Nonparametric Bayesian Models for Machine Learning

Romain Jean Thibaux

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-130
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-130.html

October 14, 2008

Nonparametric Bayesian Models for Machine Learning

by

Romain Jean Thibaux

Diplôme d'Ingénieur (Ecole Polytechnique) 2001

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science
and the Designated Emphasis
in Communication, Computation and Statistics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2008

The dissertation of Romain Jean Thibaux is approved:

Professor Michael I. Jordan, Chair	Date
Professor Jim Pitman	Date
Professor Martin Wainwright	Date

University of California, Berkeley

Fall 2008

Abstract

Nonparametric Bayesian Models for Machine Learning

by

Romain Jean Thibaux

Doctor of Philosophy in Computer Science

and the Designated Emphasis in Communication, Computation and Statistics

University of California, Berkeley

Professor Michael I. Jordan, Chair

This thesis presents general techiques for inference in various nonparametric Bayesian models, furthers our understanding of the stochastic processes at the core of these models, and develops new models of data based on these findings. In particular, we develop new Monte Carlo algorithms for Dirichlet process mixtures based on a general framework. We extend the vocabulary of processes used for nonparametric Bayesian models by proving many properties of beta and gamma processes. In particular, we show how to perform probabilistic inference in hierarchies of beta and gamma processes, and how this naturally leads to improvements to the well known naïve Bayes algorithm. We demonstrate the robustness and speed of the resulting methods by applying it to a classification task with 1 million training samples and 40,000 classes.

Professor Michael I. Jordan, Chair Date

Acknowledgements

I am particularly grateful to my research advisor, Michael Jordan, for the tremendous freedom I have enjoyed during my time at Berkeley, and for his support and advice when came time to make important decisions, in research as well as in life. He and his group have given me a fun and stimulating time in the lab and at conferences, where our late night discussions started many of my projects. I have had, in particular, countless discussions with Simon Lacoste-Julien, Guillaume Obozinski and David Rosenberg about our respective research, and I am grateful for Simon and Guillaume's help during difficult times.

I had the pleasure to collaborate with many other people at Berkeley, Intel, Google and Microsoft along the way and I am grateful for their time, expertise and friendship: Carlos Guestrin, Sam Madden, Mark Paskin, Peter Bodik, Martin Wainwright, Francis Bach, Dan Klein, Slav Petrov, Leon Barrett, Luc Vincent, Emre Kiciman, Dave Maltz, John Platt and Erik Sudderth. I received great insights from discussions with Jim Pitman, Steve Evans, Lancelot James, and Yee Whye Teh who helped me clear important roadblocks.

I also met many wonderful people during my time at Stanford. I was introduced to research by my Master's advisor Daphne Koller as well as Eran Segal, Sebastian Thrun and Andrew Ng. Daphne's amazing Artificial Intelligence class got me started in the field and a large group of students from that class have become very close friends.

I had the good fortune to be raised in a happy, loving and supportive family: my parents Thierry and Catherine, who taught me by example how to work hard and be passionate, and my brothers and friends Thomas and Guillaume, who developed my creativity and my logical mind through our many games and arguments.

Last and foremost I am grateful to my wife Audrey for her love, for providing me with what I never knew I needed, and changing me in ways that I never thought possible, and to my daughter Chloe, whose learning algorithms are an inspiration for mine.

To Audrey

Contents

1 Introduction **1**

2 Monte Carlo Methods for the Dirichlet Process **3**

2.1 Introduction 3

2.2 Split and Merge 5

2.3 Method 8

2.4 Virtual Permutations 12

2.5 Natural Markov kernels on $\mathcal{S}^n$, $\mathcal{S}^\infty$ and $\mathcal{P}$ 17

2.6 The Split-Merge algorithm 19

2.7 Split-Merge variant via the Ebb-Flow chain 22

2.8 The Exchange algorithm 26

2.9 Experimental Comparisons 27

2.10 Conclusion 29

3 Lévy Processes **30**

3.1 Introduction 30

3.2 Lévy processes 33

3.3 Beta process 35

3.4 Gamma process 39

4 Naïve Bayes Classification with Infinitely Many Features 45

4.1 Hierarchies of Lévy processes 45

4.2 Inference in hierarchies of Lévy processes 47

4.3 Posterior hierarchical beta process 51

4.4 Posterior hierarchical gamma process 52

4.5 Image application 54

4.6 Conclusion 56

5 Conclusion 57

Bibliography 59

Index 62

Chapter 1

Introduction

Bayesian statistics frames any problem of data analysis as one of updating beliefs about the world given observations. Our beliefs are represented by a *probability distribution* over possible models of the world, and *probabilistic inference* is the mechanism by which our beliefs are updated.

This approach has the advantage of being *automatic*. In principle, once we have stated our initial beliefs, any data can be observed and any question can be answered unambiguously. For this promise to be realized however, a number of challenges must be overcome, in particular: 1) What language can we use to express the set of models we consider? This language should allow us to describe extremely large sets of models. Indeed however certain we are that a model is incorrect, we want to be able to consider it if evidence strongly points in its direction. 2) What language can we use to represent our beliefs? We need a compact way to state our preferences among this large set of models. 3) How can we efficiently perform the computations implied by probabilistic inference?

In this thesis we make some progress on these questions, by placing ourselves in the framework of *nonparametric* Bayesian inference, a field dedicated to placing probabilities on spaces larger than can be described by a finite dimensional vector, and using them to represent our beliefs over large sets of models. For example, the Gaussian process places a probability on the space of continuous functions, and the Dirichlet process on the space of *discrete distributions*. Both have been used as the backbone of many types of models.

Because Dirichlet processes have been studied for a long time, they are very well understood. In Chapter 2 we take advantage of this abundant theory to derive inference algorithms for the Dirichlet process, and propose a general method to derive similar algorithms in other contexts.

Finite dimensional (parametric) models are usually based on a small vocabulary of finite dimensional distributions such as the Gaussian, Poisson, Beta, Gamma, etc. which represent a compromise between expressivity and tractability. Many other distributions have been studied, but since they are less convenient mathematically and computationally, they are used only sparingly. Similarly, nonparametric Bayesian models are based on a small set of processes: mainly the Gaussian, Dirichlet and Poisson processes. The list is far from being as complete and as well studied as in the parametric case however. In Chapter 3 we introduce the theory of Lévy processes, a family to which several of these processes belong. In particular we focus on beta and gamma processes, surveying some of their known properties and exhibiting many new ones, as well as establishing links with existing models used in machine learning.

Finally in Chapter 4 we build on these tools to create hierarchies of beta and gamma processes, with associated inference algorithms, and show how they can be used to build good "naïve Bayes" classifiers. We present some empirical results for text and image classification.

Chapter 2

Monte Carlo Methods for the Dirichlet Process

2.1 Introduction

For the line of research based on DP mixtures to realize its potential, challenging computational issues must be faced. The underlying computational problem is one faced by many (generalized) clustering methods. In particular, DP mixtures attempt to find partitions of the data; as a Bayesian methodology they aim to find a distribution over partitions of the data. Historically, the first step towards computationally-viable inference procedures for DP mixtures came from the realization that the DP induces an *exchangeable* distribution over partitions—this led to the development of a Gibbs sampler for DP mixtures [Escobar and West, 1995]. The basic idea is that exchangeability allows the conditional probability associated with the assignment of a data point to be obtained by exchanging the data point with the final data point. The prior for the final data takes the form of a Pólya urn model, which is readily sampled.

Exchangeability is but one aspect of the DP probability model. A rich literature in stochastic process theory has studied distributions on various clustering-related combinatorial objects, including partitions, permutations and cycles, and has explored stochastic processes that converge to these distributions (see, e.g., [Pitman, 2002a]). This literature would seem to provide fertile ground for the development of Bayesian inference procedures that exploit aspects of the DP model beyond exchangeability.

Indeed, the Gibbs sampler is too slow for large-scale applications of DP mixtures; the reassignment of any single data point is often highly improbable even in a suboptimal configuration. The most effective algorithms have been based on procedures—known as "split-and-merge" algorithms—that reassign multiple data points at each step [Jain and Neal, 2000; Dahl, 2003]. Currently, however, there is no theoretical guide to the development of such algorithms. Their correctness is assured by placing them within a Metropolis-Hastings procedure, but the specific choices of moves in existing algorithms are based on intuitions. It would be desirable to find theoretical support for those intuitions and to obtain a broader picture of the space of reasonable algorithms.

We make the following contributions. We provide a general probabilistic foundation to exploit the stochastic process literature when designing inference procedures. We show that the specific split-and-merge algorithms of [Jain and Neal, 2000] and [Dahl, 2003] can be justified within this theoretical framework, and we show that the same framework generates new algorithms which are competitive with and complementary to the existing algorithms.

This chapter is organized as follows. We frame the problem by introducing Dirichlet process mixtures and the Split-Merge algorithm (Sec. 2.2). We outline a general method to derive efficient algorithms from stochastic processes on a larger space (Sec. 2.3). Before we can use this method, we need to review the many properties of a particular such space with a direct relationship to Dirichlet processes: the space of *virtual permutations* [Kerov *et al.*, 1993] (Sec. 2.4 and 2.5). We then apply the method to a Fragmentation-Coagulation process on these virtual permutations, obtaining a natural derivation of Split-Merge (Sec. 2.6). Applying the method instead to a different process called Ebb-Flow, we derive a new variant of Split-Merge (Sec. 2.7). Finally we discuss in less detail several other variants (Sec. 2.8) and compare these algorithms experimentally (Sec. 2.9).

2.2 Split and Merge

For other introductions to Dirichlet process mixtures and the Split-Merge algorithm, see [Neal, 1998] and [Jain and Neal, 2000].

2.2.1 Dirichlet Process Mixtures

We consider n observations $y_{1:n} = (y_1, \dots y_n)$ drawn from a mixture distribution. Each observation t belongs to a mixture component – or *cluster* – C^i. Each cluster is characterized by a parameter η_i, drawn from a prior H, and observations from this cluster are i.i.d. $f(.|\eta_i)$. We also call c_t the cluster that observation t belongs to, so $c_t = i \iff t \in C^i$, and let $|c_{1:n}|$ be the number of clusters implied by $c_{1:n}$. We also note $n_i = |C^i|$ the size of cluster i. Since the label i of each cluster is unidentifiable, we identify any two vectors $c_{1:n}$ equal up to relabelling of the clusters. For instance,

$$(1,1,2,1,3,3) \quad = \quad (2,2,4,2,7,7).$$

In other words we consider these equivalent vectors as different names for the common partition of 1:n that they induce. All formulas involving $c_{1:n}$ could be rewritten to only involve partitions, though at great cost in readability. In summary:

$$\begin{aligned}\eta_i &\overset{\text{ind}}{\sim} H \\ y_i &\overset{\text{ind}}{\sim} f(.|\eta_{c_i})\end{aligned}$$

We restrict ourselves to f and H being *conjugate exponential families*:

$$\begin{aligned}f(y|\eta) &= h(y)\exp\left[\psi(\eta)^T T(y) - A(\eta)\right] \\ H(\eta) &= \exp\left[\mu^T\psi(\eta) - \nu A(\eta) - B(\mu,\nu)\right]\end{aligned}$$

For a cluster $A \subset \{1, \dots n\}$ we also define the marginal distribution:

$$\begin{aligned}q(y_A) &= \int f(y_A|\eta)H(\eta)d\eta \qquad (2.1) \\ &\propto \exp\left[B(\mu + \sum_{l\in A} T(y_l), \nu + |A|) - B(\mu,\nu)\right]\end{aligned}$$

The last part of the model is the prior distribution of $c_{1:n}$. Our goal in the rest of the chapter will be to obtain the posterior distribution of $c_{1:n}$ given the n observations $y_{1:n}$. Let $c_{1:n}$ be distributed according to the Chinese Restaurant Process, which can be iteratively constructed from its conditionals:

$$c_{n+1}|c_{1:n} \sim \begin{cases} c_{n+1} = i & \text{w.p. } \frac{n_i}{n+\alpha} \\ c_{n+1} = |c_{1:n}| + 1 & \text{w.p. } \frac{\alpha}{n+\alpha} \end{cases} \qquad (2.2)$$

The Chinese Restaurant Process is an instance of an *urn model*. A draw from an urn $((n_i)_{i\leq k}; \alpha)$ returns i with probability $n_i/(\alpha + \sum_i n_i)$ or returns $k+1$ with probability $\alpha/(\alpha + \sum_i n_i)$, and the urn is updated so that n_i is incremented by 1. The Chinese Restaurant Process is an urn started at $(;\alpha)$, and we will encounter other types of urns.

The distribution (2.2) is *exchangeable*: the probability of $c_{1:n}$ does not depend on the ordering. In particular, the conditional distribution of c_t given $c_{1:n\setminus\{t\}}$ is the same as that of c_n given $c_{1:n-1}$. Applying Bayes' rule to (2.2), we obtain:

$$c_{n+1}|c_{1:n}, y_{1:n+1} \sim \begin{cases} c_{n+1} = i & \text{w.p. } \propto n_i q(y_{C^i \cup \{n+1\}}) \\ c_{n+1} = |c_{1:n}| + 1 & \text{w.p. } \propto \alpha q(y_{\{n+1\}}). \end{cases} \tag{2.3}$$

This formula allows us to sample any c_t from its conditional posterior distribution. The Gibbs sampler [Neal, 1998] resamples each c_t in turn using (2.3), and in the limit yields independent samples from the posterior.

2.2.2 Split and Merge

The above Gibbs sampler can mix very slowly if two cluster parameters are similar; the chain cannot easily move from a configuration where the clusters are merged to one where they are separate. This motivated Jain and Neal [Jain and Neal, 2000] to propose a Markov chain where clusters are split and merged in one step. Here we present the simplified version by Dahl [Dahl, 2003], and discuss the original method in Sec. 2.6.

First, choose two data points r and s at random among $1{:}n$. If r and s belong to two different clusters, simply merge the two clusters while leaving all the other clusters unchanged. If r and s belong to the same cluster C^m, split it into two in the following way.

Initialize the two clusters to $C^i = \{r\}$ and $C^j = \{s\}$.

Sequentially allocate each remaining element t of C^m to either C^i or C^j with probability proportional to $|C^i|q(y_{C^i \cup \{t\}})$ and $|C^j|q(y_{C^j \cup \{t\}})$ respectively.

