterior probability.

In Figures 5.3 and 5.4, we plot the model probability and test log-likelihood versus iteration, averaged over trials (but not averaged over the chains). The error bars correspond to 1 standard deviation. NUTS and HMC do not reach the regions of parameter space with the same posterior probability or test log-likelihood as Refractive Sampling and Ref-HMC do. The MAP estimator finds a mode not found by any of the samplers above, however this mode does not correspond to higher test log-likelihood.

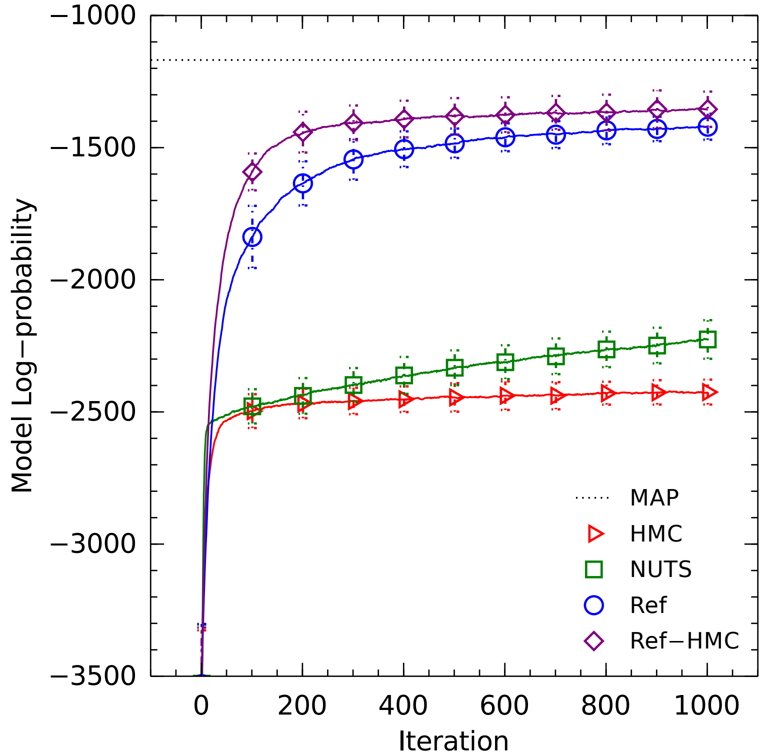

Figure 5.3: Model posterior log-probability for the Yeast dataset.

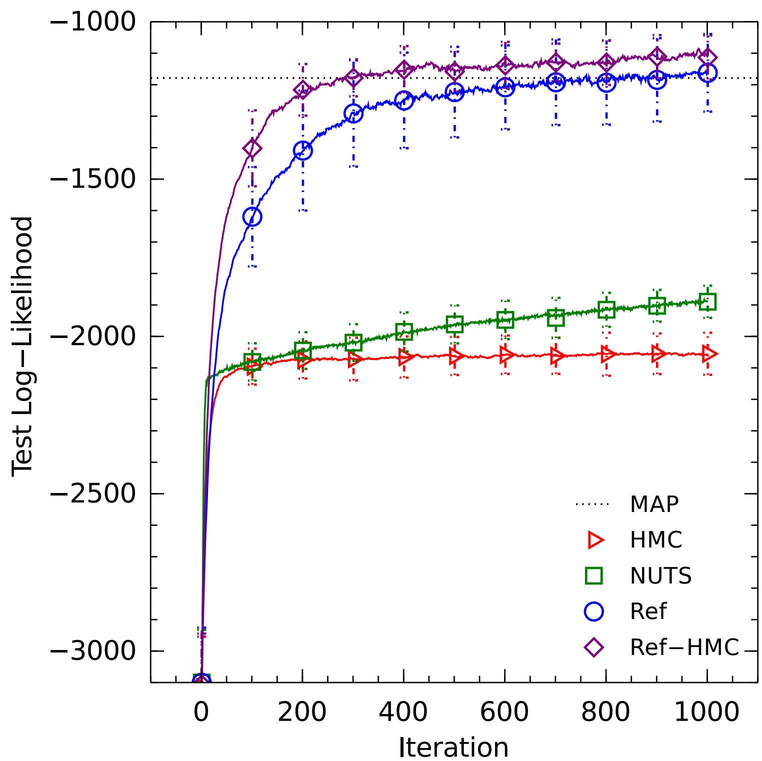

Figure 5.4: Test log-likelihood for the Yeast dataset.

Bayesian Softmax Regression

Finally, we compare all algorithms on Bayesian Softmax Regression (also known as multinomial logistic regression). The model for data $X \in \mathbb{R}^{N \times D}$, $Y \in \mathbb{Z}_C^N$, and $C > 1$ is:

$$\beta_{c,j} \sim \mathcal{N}(0, \sigma_\beta)$$
$$Y_i \sim \text{Categorical}(P(Y_i|\beta, X_i))$$
$$P(Y_i = c|\beta, X_i) = \frac{\exp(\beta_c^T X_i)}{\sum_{c'=1}^{C} \exp(\beta_{c'}^T X_i)}$$

where we set the parameters of the pivot class $\beta_C = \mathbf{0}$ as they are superfluous degrees of freedom. We apply the model to the St. Jude Leukemia dataset [60], a data set with $N = 327$ and $d > 10000$. Many of these dimensions are small-variance, so we preprocess the data using PCA to give $d = 140$, retaining about 90% of the data variance. The data are gene expression levels from 6 different diagnostic classes of leukemia, with a 7^{th} class denoting cases that were not assigned a diagnostic label. We treat each designation as its own class, including the 7^{th} "unlabeled" class, which we set as the pivot class.

Here we plot only the model probability, as there is little danger of extreme overfitting as is the case with the GMM. Again, we plot the model probability averaged over trials, but not over the chains, see Figure 5.5. Refractive Sampling and Ref-HMC reach regions of higher posterior probability more quickly than HMC and NUTS. The MAP estimate finds an even higher region of posterior probability, however this corresponds to an extremely narrow (and thus low probability) peak: in a separate experiment when Refractive Sampling was initialized to this mode it slowly escapes to a region of parameter space with the same probability as found with random initialization.

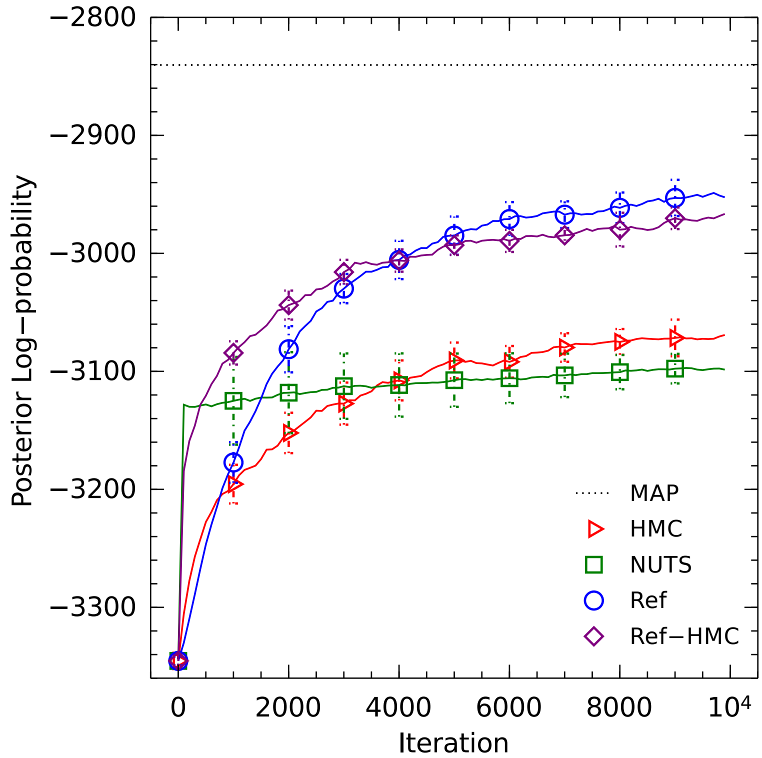

Figure 5.5: Model posterior log-probability for the Leukemia dataset.

5.5 Additional Remarks

There are some drawbacks to refractive sampling. Refractive Sampling does not fully leverage the gradient magnitude, and so performance can suffer where this information is useful. Highly elliptical problems – where an update with fixed norm may not be optimal – are another issue. We suggest standardizing or whitening data where possible so as to potentially reduce the severity of elliptical posteriors. In extremely high dimensions, Refractive Sampling may not be sample efficient.

There are possible improvements and variants of refractive sampling that have yet to be explored. One obvious direction is that any vector-valued function may be substituted for the gradient for use in the refraction transformation. Stochastic approximations of the gradient can be used while still providing a valid Markov Chain, and other schemes choosing directions other than that of steepest ascent may be fruitful. Many of the ideas in Riemannian geometry variants of HMC and Metropolis Adjusted Langevin Algorithm (MALA) can be applied to refractive sampling, as well as the automated/incorporated choice of the number of steps and stepsizes in [27] and [58].

5.6 Summary

Refractive Sampling is a Metropolis Hastings sampler which uses the normalized gradient to guide its proposals. It constructs proposals based on basic physical processes and is easy to implement. Refractive Sampling enjoys many of the benefits of other gradient-based samplers without the sensitivity to large fluctuations in gradients – in some settings, this enables refractive sampling to find regions of high probability more easily. As such, refractive sampling is less sensitive to initialization. Additionally, it can be used as a large-step sampler in conjunction with small-step samplers such as HMC in order improve overall sampler behavior.

Chapter 6

Retrospective Jump Sampling

6.1 Introduction

In this chapter, we introduce Retrospective Jump Sampling (RTJ), a general purpose sampler for model averaging inference tasks. There are several existing algorithms for MCMC inference on model averaging tasks (see Chapter 4), however not many are suitable for general purpose sampling. RTJ only requires as input the model log-density, a black box sampler suitable for MCMC sampling from arbitrary distributions in finite dimensions, and a few inference hyperparameters, making it a useful algorithm suitable for general purpose sampling frameworks.

For the sake of exposition, we will make use of a running example throughout. We consider RTJ applied to a Gaussian Mixture Model with a random number of components. Given data $X_i \in \mathbb{R}^d$:

$$
\begin{aligned}
K &\sim \text{Poisson}(\lambda) \\
w &\sim \text{Dirichlet}(\alpha \mathbf{1}^{(K)}) \\
\mu_k &\sim \mathcal{N}(0, \Sigma_\mu) \\
\Sigma_k &\sim \text{InvWishart}(\Psi, \nu) \\
z_i &\sim \text{Categorical}(w) \\
X_i &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})
\end{aligned}
\tag{6.1}
$$

where $\mathbf{1}^{(K)}$ is a vector of ones of length K. Here the model parameter is $\theta = (w, \mu, \Sigma, z)$, but in this case z can be integrated out and we take $\theta = (w, \mu, \Sigma)$. When speaking generically, we will refer to the K clusters as "objects." In the model determination context, let m_k correspond to the GMM with $K = k$, giving $P(M = m_k) = P(K = k)$.

6.2 Retrospective Jump

There are many "black box" samplers that can be used for finite dimensional inference problems that require little problem specific tuning. For example, Hamiltonian Monte Carlo (HMC) [45] performs well when a gradient is available, and slice sampling [44] is efficient in univariate settings or those in which the variables are not highly codependent. Neither of these algorithms require any special structure in the model in order to work reasonably well.

Unfortunately, sampling in the infinite dimensional setting is more complicated. Some methods are available for inference in infinite spaces. The most classic is Reversible Jump MCMC (RJMCMC) [19], in which a random walk along M is performed by proposals to higher or lower dimensional representations. This method is generally applicable but requires careful construction of a proposal distribution in order to be effective. In Bayesian nonparametrics,

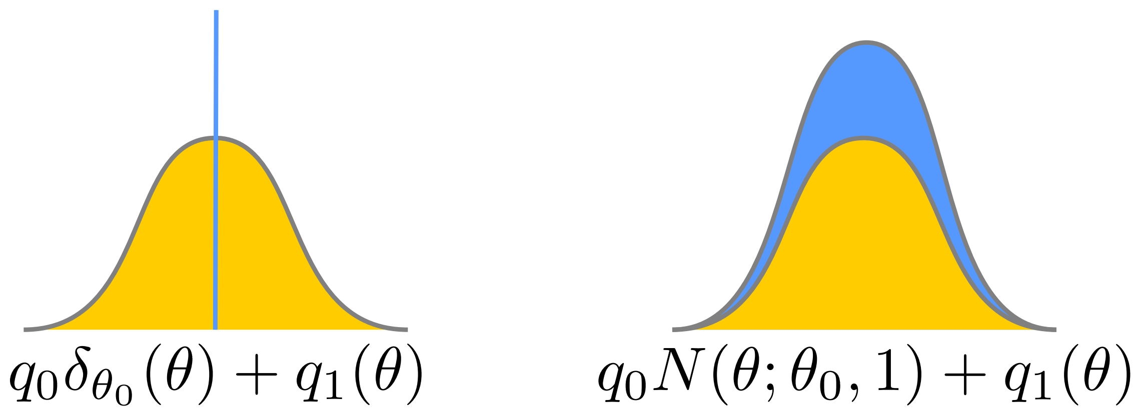

Figure 6.1: Retrospective sampling for mixed discrete and continuous distributions. Even though $q_0\delta_{\theta_0}(\theta) + q_1(\theta)$ is unnormalized, by representing the probability of the point mass q_0 as a weighted continuous pdf, we can create a single continuous distribution from which we can slice sample. Given a particular θ, we can then choose either the discrete or continuous component by sampling from the probability vector proportional to $(q_0 N(\theta;\theta_0, 1), q_1(\theta))$.

the models frequently lend themselves to inference schemes in which each data point is visited in sequence and assigned to some object (say a cluster). This assignment step allows for the creation and destruction of "active" objects, giving a random walk on finite representations. In [47] the active dimension is sampled by first sampling a uniform variate, and second (ie retrospectively) sampling the dimension using the inverse CDF over all dimensions – because the chosen uniform variate will always correspond to a finite representation, this can be done tractably. Slice sampling methods can be used in which an auxiliary slice variable is introduced which allows sampling the effective dimension in BNP models with stick-breaking representations [57]. There are also methods that allow for approximate sampling.

