The Indian Buffet Process: Scalable Inference and Extensions

Finale Doshi-Velez

A Thesis
Presented to the Fellowship of
The University of Cambridge
in Candidacy for the Degree of
Master of Science

Department of Engineering
Zoubin Ghahramani, supervisor

August 2009

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

Abstract

Many unsupervised learning problems seek to identify hidden features from observations. In many real-world situations, the number of hidden features is unknown. To avoid specifying the number of hidden features a priori, one can use the Indian Buffet Process (IBP): a nonparametric latent feature model that does not bound the number of active features in a dataset. While elegant, the lack of efficient inference procedures for the IBP has prevented its application in large-scale problems. The core contribution of this thesis are three new inference procedures that allow inference in the IBP to be scaled from a few hundred to 100,000 observations.

This thesis contains three parts:

(1) **An introduction to the IBP and a review of inference techniques and extensions.** The first chapters summarise three constructions for the IBP and review all currently published inference techniques. Appendix C reviews extensions of the IBP to date.

(2) **Novel techniques for scalable Bayesian inference.** This thesis presents three new inference procedures: (a) an accelerated Gibbs sampler for efficient Bayesian inference in a broad class of conjugate models, (b) a parallel, asynchronous Gibbs sampler that allows the accelerated Gibbs sampler to be distributed across multiple processors, and (c) a variational inference procedure for the IBP.

(3) **A framework for structured nonparametric latent feature models.** We also present extensions to the IBP to model more sophisticated relationships between the co-occurring hidden features, providing a general framework for correlated non-parametric feature models.

Acknowledgements

Finale thanks the Marshall Aid Commemoration Commission for funding her two years at Cambridge University.

Contents

Chapter 1

Introduction

A common goal in machine learning is to discover hidden, or *latent*, structure in data. In the simplest case, structure may correspond to grouping, or *clustering*, similar observations. For example, balloons may be clustered by their colour. In contrast, *multiple-membership* or *feature-based* models attempt to explain structure in observations by a collection of features (Ueda and Saito, 2003; Jolliffe, 2002; Zemel and Hinton, 1994; Spearman, 1904). For example, a balloon may have a feature describing its colour and another feature describing its shape. More generally, images may be characterised by the objects they contain; a piece of music by its notes. The structure in the data corresponds to features shared across different observations: similar objects in multiple images, similar notes played at different times.

By capturing shared features, latent feature models can provide an efficient encoding for the observations. For example, if notes are extracted from an audio recording of a musical piece, then the piece may be played again using just the information about the notes. Likewise, telling someone to buy a 'round, red' balloon may be more efficient than sending them an photograph. In data-compression situations, the identities of the features—that is, what they correspond to—are often known. For example, we may already know the frequency characteristics of each note; our goal is then to identify which notes occur at which times.

In other situations, the values of the features may not be known a priori, and the discovering the structure itself may be of inherent interest. For example, sets of genes that are often expressed together may reflect underlying biological pathways (Knowles and Ghahramani, 2007); patterns in stroke symptoms may provide information about structure of the brain (Wood et al., 2006). Patterns of software use from code traces may contain clues about the presence of software bugs (Doshi and Gael, 2008). In this case, the goal of the unsupervised learning problem is to simultaneously determine the feature identities and what features are present in each observation. Finding these patterns—latent structure in the data—may guide further research and analysis.

In this work, we focus on binary latent feature models, shown graphically in figure 1.1. Here, A_k represents the *value* of the k^{th} feature. For example, if the observations are images, then A_k might correspond to some object in the image, such as a car or tree. An arrow connecting A_k to X_n indicates that feature k is present in observation n. The model is binary in that we are only concerned about whether A_k is present (connected by an arrow) or absent (not connected) in each observation X_n.

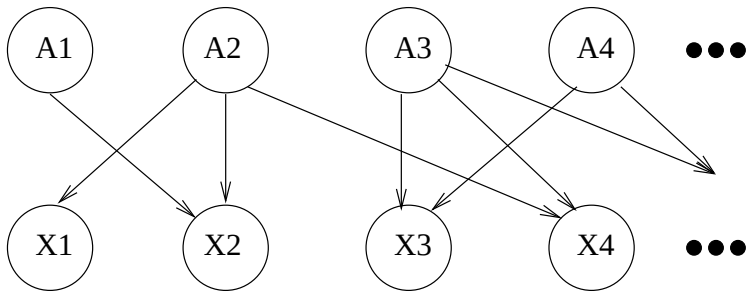


Figure 1.1: Graphical model of the binary latent feature model. Each observation X_i is generated by a combination of features A_k. For example, observation X_2 contains features A_1 and A_2. For now, we do not limit the number of observations or features.

If the features' identities—that is, the values A_k—are unknown, it seems reasonable that the total number of latent features may also be unknown. However, many current approaches to modelling latent features require the

total number of features as input. Specifying the input can be tricky: if the number of features is underestimated, the model may fail to explain the data well; if the number of features is overestimated, the model may overfit the data. Another issue in pre-specifying the number of features is that the expected number of observed features may vary based on the size of the data set. For example, given a few images, we may expect to see only a handful of unique objects. However, many more unique objects may be present in a large photo album.

A nonparametric Bayesian prior helps address the problem of specifying the number of features. The Bayesian aspect of the approach allows us to place a distribution over the expected number of features—as well as how many features are likely to be present in any particular observation—while the nonparametric aspect of the prior allows the expected number of features to grow as more data is observed. The Indian Buffet Process (IBP) of Griffiths and Ghahramani (2005) is one such non-parametric model that places a prior over binary feature matrices Z in which the number of features may be unbounded. In figure 1.1, a value of $Z(n, k) = 1$ would indicate a link between the feature A_k and the observation X_n. Instead of having to specify the total number of features present, the IBP provides a distribution over latent features. An attractive property of the IBP is that new features are expected to appear as more data is observed, but the number of features expressed in a finite data set is finite with probability one. Details on the IBP model and its construction are provided in Chapter 2.

Applications of Latent Feature Models

The properties of an IBP have made it useful in many applications, including modelling choice behaviour and collaborative filtering, protein interactions, causal graph structure, and dyadic data. Before diving into the details of the model and inference, we describe several recent applications, showing the richness in how the IBP matrix can be interpreted for a variety of purposes.

Dyadic Data. Dyadic data refers to a range of scenarios in which each observation is associated with a pair of factors. For example, for a product-customer pair may be associated with the observation of whether the customer bought the item. A user-movie pair might be associated with the observation of how well the user liked the movie. In these situations, the goal is often to predict values of certain pairs given values for other pairs: for example, we may wish to know how likely a customer is to enjoy a movie that he has not yet seen. Meeds et al. (2007) posit that both the qualities of the customers and the products may be summarised by a set of predictive features: for example, a customer might be characterised by age, gender, and wealth, while the product might be characterised by its cost, use, and reliability. By using an IBP prior, the number of these predictive features can be determined automatically.

Choice Behaviour. A particular choice behaviour model, elimination by aspects (Tversky, 1972), assumes that each of a customer's options may be described by a binary vector of features known as aspects. Each aspect represents some binary quality related to the option—whether it is expensive, pretty, useful, etc.—and has a weight associated with its importance. The number of aspects encodes the 'complexity' of the decision, that is, how many factors a customer considers when choosing amongst the different options. Görür et. al. use an IBP to automatically determine the number of aspects (Görür et al., 2006). When applied to a data set where people are asked to choose celebrities with whom they would like to spend an hour in conversation, and the model recovers the three types of celebrities in the set.

Similarity Judgements. Closely related to choice modelling is the problem of modelling similarity judgements. Given a similarity value between several pairs of items (possibly from multiple sources), the goal is to determine some underlying structure to describe the similarities. One approach, known as additive clustering (Shepard and Arabie, 1979), models each item as having a set of binary features, which may correspond to whether the object has a particular colour or weight. Next, a similarity matrix is learned between

features instead of between objects. This approach typically requires fewer parameters to be learned: for example, instead of having to learn that each pair of red items are similar to each other, one can create a feature encoding whether an item is red and use the feature similarity matrix to encode that all red items are similar to each other. Navarro and Griffiths avoid having to specify the number of features through an IBP prior.

Protein Interactions. Proteins interact to form complexes. While the levels of various proteins are easily measured, the presence of a complex is much more difficult to ascertain. Moreover, a protein may be part of several complexes, and the number of complexes is an open biological question. Chu et al. (2006) used the IBP to model which proteins take part in which complexes, where the complexes correspond to the latent features. Their results on an RNA-Polymerase data set recovers the several complexes known from biology.

Explaining Symptoms. As seen in figure 1.1, the IBP also places a distribution over directed bipartite graphs. Thus the IBP can be used to determine the causal graph structure where the observations come from a variety of hidden causes. Wood et al. (2006) use the IBP to determine the locations of a stroke within the brain. The observations are a collection of symptoms and the hidden variables are all the locations in the brain that have been blocked by the stroke. Wood et. al. found that the IBP roughly clustered together observations with common stroke locations. In particular, the IBP was able to separate stroke locations in the left and right hemispheres. Such a model could also be applied to other scenarios where one might need to consider that a patient has multiple diseases or conditions to explain his symptoms.