Accept the move with the appropriate Metropolis-Hastings probability to ensure convergence to the posterior.

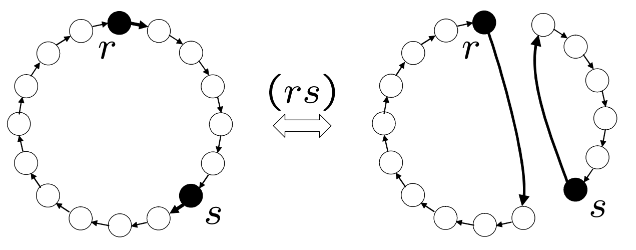

Figure 2.1: Composing a permutation with a transposition (rs) merges the clusters of r and s, or splits their common cluster.

Where does this algorithm come from? Any distribution can be used as a proposal for Metropolis-Hastings, so why this particular form? One hint comes from the similarity between this algorithm and the effect of composing a permutation with the transposition (rs) exchanging two elements r and s. Indeed a permutation consists of a set of cycles. If r and s belong to different cycles, these are merged, otherwise their common cycle is split in two (see fig. 2.1).

In fact we show (Sec. 2.6), that Split-Merge can be directly derived from this composition operation. Not only does this justify the form of the proposal, but it provides us with a general method to mechanically derive such algorithms. First we describe this method in general terms (Sec.2.3), before showing how Split-Merge can be naturally derived in this way (Sec. 2.6). Then we apply the same method to other operations as well as deriving several new algorithms (Sec. 2.7 and 2.8).

2.3 Method

A fundamental operation of Bayesian statistics is posterior inference, where we want to compute the posterior $P(x|y)$ over a quantity of interest $x \in \mathcal{X}$ given a prior $P(x)$, and a likelihood $P(y|x)$. However usually the space of x is too large for the posterior to be computed by enumerating and normalizing $P(x)P(y|x)$. We can get an estimate of the posterior with Markov chain Monte Carlo, but how can we design fast mixing Markov chains for $P(x|y)$? Since the prior usually has a much more symmetrical structure than the posterior, finding good Markov chains for $P(x)$ is usually much easier, so we consider how to transform them into Markov chains for $P(x|y)$.

2.3.1 Metropolis-Hastings

For a Markov kernel K, we note $K(\bar{x}, x)$ the transition probability from the current state $\bar{x}$ to the next state x. In many cases we will be interested in kernels K mixing to the prior that are P-*reversible*, that is

$$\forall\, x, \bar{x} \quad P(x)K(x, \bar{x}) \;=\; P(\bar{x})K(\bar{x}, x),$$

because this property provides a quick proof that P is stationary for K. In principle, we can apply the Metropolis-Hastings algorithm, which transforms K into a reversible Markov process $M(K)$ with any prespecified stationary distribution which we take to be the posterior. To sample from $M(K)$, one samples from K and accepts the move from $\bar{x}$ to x with probability $\min(1, R(\bar{x}, x))$ where

$$\begin{aligned} R(\bar{x}, x) &= \frac{K(x, \bar{x})P(x|y)}{K(\bar{x}, x)P(\bar{x}|y)} \\ &= \frac{P(y|x)}{P(y|\bar{x})} \qquad \text{by } P\text{-reversibility.} \end{aligned} \tag{2.4}$$

What we have won by starting from a P-reversible kernel rather than any other kernel is that we only need to compute the likelihood. It is usual when using Metropolis-Hastings to need the posterior only up to a multiplicative constant, but (2.4) does not even require the prior. However, simply using Metropolis-Hastings in this fashion will usually result in a very slow chain. Since any distance between K and $M(K)$ is paid for by waiting, we want to start from a K whose stationary distribution is as close to the posterior as possible. Let $\hat{K}$ be a modification of K, with the same pattern of non-zero transition probabilities (we discuss in the next section how to build $\hat{K}$). Applying Metropolis-Hastings to $\hat{K}$ gives us the following acceptance ratio:

$$\begin{aligned} R(\bar{x}, x) &= \frac{\hat{K}(x, \bar{x}) P(x|y)}{\hat{K}(\bar{x}, x) P(\bar{x}|y)} \\ &= \frac{\hat{K}(x, \bar{x})}{K(x, \bar{x})} \frac{K(\bar{x}, x)}{\hat{K}(\bar{x}, x)} \frac{P(y|x)}{P(y|\bar{x})} \qquad \text{by } P\text{-reversibility.} \end{aligned} \tag{2.5}$$

Observe that expression (2.5) only involves the ratio between K and $\hat{K}$, and that we only need to be able to compute the likelihood, not the posterior or the prior.

2.3.2 Likelihood reweighting

To really take advantage of eq. (2.5), we want to design a kernel $\hat{K}$ that is a simple modification of K, yet whose stationary distribution is as close to $P(x|y)$ as possible. We consider the following transformation, which we call *likelihood reweighting*:

$$\begin{aligned} \forall \bar{x} \in B \quad \hat{K}(\bar{x}, x) &= \frac{1}{Z(\bar{x})} K(\bar{x}, x) P(y|x) \\ \text{where} \quad Z(\bar{x}) &= \sum_x K(\bar{x}, x) P(y|x) \end{aligned} \tag{2.6}$$

If K happens to be a *Gibbs kernel* for P, then $\hat{K}$ has a stationary distribution *exactly* equal to $P(x|y)$. A Gibbs kernel relies on a decomposition of x into variables, and resamples ones of these variables from its conditional distribution given all other variables. If we call B these other variables, then $K(\bar{x}, x) = P(x|B(x) = B(\bar{x}))$ and $\hat{K}(\bar{x}, x) = P(x|B(x) = B(\bar{x}), y)$, which is $P(x|y)$-reversible.

If K is not a Gibbs kernel, there is no guarantee, but we can still use likelihood reweighting as a heuristic. In the end, the acceptance ratio corrects for any discrepancies and ensures that $M(\hat{K})$ converges exactly to $P(x|y)$. Our experience is that this likelihood reweighting is usually very beneficial.

Often likelihood reweighting itself is intractable and must be approximated, however since exact likelihood reweighting is only a heuristic, a good approximation is all we need.

2.3.3 Auxiliary variables

Distributions on discrete structures such as partitions, trees, or walks often simplify greatly in the limit of infinite size or infinite time when rescaling appropriately. Even though the mathematics involved is typically more advanced, the limit objects enjoy additional symmetries and simple closed forms while their finite counterparts often involve intractable combinatorics. This simplicity makes it easier to discover or study stochastic processes acting on these spaces.

In particular, we are interested in cases where the limit object X *contains* the finite object x as one of its parts, i.e. x is completely determined given X. For example an infinite tree X induces a finite tree x over any finite subset of its points. In this case any family of consistent distributions $P(x)$ converges to a limit $P(X)$ as x gets large. The distribution $P(x)$ can then be interpreted as the marginal on x of $P(X)$. If a Markov process K acts on X and yields samples from $P(X)$, it automatically provides samples from $P(x)$.

Since X is infinite, an algorithm cannot manipulate it explicitly. To obtain instead a process on x only, we must *project* K onto x by interleaving an operation Γ that resamples X from $P(X|x)$. Γ ensures that we “forget” X between two applications of K. We can then view $G \circ \Gamma$ as a Markov process on x that first generates an auxiliary variable X, performs a step of K on X, which modifies x, and then throws X away. It turns out that we can often marginalize out X in this sequence and remove the need to generate X even temporarily.

The overall method is summarized in fig. (2.2).

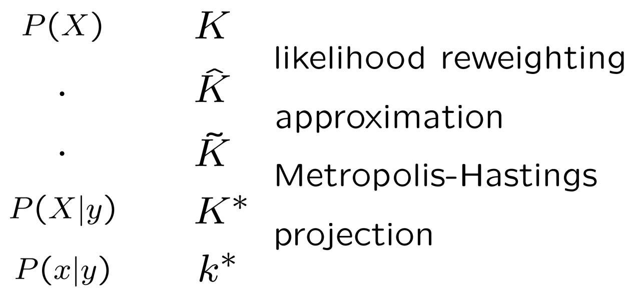

Figure 2.2: A method to derive MCMC algorithms mixing to the posterior on x from a stochastic process K mixing to the prior on X. On the left is the equilibrium distribution at each step.

2.4 Virtual Permutations

We will apply the above method to several Markov kernels to derive posterior inference algorithms for Dirichlet process mixtures, where the hidden variable x corresponds to the clustering vector $c_{1:n}$. We are faced with a choice of several auxiliary variables X: partitions of 1, and permutations. We choose to introduce and review the most general auxiliary variable, the space of *virtual permutations*, which subsumes both of these. The Markov chains we will consider are each most naturally described in one of these spaces, though they have natural counterparts in the others.

2.4.1 The space $\mathcal{S}^\infty$

Let $\mathcal{S}_A$ be the space of permutations of $A \subset \mathbb{N}$. If $A = 1{:}n$ we simply write $\mathcal{S}^n$ for $\mathcal{S}_{1:n}$. We note $|A|$ the size of A. Following [Kerov *et al.*, 1993], for any two sets $A \subset B$ we define the projection π_A^B that maps a permutation $\omega_B \in \mathcal{S}_B$ to $\omega_A \in \mathcal{S}_A$ by removing from the cycles of ω_B all elements of $B \setminus A$. When B is clear from context we simply write π_A^B as π_A. For example:

$$(1365)(247)(8) \xmapsto{\pi_{1:5}} (135)(24)$$

Two permutations are called *coherent* if one is the projection of the other. A *virtual permutation* is a coherent set of permutations $\omega = (\omega_{1:N})_N \in \mathcal{S}^\infty$. It can be thought of as a permutation of $\mathbb{N}$. In particular, ω induces a permutation on any finite subset of $\mathbb{N}$ by projection from $1{:}N$ for N large enough.

Although infinite ω has *cycles*, that is a partition of $\mathbb{N}$ into *clusters*, which we note C^i, and for each cluster a cyclic order[1]. We use the same notations C^i and c as that of Sec. 2.2.1 to represent the partition of $\mathbb{N}$. We also note c_A and C^i_A their restriction to a subset A of $\mathbb{N}$. In the common case that $A = 1{:}n$ we still use the shorter notation $n_i = |C^i_{1:n}|$.

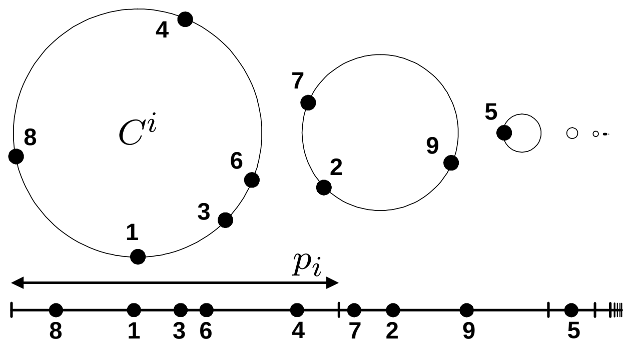

Figure 2.3: A virtual (infinite) permutation ω and its stick representation, with 9 data points highlighted. $\omega_{1:9} = (81364)(729)(5)$. p_i is the relative length of cluster C^i.

As is shown in fig. (2.3), one can think of ω as having infinitely many cycles, each isomorphic to a continuous circle, on which the elements of $\mathbb{N}$ form a dense subset.

[1]Contrary to the finite case, this cyclic order will in general not admit a successor function.

The projection of ω on $\omega_{1:n}$ can be read from the positions of the elements $1{:}n$ on the cycles. More pathological cases where this picture is inaccurate are possible, but we now define a probability distribution over this space which will ensure they occur with probability 0.

2.4.2 The μ^α Distribution

Let $\alpha > 0$ and μ^α be the distribution on $\mathcal{S}^\infty$ defined by its coherent marginals on $\mathcal{S}_A$ for each finite $A \subset \mathbb{N}$:

$$\mu^\alpha(\omega_A) = \frac{\alpha^{|\omega_A|}}{(\alpha)_{|A|}} \tag{2.7}$$

where $(\alpha)_n = \alpha(\alpha{+}1)\dots(\alpha{+}n{-}1)$. In particular μ^1 induces the uniform distribution on every $\mathcal{S}_A$, while $\alpha > 1$ encourages permutations with more clusters, and $\alpha < 1$ less. This distribution is exchangeable (or central) since it does not depend on the identity of the elements of A. This implies that we can take $A = 1{:}n$ in all subsequent formulas without loss of generality.

Permutations with the same number of cycles, and in particular permutations with the same partition, have the same probability. The marginal of μ^α on the space of partitions can therefore be obtained by counting the number of permutations inducing a given partition.

$$\mu^\alpha(c_{1:n}) = \frac{\alpha^{|c_{1:n}|}}{(\alpha)_n} \prod_i (n_i - 1)! \tag{2.8}$$

We can marginalize further by counting the number of partitions having the same set of cluster sizes. Let m^j be the number of clusters of $c_{1:n}$ of size j, and $|m| =$

$|c_{1:n}| = \sum_j m^j$. We obtain the *Ewens Sampling Formula*[2], ESF(α) [Ewens, 1972]:

$$\mu^\alpha(m) = \frac{\alpha^{|m|}}{(\alpha)_n} n! \prod_{j \le n} \left(\frac{1}{j}\right)^{m^j} \frac{1}{m^j!}$$

We can also compute its conditionals. From (2.7) we get $\mu^\alpha(\omega_{1:(n+1)}|\omega_{1:n})$:

$$\begin{cases} n+1 \text{ inserted after } t \in 1{:}n & \text{w.p. } \frac{1}{n+\alpha} \\ (n+1) \text{ is a cycle by itself} & \text{w.p. } \frac{\alpha}{n+\alpha} \end{cases} \tag{2.9}$$

and from (2.8) or (2.9) we get that $\mu^\alpha(c_{n+1}|c_{1:n})$ is equal to the Chinese Restaurant Process (2.2), which we repeat for convenience:

$$c_{n+1}|c_{1:n} \quad \sim \quad \begin{cases} c_{n+1} = i & \text{w.p. } \frac{n_i}{n+\alpha} \\ c_{n+1} = |c_{1:n}| + 1 & \text{w.p. } \frac{\alpha}{n+\alpha} \end{cases} \tag{2.10}$$

Eq. (2.9) extends (2.10) to permutations, and justifies the name *Chinese Restaurant Process* since elements inserting themselves into cycles can be thought of as customers sitting around circular tables. The connection shows that both permutations of $1{:}n$ and virtual permutations of $\mathbb{N}$ are valid auxiliary variables for the model of Sec. (2.2.1) since $\mu^\alpha(\omega)$ and $\mu^\alpha(\omega_{1:n})$ recover the Chinese Restaurant Process as one of their marginals.