The above methods are useful tools but none are quite up to the task of being a black box sampler in which only the model densities need to be specified (along with some MCMC tuning parameters) in order to sample from an infinite dimensional model. Many of these algorithms, particularly those in which data are assigned to objects, can have problems with mixing as it requires a search over a combinatorial space with many local maxima. Methods such as split-merge [20, 29, 30] can address this for some specific problems, but this also is a solution that must be tailored to the model in order to be effective.

In an aim towards a black box sampler for infinite dimensional models, we introduce the Retrospective Jump Sampler. RTJ operates by introducing an auxiliary variable L, that, when conditioned upon, allows a random walk on M. RTJ rests upon two basic principles:

1. **(Augmentation)** A distribution of dimension d can be augmented to distribution $d + k$ by introducing k independent auxiliary variables, that, when marginalized out give back the original distribution of dimension d. This is useful when relating MCMC states of different dimensionality.

2. **(Retrospective Sampling)** Given a mixture q over k pdfs such that $q = \sum_{i=1}^{k} p_i(\theta)$, one can sample the state θ directly from the mixture, and subsequently sample $i|\theta$ from normalizing the probability vector $\{p_i(\theta)\}_{i=1}^{k}$. This is a slightly different definition of Retrospective Sampling as given in [47], in which the mixture q has infinitely many components.

We begin with an observation on how one might sample from a mixture of discrete and continuous (unnormalized) measures. The straightforward approach would be to integrate out the continuous component to give another discrete component, sample from the corresponding mixture, and then pick a sample based on the mixture component that was selected. This is costly in the general case as it involves integrating over the continuous component. A useful alternative is to instead *augment* the lower-dimensional, discrete components, into the continuous space, rather than integrate the continuous component into the discrete space. Then, we can sample directly from the mixture of continuous pdfs using methods such as slice sampling or HMC, and then *retrospectively* sample the mixture component. See Figure 6.1.

This augmentation/retrospective sampling scheme allows for samplers that can sample from mixtures of distributions of different dimensions. We apply this scheme to the problem of inference for model averaging in the following.

Consider a set of models $\mathcal{M} = \{m_k\}$, which may be finite or countably infinite – let M denote a random variable whose support is $\mathcal{M}$. Let each parameter θ_j occupy a space Ω_j, and let $\Omega = \prod_{j=1}^{\infty} \Omega_j$. Each model has a set of parameters $\theta^{(k)} \in \Omega^{(k)}$, where $\Omega^{(k)}$ is subspace of Ω. The particular subspace is determined by an index set of size[1] d_k: $I_k = \{i_{k1}, \dots i_{kd_k}\}$. Let $\Omega_I = \prod_{i \in I} \Omega_i$ denote the subspace defined by restricting Ω to the indices that are in I. The index sets of different models may be disjoint or they may overlap – that is, some parameters might be shared between models. Let $\theta_I = \{\theta_i | i \in I\}$, so that $\theta_{I_k} = \theta^{(k)}$. The observations X live in the dataspace $\mathcal{X}^N$. Let A be a *cylinder* with base $B \subseteq \Omega_I$ with $I = \{i_1, \dots i_d\}$ if

$$A = \{\omega \in \Omega | (\omega_{i_1}, \dots \omega_{i_d}) \in B\} \tag{6.2}$$

Let $A \subseteq \Omega$, $C \subseteq \mathcal{X}^N$, $D \subseteq \mathcal{M}$, and A_{B_k} be a cylinder with base $B_k \subseteq \Omega_{I_k}$. Define a probability measure $\mu(A, C, D)$ such that:

$$\mu(\Omega, \mathcal{X}^N, \{m_k\}) = P(M = m_k) \tag{6.3}$$

$$\begin{aligned} \mu(A_{B_k}, C, \{m_k\}) = & P(X \in C | \theta^{(k)} \in B_k, M = m_k) \\ & P(\theta^{(k)} \in B_k | M = m_k) P(M = m_k) \end{aligned} \tag{6.4}$$

where $P(M = m_k)$ is the prior probability of model m_k, $P(\theta^{(k)} \in B_k | M = m_k)$ is the prior measure for $\theta^{(k)}$, and $P(X \in C | \theta^{(k)} \in B_k, M = m_k)$ is the data generating measure. As it stands, μ is not fully defined; we need to specify how $\mu(A_B, C, \{m_k\})$ should behave when B is a base that is contained in $\Omega_{I_{(-k)}}$ with $I_{(-k)} \subseteq \mathbb{N} \setminus I_k$. This choice determines the probabilities of parameters in "augmented" dimensions. Let $\theta^{(-k)} = \theta_{I_{(-k)}} \in \Omega_{I_{(-k)}}$ be the

[1] Note that a particular index in an index set may actually correspond to multiple parameters, that is $\dim(\Omega_j) \geq 1$. In the GMM example, we may have each index i correspond to the mean, covariance, and mixing weight for a single Gaussian component.

parameters associated with these augmented dimensions. We have

$$\mu(A_B, C, \{m_k\}) = P(X \in C | \theta^{(k)} \in \Omega^{(k)}, M = m_k)$$
$$P(\theta^{(-k)} \in B | M = m_k) P(M = m_k) \tag{6.5}$$

where $P(\theta^{(-k)} \in B|M = m_k)$ is an arbitrary probability distribution on $\theta^{(-k)} \in \Omega_{I_{(-k)}}$.

We can take Radon-Nikodym derivatives of μ restricted to particular subspaces to get interesting density functions that are useful for inference. If we take $\lambda(B_k, C) = \mu(A_{B_k}, C, \{m_k\})$, and ν to be the Lebesgue measure, then

$$\frac{1}{P(M = m_k)} \frac{d\lambda}{d\nu} = p(X, \theta^{(k)} | M = m_k) \tag{6.6}$$

is the joint density of X and θ_k conditioned on model m_k. A bit more interesting is to choose a set of indices $I_{(-k)}$ that are disjoint with I_k, and to take $\lambda(B, C)$ with $B \subseteq \Omega_{I_a}$, and $I_a = I_k \cup I_{(-k)}$:

$$\frac{1}{P(M = m_k)} \frac{d\lambda}{d\nu} = p(X, \theta^{(k)} | M = m_k) p(\theta^{(-k)} | M = m_k) \tag{6.7}$$

where $p(\theta^{(-k)}|M = m_k)$ is the density function of $P(\theta^{(-k)} \in B|M = m_k)$. Ω_{I_a} acts as the "augmented" space in which we may sample parameters $\theta^{(k)}$ and $\theta^{(-k)}$ simultaneously.

Different sampling algorithms can be derived depending on whether the I_k are disjoint or overlapping. First we consider the disjoint case.

6.2.1 Disjoint RTJ

If $\mathcal{M}$ is finite, we can take $B \subseteq \Omega$, and $A_B = B$. Let all models m_i, $i \neq k$ share the same prior for $\theta^{(k)}$ as that for m_k, so we may write $p(\theta|M = m_k) = p(\theta)$.[2] If we take $\lambda(B, C) = \mu(B, C, \mathcal{M})$, then

$$p(\theta|X) \propto \frac{d\lambda}{d\nu} = p(\theta) \sum_k p(X|\theta^{(k)}, M = m_k) P(M = m_k) \tag{6.8}$$

From this we can derive

$$\begin{aligned} p(\theta^{(k)}|X) \propto & p(X, \theta^{(k)}|M = m_k) P(M = m_k) + \\ & p(\theta^{(k)}) \sum_{j \neq k} p(X|\theta^{(j)}, M = m_j) P(M = m_j) \end{aligned} \tag{6.9}$$

And note that

$$P(M = m_k|\theta, X) \propto p(X, \theta^{(k)}|M = m_k) P(M = m_k) \tag{6.10}$$

This lends itself to a very simple inference algorithm:

1. Update the model parameters for each model according to (6.9)
2. Sample M according to (6.10)

When $\mathcal{M}$ is infinite, we may still perform inference by introduction of an auxiliary variable $L \in \mathbb{N}_0$, defined so that conditioning on L restricts the chain to considering a finite set of models. We define $L|M$ in the following way. First, we draw $\delta \sim \text{Categorical}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) - 2$ so that $\delta \in \{-1, 0, 1\}$. If $M = m_k$, then we take $L = k + \delta$. Introducing L gives a measure on

[2] This choice of augmenting distribution is not too restrictive – it simply means that a parameter θ_i has the same prior regardless of the state of M, and is in fact equal to the prior $p(\theta_i|M = m_k)$ if θ_i is a parameter of m_k. Other choices are possible but we have found that simply using the prior works well.

$(\Omega, \mathcal{X}^N, \mathcal{M}, \mathbb{N}_0)$:

$$\mu(A, C, \mathcal{M}, \{L\}) = \frac{1}{3}\mu(A, C, \{m_{L-1}, m_L, m_{L+1}\}) \tag{6.11}$$

As $\mu(A, C, \{m_k\}) = 0$ with $k \leq 0$, it is easy to confirm that the measure in (6.11) is still a probability measure. As conditioning on L is equivalent to conditioning on $M \in \{m_{L-1}, m_L, m_{L+1}\}$, sampling and then conditioning upon L renders the infinite model determination problem into a finite model determination problem. We can thus apply the algorithm for a finite $\mathcal{M}$ to the infinite case by sampling and conditioning upon L. We can then sample $\theta|L$ using (6.9), but where only models in $\{m_{L-1}, m_L, m_{L+1}\}$ are considered, and finally sample $M|\theta, L$.

This algorithm for infinite $\mathcal{M}$ may be slow to mix: if the parameters for model m_k happen to be in a poor state relative to the other models, then $M = m_k$ will not be selected easily and the overall chain may have trouble transitioning "past" $M = m_k$. If the set of models $\mathcal{M}$ are all related models, then we may instead choose to share some parameters between them. Sharing parameters will tie the models together, reducing the prevalence of such complications.

6.2.2 Nested RTJ

When the models in $\mathcal{M}$ are related, it may be sensible to share parameters between models. For example, if m_k represents a Gaussian Mixture Model with k components, we may want to tie the parameters of $k-1$ clusters in m_k to the parameters of m_{k-1}. To do this we can set $I_{k-1} = \{1, \dots k-1\}$ and $I_k = \{1, \dots k\}$, so that the parameters of $k-1$ clusters are shared between m_{k-1} and m_k.

We restrict our attention to the case where the models m_k are *nested*, that is, $I_k \subseteq I_{k+1}$. We again assume the prior on θ does not depend on M: $p(\theta|M = m_k) = p(\theta)$

Consider again the finite $\mathcal{M}$ case. As the parameters are now tied between models, we cannot update each model's parameters independently of the other models. Taking $\theta \in \Omega$, $A \subseteq \Omega$, and $\lambda(A, C) = \mu(A, C, \mathcal{M})$, we have

$$p(\theta|X) \propto \frac{d\lambda}{d\nu} = p(\theta) \sum_k p(X|\theta^{(k)}, M = m_k) P(M = m_k) \tag{6.12}$$

and

$$P(M = m_k|\theta, X) \propto p(X|\theta^{(k)}, M = m_k) P(M = m_k) \tag{6.13}$$

Thus for the nested case, we can update the full θ with (6.12), and then sample M.

For the infinite $\mathcal{M}$ case, we can introduce the variable L that restricts the conditional measure to finite sets of models in the same manner as in Section 6.2.1. Let $\lambda(B, C) = \mu(A_B, C, \mathcal{M}, \{L\})$, with $B \subseteq \Omega_{I_{(L)}}$, $\theta_{I_{(L)}} \in \Omega_{I_{(L)}}$, and $I_{(L)} = I_{L-1} \cup I_L \cup I_{L+1}$. Then

$$p(\theta_{I_{(L)}}|X, L) \propto \frac{d\lambda}{d\nu} = p(\theta_{I_{(L)}}) \sum_{k=L-1}^{L+1} p(X|\theta^{(k)}, M = m_k) P(M = m_k) \tag{6.14}$$

and

$$P(M = m_k|\theta_{I_{(L)}}, X, L) \propto p(X|\theta^{(k)}, M = m_k) P(M = m_k) \tag{6.15}$$

for $k \in \{L - 1, L, L + 1\}$.