Software Debugging. Andrzejewski et. al. propose the use of statistical methods that analyse patterns of software use to discover bugs that may otherwise be difficult to find via standard techniques (Andrzejewski et al., 2007). One such approach, applied by Doshi and Gael (2008), is to treat each line of code as an observation and each feature as a particular usage pattern, such as

clicking a mouse. Given tags on each session on whether the run was successful, one can characterise what usage patterns make a run fail.

Extensions of the IBP to other probability models allow for an even broader range of applications. Titsias (2008) suggests using a Gamma-Poisson model instead of using a Beta-Bernoulli process for creating the feature assignment matrix. Here, instead of binary values in the matrix, the matrix contains elements distributed as Poisson random variables. Such a model is useful when we expect to see multiple copies of a feature in an observation; for example, to describe an image, it may be more useful to describe the number of cars, rather than the presence of cars. The extensions described in Section 7 and Appendix C also allow the IBP to be used to separate (an unknown number) of audio channels (Gael et al., 2008) or related consumer preferences (Miller et al., 2008b).

A common theme in all the applications is that the IBP provides an elegant way of avoiding the need to pre-specify the number of latent aspects, protein complexes, or other features. However, when applied to real data, the IBP is often limited to small, often toy data sets. One of the reasons that the IBP is not applied to larger problems is that existing inference techniques for the IBP model do not scale, an issue that we address in Chapter 3.

Contribution

While elegant, scaling IBP inference to large data sets has proven to be a challenge. Current sampling approaches often do not scale beyond a few hundred observations and tens of dimensions; as the size of the datasets grow, samplers become slow and are often caught in local modes. Scaling issues are a problem because we are currently faced with very large datasets that are still sparse: for example a rating database might have millions of users and thousands of movies, but each user may have only rated a few movies. In such settings, Bayesian methods provide a principled and robust approach to drawing inferences and making predictions from sparse information. The core of this thesis is devoted to efficient algorithms for scalable inference in the IBP model. Many

of these techniques are also applicable to more general problems in Bayesian inference.

The IBP is an important nonparametric latent feature model, but in many cases, even more structure is expected in the observations. Recent work (Gael et al., 2008; Miller et al., 2008b; Wingate et al., 2009) has focused on nonparametric feature models that allow correlations between observations to be modelled—for example, we may expect subsequent frames in movie to contain similar objects, or people living in certain regions to be more likely to share certain characteristics. Existing work (Rai and Daume, 2009) has also introduced tree-structured correlations among the features in the IBP model in a somewhat limited context. A second contribution of this thesis is to extend work this fledgling area of nonparametric correlated latent feature models by providing a general framework for introducing correlations between features in infinite models.

The remainder of this thesis is laid out as follows

- Chapter 2 describes three constructions for the IBP. Chapter 3 reviews current inference techniques.
- Chapter 4 presents a more efficient Gibbs sampler. More generally applicable than just IBPs, the sampler provides orders of magnitude speed-up on a broad range of Bayesian models with some basic conjugacy properties.
- Chapter 5 builds on from Chapter 4 to develop a data-parallel inference scheme.
- Chapter 6 presents the first variational inference scheme for the IBP, as well as the first truncation bounds for modelling the IBP with a finite model.
- Chapter 7 extends the basic IBP model to handle correlations in the data.

Chapter 2

The Indian Buffet Process Model

The problem of discovering hidden features in a set of observations can naturally be decomposed into two parts: determining what features 'look-like'—the *likelihood model*—and determining which features are present in each observation. Suppose that we have N observations. Then we can encode which features are present in each observation with a binary feature-assignment matrix Z, where $Z(n, k) = 1$ if feature k is present in observation n. Recalling figure 1.1, one way to visualise the feature assignment matrix is as a matching between features and data.

The Indian Buffet Process (Griffiths and Ghahramani, 2005) is a nonparametric distribution over such binary matrices which does not bound the number of features K. However, given a finite number of observations N, the distribution ensures that the number of features K is finite with probability one. The behaviour of the process is governed by a single concentration parameter α, which governs the expected number of features in each observation. The expected number of features K in N observations is $O(\alpha \log(N))$; the generative model favours scenarios in which there exist a few popular features and many rare features (extensions to the basic IBP model, which encode different prior biases, are described in Appendix C). In this chapter, we describe three

constructions for the Indian Buffet Process. Likelihood models used in this thesis are described in Appendix A.

2.1 Restaurant Construction

The classic generative process of Griffiths and Ghahramani (2005) proceeds as follows: imagine N customers, each representing an observation, in a queue at an infinitely long buffet. The first customer tries the first Poisson(α) dishes, where each dish represents a feature. The following customers take portions from previously sampled dishes with probability m_k/n, where m_k is the number of people who sampled dish k before customer n. Each customer also tries Poisson(α/n) new dishes. We record if customer n tried dish k in $Z(n, k)$. Figure 2.1 shows a cartoon of this process.

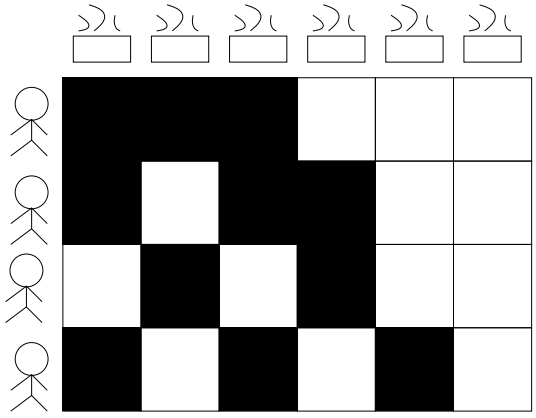


Figure 2.1: Cartoon of the generative process for the IBP. Each customer tries a combination of previously tried dishes based on their popularity and then possibly samples a few new dishes.

Even though the buffet is infinite, it follows from the construction that each customer has a finite number of dishes with probability one, and thus, given a finite number of observations, we expect only a finite number of features to be present. The buffet analogy also highlights two important properties of the Indian Buffet Process. First, we expect the number of sampled dishes—or *active* features—to grow as the number of observations increases. Second, we

expect there to exist a few popular features occurring in many observations and many rare features expressed in only a few observations.

Less obvious from the buffet construction is that the Indian Buffet Process is also infinitely exchangeable, that is, the order in which the customers attend the buffet as no impact on the distribution of Z up to permutations in the columns, and that columns are also independent. Recall that in the buffet construction, the customers simply chose dishes based solely on their popularity. Thus, the only thing special about 'feature one' is that it is (probably) one of the more popular features. The value chosen for 'feature one' is independent of its popularity. In the graphical model, switching columns corresponds to moving the nodes $A_1 \dots A_k$ to different values—but since we do not change the edges connecting the features to the observations, the model remains the same.

Once we note that permuting the columns should not affect the model, it is convenient to think of a canonical ordering for which all Z matrices that are the same up to column-permutations are equivalent. Griffiths and Ghahramani define a canonical representation called the *left-ordered form* of Z, written as $[Z] = lof(Z)$. The left-ordered form first takes the binary sequence of 0's and 1's for each column (referred to as a *history* h), treating the first customer as the most significant bit, and converts the binary sequence to a number. Thus, each column—or feature—receives a single value. We then order the columns by descending value. It is possible to directly generate left-ordered Z matrices with a variant of the buffet analogy (Griffiths and Ghahramani, 2005), but it is not necessarily useful for inference.

The Indian Buffet Process places the following prior on $[Z]$:

$$P([Z]) = \frac{\alpha^K}{\prod_{h=1}^{2^N-1} K_h!} \exp\{-\alpha H_N\} \prod_{k=1}^{K} \frac{(N-m_k)!(m_k-1)!}{N!}, \tag{2.1}$$

where K is the number of nonzero columns in Z, m_k is the number of ones in column k of Z, H_N is the N^{th} harmonic number $\sum_n^N \frac{1}{n}$, K_h is the number of columns in Z with binary representation h, and α controls the expected

number features expressed in each observation. In particular, we note the number of nonzero columns K is not bounded in equation 2.1.

2.2 Infinite Limit Construction

The exchangeability of the process is clearer in a the infinite-limit construction also presented by Griffiths and Ghahramani (2005), which derives the IBP as the infinite limit of a finite model with K_f latent features. The finite model assigns a probability π_k to each feature k. Then, each $Z(n,k)$ is sampled as independent Bernoulli random variable with parameter π_k. Since each 'customer' now samples a dish independently of the other customers, it should be clear that the ordering of the observations does not impact the distribution on Z.

The probabilities π_k are sampled from $\mathsf{Beta}(\frac{\alpha}{K_f}, 1)$. Under the finite model, the probability of the left-ordered form $[Z]$ of a binary matrix Z is given by

$$P([Z]) = \frac{K_f!}{\prod_{h=1}^{2^N-1} K_h!} \prod_{k=1}^{K_f} \frac{\frac{\alpha}{K_f}\Gamma(N - m_k + 1)\Gamma(m_k + \frac{\alpha}{K_f})}{\Gamma(N + 1 + \frac{\alpha}{K_f})}, \tag{2.2}$$

where, as in equation 2.1, m_k is the number of ones in column k of Z, and K_h is the number of columns in Z with binary representation h.