Eq. (2.10) also shows that $\frac{n_i}{n+\alpha}$ follows a bounded martingale. Indeed it is bounded by 0 and 1, and its expectation after increasing n by 1 is

$$\frac{n_i+1}{n+\alpha}\frac{n_i}{n+\alpha} + \frac{n_i}{n+\alpha}\frac{n-n_i+\alpha}{n+\alpha} \quad = \quad \frac{n_i}{n+\alpha}, \text{ i.e. its current value.}$$

Therefore it converges almost surely as $n \to \infty$ to a random limit p_i called the *relative*

[2]The ESF is also known as the Multivariate Ewens Distribution

cluster size. Applying (2.10) to $\mathbb{N} \setminus \{t\}$ leads to:

$$\mu^\alpha(c_t = i | c_{\mathbb{N}\setminus\{t\}}) \;=\; \mu^\alpha(c_t = i|p) \;=\; p_i \tag{2.11}$$

$p = (p_i)_i$ is thus the hidden variable conditionally on which the exchangeable distribution of c becomes i.i.d., in agreement with de Finetti's theorem, and is another valid auxiliary variable for our model. Since we have identified the vectors $c_{1:n}$ equal by relabelling, p is really a *set* of values.

It is known that p is *saturated*, that is $\sum_i p_i = 1$ almost surely. What is more the martingale followed by $\frac{n_i}{n+\alpha}$ is equal to posterior inference on p_i when giving it a Beta distribution. Therefore:

$$p_i | c_{1:n} \sim \mathrm{Beta}(n_i, n - n_i + \alpha) \tag{2.12}$$

However the joint distribution of p as a set of values is not convenient. Instead we use the vector $p^\downarrow = (p_i^\downarrow)_{i\in\mathbb{N}}$ of values of p ranked in decreasing order. $p^\downarrow$ has the *Poisson-Dirichlet* distribution $\mathrm{PD}(\alpha)$ [Kingman, 1975].

Alternatively, we can use the vector $\tilde{p} = (\tilde{p}_i)_{i\in\mathbb{N}}$ obtained by drawing the values p_i without replacement with probabilities given by the p_i's themselves. This operation is called a *size-biased permutation.* $\tilde{p}$ has the *Griffiths-Engen-McCloskey* distribution $\mathrm{GEM}(\alpha)$, which arises from the following stick-breaking process[3]:

$$p_i = V_i \prod_{j=1}^{i-1} (1 - V_j) \quad \text{where} \quad V_i \sim \mathrm{Beta}(1, \alpha)$$

We note $\mathcal{P}$ the space of positive vectors with sum 1, to which $p^\downarrow$ and $\tilde{p}$ belong.

[3]For this reason $\mathrm{GEM}(\alpha)$ is sometimes called the Stick-Breaking distribution, even though other stick-breaking schemes exist. Stick-breaking processes are also known as Residual Allocation Models (RAM).

2.5 Natural Markov kernels on $\mathcal{S}^n$, $\mathcal{S}^\infty$ and $\mathcal{P}$

2.5.1 The cyclic reshuffle kernel

Let Δ be the kernel $\Delta(\bar{\omega}, \omega) \overset{\text{def}}{=} P(\omega | c = \bar{c})$ that resamples ω given c. In other words, Δ resamples the cyclic order of each cluster uniformly at random, leaving clusters unchanged. It is a Gibbs kernel for any μ^α, as is clear from (2.7).

2.5.2 The transposition kernel

Let $R_{(rs)}(\omega, \omega') \overset{\text{def}}{=} \delta_{\omega \circ (rs)}(\omega')$ be the kernel that composes (deterministically) ω with a fixed transposition (rs). $R_{(rs)}$ is involutary, with the effect discussed in Sec. 2.2.2 and illustrated in fig. (2.1): if r and s belong to different clusters of ω, these are merged, otherwise their common cluster is split in two.

The uniform distribution $\mu^1(\omega_{1:n})$ is stationary for $R_{(rs)}$ on $\mathcal{S}^n$ and we can obtain μ^α using Metropolis-Hastings, i.e. accepting split moves only with probability α (if $\alpha < 1$), or merge moves w.p. $\frac{1}{\alpha}$ (if $\alpha > 1$). This can be checked using (2.7). Since this is true for any n, $\mu^1(\omega)$ is also stationary for $R_{(rs)}$ on $\mathcal{S}^\infty$.

2.5.3 Fragmentation-Coagulation

We can define the kernel R_n which draws a pair (rs) uniformly at random among $1{:}n$ then applies $R_{(rs)}$. If we extend the definition of $R_{(rs)}$ to allow r and s to be any point on the closure of a cycle of ω (the continuous circles in fig. 2.3), we can also define R_∞ which selects r and s uniformly at random on the cycles and applies $R_{(rs)}$.

Since μ^1 is invariant for any $R_{(rs)}$, it is also invariant for any R_n and R_∞, and μ^α is invariant if we accept split moves with probability $\min(1, \alpha)$ and merge moves with probability $\min(1, 1/\alpha)$. [Tsilevich, 1998] uses such random transpositions to study properties of μ^1, showing that it is the only saturated stationary distribution for all

R_n.

Since R_∞'s effect on p depends on ω only through p, p follows a Markov chain, called *Fragmentation-Coagulation* [Mayer-Wolf *et al.*, 2002]:

∘ Pick two clusters i and j at random with probabilities given by p

∘ If $i \neq j$, merge the two clusters into a new cluster $p_m = p_i + p_j$

∘ If $i = j$, split the cluster into Up_i and $(1-U)p_i$ where $U \sim \mathrm{U}[0,1]$

This chain is also a chain on $p^\downarrow$ if we re-rank at each step, in which case it mixes to PD(1), or to PD(α) when adding Metropolis-Hastings acceptance probabilities as above [Diaconis *et al.*, 2004]. We will see that this chain and the Split-Merge algorithm are not just very similar, they are essentially the same.

2.5.4 Ebb-Flow

[Gnedin and Kerov, 2001] show that GEM(α) is the unique stationary distribution for the following chain, whose equilibrium is reversible [Pitman, 2002b]:

$$\begin{aligned} \text{Draw } W &\sim \text{Beta}(1,\alpha) \\ p|\bar{p} &\sim \begin{cases} (W, \bar{p}_1 - W, \bar{p}_2, \bar{p}_3, \ldots) & \text{if } W < \bar{p}_1 \\ (\bar{p}_1 + \bar{p}_2, \bar{p}_3, \bar{p}_4, \ldots) & \text{otherwise} \end{cases} \end{aligned}$$

They call this chain split-and-merge but to avoid confusion in this article we will call it "Ebb-Flow" (interpreting W as the level of the tide leaving a mark on the beach, then a higher mark can be set only after all lower marks have been erased).

By itself Ebb-Flow mixes exceeding slowly since clusters with a large index must wait for a very unlikely event (a series of very high "tides") before being modified. There is an easy fix however which is to interleave a size-biased permutation (SBP), which leaves GEM(α) invariant and ensures that each cluster i has probability p_i of moving to the front. Since the ordering is lost at each iteration, Ebb-Flow composed with SBP induces a Markov chain on $p^{\downarrow}$, if we rank between each iteration.

This Markov chain mixes to $\mathrm{PD}(\alpha)$. However, contrary to Fragmentation-Coagulation, it does so for any α without wasting time rejecting moves. Since for $\alpha = 1$ the two chains are identical, Ebb-Flow appears to be a better way to extend Fragmentation-Coagulation to $\alpha \neq 1$ than is Metropolis-Hastings. Ebb-Flow also exemplifies the advantages of introducing the limit spaces $\mathcal{P}$ and $\mathcal{S}^{\infty}$: contrary to Fragmentation-Coagulation, Ebb-Flow has no known counterpart on $\mathcal{S}^n$.

We can extend Ebb-Flow to a Markov chain on c in a natural way: when splitting, allocate each element of $\bar{C}_1$ to C_1 or C_2 with probabilities $W/\bar{p}_1$ and $(\bar{p}_1 - W)/\bar{p}_1$, otherwise merge the two clusters $\bar{C}_1$ and $\bar{C}_2$, maintaining all other clusters constant. We call Q this kernel on c. Its equilibrium is also reversible.

2.6 The Split-Merge algorithm

Any of the previous Markov chains are possible starting points to apply the method of Sec. 2.3. In fact, any composition of them leaves μ^{α} invariant and is also a valid starting point.

In this section, we consider $S_{(rs)} = R_{(rs)} \circ \Delta$. If r and s are in the same cluster C^m, and if U is the proportion of the cycle between r and s (clockwise) after applying Δ, we have $U \sim \mathrm{U}[0,1]$. Applying $R_{(rs)}$ then cuts the cycle at r and s, forming two cycles of size $p_i = Up_m$ and $p_j = (1-U)p_m$. Conditionally on U, each element of C^m is sent to C^i or C^j with probability U or $1-U$, except for r and s which are sent deterministically to C^i and C^j respectively. Marginalizing U, elements are sent to C^i and C^j with probabilities given by an urn started at $(1,1;0)$. If r and s are in different clusters, the two clusters are merged, deterministically.

As the above description shows, $S_{(rs)}$'s effect on $c_{1:n}$ only depends on $c_{1:n\cup\{r,s\}}$, therefore $S_{(rs)}$ induces a Markov kernel on $c_{1:n\cup\{r,s\}}$, which we also call $S_{(rs)}$.

2.6.1 Approximate likelihood reweighting

For simplicity we assume that r and s do not belong to $1{:}n$ and since the merge move is trivial we turn to the split move. Let $c_r = c_s = m$, and C^m the cluster we split. Without loss of generality (by exchangeability) we rename the data points so that $C^m_{1:n} = 1{:}d$. To simplify notations, we write $S_{(rs)}(c_{1:n\cup\{r,s\}})$ for the transition probability $S_{(rs)}(\bar{c}_{1:n\cup\{r,s\}}, c_{1:n\cup\{r,s\}})$ since $\bar{c}_{1:n\cup\{r,s\}}$ is fixed throughout. Conversely, we write $S_{(rs)}(\bar{c}_{1:n\cup\{r,s\}})$ for the reverse transition probability $S_{(rs)}(c_{1:n\cup\{r,s\}}, \bar{c}_{1:n\cup\{r,s\}})$. Since $S_{(rs)}$ only modifies $c_{1:d}$, we concentrate on $S_{(rs)}(c_{1:d})$. Reweighting exactly by the likelihood would yield the following distribution:

$$S_{(rs)}(c_{1:d}|y_{1:d}) = \prod_{t=1}^{d} S_{(rs)}(c_t|c_{1:(t-1)}, y_{1:d})$$

from which we might hope to sample iteratively. Alas, $S_{(rs)}(c_t|c_{1:(t-1)}, y_{1:d})$ is intractable. We thus resort to approximating it by dropping some of the dependencies on y. We define

$$\hat{S}_{(rs)}(c_{1:d}) \stackrel{\text{def}}{=} \prod_{t=1}^{d} S_{(rs)}(c_t|c_{1:(t-1)}, y_{1:t}) \tag{2.13}$$

This approximation allows us to allocate points to C^i and C^j incrementally. Consistent with our notation, let $C^i_{1:(t-1)}$ be the set of elements already allocated to cluster i among the first $t-1$ elements of C^m. Using (2.1) we get:

$$\begin{aligned} S_{(rs)}(c_t|c_{1:(t-1)}, y_{1:t}) &\propto S_{(rs)}(c_t|c_{1:(t-1)})P(y_{1:t}|c_{1:t}) \\ &\propto \begin{cases} (|C^i_{1:(t-1)}|+1)q(y_{C^i_{1:(t-1)}\cup\{t\}}) \\ (|C^j_{1:(t-1)}|+1)q(y_{C^j_{1:(t-1)}\cup\{t\}}) \end{cases} \end{aligned} \tag{2.14}$$

since $S_{(rs)}(c_t = i|c_{1:(t-1)})$ is the expectation of $U|c_{1:(t-1)} \sim \text{Beta}(|C^i_{1:(t-1)}|+1,|C^j_{1:(t-1)}|+1)$. The probability for the reverse (merge) move is

$$S_{(rs)}(c_{1:n}, \bar{c}_{1:n}) = \hat{S}_{(rs)}(c_{1:n}, \bar{c}_{1:n}) = 1.$$

2.6.2 Metropolis-Hastings

Applying (2.5), the Metropolis-Hastings acceptance ratio to correct for the error is $R(\bar{c}_{1:n}, c_{1:n})$ where, for a split move:

$$\begin{aligned} R(\bar{c}_{1:n}, c_{1:n}) &= \frac{\hat{S}_{(rs)}(\bar{c}_{1:n})}{S_{(rs)}(\bar{c}_{1:n})} \frac{S_{(rs)}(c_{1:n})}{\hat{S}_{(rs)}(c_{1:n})} \frac{P(y|c_{1:n})}{P(y|\bar{c}_{1:n})} \\ &= \alpha \frac{q(y_{C^i_{1:n}})q(y_{C^j_{1:n}})}{q(y_{C^m_{1:n}})} \frac{n_i!n_j!}{(n_m+1)!} \frac{1}{\hat{S}_{(rs)}(c_{1:n})} \end{aligned} \tag{2.15}$$

where the last term is computed iteratively as we sample $c_{1:d}$. Expression (2.15) is for r and s non data points. If we take them among $1{:}n$, the only difference is that the first two choices (the placement of r and s) are deterministic. In particular this leads to replacing the ratio of factorials in (2.15) by $\frac{(n_i-1)!(n_j-1)!}{(n_m-1)!}$, giving the acceptance probability for the Split-Merge algorithm of [Dahl, 2003]. Choosing r and s among $1{:}n$ or not makes little difference in practice.