We now give an overview of the nested RTJ algorithm for infinite $\mathcal{M}$ with important implementation details. Let R be the set of indices associated with models m_k that have been visited in the chain so far. Thus the parameters $\theta_R \in \Omega_R$ are those that are explicitly represented. Begin at a state $(\theta_R, M = m_k)$. First, we sample $L|M$. Given L, we determine $\theta_{I_{(L)}} \in \Omega_{I_{(L)}}$. For $i \in I_k$, we keep the given value of θ_i. For $i \in R \cap I_{(L)} \setminus I_k$, we may keep θ_i or overwrite it with a draw from the prior $p(\theta_i)$. For $i \in I_{(L)} \setminus (R \cup I_k)$, we may initialize θ_i to a value arbitrarily as this is the first time θ_i is being represented. We then update $\theta_{I_{(L)}}$ according to (6.14) using a black box sampler for finite models initialized to $\theta_{I_{(L)}}$ to give a new parameter θ'. Finally, we sample a model M' conditioned on θ' using (6.15), and we have our new state θ', M'. Thus we have performed a valid set of MCMC steps that allowed the transition from $M = m_k$ to some other model. Furthermore, the sampling of $\theta'|L, \theta$ acts as an "exploration" step which seeks out (M', θ') pairs with high probability, much like split-merge sampling techniques. Pseudocode for this algorithm (with sampling from the prior rather than using previous values for $i \in I_{(L)} \setminus I_k$) is given in Algorithm 5. Note that in one complete Retrospective Jump step we may sample $k' \in \{k-2, \ldots k+2\}$, as, for example, we might first sample $L = k+1$, and then $k' = L + 1$.

Algorithm 5 Nested RTJ with infinite $\mathcal{M}$

```
Input: Model specifications M = {m_k | k ∈ ℕ}
Input: Sampler S_Ω ∈ ((Ω ↦ ℝ), Ω) ↦ Ω
Input: Initial model index k
Input: Initial parameters θ^(k)
Input: Index sets I = {I_k | k ∈ ℕ}
for j = 1 : iterations do
    δ ~ Categorical(1/3, 1/3, 1/3) − 2
    L ← k + δ
    I_(L) = I_{L−1} ∪ I_L ∪ I_{L+1}
    for i ∈ I_(L) \ I_k do
        θ_i ~ p(θ_i)
    end for
    θ_{I_(L)} ← S_{Ω_{I_(L)}}(p(θ_{I_(L)} | X, L), θ_{I_(L)})
    M ~ P(M | θ_{I_(L)}, X, L)
    k = k' such that M = m_{k'}
    Record (θ^(k), m_k)
end for
```

The idea of treating parameters of potentially new spaces as auxiliary variables is not new; see [43, 12]. However, RTJ is distinct in that these auxiliary variables can be updated using data, while still maintaining detailed balance, *before* the next model M' is sampled. This capability impacts the mixing efficiency of the algorithm. Split-merge algorithms [30] also allow the augmented space to be explored before sampling M', however it does so in a more directed and problem specific manner via construction of a proposal distribution for RJMCMC.

6.2.3 Making use of Exchangeability

The nested RTJ sampler for infinite $\mathcal{M}$ described above imposes an ordering on the dimensions i, and thus also on the parameters. When sampling $M' = m_{k-1}$, the parameters in $I_k \setminus I_{k-1}$ are "deactivated." For the nested case, this is somewhat arbitrary: if the parameters are exchangeable, why not allow any of the parameters in I_k to be removed instead? In this case, there is some flexibility as to the choice of index sets I_k, all of which define equivalent measures μ. We can treat these I_k as random variables that we should sample, which then affects the set of parameters that are removed when sampling $k' < k$. Note that the disjoint RTJ algorithm avoids this issue, as each model has its own set of parameters.

Constructions with $O(1)$ components

Let $n_k = |I_k|$, and let $I = \{I_k | k \in \mathbb{N}\}$. Given the requirement that $I_{k-1} \subseteq I_k$, there are $\binom{n_k}{n_{k-1}}$ ways to define I_{k-1} that respects I_k. Let $\mathcal{I}$ denote the set of all such consistent sets of index sets I. Extend the definition of μ to also depend on the index set: $\mu(A, C, D) = \mu(A, C, D|E)$, with $E \subseteq \mathcal{I}$. We may take the uniform measure on $\mathcal{I}$ to obtain a probability measure $\mu(A, C, D, E) = \mu(A, C, D|E)\mu(E)$. We may extend these definitions to include L as well.

Explicating the dependence on I gives the following conditionals for θ and M:

$$p(\theta|X, L, I) \propto p(\theta) \sum_{k=L-1}^{L+1} p(X|\theta^{(k)}, M = m_k, I)P(M = m_k) \tag{6.16}$$

and

$$P(M = m_k|\theta, X, L) \propto p(X|\theta^{(k)}, M = m_k, I)P(M = m_k) \tag{6.17}$$

where $k \in \{L-1, L, L+1\}$ and $\theta^{(k)} = \theta_{I_k}$. Conditioned on $M = m_k$, we can sample I constrained so that $I'_k = I_k$, effectively choosing a random $I'_{k-1}, I'_{k-2}, \ldots$ so that $I'_{k-1} \subseteq I'_k$ is still satisfied for all k. Thus on the subsequent RTJ sampling step, the model with $M = m_{k-1}$ will correspond to the random set of parameters in I'_{k-1}, rather than a fixed set. This modification can greatly improve mixing by allowing objects to be removed in any order.

Constructions with $O(K)$ components

In Section 6.2.3, we sampled I uniformly (subject to some constraints) conditioned on M. Here, we sample I along with θ conditioned on L. This allows the sampler to find index sets that correspond to suitable parameter sets for model m_{L-1}.

Let $J(I, L)$ be the set of I'_{L-1} such that $I'_{L-1} \subseteq I_L$. Note that $|J(I_L)| = \binom{n_L}{n_{L-1}}$. We have

$$\begin{aligned} p(\theta, I|X, L) &\propto p(\theta) \sum_{k=L-1}^{L+1} p(X|\theta^{(k)}, M = m_k, I)P(M = m_k) \\ &= p(\theta) \sum_{k=L-1}^{L+1} p(X|\theta_{I_k}, M = m_k)P(M = m_k) \end{aligned} \tag{6.18}$$

Thus the conditional depends on only the index sets I_{L-1}, I_L, I_{L+1}. Letting $I'_{L-1} \in J(I, L)$

and integrating,

$$p(\theta|X, L) \propto p(\theta) \sum_{k=L}^{L+1} \binom{n_L}{n_{L-1}} p(X|\theta_{I_k}, M = m_k) P(M = m_k)$$
$$+ \sum_{I'_{L-1} \in J(I,L)} p(X|\theta_{I'_{L-1}}, M = m_{L-1}) \tag{6.19}$$

This gives a total of $\binom{n_L}{n_{L-1}} + 2$ likelihood evaluations to evaluate (6.19). After updating θ, we sample M and I'. The conditional for M, I' is

$$P(M, I'|\theta, L) \propto p(X|\theta^{(k)}, M, I')P(M) \tag{6.20}$$

where $I'_{L-1} \in J(I, L)$, $I'_L = I_L$, $I'_{L+1} = I_{L+1}$, and $M \in \{m_{L-1}, m_L, m_{L+1}\}$. Again, the terms for $M \in \{m_L, m_{L+1}\}$ can be collected to save computation. Upon sampling $M = m_{L-1}$ and I'_{L-1}, I'_k with $k < L - 1$ can be set arbitrarily or at random.

6.3 Invariance and Ergodicity

In this section we explicate the correctness of the disjoint and nested RTJ samplers; similar arguments apply to the other variants of the algorithm.

First we show that RTJ follows detailed balanced if the underlying finite dimensional samplers also follow detailed balance. First recall that by definition the measure μ respects all probability measures of interest; that is the model prior probabilities $P(M = m_k)$ and model priors $p(\theta^{(k)}|M = m_k)$. Thus the density $p(\theta|X)$ respects the posteriors $P(M = m_k|X)$ and $p(\theta^{(k)}|M = m_k, X)$. In the finite $\mathcal{M}$ case, the target distribution of the finite sampler is $p(\theta|X)$, so RTJ is invariant to $p(\theta|X)$. M is then sampled conditioned on θ in a Gibbs step, leaving the chain invariant to $p(\theta, M|X)$.

For the infinite $\mathcal{M}$ case, the introduction of L does not affect the marginal distributions of θ or M. MCMC updates conditioned on L leave the conditional distribution $p(\theta|L, X)$ invariant. M and L are updated by Gibbs steps, so the overall chain is invariant to $p(\theta, M, L|X)$, and discarding L gives a chain invariant to $p(\theta, M|X)$.

Thus the invariant distribution for RTJ is the specified target distribution. The MCMC chain is also irreducible. For the finite $\mathcal{M}$ case, RTJ is irreducible by virtue of the irreducibility of finite dimensional sampler, and the Gibbs sampling of M. For the infinite $\mathcal{M}$ case, when conditioned on L, the chain is irreducible for the parameter spaces $\Omega^{(l-1)}$, $\Omega^{(l)}$ and $\Omega^{(l+1)}$. As any state $L = l'$ can be reached from $L = l$ via a random walk by a series of Gibbs steps, and likewise for M, the overall chain is irreducible.

As RTJ leaves the target distribution invariant and the chain is irreducible, the chain is positive. Positivity implies that the chain is recurrent (see Proposition 6.36 of [50]). As the updates to M are a random walk, the chain is aperiodic. Thus RTJ is ergodic.

6.4 Retrospective Sampling and Reversible Jump

Retrospective Jump gets its name from two existing sampling algorithms, Retrospective Sampling and Reversible Jump MCMC. Retrospective Sampling operates on Dirichlet Process mixture models, where a datapoint X_j is assigned to a cluster with probability proportional to

$$p(z_j = k|X, \theta) \propto q_k = w_k p(X_j|\theta_k) \tag{6.21}$$

This step is done *retrospectively*, that is, first a uniform variate u is drawn, and then z_j is selected in finite time using bounds on q_i so that a full normalization of an infinite vector is not necessary.

The introduction of the variable L simplifies the above procedure, in that conditioning on $L = l$ leaves $z_j \in \{l-1, l, l+1\}$ as the only values of z_j that have nonzero probability, thus normalization is tractable. Furthermore, this easily generalizes to arbitrary model averaging problems (and is not restricted to updates for individual assignment variables), and we may even update θ conditioned on L to improve mixing. The retrospective step comes in when we sample z_j (or, more generally, M) conditioned on θ and L.

This generalization leads to an algorithm that is in similar in some regards to Reversible Jump, most notably, the chain for M updates via a random walk. However, rather than requiring a manually specified proposal distribution that proposes jumps to higher or lower dimensional representations, Retrospective Jump augments the parameter space so that all models share the same space, and a random walk is performed on M by sampling it conditioned on L and θ.

6.5 Demonstration

We demonstrate the RTJ algorithm on variable dimension variants of two basic problems: mixture modelling and social network analysis. To demonstrate the general applicability of our sampler, we did not make use of conjugacy for the purposes of sampling any variables – all low level samplers involved are "black box" samplers.

6.5.1 Mixture Modelling

We demonstrate RTJ on our GMM running example (6.1). Gaussian mixture models with a parametric yet random number of components has been studied before in [19] using RJMCMC, however when using RTJ we do not need to provide a proposal distribution. Dirichlet Process Mixtures remain a popular choice for cluster analysis where the effective number of

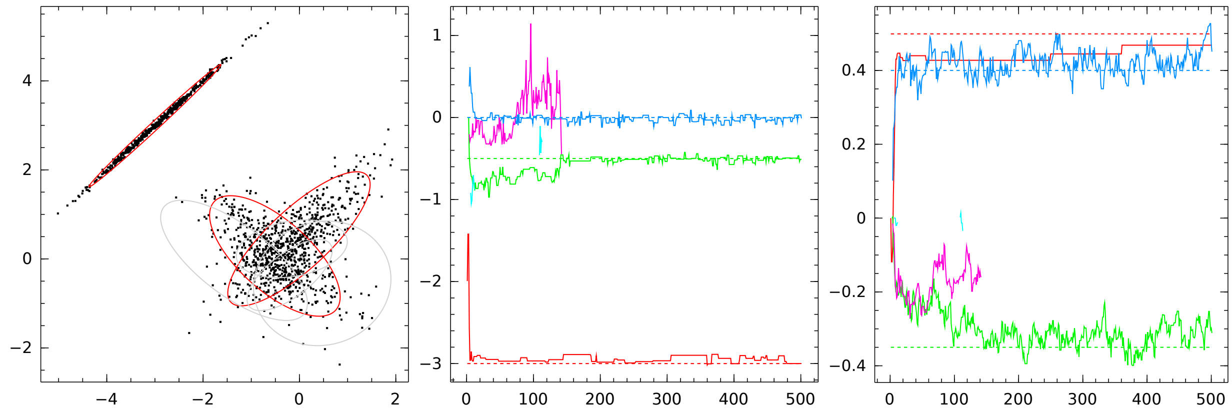

Figure 6.2: RTJ demonstration on a mixture of GMMs. The final state of our sampler **(left)** and the trajectories of the first dimension of μ_k **(middle)** and the offdiagonal term of Σ_k **(right)** for all active k. The red ellipses correspond to the 2 standard deviation level sets of the active clusters, and the light gray correspond to those of the inactive clusters (ie $k > K$), that are being explicitly represented. For the middle and right figures, the solid lines are the sampled trajectories and the dashed lines are the generating parameters. The red line corresponds to the parameters associated with the highly elliptical data.

components is inferred. For inference in the DPM, the z_is are sampled and w integrated out. However, in this parametric setting it is possible to instead integrate out the z_is and represent w. We applied this model to synthetic data generated from three Normal distributions in two dimensions. We synthesized enough points so that the generating parameters could easily be recovered. In this case we have $N = 1500$.