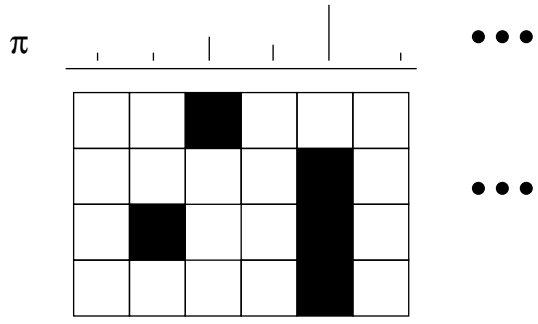


Figure 2.2: Cartoon of the infinite-limit generative model for the IBP. Since features are not ordered in any way by their popularity, so initial features may be absent from all observations.

In the limit that K_f goes to infinity, equation 2.2 becomes equation 2.1, the distribution for the IBP. Intuitively, the mean feature probability, $\frac{\alpha}{\alpha+K_f}$ becomes very small; however, a few π_k's will still be large. Figure 2.2 shows a cartoon of the process: because most of the feature probabilities are small, very few features are expressed in the finite dataset. Since each probability π_k is drawn independently, it is clear that each of the (infinite) columns are equally likely to be one of the popular features.

2.3 Stick-breaking Construction

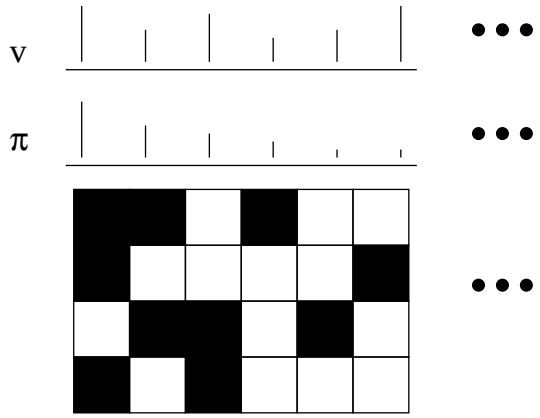


Figure 2.3: Cartoon of the stick-breaking generative model for the IBP. Unlike in the infinite limit construction (Section 2.2), features in the stick-breaking construction are ordered by on their popularity.

The finite-limit construction for the IBP does not order the features; more likely features may 'located' at some arbitrarily large index. The stick-breaking construction of Teh et al. (2007) derives a similar model that orders the features by their popularity. As with the infinite limit construction, the stick-breaking construction assigns a probability π_k to each column and samples each $Z(n,k)$ is sampled as independent Bernoulli random variable with parameter π_k.

Unlike the infinite-limit construction, however, the feature probabilities π_k are not sampled independently. Instead, a sequence of independent random variables $v_1, v_2, \ldots$, each distributed as $\mathsf{Beta}(\alpha, 1)$, are first sampled. Given the

set $v_1, v_2, \ldots$, the probability π_k of a feature k is given by

$$\pi_k = \prod_{i=1}^{k} v_i.$$

As a result, $\pi_k > \pi_{k+1}$ (see figure 2.3); the expected value of π_k is $(\frac{\alpha}{1+\alpha})^k$. For large k, the probability of any of the N observations containing that feature decreases exponentially fast. Also, large values of α cause π_k to decay more slowly in expectation. Thus, larger values of α mean that more features will be present in the data.

2.4 Beta-Bernoulli Process Construction

Finally, Thibaux and Jordan (2007) show one can also construct the Indian Buffet Process from a Beta-Bernoulli process (just as the Chinese Restaurant Process relates to the Dirichlet process). A Beta process $\mathsf{BP}(c, B_0)$ is a distribution over measures governed by a concentration parameter c and a base distribution B_0. A draw B from the Beta process $\mathsf{BP}(c, B_0)$ is a set of pairs (ω_k, π_k). The pairs (ω_k, π_k) are sampled from a Poisson process on $\Omega \times [0, 1]$. The output can be represented as

$$B = \sum_k \pi_k \delta_{\omega_k},$$

where δ_{ω_k} is an atom at ω_k. Figure 2.4 illustrates how one may visualise B as a set of weights π_k placed at atoms ω_k in some feature space Ω. The π_k values are not normalised; they will generally not sum to one. Finally, if B_0 is a discrete distribution made up of atom-weight pairs (ω_k, q_k), then for each atom ω_k, $\pi_k \sim \mathsf{Beta}(cq_k, c(1 - q_k))$.

The Beta process B gives us a (possible infinite but) countable set of atoms and weights. The corresponding Bernoulli process samples atoms from B; each draw from the Bernoulli process may be interpreted as which atoms—or features—are present in a particular observation. More formally, a Bernoulli process $Z_n \sim \mathsf{BeP}(B)$ takes in a distribution B as its base measure. If the B

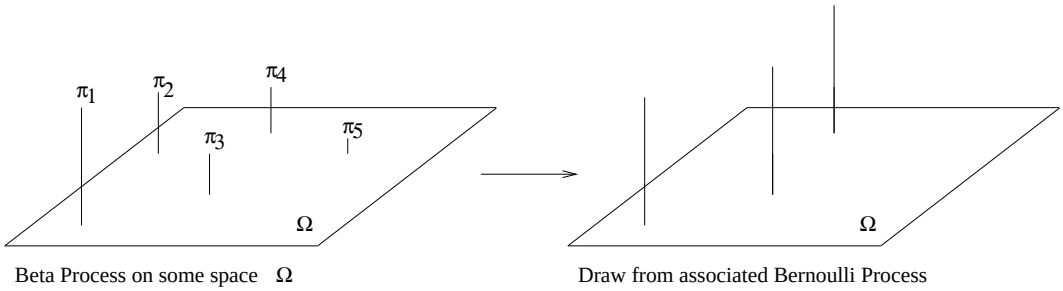


Figure 2.4: Cartoon of the Beta-Bernoulli generative model for the IBP.

is a discrete set of atom-weight pairs (ω_k, π_k), then a draw $Z_n \sim \mathsf{BeP}(B)$ is given by

$$Z_n = \sum_k b_k \delta_{\omega_k},$$

where b_k is a 0-1 Bernoulli random variable with probability π_k. If B is continuous, then Z_n is Poisson with hazard B. Thus, a draw Z_n of a Bernoulli process is a collection of features. Concatenating several draws Z_n into one matrix Z results in a binary-feature assignment matrix.

When the draw from the Beta process B is explicitly represented, the representation is similar to the stick breaking construction for the IBP: specifically, the feature probabilities π are explicitly represented in the model. Marginalising out B gives us a generative process more similar to the restaurant construction. Given a set of draws $Z_1 \dots Z_n$, a new draw Z_{n+1} can be sampled without explicitly representing B via the following equation

$$Z_{n+1}|Z_{1\dots n} \sim \mathsf{BeP}(\frac{c}{c+n}B_0 + \frac{1}{c+n}\sum_j^n Z_j).$$

where $\sum_j^n Z_j$ is shorthand for a set of pairs that places mass m_k on atom ω_k, where $m_k = \sum_j^n b_{jk}$.

Now, suppose the concentration $c = 1$ and mass of the base measure over the feature space $B_0(\Omega) = \alpha$. Then we obtain:

$$Z_{n+1}|Z_{1\dots n} \sim \mathsf{BeP}(\frac{1}{1+n}B_0 + \frac{1}{1+n}\sum_j^n Z_j).$$

The first term, with relative weight $\alpha/(n+1)$, represents drawing a new feature from the base measure. As expected, the probability of choosing a new feature decreases with n. The second term is the discrete distribution based on the counts (or popularities) of the features that have occurred at least once. A nice property of the Beta-Bernoulli construction is that it naturally provides a second parameter, c, that can be adjusted to change the 'stickiness' or concentration of the features in the IBP.

2.5 Summary

In this chapter, we described four constructions for the Indian Buffet Process. The restaurant construction is unique in that it always marginalises the feature probabilities—that is, the π_k's are only ever implicitly represented—while the remaining constructions provide a generative procedure for sampling the π_k's. In practice, the restaurant and stick-breaking constructions tend to be more useful for inference because they give the more popular features lower indices. However, the infinite-limit and Beta-Bernoulli constructions make the exchangeability and independences in the model explicitly clear. Appendix C describes constructions for extensions of the IBP, including the two-parameter IBP alluded to in Section 2.4.

Chapter 3

Inference

Chapter 2 described several generative models for the Indian Buffet Process. A generative process can be used to simulate observations: we use the IBP to sample a feature-assignment matrix Z, and then we apply a likelihood model $p(X|Z)$ to generate simulated observations X. This chapter considers the complementary problem of inference: given a set of observations X, our goal is to characterise the posterior $p(Z|X)$. This posterior may be used to compress existing data, predict properties of new data, or suggest structures to guide further research. All currently existing methods for inference in the IBP express the posterior $p(Z|X)$ as a set of samples. We provide an overview of these samplers and summarise their properties. As before, detailed descriptions of likelihood models are given in Appendix A.