The original Split-Merge algorithm of [Jain and Neal, 2000] uses a more complex proposal distribution. Even though the authors do not motivate their proposal, it is clearly an attempt to approximate $S_{(rs)}(c_{1:d}|y_{1:d})$. Rather than using (2.13), they introduce an auxiliary variable $\hat{c}_{1:n}$ approximately drawn from $S_{(rs)}(c_{1:d}|y_{1:d})$ using Gibbs sampling, and define:

$$\tilde{S}_{(rs)}(c_{1:d}) \overset{\text{def}}{=} \prod_{t=1}^{d} S_{(rs)}(c_t|c_{1:(t-1)}, c_{(t+1):n} = \hat{c}_{(t+1):n}, y_{1:n}) \tag{2.16}$$

It is unclear which of the two approximations is more accurate, but since likelihood reweighting is only a heuristic, it makes more sense to follow the simpler and somewhat faster proposal of [Dahl, 2003].

2.6.3 Projection

We note $S^*_{(rs)}$ the kernel obtained after applying MH. $S^*_{(rs)}$ is a kernel on $c_{1:n\cup\{r,s\}}$, so once we obtain samples from the posterior on $c_{1:n\cup\{r,s\}}$ we can simply drop r and s if they do not belong to $1{:}n$ to obtain samples of $c_{1:n}$. To mix well we should alternate the pair (rs) at every iteration. For instance we can cycle through all pairs of $1{:}n$ or pick one at random.

Alternatively we can pick r and s uniformly at random in the closure of the cycles of ω, in which case they will not belong to $1{:}n$. To do this, in principle we should first generate ω from $\mu^\alpha(\omega|c_{1:n})$. Since ω is independent of $y_{1:n}$ given $c_{1:n}$, this operation leaves the posterior invariant. Then we should pick r and s uniformly at random along the cycles. However we can do this operation in one step by marginalizing out ω. This results in drawing c_r and c_s from an urn started at $((n_i)_i;\alpha)$.

2.7 Split-Merge variant via the Ebb-Flow chain

In the previous section, we derived the Split-Merge algorithm from the Fragmentation-Coagulation chain, or rather its counterpart the transposition kernel. However we remarked that when mixing to the prior μ^α, the Fragmentation-Coagulation chain will need to reject some moves. At equilibrium, split and merge moves must be accepted with the same frequency, from which we can show that the rejection rate is $|1-\alpha|/(1+\alpha)$, which approaches 1 for α large or small. This motivates us to look at a the Ebb-Flow chain, which does not need to reject when mixing to the prior.

2.7.1 Approximate likelihood reweighting

As in the previous section, we abbreviate $Q(\bar{c}, c)$ to $Q(c)$ since $\bar{c}$ is fixed throughout. Reweighting Q goes essentially as for the Fragmentation-Coagulation kernel in eq. (2.13). However $Q(c_t{=}1|c_{1:(t-1)})$ is now $E(W/\bar{p}_1|W < \bar{p}_1, c_{1:(t-1)})$ which has no nice expression[4].

We must then resort to a slightly poorer approximation. Again by exchangeability we can assume without loss of generality that $C^1_{1:n} = 1{:}d$. $\hat{Q}$ decomposes as follows using the independence of c and $y_{1:n}$ given $c_{1:n}$:

$$\hat{Q}(c) \quad = \quad Q(c|c_{1:d})\hat{Q}(c_{1:d}|p)\hat{Q}(p)$$

We approximate $\hat{Q}(c_{1:d}|p)$ sequentially as in eq. (2.13), except we now condition on p:

$$\begin{aligned}
\tilde{Q}(c_{1:d}|p) &\overset{\text{def}}{=} \prod_{t=1}^{d} Q(c_t|c_{1:(t-1)}, y_{1:t}, p) \\
\text{where} & \\
Q(c_t|c_{1:(t-1)}, y_{1:t}, p) &\propto Q(c_t|p)P(y_{1:t}|c_{1:t}) \\
&\propto \begin{cases} p_1 q(y_{C^1_{1:(t-1)} \cup \{t\}}) \\ p_2 q(y_{C^2_{1:(t-1)} \cup \{t\}}) \end{cases}
\end{aligned} \tag{2.17}$$

As for $\hat{Q}(p)$, we simply drop the dependence of p on $y_{1:n}$ by setting $\tilde{Q}(p) = Q(p)$.

[4]Some closed but numerically unstable forms exist.

2.7.2 Metropolis-Hastings

Again we apply (2.5) to obtain the acceptance ratio. The factors $Q(c|c_{1:d})$ and $Q(p)$, common to Q and $\hat{Q}$, simplify.

$$\begin{aligned} a(\bar{c}, c) &= \frac{P(y|c_{1:n})}{P(y|\bar{c}_{1:n})} \frac{Q(c_{1:n}|p)}{\tilde{Q}(c_{1:n}|p)} \\ &= \frac{q(y_{C^1_{1:n}})q(y_{C^2_{1:n}})}{q(y_{C^m_{1:n}})} \frac{p_1^{|C^1_{1:n}|} p_2^{|C^2_{1:n}|}}{\bar{p}_1^{|\bar{C}^1_{1:n}|}} \frac{1}{\tilde{Q}(c_{1:n}|p)} \end{aligned}$$

where $\tilde{Q}(c_{1:n}|p)$ is given by (2.17). We note Q^* the kernel obtained when using the above acceptance ratio. The posterior $P(c|y_{1:n})$ is invariant for Q^*.

2.7.3 Size-biased permutation

As we announced when introducing the Ebb-Flow chain, we must alternate Q^* and a size-biased permutation kernel SBP. Clusters are chosen with probability p_i. To project SBP onto $c_{1:n}$ we resample $c|c_{1:n}$, and in particular $p_i|c_{1:n}$, given by (2.12). Marginalizing out p_i, clusters are chosen with probability $E(\text{Beta}(n_i, n - n_i + \alpha)) = n_i/(n+\alpha)$. Note that an empty cluster ($n_i = 0$) may be chosen. Finally this operation leaves $c_{1:n}$ and therefore the likelihood invariant, so $SBP^* = SBP$.

2.7.4 Projection

We now project Q^*'s action onto $c_{1:n}$, by composing Q^* with the Gibbs kernel that resamples c given $c_{1:n}$ and the ordering. Now however eq. (2.12) does not apply since we operate on the clusters *in size-biased order.* p_i given $c_{1:n}$ and the ordering has the posterior GEM distribution [Ishwaran and James, 2001]:

$$p_i = \hat{V}_i \prod_{j=1}^{i-1} (1 - \hat{V}_j) \quad \text{where} \quad \hat{V}_i \sim \text{Beta}(n_i, \alpha + \sum_{j>i} n_i)$$

In fact since Q^*'s action on $c_{1:n}$ depends on c only through p_1 and (if we merge) p_2, we need not sample $(p_i)_{i>2}$. This leads, to summarize, to the following algorithm:

```
**Pick** a cluster w.p. $\propto (n_1, \dots, n_k; \alpha)$, change labels to call it cluster #1
**Pick** a cluster w.p. $\propto (n_2, \dots, n_k; \alpha)$, change labels to call it cluster #2
**Sample** from the GEM posterior:
        $\hat{V}_1 \sim \text{Beta}(n_1 + 1, n - n_1 + \alpha) \quad p_1 = \hat{V}_1$
        $\hat{V}_2 \sim \text{Beta}(n_2 + 1, n - n_1 - n_2 + \alpha) \quad p_2 = (1 - \hat{V}_1)\hat{V}_2$
**With probability** $\int_{p_1}^{1} \text{Beta}_{(1,\alpha)}(W)dW = (1 - p_1)^\alpha$
    **Consider merging**
**Otherwise**
    **Increment** the name of all other clusters to leave room for a new, empty cluster 2
    **Consider splitting**
**Let** $m$ be the merged cluster, let 1 and 2 denote the two parts of the split
**Initialize** $C_1 := C_2 := \emptyset$ and $\tilde{Q} := 1$
**For each** $s \in C_m$
    **Compute** $q_1 := \int f(y_s|\eta)H(\eta|y_{C_1})d\eta$. And $q_2$.
    **Compute** $(\tilde{q}_1, \tilde{q}_2) \propto (p_1 q_1, p_2 q_2)$
    **If** splitting, **Draw** $c_s \in \{1, 2\}$ with prob. $(\tilde{q}_1, \tilde{q}_2)$
    **If** $c_s = 1$, **Let** $C_1 := C_1 \cup \{l_s\}$ and $\tilde{Q} := \tilde{q}_1 \tilde{Q}$
    **Similarly if** $c_s = 2$
**End For each**
**Let** $Q_1 := \int f(y_{C_1}|\eta)H(\eta)d\eta$ . Idem for $Q_2$ and $Q_m$
```

Let $a = \dfrac{Q_1 Q_2}{Q_m} \dfrac{p_1^{|C_1|} p_2^{|C_2|}}{\bar{p}_1^{|C_m|}} \dfrac{1}{\tilde{Q}}$

Accept Split with probability $\min(1, a)$

Or Accept Merge with probability $\min(1, \frac{1}{a})$

2.8 The Exchange algorithm

In the previous section, we derived a Split-Merge variant based on the Ebb-Flow chain. The derivation exemplified how the method of Sec. 2.3 can derive an algorithm that would be very difficult to propose based on intuition alone. Nonetheless, the resulting algorithm is very similar to Split-Merge.

Split-Merge was based on $R_{(rs)} \circ \Delta$. In this section, we apply the method to $T_{(rs)} = R_{(rs)} \circ \Delta \circ R_{(rs)}$, which randomizes the cyclic order and acts on c as follows. If r and s are in different clusters $\bar{C}^i$ and $\bar{C}^j$, they are merged and re-split into two new clusters C^i and C^j initialized to $\{r\}$ and $\{s\}$ and filled with the remaining elements of $\bar{C}^i \cup \bar{C}^j$ according to a $(1, 1; 0)$ urn. If r and s are in the same cluster, nothing happens.

2.8.1 Metropolis-Hastings

Since the transition probability of $T_{(rs)}$ starting from r and s in different clusters $\bar{C}^i$ and $\bar{C}^j$ is the same as that of $S_{(rs)}$ starting from r and s in the same cluster $\bar{C}^i \cup \bar{C}^j$, likelihood reweighting has the same effect for both kernels. The Metropolis-Hastings acceptance ratio becomes

$$\begin{aligned} R(\bar{c}_{1:n}, c_{1:n}) &= \frac{\hat{T}_{(rs)}(\bar{c}_{1:n})}{T_{(rs)}(\bar{c}_{1:n})} \frac{T_{(rs)}(c_{1:n})}{\hat{T}_{(rs)}(c_{1:n})} \frac{P(y|c_{1:n})}{P(y|\bar{c}_{1:n})} \\ &= \frac{q(y_{C^i_{1:n}}) q(y_{C^j_{1:n}})}{q(y_{\bar{C}^i_{1:n}}) q(y_{\bar{C}^j_{1:n}})} \frac{n_i! n_j!}{\bar{n}_i! \bar{n}_j!} \frac{\hat{T}(\bar{c}_{1:n})}{\hat{T}(c_{1:n})} \end{aligned} \qquad (2.18)$$

where $\hat{T}_{(rs)}$ is computed the same way as $\hat{S}_{(rs)}$ in (2.14).

Since the Exchange algorithm can never change the number of clusters, it should be used only in combination with another kernel such as Split-Merge or Gibbs sampling.

2.8.2 Other possibilities

[Tsilevich, 1998] considers the action of any finite permutation g on $\mathcal{S}^\infty$. Taking $g = (rst)$ leads to an algorithm with 3-way split, 3-way merge, and cut-reattach moves. Such complex moves could in principle improve mixing in difficult cases, but preliminary results do not point to an advantage.

We can take this to the extreme however and start from the kernel that samples ω from μ^α. The Chinese Restaurant Process gives us such a sample and therefore mixes in one step. Then applying approximation (2.13) recovers the *Predictive Recursion* algorithm of [Newton and Zhang, 1999]: sample $c_{1:n}$ by sequentially sampling c_{t+1} from (2.3), i.e. its conditional distribution given $c_{1:t}$ and $y_{1:t+1}$. It differs from the Gibbs sampler in that when sampling c_t we ignore the dependency on $y_{t+2:n}$, and while one step of the Gibbs sampler samples one c_t, predictive recursion samples $c_{1:n}$.

Another idea is to compose to the right *and* left with the same transposition (rs). This simply exchanges the role of r and s. If we pick r among the data points $1{:}n$ and s among non-data points, we obtain the Gibbs sampling algorithm [Neal, 1998], showing that it too can be derived from permutations.

2.9 Experimental Comparisons

Although when mixing to the prior Ebb-Flow rejects less moves than Split-Merge, this may or may not translate into an advantage when mixing to the posterior, for the following reasons:

1) Ebb-Flow can propose a trivial split where one of the two clusters is empty, thus wasting computation.

2) If the partition implied by the likelihood is atypical according to the prior, the prior may not be a good guide in choosing which clusters to split or merge.

3) A good rejection rate does not directly translate into a good mixing rate, which also depends on the correlation between successive moves.

Therefore it is instructive to compare Split-Merge and Ebb-Flow experimentally. While we cannot draw general conclusions from these simple datasets, they give some qualitative assessment of the effects above.

First we compare Split-Merge, Ebb-Flow, and the Gibbs sampling algorithm on the model of [Jain and Neal, 2000]. Each cluster is represented by a vector θ of dimension $d = 6$, and observations from this cluster are Binary vectors y such that each y_h is independently 1 or 0 with probability θ_h. The conjugate prior on θ is a product of independent Beta(1,1). For this experiment, 100 data points are split evenly into 5 clusters with predefined θ. The prior parameter α is set to 1 so that points (2) and (3) above are not a concern. We ran each algorithm for 5 minutes[5] and compared their performance in Table 2.9. Both Ebb-Flow and Split-Merge are much faster than Gibbs sampling, and point (1) does not appear to affect performance.

Algorithm	Autocorr. time	#iterations
Gibbs Sampling	$>40\ s$	694×100
Split-Merge	$1\ s$	24688
Ebb-Flow	$1\ s$	22614

Second we explored the behavior of Split-Merge and Ebb-Flow when $\alpha \neq 1$. We used the same Beta-Bernoulli model but in dimension 20, sampled 500 data points

[5]Simulations were run in Matlab 6.5 on a Pentium M 1.3GHz.

from a DP($\alpha = 40$) mixture and obtained 89 clusters. Since the dataset is drawn from the prior, point (2) is not a concern. We ran each algorithm for 20 minutes.