Inference

We use the RTJ algorithm as given in Section 6.2.3. For each RTJ iteration, we sampled w, μ, and Σ in turn, and for μ and Σ, we sampled each component's parameters μ_k or Σ_k in turn. We represented w as the normalization of K independent Gamma draws. As the density (6.19) may be multimodal, we used refractive sampling for all of w, μ, and Σ. With a suitable caching scheme for sharing computations between different components of (6.19), our sampler completed 500 Retrospective Jump iterations in a few hours, with six sweeps through the parameters per RTJ iteration.

Results

Figure 6.2 shows the final state recovered and the trajectory of some of the parameters during inference. As shown, the sampler begins with $K = 1$ and quickly ramps up to $K = 5$. After about 200 iterations, the 4^{th} and 5^{th} components have been removed and the sampler remains at $K = 3$.

6.5.2 Network Analysis

Given a $N \times N$ matrix of binary observations Y, the Latent Feature Relational model (LFRM) [42] defines a Bayesian nonparametric model using the Indian Buffet Process (IBP) [23] to describe Y. Rather than use the IBP, we can opt for a parametric distribution over binary matrices, which we call a Binary Matrix Prior (BMP). The model is:

$$
\begin{aligned}
K &\sim \text{Poisson}(\lambda) \\
p_k &\sim \text{Beta}(\alpha, \beta) \\
Z_{i,k} &\sim \text{Bernoulli}(p_k) \\
W_{k_1,k_2} &\sim \begin{cases} \text{Gamma}(2, \sigma_W) & \text{if } k_1 = k_2 \\ \mathcal{N}(0, \sigma_W) & \text{otherwise} \end{cases} \\
A_j &\sim \mathcal{N}(0, \sigma_A) \\
B_i &\sim \mathcal{N}(0, \sigma_B) \\
C &\sim \mathcal{N}(0, \sigma_C) \\
p(Y_{i,j} = 1) &= \sigma(Z_{i,\cdot} W Z_{j,\cdot}^T + A_j + B_i + C)
\end{aligned}
$$

K, α and β define the BMP prior on Z, which is used in the likelihood as in the LFRM. Y is the adjacency matrix of the graph describing the observed relationships between the

N individuals (or "actors"). An edge between two actors may be directed to represent relationships that are not symmetric. Here we adopt the convention $Y_{i,j} = 1$ if there is a directed edge from i to j. Z is a binary matrix that describes a latent group structure to the network. That is, $Z_{i,k} = 1$ if actor i belongs to group or "feature" k; an actor can belong to multiple features. W determines the effect of the feature interactions. As we may rewrite

$$Z_{i,\cdot} W Z_{j,\cdot}^T = \sum_{k_1 | Z_{i,k_1}=1} \sum_{k_2 | Z_{j,k_2}=1} W_{k_1,k_2} \tag{6.22}$$

W_{k_1,k_2} describes the affinity that actors in k_1 feel towards actors in k_2. We restrict the diagonal entries of W to be positive by using a Gamma prior as we desire groups to represent tightly knit sets of actors. Finally, the terms A and B are column-wise and row-wise intercepts for modelling a particular actor's "popularity" or "friendliness," respectively.

We applied this model to two datasets:

Sampson's Monastery

We applied this model to Sampson's Monastery data [51]. $N = 18$ monks were inquired as to which three of his peers he held in highest esteem, at three different time periods during a social falling out at the monastery. Four distinct factions are commonly believed to be present in the data (three main factions "Young Turks," "Loyal Opposition," "Outcasts", and a set of "Waverers" whose allegiances were unsteady during the conflict). Because a monk may be selected by his peers arbitrarily many times, but each monk can only choose 3 peers, we included the receiver effects A in the model, but no sender effects B. We trained our models on the first two snapshots of the Monastery data, holding out the third for evaluation.

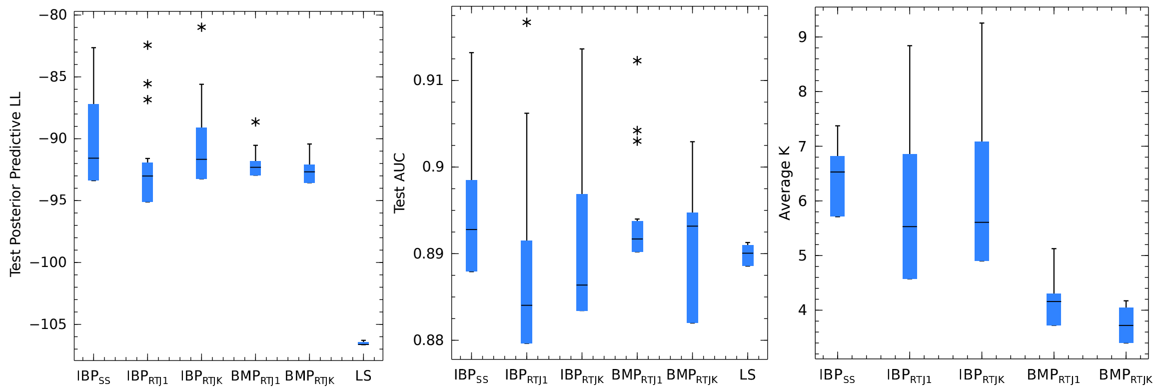

Figure 6.3: RTJ comparison on Sampson's Monastery. Each boxplot summarizes the performance among the 8 trials that were run. "LS" is the latent space model. "SS" corresponds to inference using slice sampling for the IBP. "Average K" is the average active K over a trial's chain.

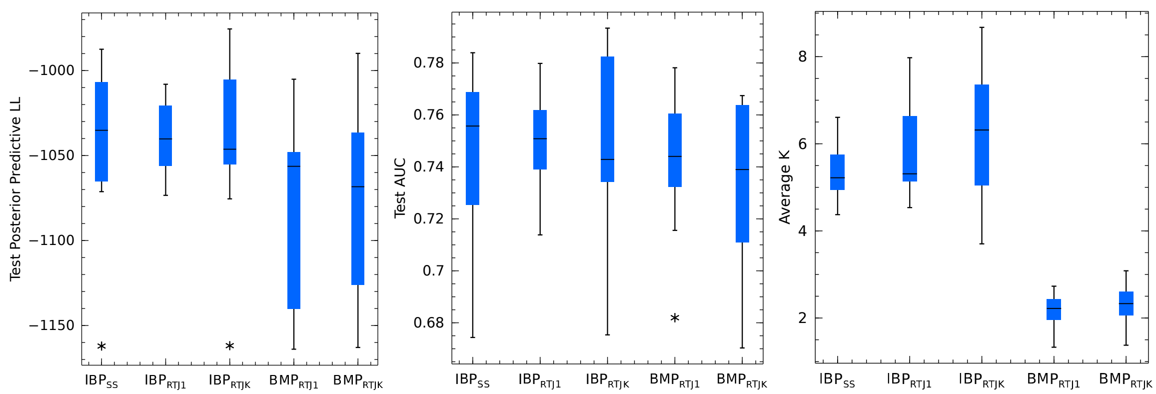

Figure 6.4: RTJ comparison on *C. elegans*. Each boxplot summarizes the performance among the 8 trials that were run. Each trial had a different train/test split of the data. "SS" corresponds to inference using slice sampling for the IBP. "Average K" is the average active K over a trial's chain.

Protein Interaction

We also applied our model to the protein interaction network for *caenorhabditis elegans* from the KONECT database [35]. We removed proteins that had fewer than 8 interactions, leaving a network among $N = 120$ proteins. The problem is symmetric, so we impose a symmetric prior on W and restrict $B = A$. We randomly split the matrix entries into train/test sets, so an entry in Y may be 1, 0, or missing.

Inference

We compared the BMP model to the LFRM with the IBP, performing inference with slice sampling for the IBP [54], which has been applied to interaction network problems previously in [14]. These models shared the above specification for W, A, C, and Y, except that they employ a different prior over Z parameterized by α_{IBP}, and the slice sampler also includes a stick-breaking representation μ of the Beta Process. We set $\lambda = 3$, and for the IBP experiments, we set $\alpha_{IBP} = \lambda/H_N$, where H_t is the t^{th} Harmonic number. With these settings both models have $E[K] = \lambda$ in the prior.

We compare the RTJ variants using (6.16) ($O(1)$ cost) and (6.19) ($O(K)$ cost), which we call RTJ1 and RTJK, respectively. We also apply both algorithms to the IBP based LFRM model. We do not provide a formal comparison to RJMCMC, as preliminary experiments using the prior as a RJMCMC proposal gave prohibitively low acceptance rates.

For each Retrospective Jump iteration, we sampled W, A, B, C, and Z in turn. We sampled W (represented in log-space), A, and B using refractive sampling, C using univariate slice sampling, and Gibbs sampled each Z_{ik} in turn. We initialized auxiliary dimensions of W and Z by sampling from the prior.

For Sampson's Monastery we ran all samplers for 6000 iterations, performing fifteen sweeps through the parameters per iteration. Using multiple sweeps through the parameters when sampling $\theta'|L, \theta$ can improve mixing. We found that using five such sweeps proved beneficial for the *c. elegans* experiments. We ran all samplers for 1000 iterations on the *c. elegans* data, giving a total of 5000 sweeps through the parameters. Burn-in iterations discarded for Sampson's Monastery and *c. elegans* where 1000 and 500, respectively.

Due to the overhead associated with evaluating (6.19) and (6.16), there is extra computational cost associated with RTJ, so the RTJ results generally took longer. The total running time for a trial also depends on the chain's trajectory through K, so the BMP-based models usually ran in relatively less time as they generally used fewer features. For the Sampson's Monastery dataset, the IBP with slice sampling finished in 30-40 minutes, IBP with RTJ1 finished in 3 hours, and with RTJK finished in 4-5 hours. BMP with RTJ1 finished in 1 hour and with RTJK finished in 2 hours. For the *c. elegans* data, IBP with slice sampling finished in approximately 3 hours, IBP with RTJK in about 12-30 hours, and RTJ1 in about 12 hours. The BMP runs with RTJ1 finished in about 3 hours and RTJK finished in about 5-10 hours. The bottleneck for all samplers is Gibbs sampling the entries of Z.

We note that RTJ was implemented for use within a general software package that can apply to a wide variety of problems, while Slice Sampling for the IBP is a much simpler implementation free from the technical details needed in a general inference framework. Thus, there is significant overhead associated with the RTJ implementation in addition to the cost of evaluating the mixed distributions. Implementations of RTJ that are designed specifically for use on models involving latent binary matrices would see a significant reduction in computational cost.

For the Sampson's Monastery experiment, we compare to the Latent Space model [26] using the `latentnet` R package [33]. We set the number of clusters equal to 3 and latent space dimension to 2. We provided the latent space model with the same sender and receiver effects provided to the latent factor models. We ran 8 independent trials using the default `latentnet` settings for the Latent Space model: 14000 iterations, discarding the first 10000 as burn-in. In addition, the Latent Space model is initialized by an optimization procedure, where as the IBP and BMP models are initialized randomly.

We found that the IBP-based models overfit on the *c. elegans* by introducing too many features. Constraining all entries of W to be positive improves this issue, thus for this dataset we use the prior $W_{k_1,k_2} \sim \Gamma(1, \sigma_w)$ for all k_1 and k_2.

Results

For each sampler, we performed 8 independent trials which were used in the Monte Carlo test in [53] to assess convergence. All samplers exhibited similar evidence of convergence[3]. For each $Y_{i,j}$, we computed its predictive log-likelihood averaged over the sampled chain for a given trial, and we also computed the test AUC using these averaged predictions. We ran RTJ on the BMP model with $\alpha = \beta = 1.1$.

On the Sampson's Monastery data, (see Figure 6.3), we find that all algorithms and models aside from the Latent Space model perform similarly well on held-out test log-likelihood, and all algorithms performed equally well on AUC. The BMP-based models use less features on average than the IBP-based models, while still providing competitive predictive performance. We note that there is significantly more variance across trials for the IBP and BMP models compared to the Latent Space model; this is not too surprising considering these models are not of fixed dimension, whereas the Latent Space model is parametric.

For the *c. elegans* trials, we split the data into 50-50 train/test splits, with different splits of the data for each trial (but shared between samplers). See Figure 6.4. Again, the BMP-based model uses less features on this larger dataset, while still maintaining competitive performance with the IBP-based model.