3.1 Gibbs Sampler

Gibbs sampling is a Markov-Chain Monte Carlo (MCMC) technique for drawing samples from a joint distribution over several variables $z_1, z_2, \ldots, z_K$. New values for each variable z_k are sampled one at a time, conditioned on the current values of the remaining variables and observed quantities x

$$z_k \sim p(z_k|z_{-k}, x) \tag{3.1}$$

where z_{-k} refers to the set of all variables excluding z_k: $\{z_1, z_2, \ldots, z_{k-1}, z_{k+1}, \ldots, z_K\}$. Usually each variable z_k is sampled in a deterministic order, but the Gibbs sampler will produce samples from the true posterior $p(z|x)$ as long as each variable z_k is sampled infinitely often during an infinite run of the sampler. A detailed description of Gibbs sampling (including a formal derivation) is provided in (Gelman et al., 2003; Neal, 1993).

In Chapter 1, A_k represented the value of feature k. More generally, A may represent any parameters needed to generate the data X given the feature-assignments Z. Given a likelihood function $P(X|Z, A)$ and data X, the goal of the inference is to draw samples from the joint distribution $p(Z, A|X)$. The Gibbs sampler alternates between sampling elements of A and sampling elements of Z. The exchangeability properties of the IBP imply that we can always imagine that the row we are sampling corresponds to the last customer to have entered the buffet. Given the feature matrix A, the IBP matrix Z is sampled element-by-element

$$p(Z_{nk}|Z_{-nk}, X, A) \propto \frac{m_k - Z_{nk}}{N} p(X|Z, A) \tag{3.2}$$

where m_k is the number of observations using feature k (if observation n is the only observation using feature k, then the Gibbs sampler will always remove it). In the linear-Gaussian model, the likelihood $p(X|Z, A)$ factors into $\prod_n p(X_n|Z_n, A)$. Sampling a new value for Z_{nk} is computationally efficient because changing Z_{nk} only $p(X_n|Z_n, A)$ changes in $p(X|Z, A)$.

Following the buffet analogy, unused features become *active* if the customer decides to try new dishes at the buffet. The probability that customer n samples k_{new} features is given by

$$p(k_{new}) \propto \mathsf{Poisson}(k_{new}, \frac{\alpha}{N}) \int_{A_{new}} p(X|Z_{new}, A_{new}) p(A_{new}) dA_{new} \tag{3.3}$$

where Z_{new} is an IBP matrix with k_{new} 1's appended to row n, and A_{new} has k_{new} rows drawn from the prior $P(A)$. (Note that the integral in equation 3.3 depends on k_{new} through the size of A_{new}.)

In nonconjugate models, the integral in equation 3.3 is intractable, and Gibbs sampling is no longer possible. However, new features can still be sampled with Metropolis-Hastings (MH) methods. Like Gibbs sampling, MH samplers are a form of MCMC that produce a sequence of samples from a desired distribution. To sample a new value given a current value z, we first first draw a value $\hat{z}$ from some proposal distribution $q(\hat{z}|z)$. The value $\hat{z}$ is accepted with probability

$$P(z \to \hat{z}) = \min(1, \frac{p(x|\hat{z})p(\hat{z})q(\hat{z}|z)}{p(x|z)p(z)q(z|\hat{z})}) \tag{3.4}$$

where $p(x|z)$ is the likelihood of z given the data x.

Specifically for sampling new features, Meeds et al. (2007) describe a procedure in which first, values for k_{new} and A_{new} are sampled from their respective priors. Next, a proposed Z_{new} is created which *deletes* any current singleton features in Z_n and adds the k_{new} features to Z_n. The proposed matrix Z_{new} is accepted with probability

$$P(Z \to Z_{new}) = \min(1, \frac{P(X|Z_{new}, A_{new})}{P(X|Z, A)}).$$

Depending on the model for the features A, the feature matrix may be resampled analytically at once (for the linear-Gaussian model) or element-by-element (exponential-Gaussian and Bernoulli models). In some cases, A may be resampled by using a Metropolis step to propose a new value. In other cases, A may be marginalised out so that only elements of Z need be sampled; this special case is detailed in Section 3.2. To differentiate the two Gibbs samplers, we call the sampler in this section the *uncollapsed* Gibbs sampler—since A is explicitly represented—and the sampler where A is marginalised out the *collapsed* Gibbs sampler.

In all MCMC-based sampling methods, the value of the current sample depends on the value of the previous sample. An important consideration when employing these methods is how quickly the underlying Markov chain mixes, which roughly corresponds to how many 'independent' samples we obtain from this set of dependent samples (see Gelman et al., 2003). While the uncollapsed Gibbs sampler is computationally efficient per iteration, it mixes relatively

slowly. In particular, since new features are sampled from the prior, which may not be a good fit for the data, many proposals may be needed before a needed feature is added. Sampling A and Z in two separate stages also decreases the sampler's flexibility, preventing observations from making adjustments to A when determining the feature assignment Z.

Fortunately, mixing can be improved in several ways. In conjugate models, we find that instantiating initialised features but integrating out new features usually speeds up mixing, especially in high-dimensional datasets where samples from the prior were unlikely to be good proposals. In non-conjugate models, mixing may be improved by drawing the new features in A_{new} from some intelligent proposal distribution $p^*(A)$ and weighting samples by $\frac{p(A)p(X|Z_{new},A_{new})}{p^*(A)}$. Finally, Courville (2008) notes that mixing can be improved by running the Gibbs sampler at several different temperatures, a technique known as parallel tempering (Earl and Deem, 2005). Chains are allowed to exchange samples via Metropolis steps, and the idea is that chains running at higher temperatures, that is, with smoothed probability distributions, will be able to move more effectively from local hills.

Finally, we observe that mixing is generally not a problem in lower dimensional datasets. Combined with a fast per-iteration runtime, this simple sampler is often difficult for the other methods to beat. Two advantages of the uncollapsed Gibbs sampler are that it rarely proposes many new features and it readily prunes features that are poor matches for the data. Always maintaining a small set of features improves its already fast runtime.

3.2 Collapsed Gibbs Sampler

We now consider the special case where the parameters of the likelihood function can be integrated out, that is, where we have an analytic expression for $p(X|Z)$:

$$p(X|Z) = \int_A p(X|Z,A)p(A)dA. \tag{3.5}$$

In this situation, we can sample feature-assignment matrices Z from the posterior $p(Z|X)$ without having to represent A explicitly. We refer to this as the collapsed Gibbs sampler (Griffiths and Ghahramani, 2005).

As with the uncollapsed Gibbs sampler, the collapsed Gibbs sampler samples elements of Z one at a time. We use the exchangeability property of the IBP to imagine that the current observation n is the last customer to have entered the buffet. For all active features k, we sample Z_{nk} via

$$p(Z_{nk}|Z_{-nk}, X) \propto \frac{m_k - Z_{nk}}{N} p(X|Z) \tag{3.6}$$

where m_k is the number of observations containing feature k.

The probability of observation n containing k_{new} features is given by

$$p(k_{new}) \propto \mathsf{Poisson}(k_{new}, \frac{\alpha}{N}) p(X|Z_{new}) \tag{3.7}$$

where Z_{new} is the feature-assignment matrix with k_{new} additional columns set to one for feature n. The likelihood term $p(X|Z_{new})$ is computed using equation 3.5. To Gibbs sample k_{new}, we compute $p(k_{new})$ for $k_{new} = 1 \ldots k_{max}$, where k_{max} is some truncation level, and then sample a value for k_{new} proportionally from the probabilities $p(k_{new})$.[1]

The advantage of the collapsed Gibbs sampler is that integrating out the features A gives the sampler faster mixing rates (by the Rao-Blackwell theorem). However, integrating out the features correlates all of the data, so computing the likelihood $P(X|Z)$ with one element Z_{nk} involves all of the observations $X_1 ... X_N$, not just the associated observation X_n. As a result, the runtime of the collapsed Gibbs sampler (as described in Griffiths and Ghahramani (2005)) is significantly slower than its uncollapsed counterpart. One of the contributions of this thesis is novel collapsed Gibbs sampler that achieves an identical runtime (up to constant factors) to the uncollapsed Gibbs sampler (Chapter 4). Empirically, we also observe that collapsed Gibbs sampler is

[1] One may also use the Metropolis approach of Section 3.1. In practice, we find that the Metropolis method prevents the sampler from adding many features at once, a useful computational quality of in a sampler.

quicker to add features and slower to delete them during the burn-in period directly following initialisation. The initial feature explosion can significantly affect running times, especially in large datasets.

3.3 MH Proposals and Reversible Jump MCMC

Because it makes very local moves—changing only one variable at a time—Gibbs samplers often get trapped in a single mode, and it may take exponential time (in the number of variables) for the Gibbs sampler to move from one part of the posterior to another. Using MH moves (equation 3.4), particularly with a well-chosen proposal distribution $q(\hat{z}|z)$, allows the sampler to make large, global moves such as adding or deleting entire features.

The choice of proposal distribution depends heavily on the application, but there are a few basic moves that are useful in a variety of applications. Meeds et al. (2007) introduce split-merge proposals, where two features are either combined (via an 'or'-operation) or a single feature is split, with each '1' in the original feature column choosing to go with one of the new features. The probability of acceptance in these Metropolis steps may be improved by using ideas from Jain and Neal (2000), which introduces a novel split-merge MCMC for the Dirichlet Process. Their approach incorporates Gibbs sampling within the MH proposal distribution to produce better proposals. Create-destroy steps (Doshi and Gael, 2008) may also be posited to help form or remove individual features and be optimised in a similar way.