Ebb-Flow and Split-Merge still mix at about the same rate, the different factors affecting performance apparently cancelling each other. We do observe that Ebb-Flow accepts many more move (27%) than Split-Merge (4%), which was the goal. However when $\alpha > 1$ Ebb-Flow tends to focus on the larger clusters, therefore iterations of Ebb-Flow are a little slower (33ms) than those of Split-Merge (19ms), and they are also a little more correlated.

2.10 Conclusion

We introduced a method to derive MCMC algorithms, and showed how it applied to a variety of Markov chains for Dirichlet Process mixtures. In the process we recovered several known state of the art methods as well as new algorithms. Our method is not entirely automatic since it relies on a good choice of initial Markov chain for the prior, and on a tractable surrogate for likelihood reweighting. However it greatly reduces the role of creativity and intuition to derive very reasonable algorithms.

While we have focused narrowly on Dirichlet process mixtures, our method could potentially be applied more generally. The properties of the Dirichlet process are well understood, and we take advantage of them, but other models rely on nonparametric priors whose symmetry and limit properties could be exploited, such as Kingman's coalescent [Kingman, 1982] and Dirichlet Diffusion Trees [Neal, 2003]. In fact, our work provides further encouragement to study the properties of these objects.

Chapter 3

Lévy Processes

3.1 Introduction

Let $X_{1:n}$ denote a set of observations $\{X_1, \ldots, X_n\}$. Various processes have been used as the distribution of a hidden parameter B in a model where observations X_i are drawn identically and independently given B, leading to an exchangeable sequence of random variables. In principle, to sample $X_{1:n}$, we could first sample B, then sample each $X_i|B$ independently. In practice, if B is a measure, it can't be represented exactly. Instead, if the distributions of B and $X_i|B$ are conjugate, we can consider increasing values of n, and sample each new value X_{n+1} from the posterior distribution $X_{n+1}|X_{1:n}$. We call this a *posterior sampling process.* Not only do such processes arise as natural models of data, but they play a very important role in the analysis of their parent process. For example the conjugacy of Dirichlet processes to multinomial sampling yields the *Chinese restaurant process*, which forms the basis of many algorithms.

In the previous chapter we explored inference techniques for Dirichlet process mixtures. While mixtures have found many natural applications, many other applications exhibit structure that cannot be expressed in that way. As an alternative to the multinomial representation underlying classical mixture models, *factorial models* associate to each data point a set of latent Bernoulli variables. The factorial representation has several advantages. First, the Bernoulli variables may have a natural interpretation as "featural" descriptions of objects. Second, the representation of objects in terms of sets of Bernoulli variables provides a natural way to define interesting topologies on clusters (e.g., as the number of features that two clusters have in common). Third, the number of clusters representable with m features is 2^m, and thus the factorial approach may be appropriate for situations involving large numbers of clusters.

As in the mixture model setting, it is desirable to consider nonparametric Bayesian approaches to factorial modeling that remove the assumption that the cardinality of the set of features is known a priori. An important first step in this direction has been provided by Griffiths and Ghahramani [Griffiths and Ghahramani, 2006], who defined a stochastic process on features that can be viewed as a factorial analog of the Chinese restaurant process. This process, referred to as the *Indian buffet process*, involves the metaphor of a sequence of customers tasting dishes in an infinite buffet. Let Z_i be a binary vector where $Z_{i,k} = 1$ if customer i tastes dish k. Customer i tastes dish k with probability m_k/i, where m_k is the number of customers that have previously tasted dish k; that is, $Z_{i,k} \sim \text{Ber}(m_k/i)$. Having sampled from the dishes previously sampled by other customers, customer i then goes on to taste an additional number of new dishes determined by a draw from a Poisson(α/i) distribution. Modulo a reordering of the features, the Indian buffet process can be shown to generate an exchangeable distribution over binary matrices (that is, $P(Z_1, \ldots Z_n) = P(Z_{\sigma(1)}, \ldots Z_{\sigma(n)})$ for any permutation σ).

Given such an exchangeability result, it is natural to inquire as to the underlying distribution that renders the sequence conditionally independent. Indeed, de Finetti's theorem states that the distribution of any infinitely exchangeable sequence can be written

$$P(Z_1, \dots Z_n) = \int \left[\prod_{i=1}^{n} P(Z_i|B) \right] dP(B),$$

where B is the random element that renders the variables $\{Z_i\}$ conditionally independent and where we will refer to the distribution $P(B)$ as the "de Finetti mixing distribution." For the Chinese restaurant process, the underlying de Finetti mixing distribution is known—it is the Dirichlet process. As we have seen in the previous chapter, identifying the de Finetti mixing distribution behind a given exchangeable sequence is important; it greatly extends the range of statistical applications of the exchangeable sequence and offers more angles of attack to design inference algorithms. It also gives an important tool in the theoretical analysis of the sequence.

Soon after the Indian buffet process, another exchangeable process called the *infinite gamma-Poisson process* was proposed [Titsias, 2008] to generate feature vectors with arbitrary integer counts rather than binary indicators. Again, de Finetti's theorem implies the existence of a distribution for a hidden variable conditionally on which this sequence becomes independent, and finding this distribution would greatly increase our ability to use and extend this model.

In this chapter, we identify the de Finetti mixing distribution behind the Indian buffet process and the infinite gamma-Poisson process. As expected, establishing this connection allows us to develop many analogs for the beta and gamma process to the tools we used in the previous chapter for the Dirichlet process.

We show that these advances can be understood within the general framework of Lévy processes, a framework that includes beta processes, gamma processes and many additional stochastic processes of interest. We use the framework to show that the conjugacy of the *beta process* [Hjort, 1990] to Bernoulli sampling recovers the Indian buffet process as a posterior sampling process. This allows us to derive a size-biased construction for the beta process and to generate an interesting new model based on the conjugacy to geometric sampling (Sec. 3.3). Similarly we show that the conjugacy of gamma processes to Poisson sampling [Wolpert and Ickstadt, 1998b] recovers the gamma-Poisson model of Titsias [Titsias, 2008] and leads to a size-biased construction for the gamma process. We also establish an alternative posterior sampling process for the gamma process by taking advantage of its conjugacy to continuous Poisson sampling, leading to a second size-biased construction for the gamma process (Sec. 3.4). For the beta process, most of this research first appeared in [Thibaux and Jordan, 2007]. Other properties of the Indian buffet process are also found in [Teh *et al.*, 2007].

The unified point of view of Lévy processes also allows us to construct various kinds of nonparametric Bayesian hierarchies and corresponding inference algorithms (Sec. 4.1). This in turn allows us to study highly-flexible, large-scale generative probabilistic models. To illustrate the scalability of our methods, we study the performance of classifiers based on hierarchies of beta and gamma processes on a classification task with 1 million training samples and 40,000 classes, taken from the *80-million tiny images* dataset of Torralba, Fergus and Freeman [Torralba *et al.*, 2007] (Sec. 4.5).

But first, we begin with an introduction to Lévy processes, random measures that give independent mass to disjoint sets, and whose properties are at the core of essentially all current nonparametric Bayesian methods (Sec. 3.2).

3.2 Lévy processes

Definition. A positive random measure B on a space Ω (e.g., $\mathbb{R}$) is a *Lévy process*, or independent increment process, if the masses $B(S_1), \dots B(S_k)$ assigned to disjoint subsets $S_1, \dots S_k$ of Ω are independent[1]. Up to a constant that we can always remove,

[1]Positivity is not required to define Lévy processes but greatly simplifies their study, and positive

a positive Lévy process is discrete[2], that is it can be written

$$B = \sum_i p_i \delta_{\omega_i} \tag{3.1}$$

where δ_ω is a unit point mass (or *atom*) at ω, for a set of random locations ω_i and random positive weights p_i. Equivalently, $B(S) = \sum_{i:\omega_i \in S} p_i$ for all $S \subset \Omega$. In particular, $B(\{\omega_i\}) = p_i$.

A point $\omega \in \Omega$ such that $P(B(\{\omega\}) > 0) > 0$ corresponds to an atom of B whose location is constant. It is called a *fixed point of discontinuity.* Let $\mathcal{D}$ be the set of all fixed points of discontinuity of B. By the definition of Lévy processes, the restrictions of B to $\mathcal{D}$ and $\Omega \setminus \mathcal{D}$ are independent. Therefore B decomposes into a sum of two independent Lévy processes

$$B \overset{\mathrm{d}}{=} C + D \tag{3.2}$$

where C has no fixed point of discontinuity, and the support of D is $\mathcal{D}$, which is countable. If we note B^S the *restriction* of B to S, then $C = B^{\Omega\setminus\mathcal{D}}$ and $D = B^{\mathcal{D}}$. On singletons $\{\omega\}$, we slightly abuse notation and write B^ω for $B(\{\omega\})$.

Lévy measure. The Lévy-Khinchine theorem [Khinchine, 1934; Lévy, 1937] implies that a positive pure-jump Lévy process is uniquely characterized by its *Lévy measure*[3] (or *compensator*), a measure on $\Omega \times (0, \infty)$.

For a process C with no fixed point of discontinuity, its Lévy measure ν has the following elegant interpretation. To draw C, draw a set of points $(\omega_i, p_i) \in \Omega \times (0, \infty)$ from a Poisson process with base measure ν, and define C as in (3.1). If, as is the

Lévy processes are sufficient here. On $\Omega = \mathbb{R}$, positive Lévy processes are also called *subordinators.* See Jacod and Shiryaev [Jacod and Shiryaev, 1987] for a complete treatment.

[2]A random discrete measure is also called a *pure-jump process.*

[3]Homogeneous Lévy processes have a Lévy measure of the form $\nu = \Lambda \otimes \nu'$ where Λ is the Lebesgue measure on Ω and ν' is a measure on $(0, \infty)$, usually also called the Lévy measure.

case for the beta and gamma processes, $\nu(\Omega \times (0,\infty)) = \infty$ the Poisson process generates infinitely many points, making (3.1) a countably infinite sum. The *inverse Lévy measure algorithm* [Wolpert and Ickstadt, 1998b] is a general method to draw from this Poisson process by generating atoms in decreasing order of weights.

A process D with fixed discrete support $\mathcal{D}$ can be written simply as $D = \sum_{\omega \in \mathcal{D}} p_\omega \delta_\omega$ where each p_ω is drawn from some distribution on $[0,\infty)$. The Lévy measure still exists, but no longer corresponds to a Poisson process. Its support is $\mathcal{D} \times (0,\infty)$, and it determines the distribution of p_ω via:

$$P(p_\omega \in A) = \nu(\{\omega\} \times A) \quad \forall\ A \subset (0,\infty) \quad \text{and} \quad P(p_\omega = 0) = 1 - \nu(\{\omega\} \times (0,\infty)).$$

To specify a Lévy process, rather than writing its full Lévy measure, it is enough and often more convenient to specify its Lévy measure outside of fixed points of discontinuity, and the distribution of mass at these fixed points.

3.3 Beta process

Definition. A *beta process* [Hjort, 1990] $B \sim \text{BP}(c, B_0)$ is a positive Lévy process whose Lévy measure depends on two parameters: a function c called the *concentration function* or the *concentration parameter* when it is constant, and a fixed measure B_0 over Ω, called the *base measure*. We also call $\gamma = B_0(\Omega)$ the *mass parameter*.

If B_0 is continuous, the Lévy measure of the beta process is

$$\nu(d\omega, dp) \quad = \quad c(\omega)p^{-1}(1-p)^{c(\omega)-1} dp B_0(d\omega) \tag{3.3}$$

on $\Omega \times [0,1]$. As a function of p, it is a degenerate beta distribution, justifying the name. Atoms of B_0 correspond to fixed points of discontinuity, which are also beta distributed:

$$B^{\omega} \sim \text{Beta}(c(\omega)B_0^{\omega}, c(\omega)(1-B_0^{\omega})) \qquad \text{which imposes } B_0^{\omega} < 1 \text{ for all } \omega.$$

In the special case where B_0 is continuous and $c = 1$, (3.3) takes a simpler form and we can generate the weights p_i of the atoms of B via the following *stick-breaking construction* [Teh *et al.*, 2007]:

$$V_i \overset{\text{i.i.d.}}{\sim} \text{Beta}(\alpha, 1) \qquad p_i = \prod_{k=1}^{i} V_i$$

B is then defined as in (3.1) with atom locations ω_i drawn i.i.d. from B_0/γ.

3.3.1 Conjugacy and the Indian buffet process

We now introduce the Bernoulli process, whose conjugate prior is the beta process. This conjugacy extends the conjugacy between the Bernoulli and beta distributions.

Definition. Let B be a measure on Ω. We define a *Bernoulli process* with *hazard measure* B, written $X \sim \text{BeP}(B)$, as the Lévy process with Lévy measure

$$\mu(dp, d\omega) = \delta_1(dp)B(d\omega). \tag{3.4}$$

If B is continuous, X is simply a Poisson process with intensity B: $X = \sum_{i=1}^{N} \delta_{\omega_i}$ where $N \sim \text{Poi}(B(\Omega))$ and ω_i are independent draws from the distribution $B/B(\Omega)$. If B is discrete, of the form $B = \sum_i p_i \delta_{\omega_i}$, then $X = \sum_i b_i \delta_{\omega_i}$ where the b_i are independent Bernoulli variables with the probability that $b_i = 1$ equal to p_i. In summary, a Bernoulli process is similar to a Poisson process, except that it can give weight at most 1 to singletons, even if the base measure B is discontinuous[4].

Conjugacy. Consider the following model

$$\begin{aligned} B &\sim \mathrm{BP}(c, B_0) \\ X_i|B &\overset{\text{i.i.d.}}{\sim} \mathrm{BeP}(B) \quad i = 1, \dots n \end{aligned} \tag{3.5}$$

Applying theorem 3.3 of Kim [Kim, 1999] to the beta and Bernoulli processes, the posterior distribution of B after observing $X_{1:n}$ is still a beta process with modified parameters.

$$B|X_{1:n} \sim \mathrm{BP}\left(c+n, B_n\right) \tag{3.6}$$

$$\text{where} \quad B_n = \frac{c}{c+n}B_0 + \frac{1}{c+n}\sum_{i=1}^{n} X_i = \frac{c}{c+n}B_0 + \sum_j \frac{m_{n,j}}{c+n}\delta_{\omega_j} \tag{3.7}$$

Here ω_j are the locations of the atoms of $X_{1:n}$, and $m_{n,j} = \sum_{i=1}^{n} X_i(\{\omega_j\})$ is the number of X_i's with an atom at location ω_j.