6.6 Summary

Retrospective Jump sampling can be easily applied to a wide variety of problems, as models can be specified to RTJ simply by specifying the relevant density. While more computational demanding than model specific algorithms, RTJ is a general purpose algorithm that can

[3]Note that algorithms that have tendencies to fall into similar modes across trials will still report evidence for convergence.

effectively perform inference in a wide variety of problems. When updating $\theta'|L, \theta$, RTJ is able to seek M, θ pairs with high posterior probability; this is an "exploration" step that is critical to transitioning between spaces of differing dimension. RTJ opens the door for many interesting models that would otherwise require sophisticated RJMCMC proposals.

Chapter 7

Application: Infinite Sites Feature Prior

In this chapter, we apply Retrospective Jump sampling to a novel model that would otherwise require sophisticated RJMCMC proposals.

7.1 Introduction

It is normally understood that interpretability and predictive accuracy are two properties of statistical models that are at odds with each other. A simpler, more interpretable model will give an elegant interpretation to the data, but fail to capture the more subtle patterns that may be present. A more complex model can capture these tendencies, but may report them in an obscure manner that is difficult to interpret.

Hierarchical clustering models, on the other hand, are models with varying levels of resolution; parameters higher up in the hierarchy explain the overall trends in the data, and lower-level parameters explain local trends. After a hierarchical clustering model is learned, the hierarchy can be pruned at different levels to give clusterings of different resolutions. Thus these models allow parameters for a range of high-level to low-level effects to explain all trends in the data, while also allowing that the level of interpretability of the model can be chosen (or varied) after inference is performed.

There has been growing interest in nonparametric latent feature models such as the Indian Buffet Process (IBP) [23] which provide high predictive accuracy due to their flexible modeling capacity. Several extensions and modifications to the IBP have been introduced to give latent feature models with varying properties. [7] provides a nonparametric latent feature model where the rows are not marginally Poisson (as is the case with the IBP). [59] extends this capability, by "restricting" the IBP (or other nonparametric distributions) to have certain properties while still maintaining exchangeability – these restrictions may include user-specified marginal distributions for the number of features assigned to a datapoint. The phylogenetic IBP [41] modifies the IBP with a given tree structure which expresses *apriori* dependencies between the data.

In this chapter we combine hierarchical clustering with latent feature modelling to give a latent feature model whose features have varying scopes. Specifically, we combine a prior over trees known as the Beta-Splitting prior [2] with the Infinite Sites model from population genetics [32, 11] to obtain a distribution over binary matrices with hierarchical column structure.

First, we review the Beta-Splitting model and Infinite Sites model, and define the Infinite Sites Feature Process (ISFP). Next we detail the inference scheme that we use for this model. Then, we detail our application of interest, namely social network analysis, and finally we give experimental results and conclusions.

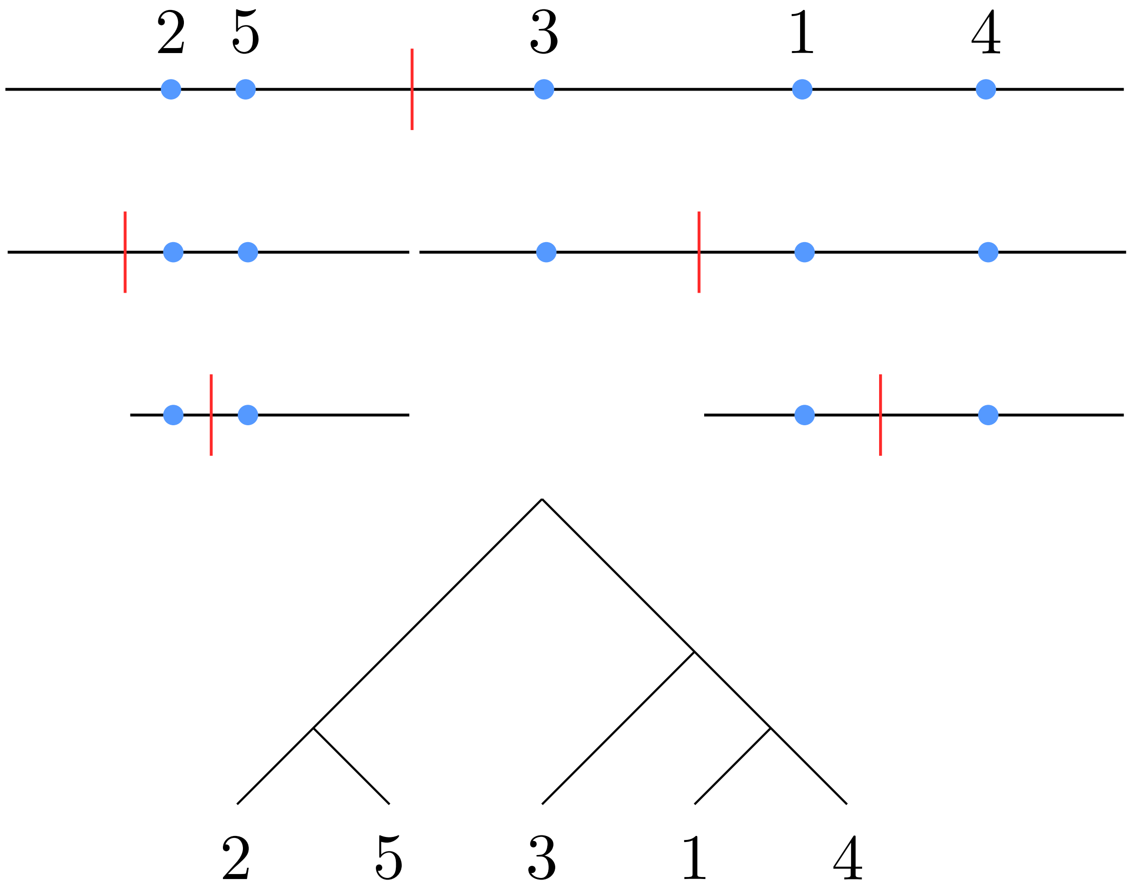

Figure 7.1: Aldous Beta-Splitting. **(top)** Five points are given locations on the unit interval uniformly at random. A symmetric random variable is drawn that splits the interval into two. The remaining intervals are recursively split until each point is isolated. **(bottom)** The hierarchy of the five points generated by the beta-splitting procedure.

7.2 Infinite Sites Feature Process

In order to define a prior over binary matrices with underlying hierarchical structure, we first define a distribution over trees using Aldous' beta-splitting model [2, 38], and then define a distribution over binary matrices given the tree.

We denote tree structures (or hierarchical partitions) with ψ. Given a set X with $N = |X|$, ψ is an ordered set of partitions of X. That is, $\psi = \{\Lambda_i | i \in \{1, \ldots M\}\}$, with Λ_i a partition of X, $|\Lambda_i| < |\Lambda_j|$ if $i < j$, and $\Lambda_1 = \{X\}$ is the trivial partition. As each Λ_i is a partition of X, it is exhaustive and pairwise mutually exclusive: $\bigcup_{a \in \Lambda_i} a = X$ and $a \cap b = \emptyset$ for all $a, b \in \Lambda_i$ and for all i. Furthermore, Λ_i is constructed from Λ_{i-1} by splitting one or more of its clusters. In the case of a binary tree, as in this work, we have $|\Lambda_{i+1}| = 1 + |\Lambda_i|$ and $M = N - 1$.

ψ defines parenthood and childhood relationships among nodes in a tree. There is an internal node i for each unique subset $b_i \subset X$ with $b_i \in \Lambda_j$ for any j. ψ is binary, so there are N leaf nodes, and $N - 1$ internal nodes. Every internal node has two children, and every node except the root has one parent. A node p is the parent of l and r if l and r were formed by splitting p. We also allow the root to have a "parent," which we call the supraroot. The supraroot has only one child, the root.

7.2.1 Beta-Splitting Trees

Beta-splitting defines an infinite tree-structure in a top-down fashion. Define a symmetric density $f(x)$ on the unit interval. Begin by placing N points, or individuals, uniformly at random on the unit interval. Then repeat until all points are isolated:

1. Draw $x \sim f$.

2. Draw $l_i \sim \text{Bernoulli}(x)$ for each point i

3. Those i with $l_i = 1$ are designated to the left branch, all others designated to the right

4. Recurse on each subtree, until all points are assigned their own branches as leaves.

See Figure 7.1 for an illustration. This process defines an infinite tree if we let $N \to \infty$. [2] specializes f to a one-parameter Beta distribution, so that $x \sim \text{Beta}(\beta + 1, \beta + 1)$. Picking β and marginalizing out x gives rise to many familiar priors, for example $\beta = 0$ corresponds to Yule trees or Kingman's Coalescent prior, and $\beta = -\frac{3}{2}$ corresponds to the uniform distribution on trees.

Typically, x is integrated out and the resulting discrete distribution over tree structures remains, with an additional time variable for each internal node representing the time at which a set of leaves' lineages split. However, we will instead represent the splitting proportions x as it gives a convenient way to define the times of the internal nodes of the tree.

For a tree with N leaves, let the indices $i \in \{1, \ldots 2N-1\}$ index the internal and leaf nodes, where $i \in \{1, \ldots, N\}$ correspond to leaf nodes and $i \in \{N+1, \ldots 2N-1\}$ correspond to internal nodes. No special structure is assumed for the indices of the internal nodes. Let $\nu_i \sim \text{Beta}(\beta + 1, \beta + 1)$ be the splitting proportion associated with node i, and μ_i be the total proportion of mass associated with the subtree rooted at node i, as drawn from the beta-splitting process. Here, an internal node represents the most recent common ancestor (MRCA) of all its descendant leaf nodes.

In the beta-splitting process, a finite set of N points may sometimes all draw equal values of l, thus causing a split in the infinite tree that is not represented in the tree restricted to the N points. Thus, with finite N, μ_i is not simply $\nu_i \prod_{j \in \text{An}(i)} \nu_j$, where $\text{An}(i)$ is the ancestor set of i – we must account for these unobserved splits in the infinite tree. Consider node i

with children l and r, where N_i points split into two sets of sizes N_l and N_r. Let $\tilde{\nu}_i$ be the total split proportion accumulated by repeated beta-splitting of the N_i points until a split actually occurs. We derive the distribution of $\tilde{\nu}_i$ next.

We restrict our attention to the Coalescent model, $\beta = 0$, so that $x \sim \text{Uniform}(0,1)$. The marginal probability of a set of N points failing to split apart is

$$2\int x^N(1-x)^0 f(x)dx = \frac{2}{N+1} \tag{7.1}$$

Let $\xi_N = 1 - \frac{2}{N+1}$. Then $D \sim \text{Geometric}(\xi_N)$ gives the number of failures until the N points are split. The distribution of $\tilde{\nu}_i$ is thus the product of D uniform draws. If we take $U_i \sim \text{Uniform}(0,1)$, and $U^{(d)} = \prod_{i=1}^{d} U_i$, then $-\ln U^{(d)}$ is the sum of d Exponential draws, and

$$-\ln U^{(d)} \sim \Gamma(d,1) \tag{7.2}$$

And so

$$\begin{aligned}
p(-\ln\tilde{\nu}_i = y) &= \sum_{d=1}^{\infty} p(D=d)p(-\ln U^{(d)} = y) + p(D=0)\delta_0(y) &(7.3)\\
&= \sum_{d=1}^{\infty}(1-\xi_N)^d\xi_N\frac{1}{(d-1)!}y^{d-1}e^{-y} + \xi_N\delta_0(y) &(7.4)\\
&= (1-\xi_N)\xi_N e^{-y}\sum_{k=0}^{\infty}\frac{((1-\xi_N)y)^k}{k!} + \xi_N\delta_0(y) &(7.5)\\
&= \xi_N\delta_0(y) + (1-\xi_N)\xi_N e^{-\xi_N y} &(7.6)
\end{aligned}$$

Which gives

$$\tilde{\nu}_i \sim \xi_N\delta_1(\tilde{\nu}_i) + (1-\xi_N)\text{Beta}(\xi_N,1) \tag{7.7}$$

This result agrees with intuition; if N is large, then ξ_N is close to 1, and $\tilde{\nu}_i$ approaches a point mass at 1. ν_i is simply Uniform$(0,1)$, and if nodes l and r are siblings, then $\nu_r = 1-\nu_l$. If N points are split between siblings l and r into groups of sizes N_l and N_r, then

$$p(\nu_l, N_l, N_r) \propto \frac{1}{N(N-1)} \nu_l^{N_l-1}(1-\nu_l)^{N_r-1} \tag{7.8}$$

This form arises from the fact that we are conditioning on a split occurring, that is, $N_l > 0$ and $N_r > 0$. Finally, we can write μ_i:

$$\mu_i = \nu_i \tilde{\nu}_i \prod_{j \in A(i)} \nu_j \tilde{\nu}_j \tag{7.9}$$

Introducing Time Variables

Variables that denote the time from leaf to internal node are typically employed in hierarchical clustering models. If these variables have the property that internal nodes that have few descendants are strongly encouraged to have smaller times (and thus are "closer" to the leaves), then the clustering model is less likely to be overly flexible and in danger of overfitting. Kingman's Coalescent the Dirichlet Diffusion Trees [46] both employ time variables with this property.