Reversible jump MCMC (RJMCMC) is a variant of MH designed to explore models of different dimensionalities (Green, 1995). In the context of the IBP, changing "dimension" corresponds to changing the number of active features in a dataset. Wood and Griffiths (2007) suggest the an RJMCMC procedure for the IBP that introduces create-destroy proposals between standard Gibbs sampling steps. The proposals have clean, closed-form acceptance probabilities for the binary likelihood models. In practice, Wood et. al. found the RJMCMC sampler much slower to mix than the Gibbs sampler, performing poorly until a very large number of iterations had been run. Our experience with MH

proposals is similar: especially as the dimensionality of the data grows and features become more distinct, simple create/destroy and split/merge moves rarely accept. A good RJMCMC or MH sampler should leverage domain knowledge about the data.

3.4 Slice Sampler

Slice sampling is a two-step sampling technique well-suited to drawing samples from distributions with a few tall modes and many regions of low probability mass (Neal, 2003). Suppose we wish to sample a new value for the variable of interest z from some distribution $p(z)$. The key concept is to introduce an auxiliary variable u that does not change the underlying distribution, that is $\int_u p(z,u)du = p(z)$, for which sampling $p(z|u)$ is efficient. Gibbs sampling is performed over each of the variables z and u in turn, and the resulting sequence of variables z are samples from $p(z)$.

In the context of slice sampling, the auxiliary variable u is used to focus computational effort on more probable values of z, that is, "slice" away less likely z. More concretely, u is drawn from $(0, p(z))$, $p(z) < 1$, from some distribution that has full support over the the interval. Next, we sample a new value for the variables of interest z, considering only z such that $p(z) > u$. If we write the joint as $p(z,u) = p(u|z)p(z)$, we see that the marginal distribution of z is unchanged:

$$\begin{aligned}
\int_u p(z,u)du &= \int_u p(u|z)p(z)du \\
&= \int_u \frac{1}{p(z)} I(u \in (0,p(z))p(z)du \\
&= \int_u I(u \in (0,p(z))du \\
&= p(z)
\end{aligned}$$

In this way, the slice sampler provides a principled approach to largely ignore regions of low probability mass.

The slice sampler for the IBP (Teh et al., 2007) leverages the stick-breaking construction of the IBP, explicitly representing the probability π_k of each feature. The sampling process first draws an auxiliary slice variable u from the distribution

$$u \sim \mathsf{Uniform}(0, \pi^*)$$

where $\pi^* = \min_{k \leq K}(\pi_k)$ and K is the number of active features. (If there are no active features, $\pi^* = 1$). The slice u corresponds to the least likely feature we will consider—any feature with probability less than u is ignored in the following sampling round.

Without loss of generality, the slice sampler assumes that the inactive features are ordered by their feature probabilities $\pi_{K+1} > \pi_{K+2} > \pi_{K+3} \ldots$. New feature probabilities—and feature values A_k—are sampled until $\pi_k < u$. Next, standard conditionals (see Teh et al., 2007) are applied to resample elements of the feature assignment matrix Z, feature values A, and feature probabilities π_k. Because the slice variable has fixed the number of features, expensive operations to add or delete features are not needed at this stage. Finally, any unused features are deleted, and a new slice variable u is sampled. Teh et. al. find that the slice sampler mixes almost as quickly as the collapsed Gibbs sampler, even though it does not integrate out the feature matrix A. Thus the slice sampler is particularly useful for nonconjugate likelihood models.

3.5 Particle Filtering

Wood and Griffiths (2007) propose a very different approach to sampling using sequential Monte Carlo, or particle filtering. Here, even though the data arrives as a batch, we pretend that it arrives sequentially. Thus, the particle filter samples based on successively larger data sets. Processing the data sequentially provides similar advantages as tempering: the posterior is initially less peaked because the number of observations is small. By adding data gradually—and thus peaking the posterior gradually—the hope is that the particle filter will follow general trends in the observations and be less likely to get caught in poor modes early on.

Applied to the IBP, we can think of particle filtering as considering a number of possibilities for what each new customer will do when entering the buffet and keeping the most likely possibilities. The algorithm begins with I particles $\{Z^i : i = 1...I\}$ with zero rows and zero columns. The first customer—or observation—arrives and each particle Z^i samples $\mathsf{Poisson}(\alpha)$ features for it (and, if the model is not conjugate, corresponding feature values A^i). Each particle is weighted by the likelihood $w_i = P(X_1|Z^i, A^i)$. A new set of I particles Z^i is then drawn with replacement from the discrete distribution based on normalising the weights w_i.

During the n^{th} transition step, an additional row n is added to each Z^i matrix based on the IBP generative model and the previous rows. For non-conjugate models, a new feature matrix A^i is also sampled based on Z^i. Particles are weighted by their likelihood $w_i = P(X_{1:n}|Z^i_{1:n}, A^i)$, and a new set of I particles is drawn with replacement from the discrete distribution based on the weights w_i. The procedure continues until all N rows have been added to each Z^i. After processing all N observations, the set $\{Z^i\}$ represents an approximation to the true posterior $P(Z, A|X)$. In the limit as $I \to \infty$, this approximation converges to the true distribution.

For certain problems, Wood and Griffiths show that particle filtering provides orders of magnitude of speed-up. However, we found that because each row of the Z matrix is being sampled from the prior, which may not well match the posterior, the particle filter often requires a large number of particles, especially as the number of observations and the dimensionality of the dataset become large.

3.6 Summary

This chapter described several sampling methods for inference in the Indian Buffet Process. Gibbs sampling is the most simple and, in our experience, effective on broad variety of problems (especially when models are conjugate). The remaining methods—MH sampling, slice sampling, and particle filtering—have more parameters and work well in the applications presented by their

developers. In particular, MH sampling can be quite effective in situations when a one has a method for generating good proposals.

Chapter 4

Accelerated Sampling in Conjugate Models [1]

In chapter 3, we noted that while the collapsed Gibbs sampler mixed faster than the uncollapsed Gibbs sampler, it required significantly more computation. In the context of IBPs, the collapsed Gibbs sampler of Griffiths and Ghahramani (2005) is $O(N^3)$ even with rank-one optimisations, whereas the uncollapsed Gibbs sampler is $O(N)$. However, the mixing-runtime tradeoff is not limited to IBPs: many models—such a factor analysis, probabilistic PCA, and probabilistic matrix factorisation—contain a core variable which, if explicitly represented, renders the data independent at the cost of losing flexibility in the sampler. The contribution of this chapter is an accelerated Gibbs sampler for models such at the IBP, FA, and PCA with conjugate likelihoods. The accelerated sampler retains the benefits of a collapsed sampler but has a linear per-iteration running time.

For the purposes of this chapter, we focus on the linear-Gaussian model for the IBP. In the linear-Gaussian model, the data X is generated by a matrix product $X = ZA + \epsilon$, where each binary Z_{nk} denotes whether feature k is present in observation n, each A_{kd} indicates the value of feature k along dimension d, and ϵ_{nd} is Gaussian white noise. An independent Gaussian prior is placed on each A_{kd} (see Appendix A for more details). Given the data X,

[1] Parts of this chapter were previously published as Doshi-Velez and Ghahramani (2009).

our goal is to infer Z and A. However, our method is broadly applicable to a much wider class of conjugate models.

Inspired from Maximisation-Expectation clustering (Welling and Kurihara, 2006), where hard cluster assignments are combined with distributions over cluster indices, our accelerated sampler samples the feature assignments Z but keeps a posterior over the feature identities A. We use this posterior to efficiently compute the likelihood of an observation without being constrained by a single, sampled feature identity. Our approach easily scales to datasets of 10,000 observations or 1,500 dimensions. The derivations and experiments that follow focus on the IBP, but our method can provide significant speed-ups in any conjugate model where the following conditions hold:

- We can marginalise out the likelihood parameters A to compute $p(X|Z)$, but it is computationally expensive.
- We have an exponential family form for the posterior on the likelihood parameter $p(A|Z, X)$.
- We do not have an analytic form for the joint $p(Z, A|X)$.

In the general case, Z would correspond to variables associated with specific observations and A would correspond to global parameters of the likelihood. For example, in factor analysis, Z would correspond to the factor loadings of each observation and A to the set of factors. In probabilistic PCA, Z would correspond to the low dimensional representation of each input, and A would be the set of eigenvectors defining the projection.

4.1 Intuition

The collapsed Gibbs sampler is slow because the collapsed likelihood computation (equation A.2) depends on the entire dataset. The graphical model in figure 4.1 illustrates this dependence, where we have split the observations into two parts. The "bottom" X_W matrix represents a window containing the last W observations and X_{-W} represents the other observations. If A is not

observed, inference on Z_W depends on both X_W and X_{-W}. In contrast, if A is observed—as in the uncollapsed Gibbs sampler—inference on Z_W depends only on the corresponding data X_W. (The Z_{-W} dependence is easy to compute in the IBP; it may not exist in other models.)