Censoring. Lévy processes are conjugate to censoring since we can partition Ω into regions that have been observed at the same times. Thus (3.6) holds more generally when n is a piecewise constant function counting the number of observations at each point.

Sampling. Marginally, we can show that $X_1 \sim \mathrm{BeP}(B_0)$. Indeed, since both X_1 and B are Lévy processes, marginally X_1 gives independent mass to disjoint sets and is therefore a Lévy process. Moreover since X_1 gives mass 0 or 1 to singletons it is a Bernoulli process. All we need to characterize the distribution of X_1 is therefore its hazard measure, which is also its expectation. Since $E(X_1) = E(E(X_1|B)) = E(B) = B_0$, X_1 is indeed the Bernoulli process with hazard measure B_0. In particular, we can

[4]Some authors (e.g. [Kim, 1999]) call Poisson process what we call Bernoulli process. We reserve the term Poisson process for the Lévy process whose marginals are Poisson.

easily sample X_1 (that is, without first sampling B).

Sampling X_{n+1} reduces to the case of X_1 since $X_{n+1}|X_{1:n}$ is also a Bernoulli process whose hazard measure is drawn from a beta process, namely (3.6). Therefore

$$X_{n+1}|X_{1:n} \sim \mathrm{BeP}\left(B_n\right)$$

If B_0 is continuous, the two terms of (3.7) are the continuous and discrete parts of B_n, so using (3.2) we get that $X_{n+1}|X_{1:n} \overset{\mathrm{d}}{=} C + D$ where $C \sim \mathrm{BeP}\left(\frac{c}{c+n}B_0\right)$ and $D \sim \mathrm{BeP}\left(\sum_j \frac{m_{n,j}}{c+n}\delta_{\omega_j}\right)$. Therefore to sample $X_{n+1}|X_{1:n}$ we include an atom at ω_j with probability $m_{n,j}/(c+n)$, and add a $\mathrm{Poi}(c\gamma/(c+n))$ number of new atoms at locations independently drawn from B_0/γ. This posterior sampling process is the two-parameter version of the Indian buffet process [Griffiths and Ghahramani, 2006; Griffiths and Ghahramani, 2005], proving that the beta process is the de Finetti mixing distribution for the Indian buffet process [Thibaux and Jordan, 2007].

Note that in the above derivation, we sample X_{n+1} by averaging over C and D. If instead we average over C and sample D we obtain the following *size-biased construction* [Thibaux and Jordan, 2007]:

$$B = \sum_{n=0}^{\infty}\sum_{j=1}^{K_n} p_{n,j}\delta_{\omega_{n,j}}$$
$$\text{where} \quad K_n \overset{\text{ind}}{\sim} \mathrm{Poi}\Big(\frac{c\gamma}{c+n}\Big), \quad p_{n,j} \overset{\text{ind}}{\sim} \mathrm{Beta}(1, c+n) \quad \text{and} \quad \omega_{n,j} \overset{\text{i.i.d.}}{\sim} B_0/\gamma \tag{3.8}$$

3.3.2 The beta process is also conjugate to the Geometric process

The previous model is based on the conjugacy between the Bernoulli and beta distribution, but the beta distribution is also conjugate to the geometric distribution. This leads to a different but related posterior sampling process.

Definition. Let B be a measure on Ω. We define a *geometric process* with *hazard measure* B, written $X \sim \mathrm{GeoP}(B)$, as the Lévy process with Lévy measure

$$\mu(dp, d\omega) \quad = \quad (1 - B(d\omega)) \sum_{k=1}^{\infty} \delta_k(dp) B(d\omega)^k. \tag{3.9}$$

Definition (3.9) says that for each atom $p_i \delta_{\omega_i}$ of B, X has an atom $N_i \delta_{\omega_i}$ where $N_i \sim \mathrm{Geom}(1 - p_i)$. The beta process is also conjugate to the geometric process, with posterior

$$B | X_{1:n} \sim \mathrm{BP}\left(c_n, \frac{c}{c_n} B_0 + \frac{1}{c_n} \sum_{i=1}^{n} X_i\right) \quad \text{where} \quad c_n = c + \sum_{i=1}^{n} X_i + n. \tag{3.10}$$

Eq. (3.10) can be obtained from (3.6) by the construction of the geometric distribution from repeated Bernoulli trials. This model can be useful as a model of data, as we show in experiments (Sec. 4.5).

3.4 Gamma process

Definition. A gamma process $B \sim \Gamma\mathrm{P}(c, B_0)$ over a space Ω with *base measure* B_0 and *concentration function*[5] c is a Lévy process whose Lévy measure is a degenerate gamma distribution

$$\nu(d\omega, du) \quad = \quad c(\omega) \frac{e^{-c(\omega)u}}{u} du B_0(d\omega)$$

when B_0 is continuous, and whose fixed points of discontinuity, corresponding to atoms of B, are gamma distributed with shape $c(\omega) B_0(\{\omega\})$ and rate $c(\omega)$. From now on we restrict ourselves to c being a constant. In this case B can equivalently

[5]An alternative parameterization is to call $\alpha = cB_0$ the shape measure*shape measure* and $\beta = c$ the *rate function* to emphasize the connection to the gamma distribution.

be defined as the Lévy process with marginals

$$B(U) \sim \text{Gamma}(cB_0(U), c) \qquad \forall\, U \subset \Omega \tag{3.11}$$

3.4.1 A first size-biased construction for the gamma process

Conjugacy. Let $X_{1:n}$ be drawn i.i.d. from a Poisson process with base measure B. Wolpert and Ickstadt [Wolpert and Ickstadt, 1998a] show that the conjugacy of the Poisson and gamma distribution extends to this more general setting, and use it to derive a Monte Carlo inference algorithm.

$$B|X_{1:n} \sim \Gamma\text{P}\left(c+n, B_n\right) \quad \text{where} \quad B_n = \frac{c}{c+n}B_0 + \frac{1}{c+n}\sum_{i=1}^{n} X_i. \tag{3.12}$$

The model of Ihler and Smyth [Ihler and Smyth, 2007] samples a Poisson(γ) number of points from a Dirichlet process $G \sim \text{DP}(\alpha G_0)$, with a Gamma$(a, b)$ prior on γ. Because a Dirichlet process is a renormalized gamma process, their measure γG is a gamma process whenever $\alpha = a$.

Sampling. From (3.11), the total mass of B is $B(\Omega) \sim \text{Gamma}(cB_0(\Omega), c)$ and since $X_1(\Omega)|B \sim \text{Poi}(B(\Omega))$, marginally the total mass $X_1(\Omega)$ of X_1 has a negative binomial distribution $X_1(\Omega) \sim \text{NB}(cB_0(\Omega), c/(c+1))$. To draw the location of those atoms, observe that X_1 given $X_1(\Omega)$ and B is a set of $X_1(\Omega)$ independent draws from $B/B(\Omega)$, itself is a renormalized gamma process, i.e. a Dirichlet process, with concentration parameter $\alpha = cB_0(\Omega)$ and base measure $B_0/B_0(\Omega)$. Therefore we can draw the locations of the atoms of $X_1|X_1(\Omega)$ by drawing $X_1(\Omega)$ samples from the Chinese restaurant process.

This leads to the following posterior sampling process when separating (3.12) into its continuous and discrete parts. To sample $X_{n+1}|X_{1:n}$, independently sample

$$X_{n+1}(\{\omega_j\}) \sim \mathrm{NB}(m_{n,j}, (c+n)/(c+n+1)) \tag{3.13}$$

for each location ω_j of the atoms of $X_{1:n}$, and add a $\mathrm{NB}(cB_0(\Omega), (c+n)/(c+n+1))$ distributed additional mass, partitioned according to the Chinese restaurant process with concentration $cB_0(\Omega)$, and located independently at random according to $B_0/B_0(\Omega)$. This is the *infinite gamma-Poisson* model of Titsias [Titsias, 2008], which he derives as the limit of a finite model[6]. This alternative derivation demonstrates that the gamma process is in fact the de Finetti mixing distribution conditionally on which the exchangeable sequence $X_{1:n}$ becomes independent.

In the above construction, if instead of (3.13) we sample $B(\{\omega_j\}) \sim \mathrm{Gamma}(m_{n,j}, c+n)$ we have a size biased construction for B, a method to build the atoms of B in the order in which they are discovered by $X_{1:n}$. However we can derive a more elegant size-biased construction from the conjugacy of gamma processes to continuous Poisson sampling. We now turn to this alternate point of view.

3.4.2 A second size-biased construction for the gamma process

Continuous Sampling. Rather than observations being indexed by discrete indices, we consider observations arriving along a continuous time. Let Λ be the Lebesgue measure on $\mathbb{R}^+$, thought of as time, and let

$$N \;\sim\; \mathrm{PP}(B \times \Lambda).$$

This model subsumes the model presented above where we sample $X_{1:n}$ from $X_i \overset{\text{i.i.d}}{\sim}$

[6]The distribution of $X_1|X_1(\Omega)$ is never specified in [Titsias, 2008], resulting in a missing factor in Eq. (16) since the probability of a row depends on the full configuration of new features, not just on their sum.

$\mathrm{PP}(B)$ since we can recover $X_{1:n}$ from N by letting $X_i = N(. \times [i-1, i])$. It is based on the following conjugacy. Let $b \sim \mathrm{Gamma}(\alpha, \beta)$ and $x \sim \mathrm{Poi}(\lambda b)$, then $b|x \sim \mathrm{Gamma}(\alpha + x, \beta + \lambda)$.

Let (T_i, Z_i) be the atoms of N ranked by T_i. Since the base measure of N is a cross product, the times T_i and locations Z_i form independent Poisson processes $T \sim \mathrm{PP}(B(\Omega)\Lambda)$ and $Z \sim \mathrm{PP}(B/B(\Omega))$. In particular, once $B/B(\Omega)$ is marginalized out, the locations Z_i are drawn from a Chinese restaurant process with concentration parameter $cB_0(\Omega)$.

Since $T_1|B \sim \mathrm{Exp}(B(\Omega))$, the marginal distribution of T_1 is

$$\begin{aligned} P(T_1 = t) &= \int P(T_1 = t|B(\Omega))dP(B(\Omega)) \\ &= \alpha \frac{\beta^\alpha}{(\beta + t)^{\alpha+1}} \qquad \text{for } \ t \geq 0 \end{aligned}$$

where $\alpha = cB_0(\Omega)$ and $\beta = c$. We can sample from this distribution by the inverse cdf method:

$$\begin{aligned} T_1 &\overset{\mathrm{d}}{=} \beta(U^{-1/\alpha} - 1) \qquad (3.14) \\ \text{where} \quad U &\sim \mathrm{Uniform}([0,1]). \end{aligned}$$

Since $B(\Omega)|T_n \sim \mathrm{Gamma}(cB_0(\Omega)+n, c+T_n)$, we can sample $T_{n+1} - T_n|T_n$ from (3.14) with $\alpha = cB_0(\Omega) + n$ and $\beta = c + T_n$.

Size-biased construction. Let N^* be obtained from N by keeping only the earliest atom at each location, and let (T_i^*, ω_i^*) be the atoms of N^* ranked by their time T_i^*. Conditioned on the gamma process B, which we know can be written

$$B = \sum_i p_i \delta_{\omega_i},$$

we have $T_i|B \sim \text{Exp}(p_i)$. Since the Poisson construction of Lévy processes states that the set of pairs (ω_i^*, p_i) is drawn from a Poisson process with base measure the Lévy measure, the set of triplets (T_i^*, ω_i^*, p_i) is also drawn from a Poisson process, with base measure $ue^{ut}dt\nu(d\omega, du) = ce^{(c+t)u}duB_0(d\omega)$. This implies in particular after marginalizing out ω and u, and looking at the conditional distribution of p_i given T_i^* that

$$\begin{aligned} T^* &\sim \text{PP}\left(\frac{cB_0(\Omega)}{c+t}\right) \\ p_i|T_i^* &\sim \text{Gamma}(1, c+T_i^*) = \text{Exp}(c+T_i^*). \end{aligned} \tag{3.15}$$

We can draw T^* by drawing $T_{n+1}^* - T_n^*|T_n^*$ from (3.14) with $\alpha = cB_0(\Omega)$ and $\beta = c+T_n^*$. Finally, atom locations Z_i^* are i.i.d. samples from $B_0/B_0(\Omega)$. We remark that since a Dirichlet process is a renormalized gamma process, and since this construction discovers features in the same order as the Chinese restaurant process, the set $(Z_i^*, p_i/\sum_j p_j)$ is a draw from a $\text{DP}(cB_0(\Omega), B_0/\gamma)$ and the sequence $(p_i/\sum_j p_j)$ is a draw from the stick-breaking construction of the Dirichlet process.

This size-biased construction is more natural since it does not rely on the Dirichlet process or Chinese restaurant process, which artificially introduces coupling between parts of the space that are in fact independent. As we will see, a size-biased construction offers more than simply a way to sample from a gamma process. Nonetheless, even if the goal is only to obtain a sample, this method may provide a simpler alternative to the very general *inverse Lévy measure algorithm* used by Wolpert and Ickstadt [Wolpert and Ickstadt, 1998b]. In particular, we do not need to invert the exponential integral function.

From (3.15), we can derive a non-trivial fact. Let $m(t) = N^*(\Omega \times [0, t])$ be the number of unique features discovered after time t. Then

$$\begin{aligned} m_t &\sim \mathrm{Poi}\left(\int_0^t \frac{cB_0(\Omega)}{c+t}\right) \\ &= \mathrm{Poi}\left(cB_0(\Omega)\log\left(1+\frac{t}{c}\right)\right) \end{aligned} \tag{3.16}$$

Non-constant concentration. We can extend this construction to sample from any gamma process, even with non-constant c, by reducing it to the constant case. If c is piecewise constant we can sample each piece separately. Otherwise, let $B_0' = cB_0$, that is $B_0'(A) = \int_A c(\omega)B_0(d\omega)$. Draw a gamma process $\Gamma\mathrm{P}(1, B_0')$ as a set of pairs (ω_i^*, p_i). The set of pairs (ω_i^*, cp_i) is a draw from $\Gamma\mathrm{P}(c, B_0)$

Chapter 4

Naïve Bayes Classification with Infinitely Many Features

4.1 Hierarchies of Lévy processes

We introduce hierarchies of Lévy processes in the context of the naïve Bayes technique, where they arise naturally. Naïve Bayes is a simple, commonly used and surprisingly effective type of classifier. Using the vocabulary of text classification, for each *category* $j \leq n$ we are given n_j *documents* x_{ij} as training data, and want to classify a new document y into one of these categories. Documents are represented by a list of features x_{ij}^{ω}, and naïve Bayes assumes that features are independent given the category. The goal is then to estimate these category-specific feature distributions so that we can apply Bayes' rule to classify a new document.