The choice of time variables is a delicate one, as any choice should leave the overall distribution Kolmogorov consistent. One way to ensure that the times retain a consistent prior distribution is to show that the time to the most recent common ancestor (MRCA) for a pair of nodes is the same in the infinite tree as it is in a finite projection. If we take $\tilde{\rho}_p = \tilde{\nu}_l = \tilde{\nu}_r$ for children l and r of p, then we may define the time t_p

$$t_p = \mu_p^\gamma \tag{7.10}$$

where $\gamma > 0$. As μ_p is the beta-splitting proportion of the MRCA of l and r in the infinite tree, using it to define a time variable gives a consistent prior over times. As $0 \leq \mu_p \leq 1$, choosing larger values of γ will correspond to nodes that are closer to the leaves, and thus to a less flexible prior.

This choice of time variable has two main advantages over the times used for the Coalescent model. First, the time for a node p is dependent only on the number of individuals delegated to each of its children, l and r. Coalescent times, on the other hand, are determined by starting with N individuals and recursively joining pairs of with exponential waiting times – the time for node p is directly dependent on the times of many other nodes throughout the tree. Second, this construction allows for more choice in the specification of how the times are distributed for nodes that are deeper in the hierarchy, for example the choice of γ can be modified to push internal nodes leaf-wards or root-wards.

This particular choice of time variable is also convenient for inference when used in conjunction with the Infinite Sites model, which we review next.

7.2.2 The Infinite Sites Model

The infinite sites model from population genetics originated as a model for the evolution and mutation of the genetic sequences of a population [32]. Given an ancestral hierarchy ψ, the infinite sites model gives a prior over binary matrices Z, which can be used as a latent feature model akin to the Indian Buffet Process. The infinite sites model is a mutation process in which mutations can only occur in a locus at most once, such that there is no back mutation. That is, if we assume that each point starts at a state of an infinite length vector of all 0s, elements are flipped to 1 at each mutation event, and no 1 is ever flipped back to 0. Mutation events occur according to a Poisson Process of rate λ down the branches of the tree. Thus all mutations that occur on the path from leaf-to-root are features that

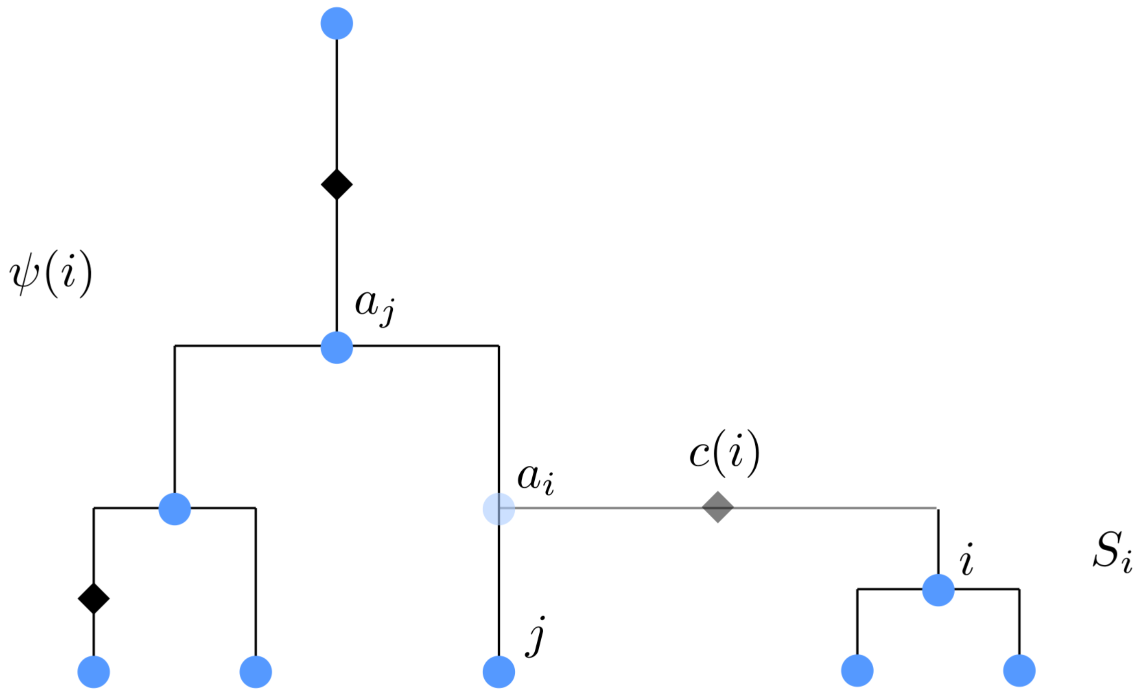

Figure 7.2: Grafting step of the prune/graft MCMC inference for ψ. Diamonds represent features, circles represent nodes. When considering attaching a_i between j and a_j, the features in $c(i)$ – in this case a single feature – can be moved to any of the edges incident to a_i.

help describe the datapoint associated with that particular leaf of ψ. The binary matrix Z can be characterized by the number of mutations u found on each branch of the tree.

The number of mutations u_i on a particular branch of length v_i is Poisson(λv_i), where the branch lengths are equal to the time from parent to child: $v_i = t_{a_i} - t_i = t_{a_i}(1 - (\nu_i \tilde{\nu}_i)^\gamma)$, where a_i is the parent of i. To construct Z, we can traverse the tree from supraroot to leaf, adding a column to Z for each mutation we encounter, and setting Z_{ij} to one if leaf i lives below mutation j. Fig 7.3 shows an example Z drawn from the ISFP. As a prior over Z, a smaller mutation rate λ encourages an overall smaller number of features, while a larger branching time parameter γ encourages less new features near the leaves, and more near the root.

This prior over Z defines what we call the Infinite Sites Feature Prior (ISFP).

7.3 Inference

Let $p(X|Z, \theta)$ denote the likelihood of a particular data model, where θ denotes additional parameters associated with the likelihood model. To perform posterior inference on $p(Z, \psi, \theta|X)$, we perform blocked Gibbs sampling, drawing ψ, Z, and θ in turn.

7.3.1 Sampling $\nu, \tilde{\nu}|\psi, Z, \theta, X$

Given a fixed tree ψ, each branch l (associated with node l) has $u_l \sim \text{Poisson}(\lambda(t_p - t_l))$ mutation events, where p is the parent of l. Let r be the sibling of l. Note that

$$\lambda(t_p - t_l) = \lambda(1 - (\tilde{\nu}_l \nu_l)^\gamma) \prod_{j \in \mathrm{An}(l)} (\tilde{\nu}_j \nu_j)^\gamma \tag{7.11}$$

$$\nu_l = 1 - \nu_r \tag{7.12}$$

We can take $\rho_p = \nu_l$, if l is the left child of p. Then,

$$\begin{aligned} p(\tilde{\nu}_p|-) \propto & R_p(\tilde{\nu}_p \nu_p)^{k_p} \tilde{\nu}_p^{\gamma K_p} \\ & \exp\left(-\lambda \left[R_p(\tilde{\nu}_p \nu_p) T_p + \tilde{\nu}_p^\gamma \nu_p^\gamma S_p\right]\right) \\ & \left[\xi_{N_p} \delta_1(\tilde{\nu}_p) + (1 - \xi_{N_p}) \xi_{N_p} \tilde{\nu}_p^{\xi_{N_p} - 1}\right] \end{aligned} \tag{7.13}$$

$$\begin{aligned} p(\rho_p|-) \propto & R_l(\tilde{\nu}_l \rho_p)^{k_l} R_r(\tilde{\nu}_l (1 - \rho_p))^{k_r} \rho_p^{\gamma K_l + N_l - 1} (1 - \rho_p)^{\gamma K_r + N_r - 1} \\ & \exp\left(-\lambda \left[R_l(\tilde{\nu}_l \rho_p) T_l + R_r(\tilde{\nu}_l (1 - \rho_p)) T_r + \tilde{\nu}_l^\gamma \rho_p^\gamma S_l + \tilde{\nu}_r^\gamma (1 - \rho_p)^\gamma S_r\right]\right) \end{aligned} \tag{7.14}$$

where there are N_j leaves in the subtree rooted at j, and

$$K_i = \sum_{j \in \mathrm{De}(i)} k_j \tag{7.15}$$

$$T_i = \prod_{k \in \mathrm{An}(i)} (\tilde{\nu}_k \nu_k)^\gamma \tag{7.16}$$

$$S_i = \sum_{j \in \mathrm{De}(i)} R_j(\tilde{\nu}_j \nu_j) \prod_{k \in \mathrm{An}(j)\setminus\{i\}} (\tilde{\nu}_k \nu_k)^\gamma \tag{7.17}$$

$$R_i(x) = 1 - x^\gamma \mathbb{1}[i \notin \mathrm{Leaves}(\psi)] \tag{7.18}$$

where $\mathrm{Leaves}(\psi)$ are the leaf nodes of ψ. ρ_p can be updated easily enough using slice sampling. The measure of $\tilde{\nu}_p$, however, is mixed discrete and continuous:

$$p(\tilde{\nu}_p|-) \propto f_1 \delta_1(\tilde{\nu}_p) + f_2(\tilde{\nu}_p) \tag{7.19}$$

$$f_1 = \xi_{N_p} f(1)$$

$$f_2(\tilde{\nu}_p) = (1 - \xi_{N_p}) \xi_{N_p} \tilde{\nu}_p^{\xi_{N_p} - 1} f(\tilde{\nu}_p)$$

$$f(\tilde{\nu}_p) = R_p(\tilde{\nu}_p \nu_p)^{k_p} \tilde{\nu}_p^{\gamma K_p} \exp\left(-\lambda \left[R_p(\tilde{\nu}_p \nu_p) T_p + \tilde{\nu}_p^\gamma \nu_p^\gamma S_p\right]\right)$$

To sample $\tilde{\nu}_p$, we define variables $x \in [0,1]$, $u \in [0, \infty)$ with joint density

$$p(x,u) \propto \mathbb{1}(u < f_1 + f_2(x)) \tag{7.20}$$

$p(x,u)$ specifies a uniform distribution on the set $u < f_1 + f_2(x)$. Sampling from $p(x,u)$ and discarding u generates samples from $p(x) = f_1 + f_2(x)$ – this is the same trick used in slice sampling. However, we note that $\int \mathbb{1}(u < f_1 + f_2(x)) dx du = \int \mathbb{1}(u < f_1) \vee \mathbb{1}(f_1 < u < f_2(x) + f_1) dx du = f_1 + \int f_2(x) dx$, so sampling uniformly from $p(x,u)$, and then setting

$$\tilde{\nu}_p = \begin{cases} x & \text{if } f_1 < u < f_1 + f_2(x) \\ 1 & \text{if } u < f_1 \end{cases} \tag{7.21}$$

will sample correctly from $p(\tilde{\nu}_p|-)$. To make use of this within a MCMC algorithm, we can slice sample from $p(x)$ and then draw a uniform $u|x \sim \text{Uniform}(0, p(x))$, and then set $\tilde{\nu}_p$ as in (7.21). To initialize the sampler, we should set $x = \tilde{\nu}_p$ if $\tilde{\nu}_p < 1$, and $x \sim \text{Uniform}(0, 1)$ if $\tilde{\nu}_p = 1$. This is similar to the trick used in Retrospective Jump sampling for traversing dimensions.

Thus we can cycle through all internal nodes p, update ρ_p and $\tilde{\nu}_p$, setting $\nu_l = \rho_p$, $\nu_r = 1 - \rho_p$.

7.3.2 Sampling $\psi, Z|\tilde{\nu}, \nu, \theta, X$

Keeping the total number of features fixed, we can perform MCMC moves consisting of pruning a branch from the tree ψ and grafting it to another location. Let $c(i)$ be the set of columns (mutations) that are introduced directly above node i ("on branch i"), a_i the parent index of node i in ψ, and $\text{De}(i)$ all descendants of i.

A particular jump for ψ, Z proceeds as follows: First, an arbitrary node i is pruned from the tree ψ, so that a_i only has one child and no parents. This splits ψ into two structures, $\psi(i)$ is the original ψ with i pruned from it, and S_i is the subtree rooted at a_i. Then we can consider grafting S_i back into $\psi(i)$ above a particular node j so that j's new parent becomes a_i. The resulting tree is denoted $\psi(i, j)$. See Figure 7.2.

This move preserves the number of columns K, and only changes the assignment of features to datapoints. When we consider joining S_i into $\psi(i)$ above j, we can reassign the features in $c(j) \cup c(i)$ to be above a_i, above i, or above j[1].