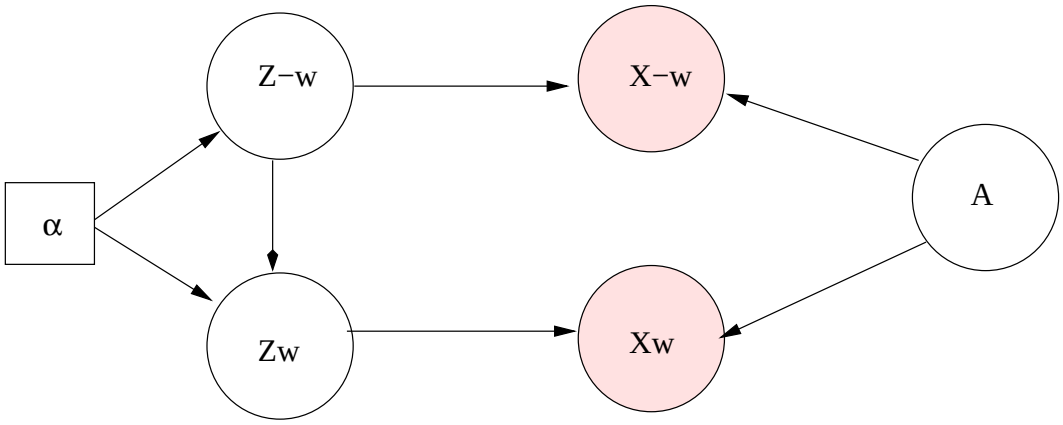


Figure 4.1: Graphical model for the IBP, showing the observations and the feature-assignment matrix split into two (arbitrary) parts. The observations corresponding to Z_W occur "after" Z_{-W} and depend on the counts of Z_{-W}.

Our accelerated sampler maintains the posterior $p(A|X_{-W}, Z_{-W})$. By keeping the posterior, instead of sampling a fixed value for A, the accelerated sampler retains the flexibility—that is, the mixing properties—of the collapsed Gibbs sampler (regardless of W). However, similar to the uncollapsed Gibbs sampler, the posterior blocks the dependence of Z_W on X_{-W}. As a result, the accelerated Gibbs sampler has similar runtime to the uncollapsed sampler.

4.2 Formal Derivation

We formally show how the intuition in the previous section produces a sampler that is exactly equivalent to the collapsed Gibbs sampler. Let X_W denote some window of observations containing observation n. The exchangeability of the IBP allows us to imagine that X_W are the final W observations and X_n is the last observation. Using Bayes rule, we write the probability $p(Z_{nk} =$

$1|Z_{-nk}, X)$ as:

$$\begin{aligned} p(Z_{nk} = 1|Z_{-nk}, X) &\propto p(Z_{nk}|Z_{-nk})p(X|Z) \\ &= \frac{m_k}{n}\int_A p(X|Z, A)p(A)dA. \end{aligned}$$

We split the data into sets X_W and X_{-W} and apply the conditional independencies from figure 4.1 to get

$$\begin{aligned} &p(Z_{nk} = 1|Z_{-nk}, X) \\ &= \frac{m_k}{n}\int_A p(X_W, X_{-W}|Z_W, Z_{-W}, A)p(A)dA \\ &= \frac{m_k}{n}\int_A p(X_W|Z_W, A)p(X_{-W}|Z_{-W}, A)p(A)dA. \end{aligned}$$

Finally, we apply Bayes rule again to $p(X_{-W}|Z_{-W}, A)$:

$$p(Z_{nk} = 1|Z_{-nk}, X) \propto \frac{m_k}{n}\int_A p(X_W|Z_W, A)p(A|X_{-W}, Z_{-W})dA \tag{4.1}$$

Thus, given $p(A|X_{-W}, Z_{-W})$, we can compute $p(Z_{nk} = 1|Z_{-nk}, X)$ exactly without touching X_{-W}.

In the linear-Gaussian model, the feature posterior $p(A|X_{-W}, Z_{-W})$ and likelihood $p(X_W|Z_W, A)$ are both Gaussian, and the integral in equation 4.1 yields

$$P(Z_{nk}|Z_{-nk}, X) \propto \frac{m_k}{n}\mathcal{N}(X_W; Z_W\mu^A_{-W}, Z_W\Sigma^A_{-W}Z_W^T + \Sigma^X), \tag{4.2}$$

where $(\mu^A_{-W}, \Sigma^A_{-W})$ is the mean and covariance of the feature posterior and $\Sigma^X = \sigma_x^2 I$ is the noise variance.

The accelerated sampling procedure, summarised in Algorithm 1, only uses a small window of observations X_W at one time. We first compute the feature posterior given all of the data using equation A.3. Next, we randomly choose W observations as our window. Given the full feature posterior $p(A|X, Z) = \mathcal{N}(\mu^A, \Sigma^A)$, the posterior $p(A|X_{-W}, Z_{-W})$ with the window X_W removed is

Algorithm 1 Accelerated Gibbs Sampler

```
Initialise Z.
Compute p(A|Z, X).
for each iteration do
  S ← {1 ... N}
  while S is not empty do
    Sample and remove W elements from S.
    Compute p(A|Z_{-W}, X_{-W}) from p(A|Z, X).
    for each observation n in W do
      Sample Z_{nk} according to equation 4.2.
      Sample new features for each observation n according to equation 3.7.
    end for
    Update p(A|Z, X) from p(A|Z_{-W}, X_{-W}).
  end while
end for
```

given by

$$\begin{aligned}\Sigma^A_{-W} &= (I-\Sigma^A Z_W^T(Z_W\Sigma^A Z_W^T-\Sigma^X)^{-1}Z_W)\Sigma^A \\ \mu^A_{-W} &= b(\mu^A-\Sigma^A Z_W^T(Z_W\Sigma^A Z_W^T+\Sigma^X)^{-1}X_W)\end{aligned}$$

where $b=(I-\Sigma^A Z_W^T(Z_W\Sigma^A Z_W^T+\Sigma^X)^{-1}Z_W\Sigma^A)^{-1}$. Now each Z_{nk} in Z_W may be sampled using equation 4.2. New features are sampled using equation 3.7. In practice, using the information form $P^A = (\Sigma^A)^{-1}$, $h^A = P\mu^A$ results in slightly more stable updates. We convert P^A_{-W} and h^A_{-W} to Σ^A_{-W} and μ^A_{-W} before computing likelihoods, so the computational complexity is unchanged.

Each sampling step requires the computation of $P(X_W|Z_W)$, given by $\mathcal{N}(X_W; Z_W\mu^A_{-W}, Z_W\Sigma^A_{-W}Z_W^T + \Sigma^X)$. If only one element Z_{nk} of Z_W changes, the new $(\Sigma^X_W)^{-1}$ and its determinant can be computed by a pair of rank-one updates; new features may also be incorporated into the covariance via a single rank-one update. If the Woodbury formulas are used, the updates require $O(W^2)$ elementary operations. Griffiths and Ghahramani (2005) show an $O(K^2)$ update is also possible. However, the $O(W^2)$ inversions will be faster because we expect K to grow with the number of observations while W can be fixed to remain small.

Once we have completed sampling the window, we recompute the full feature posterior $p(A|X,Z)$:

$$\begin{aligned}\mu^A &= \mu^A_{-W} + \Sigma^A_{-W} Z_W^T \cdot (Z_W \Sigma^A_{-W} Z_W^T + \Sigma^X)^{-1}(X_i - Z_W \mu^A_{-W}) \\ \Sigma^A &= (I - \Sigma^A_{-W} Z_W^T (Z_W \Sigma^A_{-W} Z_W^T + \Sigma^X)^{-1} Z_W) \Sigma^A_{-W}\end{aligned}$$

For each iteration, the sampler then chooses W new observations without replacement and repeats until all N observations have been updated.

4.3 Per-Iteration Running Times

We consider the number of addition and multiplication operations for one sweep through an NxK feature-assignment matrix Z under a linear-Gaussian likelihood model (ignoring the addition of new features). The running time of the collapsed Gibbs sampler is dominated by the computation of the exponent $X^T(I - Z(Z^TZ + I\sigma_x^2/\sigma_a^2)^{-1}Z^T)X$. If only one element of Z is changed, then the inverse $(Z^TZ + I\sigma_x^2/\sigma_a^2)^{-1}$ may be updated in $O(K^2)$ time. However, the remaining matrix products require $O(N^2K)$ and $O(N^2D)$ operations respectively. Thus, for NK elements, the running time for large N is $O(NK(N^2K + N^2D))$.

The uncollapsed Gibbs sampler requires many fewer operations per element: $p(X_n|Z_n, A)$ is independent of the remaining observations and only requires the computation of Z_nA, which is $O(KD)$. The features A are resampled once per iteration. Computing the mean and variance for A given Z requires steps that $O(K^3 + NK^2 + NKD)$, but the $O(NK^2D)$ time required to sample NK elements dominates these terms.

Finally, as with the collapsed Gibbs sampler, the accelerated Gibbs sampler's per-iteration running time also has a dominant term from computing the likelihood. If Woodbury inversions are used, the likelihood computations are $O(W^2+DW^2)$, for a complexity $O(NKDW^2)$ for sampling all the Z_{nk}. However, the feature posterior must also be updated N/W times. Each update requires steps of complexity $O(W^3+W^2K+WK^2)$, corresponding to inverting

the variance of the data X_W and expanding it to compute the variance of the parameters. Thus, the overall complexity is $O(N(KDW^2+K^2))$, and the optimal value for W is 1.[2] Like the uncollapsed Gibbs sampler, the accelerated sampler's complexity is linear in N.