Smoothing. The simplest approach is maximum likelihood, but if one feature does not appear in one category, maximum likelihood gives this value an unreasonable zero probability. A radical, often-used method to handle this problem is to discard all such features. This may discard a great deal of data. Instead, we can add a small constant, usually 1, to all feature value counts. This is called *Laplace smoothing* and it allows the use of all features. Unfortunately it can seriously deteriorate our probability estimates.

Hierarchical Bayes. The goal of smoothing is to fall back to a reasonable default when evidence is weak. The problem with Laplace smoothing is that its default is some fixed arbitrary distribution, such as the uniform. Instead, the default should be that the feature value is *independent* of the class. If a_j^ω is a parameter representing the distribution of x_{ij}^ω, then $a_{1:n}^\omega$ should be shrunk towards equality rather than towards a fixed value. Hierarchical Bayesian smoothing achieves this by introducing a hidden random variable b^ω on which a_j depends. For instance

$$\begin{aligned} b^\omega &\sim \text{Gamma}(c_0 b_0^\omega, c_0) \\ a_j^\omega | b^\omega &\sim \text{Gamma}(c_j b^\omega, c_j) \\ x_{ij}^\omega | a_j^\omega &\sim \text{Poi}(a_j^\omega) \end{aligned} \tag{4.1}$$

where b_0 represents our prior for the mean of x_{ij}.

Infinite feature sets. In many applications, the set of features is effectively infinite. Nonetheless for any document only a finite number of features is non-zero. For this, the sum of b_0^ω for all features must be finite. If all features are equally likely a priori, this implies that $b_0^\omega = 0$ and that (4.1) is in fact a slice of the following hierarchy of Lévy processes over the space of features

$$\begin{aligned} B &\sim \Gamma\text{P}(c_0, B_0) \\ A_j | B &\sim \Gamma\text{P}(c_j, B) \\ X_{ij} | A_j &\sim \text{PP}(A_j). \end{aligned} \tag{4.2}$$

For instance the above model can represent documents as a vector of counts for each word. If the geometric distribution is a better fit than the Poisson to the feature counts, we can use instead

$$\begin{aligned} B &\sim \mathrm{BP}(c_0, B_0) \\ A_j|B &\sim \mathrm{BP}(c_j, B) \\ X_{ij}|A_j &\sim \mathrm{GeoP}(A_j), \quad \text{or} \quad X_{ij}|A_j \sim \mathrm{BeP}(A_j) \quad \text{if our features are binary.} \end{aligned} \tag{4.3}$$

We will assume that the concentrations c_0 and c_j of all processes in the hierarchy are piecewise constant. In each case, we can more generally arrange categories as the leaves of a tree whose internal nodes are beta or gamma processes.

4.2 Inference in hierarchies of Lévy processes

Let X denote $(X_{i,j})_{i=1,\dots n_j, j=1,\dots n}$. To classify a new document Y, we compute its probability $P_j = P(X_{n_j+1,j} = Y|X)$ under each category. Since restrictions of the entire hierarchy to disjoint sets are independent, P_j can be obtained as a product over any partition of Ω. In particular, let $\mathcal{D} = \{\omega : X^\omega \neq 0 \text{ or } B^\omega \neq 0\}$ be the (finite) set of *known features*, that is features that are known to have non-zero probability either a priori or because we have observed them at least once. Then

$$P_j = P(X^{\Omega\setminus\mathcal{D}}_{n_j+1,j} = Y^{\Omega\setminus\mathcal{D}}|X^{\Omega\setminus\mathcal{D}}) \prod_{\omega\in\mathcal{D}} P(X^\omega_{n_j+1,j} = Y^\omega|X^\omega).$$

4.2.1 Known features

Each factor in the product over $\mathcal{D}$ corresponds to inference in a slice, that is model (4.1) or its equivalent for (4.3). Observe that this is simply inference in a tree of 1 dimensional random variables. Not only is inference in a tree generally easy, but the absence of dependency between features means that inference can be run in parallel. Given the posterior distribution $P(A_j^\omega|X^\omega)$ we can compute

$$P(X_{n_j+1,j}^\omega = Y^\omega|X^\omega) = \int P(X_{n_j+1,j}^\omega = Y^\omega|A_j^\omega)P(A_j^\omega|X^\omega)dA_j. \qquad (4.4)$$

The posterior of the hidden variables given the data has no closed form, but we can obtain samples from it via Gibbs sampling, allowing us to compute (4.4) by Monte Carlo integration. We can sample A_j^ω from its conditional distribution using conjugacy, and we can sample B^ω or any other node in a larger hierarchy using adaptive rejection sampling [Gilks and Wild, 1992] since its conditional distribution is log concave. Indeed it is proportional to $e^{f(B^\omega)}$ where

$$\begin{aligned} f(b) &= (c_0 B_0^\omega - 1)\log b + (c_0 B_0^\omega - 1)\log(1-b) \\ &\quad + \sum_{j=1}^{n}\left[c_j b \log A_j^\omega + c_j(1-b)\log(1-A_j^\omega) - \log\Gamma(c_j b) - \log\Gamma(c_j(1-b))\right] \\ \text{or} \quad f(b) &= (c_0 B_0^\omega - 1)\log b - c_0 b + \sum_{j=1}^{n}\left[c_j b\log A_j^\omega + c_j b \log c_j - \log\Gamma(c_j b)\right]. \end{aligned}$$

In both cases, the concavity of $-\log x - \log\Gamma(x)$ implies that f is concave. See [Thibaux and Jordan, 2007] for an alternative method using gamma upper bounds.

4.2.2 Unknown features

The last factor $P(X_{n_j+1,j}^{\Omega\backslash\mathcal{D}} = Y^{\Omega\backslash\mathcal{D}}|X^{\Omega\backslash\mathcal{D}})$ is the probability of observing *new* features, those features of Y that do not appear in the training data. In particular, if Y has many unknown features, it is more likely to come from a category with little training data. Usually the impact of this factor on the likelihood is so small that it does not change the classification decision; this is the case in all our experiments. Discarding this factor, which is almost always implicitely done in practice, amounts to censoring by only observing $Y^{\mathcal{D}}$.

In some cases however this may lead to throwing away important data, in particular when there are large differences between the number of training examples available for each category. Besides, computing this factor is of theoretical interest because it involves the size-biased constructions presented above, and illustrates the ways in which the independence property of Lévy processes can be exploited. We now examine how to compute it.

We assume that the concentration of all processes in the hierarchy is constant on $\Omega \setminus \mathcal{D}$. If they are only piecewise constant we can decompose the factor as a product over each piece. To further simplify notations we also assume without loss of generality that $\mathcal{D} = \emptyset$, that is we assume that not a single feature appears in the training data, and that B_0 is continuous. The case $\mathcal{D} \neq \emptyset$ reduces to this since the restriction of our hierarchy to $\Omega \setminus \mathcal{D}$ is itself a hierarchy of Lévy processes, with top level base measure $B_0 - B_0^{\mathcal{D}}$, which is continuous, and with observations $X^{\Omega\setminus\mathcal{D}} = 0$ and $Y^{\Omega\setminus\mathcal{D}}$.

Let $\hat{X}_{n_j+1,j}$ represent the number and mass of the atoms of $X_{n_j+1,j}$, that is $\hat{X}_{n_j+1,j}$ is the equivalence class of $X_{n_j+1,j}$ under change of atom locations. Since the concentrations are constant, the conditional distribution $X_{n_j+1,j}|\hat{X}_{n_j+1,j}$ gives to each atom a location drawn independently from $B_0/B_0(\Omega)$, and since this conditional distribution is the same for each category it only affects $P(X_{n_j+1,j} = Y|X = 0)$ by a constant that we can ignore for the purpose of classification. Thus we only need to compute $P(\hat{X}_{n_j+1,j} = \hat{Y}|X = 0)$. The idea is then to obtain samples from the posterior process $A_j|X = 0$ and use Monte Carlo integration to compute

$$P(\hat{X}_{n_j+1,j} = \hat{Y}|X = 0) = \int P(\hat{X}_{n_j+1,j} = \hat{Y}|A_j)P(A_j|X = 0).$$

The size biased constructions of beta and gamma processes give us an explicit prior distribution for the entire hierarchy. Indeed once we generate the atoms (ω, b^ω) of the top level process B, the atoms of all other processes have the same locations and follow the slice distribution given b^ω. All these atoms are also given an ordering, therefore it makes sense to talk about the i^{th} atom.

But before we consider how to sample from $A_j|X = 0$, which is the subject of Sec. 4.3 and 4.4, observe that the first factor $P(\hat{X}_{n_j+1,j} = \hat{Y}|A_j)$ is combinatorial in nature. Since the association between the atoms of A_j and those of $\hat{Y}$ is unknown, we must sum over all associations, which is intractable. Fortunately, we can use the independence property of Lévy processes to simplify this calculation. Let $S_1, \ldots S_N$ be a partition of Ω into N regions of equal base measure mass $B_0(S_i) = B_0(\Omega)/N$, such that each region S_i contains a single atom of Y. Such a partition exists since B_0 is continuous, and by independence we have

$$P(X_{n_j+1,j} = Y|X = 0) = \prod_{i=1}^{N} P(X^{S_i}_{n_j+1,j} = Y^{S_i}|X^{S_i} = 0).$$

Therefore without loss of generality we can assume that Y has a single atom, greatly simplifying our problem. Let $(a^i_j)_{i=1}^\infty$ be the mass of the atoms of A_j, $(x^i_{n_j+1,j})_{i=1}^\infty$ the mass of the corresponding atoms of $X_{n_j+1,j}$, and y the mass of the unique atom of Y. Then,

$$P(\hat{X}_{n_j+1,j} = \hat{Y}|A_j) \quad = \quad \prod_{i\in\mathbb{N}} P(x^i_{n_j+1,j} = 0|a^i_j) \sum_{i\in\mathbb{N}} \frac{P(x^i_{n_j+1,j} = y|a^i_j)}{P(x^i_{n_j+1,j} = 0|a^i_j)}.$$

If we truncate our process the sum and product become finite and the computational complexity is linear in the truncation level. Since the weights of beta and gamma processes decrease exponentially, the truncation can be small. Additionally, since concentrations are constant and we have chosen regions S_i of equal prior mass, the posterior over A_j is the same over each region except for the distribution of the location, therefore we can reuse our samples. All we need now is a method to obtain samples from the posterior $A_j|X = 0$. We do not have a general method for all Lévy processes, so in the next sections we develop specialized algorithms that rely on specific properties of beta and gamma processes.

4.3 Posterior hierarchical beta process

For hierarchies of beta processes, observe that the size biased construction of the beta process groups atoms of B by level and each level is independent. Since the observation $X = 0$ entails no data association ambiguity between levels, posterior inference can take place separately in each level. In each level n, the posterior over K_n is

$$K_n|X = 0 \quad \sim \quad \text{Poi}\left(\frac{c\gamma}{c+n} q_n\right) \tag{4.5}$$

where q_n is the probability that observations from an atom of level n are 0. We can compute q_n by sampling from the prior in the slice model of level n, that is the slice model where, at the top level, $b^\omega \sim \text{Beta}(1, c+n)$. Therefore to sample from B and A_j given $X = 0$, for each level n we draw K_n from (4.5), then we draw K_n samples from the posterior slice model of level n given $x^\omega = 0$, using Gibbs sampling. In particular we want samples of a_j^ω. The atoms of A_j are the union of all these atoms for all levels. In practice of course we truncate A_j by only considering a finite number of levels.

For the special case of hierarchies of beta processes with Bernoulli observations, [Thibaux and Jordan, 2007] show how this method can in fact be used to compute $P(X_{n_j+1,j} = Y|X = 0)$ without first partitioning the space to isolate the atoms of Y.

4.4 Posterior hierarchical gamma process

We will take an entirely different route for hierarchies of gamma processes. Neither of the two size biased constructions lead to easy posterior inference. Instead we will use a remarkable property of hierarchies of gamma variables. Consider the following slice model (4.6), where we omit the location ω.

$$\begin{aligned} b &\sim \text{Gamma}(c_0 b_0, c_0) \\ a_j | b &\sim \text{Gamma}(c_j b, c_j) \\ x_{ij} | a_j &\sim \text{Poi}(a_j) \end{aligned} \tag{4.6}$$

Then we have the following surprising result: the posterior of b and (a_j) given $x = 0$ is a hierarchy of gamma distribution. Note that this is not true for other values of x. To prove this, we prove by induction on the depth of the hierarchy that the log likelihood of $x = 0$ is a linear function of the top level variable. To initialize the induction, we look at a 1 level hierarchy where we use conjugacy

$$\begin{aligned} \log P(x_{1:n_j,j} = 0 | a_j) &= -a_j n_j \\ \text{and} \quad a_j | b, x_{1:n_j,j} = 0 &\sim \text{Gamma}(c_j b, \hat{c}_j) \\ \text{where} \qquad \hat{c}_j &= c_j + n_j \end{aligned} \tag{4.7}$$

At the next level, which is the general case for a larger hierarchy,

$$\begin{aligned} P(x = 0 | b) &= \prod_j P(x_{1:n_j,j} = 0 | b) \\ P(x_{1:n_j,j} = 0 | b) &= \int P(x_{1:n_j,j} = 0 | a_j) p(a_j | b) da_j \\ &= \frac{c_j^{c_j b}}{\Gamma(c_j b)} \int a_j^{c_j b - 1} e^{-\hat{c}_j a_j} da_j = \left(\frac{c_j}{\hat{c}_j} \right)^{c_j b} \end{aligned}$$

Therefore

$$\begin{aligned}\log P(x=0|b) &= -b\sum_j c_j \log\frac{\hat{c}_j}{c_j} \\ b|x=0 &\sim \text{Gamma}(c_0 b_0, \hat{c}_0) \\ \text{where} \qquad \hat{c}_0 &= c_0 + \sum_j c_j \log\frac{\hat{c}_j}{c_j}. \end{aligned} \tag{4.8}$$

Using the definition of gamma processes by marginals and reasoning on partitions, this result immediately extends to processes. The posterior of B and (A_j) given $X=0$ is a hierarchy of gamma processes. The base measure of a child is its parent scale by a factor $c_j/\hat{c}_j$ rather than the parent itself as in our original hierarchy. Therefore to obtain samples from $A_j|X=0$ we compute the coefficients $\hat{c}_j$ recursively up the tree using (4.8), sample the top level process B from its posterior using the size biased construction (3.15), and sample the rest of the hierarchy under each atom of B using (4.7). This method generates exact samples from the posterior, contrary to Gibbs sampling which only approaches the posterior as the Markov chain mixes.