The $\tilde{\nu}$ and ν variables can be kept fixed while we sample ψ. For a given adjoining edge j, with a particular assignment of features to edges, we need to evaluate $p(\psi, Z|-) \propto p(\psi|\tilde{\nu}, \nu)p(X|\theta, Z)p(Z|\psi)$. Attaching S_i above j changes the times (but not the ν and $\tilde{\nu}$) for

[1]In practice we only allow assigning above i or a_i, effectively considering all "splits" of the features on the adjoining branch, and not pulling them away from data that they currently help explain

all descendants of j. Thus the change in probability for attaching i above j with a particular assignment of features is

$$p(\psi(i,j), Z|\psi(i), S_i, \tilde{\nu}, \nu) \propto \prod_{k \in \mathrm{De}(a_i)} \frac{\mathrm{Poisson}(u'_k; \lambda(t'_{a_k} - t'_k))}{\mathrm{Poisson}(u_k; \lambda(t_{a_k} - t_k))} \tag{7.22}$$

$$\prod_{k \in \mathrm{An}(j)} \frac{N_k(N_k - 1)}{N'_k(N'_k - 1)} \nu_{l_k}^{N'_{l_k} - N_{l_k}} \nu_{r_k}^{N'_{r_k} - N'_{r_k}} \tag{7.23}$$

$$\prod_{k \in \mathrm{An}(j)} \frac{p(\tilde{\nu}_k|N'_k)}{p(\tilde{\nu}_k|N_k)} \tag{7.24}$$

where the primed variables are those associated with $\psi(i,j)$ and unprimed variables are associated with $\psi(i)$, l_k and r_k denote the left and right children of k, respectively, and $p(\tilde{\nu}_k|N_k)$ is determined by (7.7) . Here De() and An() operate on $\psi(i,j)$, so that the nodes in S_i are included in the computation as necessary.

$p(\psi(i,j), Z|\psi(i), S_i, \tilde{\nu}, \nu)$ can be computed efficiently for all nodes j via memoization. Combining with the likelihood $p(X|\theta, Z)$ gives a discrete conditional posterior for ψ.

7.3.3 Sampling $\theta|Z, \psi, X$

Sampling for θ is straightforward, as we have

$$p(\theta|Z, \psi, X) \propto p(\theta)p(X|\theta, \psi, Z) \tag{7.25}$$

Thus we may use a black-box sampler such as HMC for updating θ.

7.3.4 Sampling $Z, \theta | \psi, X$

To allow moves which can add and remove features, we use Retrospective Jump sampling. To apply Retrospective Jump to ISFP, we sample the features for each of the $2N - 1$ branches in turn, treating each as its own model determination problem. Consider the inference task on branch b if we are given $Z \in \mathbb{Z}^{N \times K}$. If there are u_b features on branch b, then we take $L = u_b + \delta$ with $\delta \sim \text{Categorical}\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) - 2$. Then RTJ will consider transitions to $u'_b \in \{L-1, L, L+1\}$, giving $K' \in \{K+\delta-1, K+\delta, K+\delta+1\}$. θ will need to be augmented accordingly. In the experiments that follow, θ is a $K \times K$ matrix. We then need to augment θ to a $(L+1) \times (L+1)$ matrix[2]. We used the nested RTJ-K algorithm in 6.2.3, so we also introduce the index set variable I where $I = \{I_u | u \in \mathbb{N}_0\}$ is sequence of index sets with $I_u \subseteq I_{u+1}$. Recall that I determines which features are removed upon sampling $u'_b < u_b$. Thus we need to be able to evaluate

$$p(\theta, Z^{(u,I)}, I | \psi, X, L) \propto p(\theta) p(X | \theta^{(u,I)}), \psi, Z^{(u,I)}) p(Z^{(u,I)} | \psi) \tag{7.26}$$

where $\theta^{(u,I)}$ and $Z^{(u,I)}$ map θ and Z to updated parameters with the appropriately removed features. If θ is a $(L+1) \times (L+1)$ matrix, then $\theta^{(u,I)}$ is a $(L+1+u-u_b) \times (L+1+u-u_b)$ matrix:

$$\theta^{(u,I)} = \theta_{I_u, I_u} \tag{7.27}$$

Also, we have

$$Z^{(u,I)} = Z_{\cdot, I_u} \tag{7.28}$$

[2] To do this, we did not take draws from the prior and instead used the "initialize once arbitrarily" method for instantiating the augmented θ as we found it worked better in practice. This is due to the fact that on each iteration $2N-1$ possible updates to Z are made. During the transient phase, proposals are more easily accepted, and initializing from the prior leads to poor modes with extremely large K.

where A_{I_k,I_j} is the submatrix constructed by taking the rows I_k and columns I_j from A. We need to evaluate (7.26) for $u \in \{L-1, L, L+1\}$ and $I'_{L-1} \in J(I,L)$, where $J(I,L)$ is the set of I_{L-1} consistent with I_L. We then update θ from the mixture (6.19), and finally sample u and I from (6.20), which gives $u'_b = u$ the number of features on branch b and $\theta' = \theta^{(u,I)}$.

7.4 Social Network Analysis

We apply the ISFP to social network analysis task, using again the likelihood:

$$p(Y_{ij} = 1) = \sigma(Z_{i,\cdot} W Z_{j,\cdot}^T + A_j + B_i + C) \tag{7.29}$$

but now Z is drawn from the ISFP. The ISFP imposes an important restriction on the structure of Z – if feature k_1 is shared between two actors, but k_2 only belongs to one of the actors, then a third actor may not take up k_2 without also taking k_1. This can be interpreted as a hierarchical restriction on the structure of Z. This gives an interesting interpretation to the groups or features inferred by the model: if feature k_2 occurs below k_1 in the hierarchy, then we may expect k_1 to represent a large faction and k_2 a subpopulation within k_1.

7.5 Demonstration

We evaluate the ISFP on synthetic and real data. In the real data experiments, we compare to the LFRM, using slice sampling for the IBP for inferring Z. In all experiments we run 10 independent trials each with a different 80/20 split of the adjacency matrix entries into training and test sets. The diagonal entries of Y were ignored for both training and evaluation in all experiments. Hyperparameters were set to give a “reasonable” number of features.

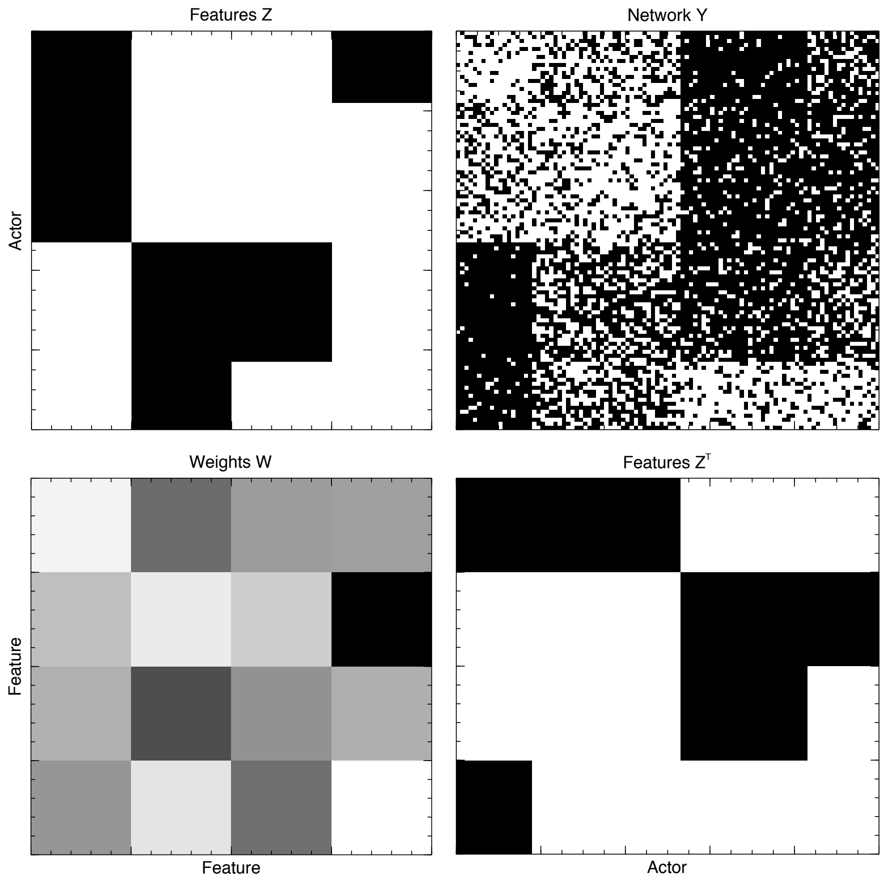

Figure 7.3: A network generated by the ISFP-based LFRM.

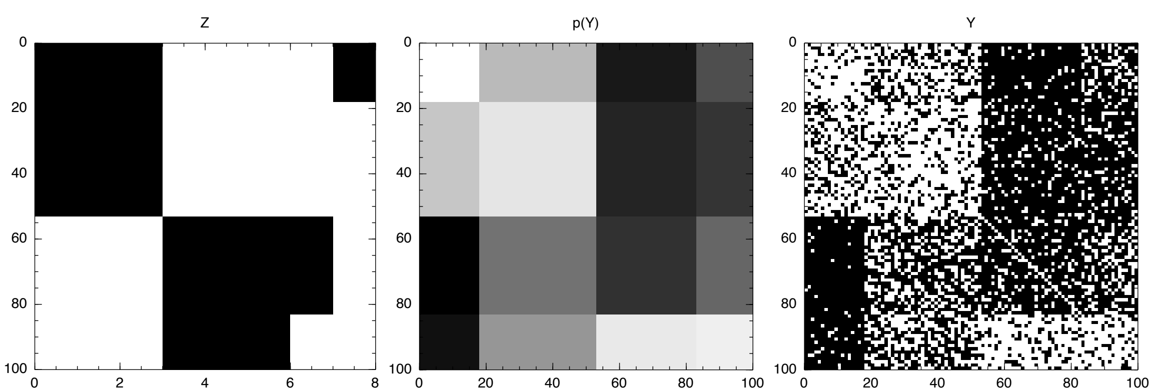

Figure 7.4: A posterior sample from the synthetic data experiment. There are some duplicate features which could be represented as single features with appropriate modifications to W.

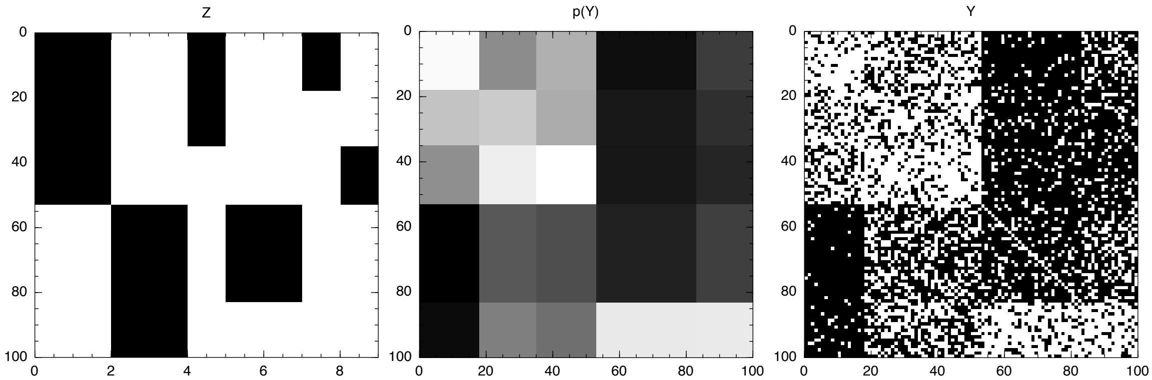

Figure 7.5: Another posterior sample from the synthetic data experiment. Here, not only are there duplicate features, there are also a few extra features near the leaves.

7.5.1 Synthetic Data

We first explore the behavior of the inference algorithm on synthetic data. We generated a network from the ISFP with 100 actors using $\gamma = 2.0$, $\lambda = 0.1$, and $\sigma_W = 1.0$, see Figure 7.3. We attempted to infer back the Z and W that generated the data, using the same hyperparameters for inference as were used for generation. We ran our MCMC chain for 500 iterations.

Despite the constrained prior on Z that the ISFP gives, the overall model is not identifiable. In particular, with the LFRM likelihood, one branch segment may contain multiple features that could be represented with a single feature. Despite this identifiability issue, the model is able to recover the structure of $p(Y)$, see Figures 7.4 and 7.5. Note that this identifiability issue is dependent on the choice of likelihood; it is certainly possible that, for some models, multiple features on a single branch segment could not be reduced and would correspond to meaningfully different aspects of the underlying data.

Table 7.1: Sampson's Monastery predictive results.

	Train Error	Test Error	Avg. Test LL	AUC	K
LFRM	0.102 ± 0.011	0.179 ± 0.052	-0.417 ± 0.084	0.760 ± 0.065	5.870 ± 0.460
LFRM $W > 0$	0.123 ± 0.016	0.194 ± 0.053	-0.446 ± 0.091	0.713 ± 0.055	5.646 ± 0.956
ISFP	0.146 ± 0.012	0.174 ± 0.058	-0.405 ± 0.103	0.736 ± 0.085	2.520 ± 0.453
ISFP $W > 0$	0.142 ± 0.014	0.183 ± 0.049	-0.388 ± 0.076	0.767 ± 0.053	2.651 ± 0.394

7.5.2 Sampsons's Monastery

We again evaluate our model on Sampson's Monastery data [51, 5], treating the sociomatrices from the 3 timepoints of the conflict as iid draws from the model to see if the ISFP can recover the structure of these networks. We ran our model with $\sigma_W = 1$, $\lambda = .1$, and $\gamma = 2.0$. As with any nonparametric model, inference can be sensitive to the choice of hyperparameters; in this case choosing $\lambda = 1$ or $\gamma = 1$ would produce an unreasonable number of latent features.