Table 4.1 summarises the running times. Both the uncollapsed and accelerated Gibbs sampler are linear in N, but the accelerated Gibbs sampler has a slightly better complexity in K and D. The lower order dependence is beneficial for scaling because we expect K to grow with N. While constants and lower-order terms are important, our experiments confirm the lower complexity is significant in the larger datasets.

Table 4.1: Per-iteration running times for large N and a linear-Gaussian likelihood model, given an NxK matrix Z and D-dimensional data.

Algorithm	Running Time
Collapsed Gibbs	$O(N^3(K^2+KD))$
Uncollapsed Gibbs	$O(NDK^2)$
Accelerated Gibbs	$O(N(KDW^2+K^2))$
Accelerated Gibbs, W=1	$O(N(K^2+KD))$

4.4 Experiments

We compared the computational efficiency and inference quality of the collapsed Gibbs sampler, the semi-collapsed Gibbs sampler (an uncollapsed Gibbs sampler that integrates over the new features when sampling k_{new}), and the accelerated Gibbs sampler. All samplers were heavily optimised to take advantage of rank-one updates and Matlab vectorisation.

Per-iteration runtimes were used to evaluate computational efficiency. Since Gibbs sampling produces a sequence of correlated samples, another important metric was the degree of independence between successive samples: more independent samples indicate faster mixing. With the synthetic data, we ran 5

[2]We included W in our original formulation because in general the optimal choice of W will depend on the relative costs of computing the feature posterior and the likelihood.

chains in parallel to evaluate the effective number of independent samples per actual sample (Gelman et al., 2003). The effective number of samples per sample will be 1 if successive samples are independent and less than 1 otherwise. Experiments with real-world data took longer to complete, so we estimated the burn-in time instead. In both cases, the number of features K and the training log-likelihood were the chain statistics used to evaluate mixing.

We evaluated the inference quality by first holding out approximately $100D$ X_{nd} values during the initial inference. No observation had all of its dimensions missing, and no dimension had all of its observations missing. The quality of the inference was measured by evaluating the average L2 reconstruction error over the missing elements x^m_{nd},

$$e_{l2} = \sum_i \sum_{x^m_{nd}} (x^m_{nd} - z^i_n a^i_d)^2$$

where i denotes the sample, and the average test log-likelihood treating each missing value as a separate test point

$$\begin{aligned} e_{ll} &= \sum_i \sum_{x^m_{nd}} -\frac{1}{2}\log(2\pi\sigma^2_{nd}) - \frac{1}{2\sigma^2_{nd}}(x^m_{nd} - z^i_n a^i_d)^2, \\ \sigma^2_{nd} &= \sigma^2_x + z_n \Sigma_A z_n^T \end{aligned}$$

The metrics were averaged over the final 50 samples.

Synthetic Data The synthetic data were generated from the IBP prior ($\alpha = 2$) and the linear-Gaussian model ($\mu^A = 0, \sigma_a = 2, \sigma_x = .2$, $D = 10$). We ran 5 chains from different starting positions for each sampler with N equal to 50, 100, 250, and 500 for 1000 iterations. Figure 4.2 shows the evolution of the per-iteration runtime as the number of observations grows (note the log-log scale). The slopes of the lines indicate the order of the runtime polynomial. The semi-collapsed and accelerated Gibbs samplers have nearly identical slopes, while the per-iteration running time of the collapsed Gibbs sampler scales much more poorly.

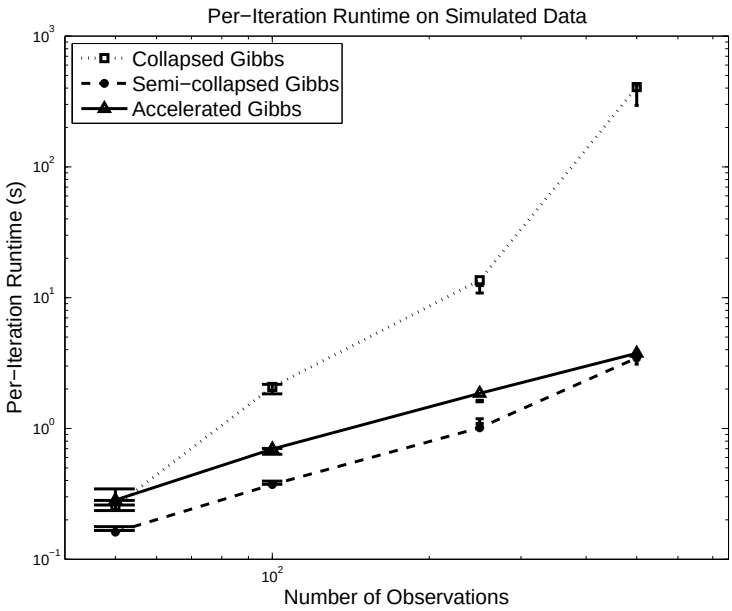


Figure 4.2: Per iteration runtime vs. number of observations (note the log scale on the x-axis).

Table 4.2: Performance statistics on data from the prior. Each value is an average over 5 chains.

Evaluation Statistic	Collapsed Gibbs	Semi-collapsed Gibbs	Accelerated Gibbs
Iteration Time	90.273	**1.302**	1.566
Effective N	0.020	0.019	**0.020**
L2 Test Error	**0.710**	4.534	1.780
Test log Likelihood	-24.557	**-13.977**	-26.936

Figure 4.3 plots the effective sample count (per sample) of for each sampler. Again, as expected, the semi-collapsed Gibbs sampler had the lowest effective sample count, and the accelerated Gibbs sampler and the collapsed Gibbs sampler had similar counts. From these two plots, we see that the accelerated Gibbs sampler mixes like the collapsed Gibbs sampler but has a run-time like the uncollapsed Gibbs sampler.

The running times and quality criteria are summarised in table 4.2. The accelerated Gibbs sampler's running time is on par with the semi-collapsed Gibbs sampler and nearly two orders of magnitude faster than the collapsed Gibbs sampler. It does not achieve the best test likelihoods or reconstruction errors, but they are always on par with collapsed Gibbs sampler.

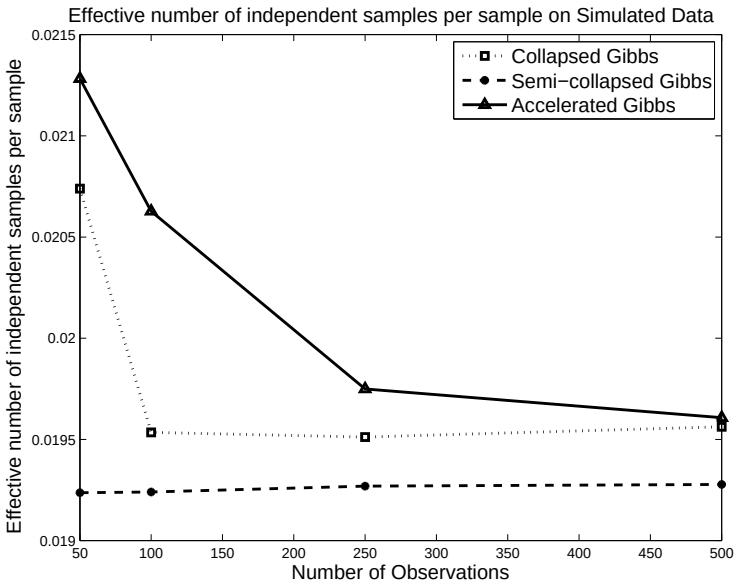
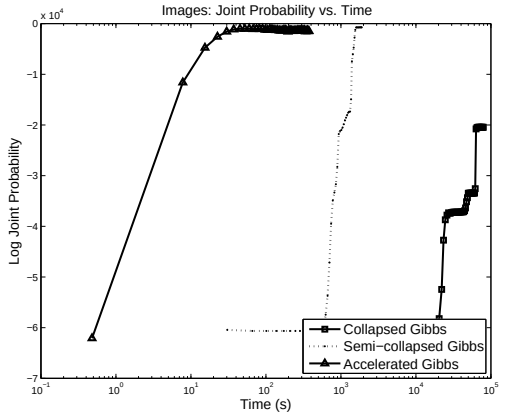


Figure 4.3: Effective number of samples vs. dataset size.

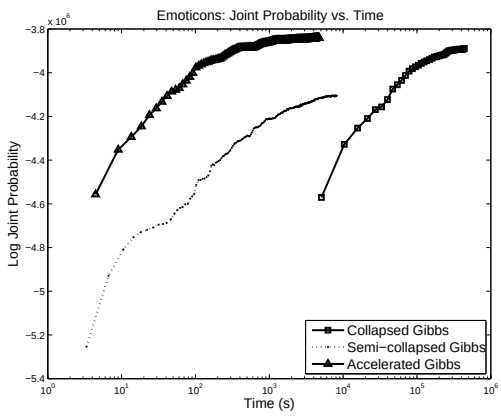
Real Data Table 4.3 summarises the real-world data sets.[3] All datasets were first centred to have 0-mean. Values for the hyperparameters σ_a and σ_x were fixed to $.75\sigma$ and $.25\sigma$, respectively, where σ was the standard deviation of the observations across all dimensions. We set $\alpha = 2$ in all experiments. Each sampler ran for 500 iterations or up to 150 hours.