Rao-blackwellization. In each of these methods to sample from the posterior process, the mass of the atoms of A_j are independent given $X=0$ and given the mass of the atoms of B. More generally, the slices below each atom of the top level process are independent given their number per level K_n in the case of beta processes, or given the times T_i^* in the case of gamma processes. Therefore we get a more efficient estimator if we integrate a_j^i in closed form using conjugacy:

$$\begin{aligned}P(\hat{X}_{n_j+1,j} = \hat{Y}|B_j) = \prod_{i\in\mathbb{N}} & P(x^i_{n_j+1,j}=0|b^i_j, x_{1:n_j,j}=0) \\ &\times \sum_{i\in\mathbb{N}} \frac{P(x^i_{n_j+1,j}=y|b^i_j, x_{1:n_j,j}=0)}{P(x^i_{n_j+1,j}=0|b^i_j, x_{1:n_j,j}=0)}.\end{aligned}$$

4.5 Image application

The 80 Million Tiny Images dataset [Torralba *et al.*, 2007] is a large collection of 32x32 pixel images, obtained by entering a list of keywords in several search engines and resizing the resulting images. Keywords were obtained by selecting all non-abstract English words listed in the Wordnet [Fellbaum, 1998] database and provide a noisy label for the images it returned. We focus on a publicly available subset of this dataset[1] from which we obtain a classification task with 44278 categories and 986,944 training examples selected at random. Understanding this dataset has implications for human vision since Torralba, Fergus and Freeman [Torralba *et al.*, 2007] demonstrate that humans are able to understand scenes and recognize objects even on such small images, but that 32x32 is near the limit where this is possible.

It is also an interesting and challenging dataset for machine vision since its large size and the small size of the images makes many techniques inapplicable. For example Lazebnik, Schmid and Ponce [Lazebnik *et al.*, 2006] use a one-vs-all support vector machine (SVM) to classify images into up to 101 categories. This method become very expensive in large datasets with large numbers of classes; in contrast, in our method we only need to sum the sufficient statistics for each class, after which our training time only depends on the number of classes, not the number of training examples. Note also that in our dataset each category contains fewer than 20 training examples, resulting in a set of extremely unbalanced classification tasks for the SVM.

Given the specificity of each category in this dataset ("rheumatologist," "vertebral artery," etc.), measuring classification performance by the number of exact matches is an overly strict measure of the quality of an algorithm. Instead we rely on the Wordnet database, which provides semantic relationships between words. In particular, the hypernym relationship forms a directed acyclic graph, which can be reduced to a

[1]This subset is available at http://people.csail.mit.edu/torralba/tinyimages/

tree rooted at "entity." Torralba, Fergus and Freeman [Torralba *et al.*, 2007] used this tree to perform classification into broader categories corresponding to internal nodes of the tree. Instead we propose to classify at the most specific level, but to use the number of hypernyms in common (not counting the root) between two words as a measure of similarity. We call *hypernym score* the average hypernym similarity between the prediction and the truth on test data.

Motivated by the relationship between color and category observed in [Torralba *et al.*, 2007], we extract for each image a very simple set of color features. We first convert the image to HSV color space, then split this space into 8 Value levels, split each Value level i into i Saturation levels, and each Saturation level (i, j) into j Hue levels, resulting in 120 coarse colors, roughly corresponding to human perception of similarity. Our 120 features count the number of pixels of each color.

We compared several naïve Bayes models, differing only in the method employed to estimate the feature distributions for each category. In the model of eq. (4.2), feature counts have a Poisson distribution whose intensity parameters are drawn from a hierarchy of gamma processes. In particular, eq. (4.2) refers to a flat hierarchy where all categories share a common parent. Alternatively, we can arrange the gamma process hierarchy to mirror the Wordnet tree. This prior expresses our intuition that semantic similarity is reflected to some extent in visual similarity and we expect such sharing of statistical strength to mitigate the considerable noise in the labels. Finally we also build models using the same two structures but for a hierarchy of beta processes with geometric observations. We compare the performance of these 4 models on a test set of 1000 images chosen at random. The concentration parameter was chosen to be $c = 10$ throughout the hierarchy.

From Table 4.1 we see that the hierarchical methods ("Wordnet") yield better performance than the flat methods ("Flat"). Note also that even though these methods are far from classifying images at the most precise level, they do extract some

Table 4.1: Performance of each method on the 80 Million Tiny Images dataset

Method	**Hypernym Score**
Baseline (random)	1.861
Gamma-Poisson Flat	2.273
Gamma-Poisson Wordnet	2.313
Beta-Geometric Flat	2.397
Beta-Geometric Wordnet	**2.447**

meaningful information even based on these rudimentary features. Training and testing took about 13 hours on a Sun X2200 M2 server, and this time was further reduced by a factor of six by attributing 20 features to each of 6 machines.

4.6 Conclusion

Bayesian methods tend to rely on combinations of a few dozen well understood distributions such as the Gaussian, the gamma, or the Poisson. These distributions are canonical in the sense that they arise naturally in a variety of contexts and they have simple properties that form the basis of many inference algorithms. Similarly nonparametric Bayesian methods rely on a small set of processes, and much work has gone into clarifying their properties and how they can be used for inference. However so far the lion's share of this effort has focused on only two objects, the Gaussian process and the Dirichlet process. This work furthers our theoretical understanding of the properties of two other processes: the beta process and the gamma process.

Our application of these techniques to a large dataset shows that nonparametric Bayesian methods are not necessarily associated with hard inference problems. The ease with which our method can be parallelized also should make this method a natural choice for the largest datasets.

Chapter 5

Conclusion

This thesis provides general tools for the analysis and the application of nonparametric Bayesian methods. One goal has been to clarify the field, to ensure that progress in one of these models can be understood from a more general point of view and applied to all the other models. Indeed we were able to transfer much of the theory of Dirichlet processes to beta and gamma processes. We have also shown how a deeper understanding of Dirichlet processes (via virtual permutations), of the Indian buffet process (via the beta process), and of the infinite gamma-Poisson process (via the gamma process) leads to new models and new algorithms.

The flexibility of nonparametric Bayesian methods comes at the cost of mathematical sophistication, yet it leads to algorithms that are neither computationally more difficult nor harder to program than their parametric counterparts. In fact, the infinite limit often makes explicit difficulties already present but hidden in the finite case, leading to better algorithms. For example, hierarchical Dirichlet processes [Teh *et al.*, 2006] came about because multiple draws from a Dirichlet process with continuous base measure share no atom, preventing any sharing. This phenomenon is hidden in the case of a finite Dirichlet, manifesting itself as a very strong disincentive for sharing.

Therefore the advantage of nonparametric methods is two-fold. The first advantage and the primary motivation for their development is their flexibility: not having to specify the number of clusters or features, allowing new structures to emerge with more data, reducing the burden on the model designer. The second is simplicity. The objects at the core of these methods are *mathematically advanced*, but they are *mathematically simple*. To quote David Aldous refering to the continuous random tree, “the infinite limit is a good warm-up for the finite case”.

This simplicity manifests itself in the many fascinating properties that we have exploited in this work. Some properties, such as the stick-breaking construction of the Dirichlet process or the fragmentation-coagulation process, are more elegant versions of similar properties of the finite case. Some properties however, such as the ebb-flow process or the second size-biased construction of the gamma process, have no known finite counterpart. This suggests that the limit object is *fundamentally* simpler. This provides a motivation for nonparametric Bayesian methods even when parametric methods seem sufficient.

Bibliography

[Dahl, 2003] D. Dahl. An improved merge-split sampler for conjugate Dirichlet process mixture models. Technical Report 1086, Department of Statistics, University of Wisconsin, 2003.

[Diaconis *et al.*, 2004] P. Diaconis, E. Mayer-Wolf, O. Zeitouni, and M. Zerner. The Poisson-Dirichlet law is the unique invariant distribution for uniform split-merge transformations. *The Annals of Probability*, 32(1B):915–938, 2004.

[Escobar and West, 1995] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, June 1995.

[Ewens, 1972] W. J. Ewens. The sampling theory of selectively neutral alleles. *Theoretical Population Biology*, 3:87–112, 1972.

[Fellbaum, 1998] C. Fellbaum. *Wordnet, an electronic lexical database.* Bardford Books, 1998.

[Gilks and Wild, 1992] W. R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, 41(2):337–348, 1992.

[Gnedin and Kerov, 2001] A. Gnedin and S. Kerov. A characterization of GEM distributions. *Combin. Probab. Comput.*, (10):213–217, 2001.

[Griffiths and Ghahramani, 2005] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit, 2005.

[Griffiths and Ghahramani, 2006] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2006.

[Hjort, 1990] N. L. Hjort. Nonparametric Bayes estimators based on Beta processes in models for life history data. *The Annals of Statistics*, 18(3):1259–1294, 1990.

[Ihler and Smyth, 2007] A. T. Ihler and P. J. Smyth. Learning time-intensity profiles of human activity using non-parametric Bayesian models. In *Neural Information Processing Systems 19*, 2007.

[Ishwaran and James, 2001] Hemant Ishwaran and Lancelot F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453), 2001.

[Jacod and Shiryaev, 1987] J. Jacod and A. N. Shiryaev. *Limit theorems for stochastic processes.* Springer-Verlag, 1987.

[Jain and Neal, 2000] 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.

[Kerov *et al.*, 1993] S. Kerov, G. Olshanski, and A. Vershik. Harmonic analysis on the infinite symmetric group. Comptes Rend. Acad. Sci. Paris, Série I, t.316, 1993.

[Khinchine, 1934] A. Y. Khinchine. Korrelationstheorie der stationären stochastiscen prozesse. *Mathematische Annalen*, 109:604–615, 1934.

[Kim, 1999] Y. Kim. Nonparametric Bayesian estimators for counting processes. *The Annals of Statistics*, 27(2):562–588, 1999.

[Kingman, 1975] J. F. C. Kingman. Random discrete distributions. *Journal of the Royal Statistical Society, Series B*, (37):1–22, 1975.

[Kingman, 1982] J. F. C. Kingman. The coalescent. *Stochastic Processes Appl.*, (13):235–248, 1982.

[Lazebnik *et al.*, 2006] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[Lévy, 1937] P. Lévy. Théorie de l'addition des variables aléatoires. Gauthiers-Villars, Paris, 1937.

[Mayer-Wolf *et al.*, 2002] E. Mayer-Wolf, O. Zeitouni, and M. Zerner. Asymptotics of certain coagulation-fragmentation processes and invariant Poisson-Dirichlet measures. *Electronic J. Probab*, 7, 2002.

[Neal, 1998] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Technical Report 9815, Dept. of Statistics, University of Toronto, 1998.

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

[Newton and Zhang, 1999] M. Newton and Y. Zhang. A recursive algorithm for nonparametric analysis with missing data. *Biometrika*, 86(1):15–26, 1999.

[Pitman, 2002a] J. Pitman. Combinatorial stochastic processes. Technical Report 621, Dept. Statistics, U.C. Berkeley, 2002. Lecture notes for St. Flour course, July 2002.

[Pitman, 2002b] J. Pitman. Poisson-Dirichlet and GEM invariant distributions for split-and-merge transformations of an interval partition. *Combinatorics, Probability and Computing*, 11:501–514, 2002.

[Teh *et al.*, 2006] Y. W. Teh, M. I. Jordan, M. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006.

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

[Thibaux and Jordan, 2007] R. Thibaux and M. I. Jordan. Hierarchical beta processes and the Indian buffet process. In *Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[Titsias, 2008] M. Titsias. The infinite gamma-poisson feature model. In *Advances in Neural Information Processing Systems 20*. 2008.

[Torralba *et al.*, 2007] A. Torralba, R. Fergus, and W. T. Freeman. Tiny images. Technical Report MIT-CSAIL-TR-2007-024, Computer Science and Artificial Intelligence Lab, MIT, 2007.

[Tsilevich, 1998] N. V. Tsilevich. Stationary random partitions of positive integers. *Theory Probab. Appl.*, 44(1):60–74, 1998.

[Wolpert and Ickstadt, 1998a] R. Wolpert and K. Ickstadt. Poisson/gamma random field models for spatial statistics. *Biometrika*, 85(2):251–267, 1998.

[Wolpert and Ickstadt, 1998b] R. Wolpert and K. Ickstadt. Simulation of Lévy random fields. In *Practical Nonparametric and Semiparametric Bayesian Statistics.* Springer-Verlag, 1998.

Index

acceptance ratio, 10
adaptive rejection sampling, 48

base measure, 35, 39
Bernoulli process, 36
beta process, 35

category, 45
Chinese Restaurant Process, 6, 15
cluster, 5
compensator, 34
concentration function, 35, 39
concentration parameter, 35
conjugacy, 6

exchangeability, 7

fixed point of discontinuity, 34

gamma process, 39
geometric process, 39
Gibbs kernel, 10
Gibbs sampler, 7

hazard measure, 36
hierarchical Bayes, 46

identifiability, 5
infinite gamma-Poisson model, 41

Lévy measure, 34
Lévy process, 33
likelihood, 9
likelihood reweighting, 10

Markov kernel, 9
mass parameter, 35
Metropolis-Hastings, 8, 9

naïve Bayes, 45
negative binomial, 40

partition, 5
permutation, 8, 13
Poisson-Dirichlet, 16
posterior, 8
posterior inference, 8
predictive recursion, 27
prior, 8

Rao-Blackwellization, 53
rate function, 39
reversibility, 9

shape measure, 39
size-biased construction, 38, 42
size-biased permutation, 16
stationarity, 9
stick-breaking construction, 36
stick-breaking process, 16
support vector machine, 54

transposition, 8

urn model, 6

virtual permutation, 12

Wordnet, 54