We compared to the LFRM with $\sigma_W = 1$, and IBP concentration parameter $\alpha = 3/H_{18}$, giving $E[K] = 3$ in the prior. For this dataset, we might expect that models that allow negative weights are too flexible, so we also tried a positive prior on the weights: $W_{ij} \sim$ Exponential(1). We ran all trials for 1000 iterations, discarding the first 500 as burn-in. The predictions from the remaining 500 samples were averaged for evaluation on test data, where we report train classification error, test classification error, posterior predictive log-likelihood, and AUC; see Table 7.1. In this case, restricting the weights to be positive does not improve performance, nor does it significantly affect the number of clusters used for either the LFRM or the ISFP. The LFRM gives models with around 6 features, whereas the ISFP typically gave 2-4. We were not able to reduce the number of features used by the LFRM with moderate alterations of the hyperparameters.

Example clusterings from our model are given in Figure 7.6. These are the final samples given in 3 of the 10 independent runs. The first two show the Loyal Opposition and the Young Turks clearly separated, the Outcasts in their own cluster, and with the Waverers mixed in. The third sample shows the Outcasts the Young Turks in one group and the Loyal Opposition in another.

In Table 7.1, we have compared LFRM to ISFP, and we see that ISFP performs better on test error, average test log likelihood, and area-under-the-curve (AUC). Here we see that

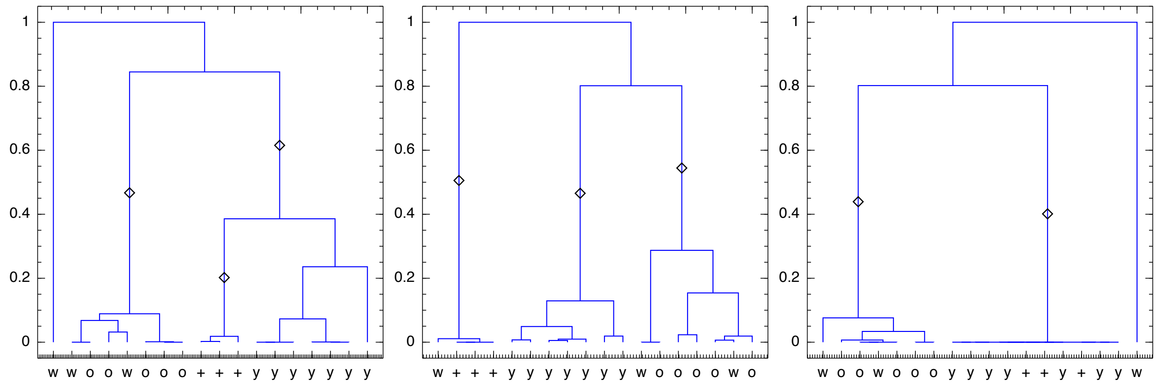

Figure 7.6: Example posterior trees and feature assignments from our model on the Monastery data. Diamonds correspond to features (that is, mutation events). The leaves are marked with labels assigned to the monks in Sampson's analysis: 'o' corresponds to Loyal Opposition, 'y' to Young Turks, '+' to Outcasts, and 'w' to Waverers.

$$
\begin{pmatrix} 1.3 & -0.2 & -1.7 \\ -1.0 & 1.8 & -2.2 \\ -1.5 & -0.4 & 1.2 \end{pmatrix}
\qquad
\begin{pmatrix} 1.5 & -1.1 & -0.8 & -1.6 & 1.2 \\ 1.1 & -0.1 & -3.4 & 1.1 & -1.0 \\ -0.2 & -2.0 & 1.2 & 0.2 & -1.5 \\ -1.6 & 1.3 & 1.1 & -0.7 & 0.2 \\ 1.7 & 1.2 & -0.4 & -2.5 & -0.2 \end{pmatrix}
$$

Figure 7.7: Example weight matrices W learned by the ISFP **(left)** and the LFRM **(right)** on the Monastery data.

there was little effect for restricting the weights to be positive, and that the ISFP used less features than the LFRM, but maintained the same predictive capability.

We show example W matrices from the ISFP and the LFRM (with prior $W \sim \mathcal{N}(0, \sigma_W)$) in Figure 7.7. The ISFP gives a weight matrix with positive diagonal entries, and negative off diagonal entries, signifying that the features found from the ISFP correspond to groups in which members like members who share their features and dislike those that do not. The three features we find are consistent with Sampson's analysis. The LFRM, on the other hand, includes features with negative weights on the diagonal, signifying that individuals sharing these features dislike each other, giving a less interpretable result.

7.6 Summary

The ISFP combines latent feature modelling with hierarchical clustering, giving a nonparametric latent feature model with hierarchical column structure. This more restrictive prior over binary matrices can improve the interpretability of the learned latent factors, and allows a hierarchical clustering interpretation as well. The beta-splitting prior used to construct the ISFP has potential applications to other hierarchical clustering problems; one advantage that it has over the Coalescent prior is that time variable of an internal node is only dependent on the number of datapoints that have split to each child of that node, thus improving tractability and allowing a wider range of MCMC procedures.

Chapter 8

Conclusion

There are many possible avenues for future research. For the RTJ sampler, we have used the prior for representing auxiliary parameters, though any probability density may be used. We have not yet explored the practicality of the Disjoint RTJ algorithm; this may be interesting as it would allow any sensible move to be made every iteration as in RTJK, while using roughly the same computational cost as RTJ1.

We have focused on Refractive Sampling as the black-box sampler for RTJ as it is simple to implement and efficient: however if computational resources are not an issue, then other sampling methods designed for multimodal densities such as parallel tempering will likely do well. There is also the opportunity for designing samplers specialized for densities such as (6.19).

RTJ relies on sampling from density functions that may be multimodal, thus it is important to ensure that we sample from these density functions efficiently. Refractive Sampling works well in this context, and future advances in such samplers will further improve the usefulness of RTJ.

Nevertheless, RTJ can be applied to nearly any model determination problem. Although there is computational overhead associated with using RTJ, this can be mitigated by caching the results of expensive computations that are shared across models. On a per-iteration basis, RTJ has been shown to perform as well as specialized samplers. Thus, RTJ is can be easily applied to any problem where computational cost is not an issue. Even so, implementations of RTJ that are designed for use on particular classes of models would see significantly reduced computational overhead.

RTJ opens the door for many models which otherwise would require RJMCMC, for example the ISFP. As RTJ may be applied to a large variety of problems, it is suitable for use within software packages for MCMC inference, and may serve as a powerful tool for simplifying the process of performing Bayesian inference for model averaging.

Bibliography

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, Dec. 1974.

[2] D. Aldous. Probability distributions on cladograms. *IMA Volumes in Mathematics and its Applications*, 76, 1996.

[3] R. Ash and C. Doleans-Dade. *Probability and measure theory.* 2000.

[4] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

[5] R. Breiger, S. Boorman, and P. Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of Mathematical Psychology*, 1975.

[6] T. Broderick, J. Pitman, and M. I. Jordan. Feature allocations, probability functions, and paintboxes. *Bayesian Analysis*, page 37, Jan. 2013.

[7] F. Caron. Bayesian nonparametric models for bipartite graphs. *Neural Informaiton Processing Systems*, 2012.

[8] J. Chan and I. Jeliazkov. MCMC estimation of restricted covariance matrices. *Journal of Computational and Graphical Statistics*, 2009.

[9] P. Damien, J. Wakefield, and S. Walker. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(2):331–344, Apr. 1999.

[10] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 1987.

[11] S. Ethier and R. Griffiths. The infinitely-many-sites model as a measure-valued diffusion. *The Annals of Probability*, 1987.

[12] S. Favaro and Y. Teh. MCMC for normalized random measure mixture models. *Stat. Sci*, 2012.

[13] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230, 1973.

[14] J. Foulds, C. DuBois, A. Asuncion, C. Butts, and P. Smyth. A dynamic relational infinite feature model for longitudinal social networks. *AI and Statistics*, 2011.

[15] D. Freedman. On the Asymptotic Behaviour of Bayes Estimates in the Discrete Case. *Annals of Mathematical Statistics*, 34(4):1386–1403, 1963.

[16] S. Ghosal, J. K. Ghosh, and R. V. Ramamoorthi. Posterior consistency of dirichlet mixtures in density estimation, Feb. 1999.

[17] S. Ghosal, J. K. Ghosh, and A. W. van der Vaart. Convergence rates of posterior distributions, Apr. 2000.

[18] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2011.

[19] P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 1995.

[20] P. Green and S. Richardson. Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, 28:355–375, 2001.

[21] J. E. Griffin. An adaptive truncation method for inference in Bayesian nonparametric models. *arXiv:1308.2045*, Aug. 2013.

[22] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. *Advances in Neural Information Processing Systems*, 2005.

[23] T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Technical Report 2005-01, University College London, 2005.

[24] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, Apr. 1970.

[25] N. Hjort. Nonparametric Bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, 1990.

[26] P. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

[27] M. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *arXiv preprint arXiv:1111.4246*, 2011.

[28] H. Ishwaran and L. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 2001.

[29] S. Jain and R. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 2000.

[30] S. Jain and R. M. Neal. Splitting and Merging Components of a Nonconjugate Dirichlet Process Mixture Model. Technical Report 0507, Department of Statistics, University of Toronto, 2005.

[31] H. Jeffreys. *The theory of probability.* 1961.

[32] M. Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 1969.

[33] P. Krivitsky. The latentnet package. *Statnet project*, 2009.

[34] W. Kruijer, J. Rousseau, and A. van der Vaart. Adaptive Bayesian density estimation with location-scale mixtures. *Electronic Journal of Statistics*, 4:1225–1257, 2010.

[35] J. Kunegis. KONECT: the Koblenz network collection. *Proceedings of the 22nd international Web Observatory Workshop*, 2013.

[36] A. Lijoi, I. Prünster, and S. G. Walker. Extending Doob's consistency theorem to nonparametric densities. *Bernoulli*, 10(4):651–663, Aug. 2004.

[37] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, Aug. 1989.

[38] P. McCullagh, J. Pitman, and M. Winkel. Gibbs fragmentation trees. *Bernoulli*, 14(4):988–1002, Nov. 2008.

[39] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087, Dec. 1953.

[40] J. W. Miller and M. T. Harrison. A Simple Example of Dirichlet Process Mixture Inconsistency for the Number of Components. *Neural Informaiton Processing Systems*, 2013.

[41] K. Miller, T. Griffiths, and M. Jordan. The phylogenetic indian buffet process: A non-exchangeable nonparametric prior for latent features. *arXiv preprint arXiv:1206.3279*, 2012.

[42] K. T. Miller, T. L. Griffiths, and M. I. Jordan. Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[43] R. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 2000.

[44] R. Neal. Slice sampling. *Annals of statistics*, 2003.

[45] R. Neal. MCMC for Using Hamiltonian Dynamics. *Handbook of Markov Chain Monte Carlo*, 2011.

[46] R. M. Neal. Density modeling and clustering using Dirichlet diffusion trees. *Bayesian Statistics*, 7:619–629, 2003.

[47] O. Papaspiliopoulos and G. Roberts. Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models. *Biometrika*, 2008.

[48] J. Pitman. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 1995.

[49] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

[50] C. Robert and G. Casella. *Monte Carlo statistical methods*. 2004.

[51] S. Sampson. Crisis in a cloister. *Unpublished doctoral dissertation, Cornell University*, 1969.

[52] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 1991.

[53] S. A. Sisson and Y. Fan. A distance-based diagnostic for trans-dimensional Markov chains. *Statistics and Computing*, 17(4):357–367, Aug. 2007.

[54] Y. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2007.

[55] Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2007.

[56] R. Thibaux and M. Jordan. Hierarchical beta processes and the Indian buffet process. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[57] S. G. Walker. Sampling the Dirichlet Mixture Model with Slices. *Communications in Statistics - Simulation and Computation*, 36(1):45–54, Jan. 2007.

[58] Z. Wang, S. Mohamed, and N. de Freitas. Adaptive Hamiltonian and Riemann Manifold Monte Carlo Samplers. *arXiv preprint arXiv:1302.6182*, page 10, Feb. 2013.

[59] S. Williamson. Restricting exchangeable nonparametric distributions. *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[60] E. J. Yeoh, M. E. Ross, S. A. Shurtleff, W. K. Williams, D. Patel, R. Mahfouz, F. G. Behm, S. C. Raimondi, M. V. Relling, A. Patel, and Others. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer cell*, 1(2):133–143, 2002.

Appendices

A Notation

Common notation used throughout this thesis:

$\mathbb{R}$	the real line
$\mathbb{N}$	the natural numbers
$\mathbb{N}_0$	the nonnegative integers
$\mathbb{Z}$	the integers
$\mathbf{1}$	a vector of ones
$\mathbf{1}^{(K)}$	a vector of ones of length K
$\mathbb{1}$	the indicator function
δ_a	the measure degenerate at a: $\delta_a(A) = 1$ iff $a \in A$
H_t	the t^{th} Harmonic number
$p()$	density function
$P()$	probability function
(X_T)	sequence or Markov chain