Figure 4.4 plots how the log-joint probability $\log p(X, Z)$ of the training data evolves for all six datasets (note the log-scale on the time axis). In almost all the datasets, the accelerated Gibbs sampler equilibrates to the same log-joint probability as the semi-collapsed Gibbs sampler but orders of magnitude faster. The plots suggest the time allowed was not sufficient for the collapsed Gibbs sampler to complete burning-in, though it seems to be approaching a similar joint probability. For the higher dimensional and larger datasets in the bottom row, the collapsed Gibbs sampler often failed to complete even one iteration in 150 hours. The semi-collapsed sampler also suffered in the higher dimensional datasets. The accelerated Gibbs sampler always completed its run in less than two hours.

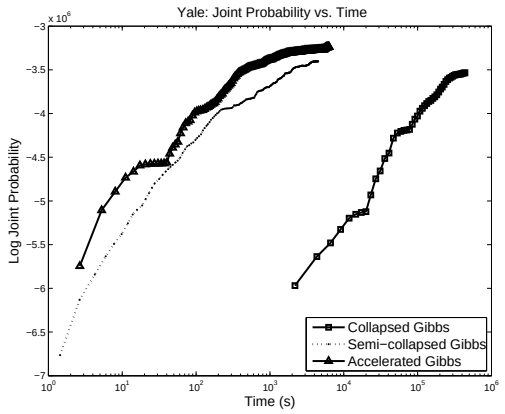
[3]The block images are not a real-world dataset, but they do not come from the linear-Gaussian model. The emoticons were obtained from www.smileyset.com and post-processed to normalise image size and centring.



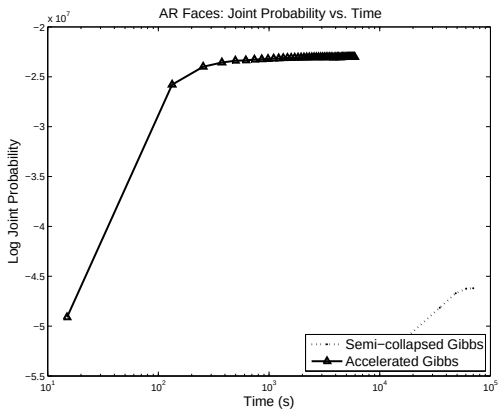
(a) Block Images: N = 1000, D = 36



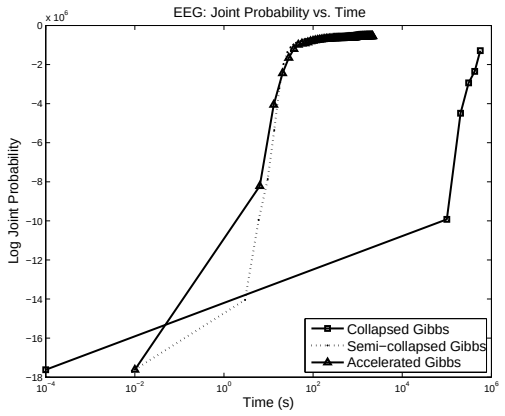
(b) Emoticons: N = 702 , D = 1032



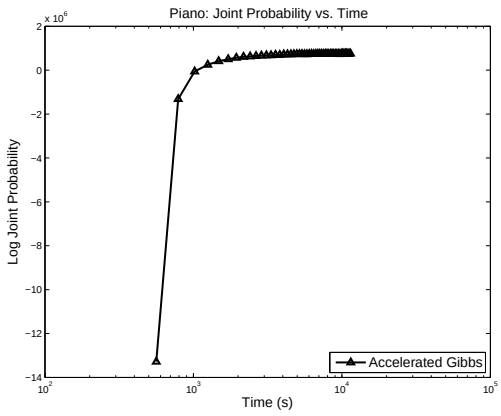
(c) Yale Faces: N = 722 , D = 1032



(d) AR Faces: N = 2600 , D = 1598



(e) EEG: N = 4400 , D = 32



(f) Piano: N = 10000 , D = 161

Figure 4.4: Evolution of the log joint-probability vs. time for several datasets.

Table 4.3: Descriptions of datasets.

Dataset	N	D	Description
Block-Images (Wood and Griffiths, 2007)	1000	36	noisy overlays of four binary shapes on a grid
Emoticons	702	1032	different smiley faces
Yale (Georghiades et al., 2001)	722	1032	faces with various lighting
AR (Mart'inez and Kak, 2001)	2600	1598	faces with lighting, accessories
EEG (Hoffmann et al., 2008)	4400	32	EEG recording on various tasks
Piano (Poliner and Ellis, 2007)	10000	161	DFT of a piano recording

Table 4.4 shows the per-iteration running times and the measures on the missing data for the six datasets. A dash indicates insufficient iterations were completed to compute the statistic. The speed benefit of the accelerated Gibbs sampler is particularly apparent on these larger datasets: often it is two orders of magnitude faster per-iteration than the collapsed sampler. However, its test L2 reconstruction error and likelihood remains on-par or better than either sampler. For slightly troublesome EEG dataset, most of the likelihood loss came from 4 dimensions of 4 observations. These four troublesome dimensions had a variance about two orders of magnitude larger than the remaining dimensions (the other datasets were more homogenous). All samplers had to manage this issue, but the accelerated Gibbs sampler's heavy reliance on rank-one updates made it particularly sensitive to ill-conditioning from extreme situations.

Finally, figures 4.5 and 4.6 show qualitative results on the emoticon and AR faces datasets. Our primary objective here was to show that the accelerated sampler finds reasonable features (rather than discover previously unknown structure in the data). Each figure decomposes three observations into features found by the sampler (all features contribute equally in the additive model). The sampler finds features associated with the emoticon's expression (first

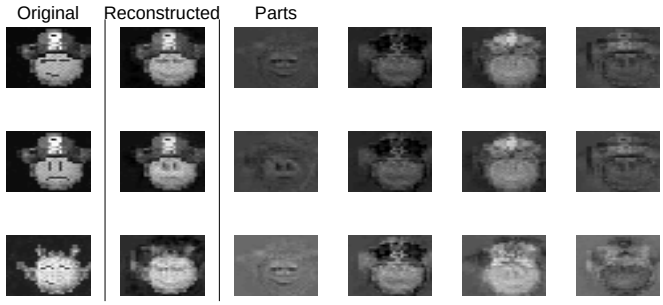


Figure 4.5: Some reconstructed emoticons: the expression is a different feature than the type of the face.

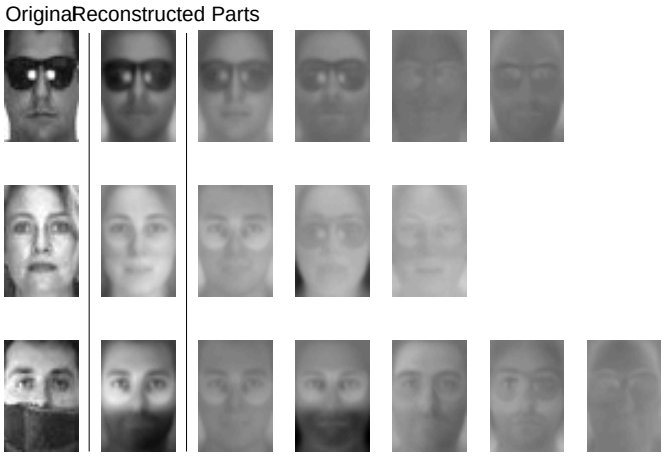


Figure 4.6: Some reconstructed AR faces: faces contain features for the face, lighting, scarves, and sunglasses.

column of 'Parts') and features that make up emoticon's accessories. In the larger AR Faces dataset, the sampler finds features corresponding to lighting conditions, facial features, and accessories such as scarves or sunglasses. These decompositions are similar to those found by the other samplers.

4.5 Discussion

The accelerated Gibbs sampler attained similar test likelihood and reconstruction errors as the existing Gibbs samplers but did so faster than either approach. Its per-iteration runtime was two orders of magnitude faster than

the collapsed Gibbs sampler. While it is difficult to judge the convergence of the samplers from a single run, the accelerated Gibbs sampler's training likelihoods were generally on par with the other samplers, suggesting that the gains in speed were achieved without sacrifices in performance.

The difference between the NK^2D term in the running time of the uncollapsed Gibbs sampler and the $N(K^2 + KD)$ of the accelerated Gibbs sampler becomes more pronounced as the dimensionality of the observations and the number of features increases. If K grows logarithmically with N, as posited by the IBP prior, then large datasets will impact the complexity because K will be larger. For example, in the AR Faces dataset, where D is 1598 and feature counts ranged between 15-80, the difference between the two expected running times is one to two orders of magnitude. (Of course, implementation details affect the differences in per-iteration running times, but the experiments confirm this speed-up factor.[4])

Loss of precision during the rank-one updates is the primary implementation concern for the accelerated sampler. This effect was particularly apparent in the EEG dataset because the dimensions had widely varying variances, leading to ill-conditioned matrices. An important question is how often expensive full-updates of the feature posterior and data posterior must be completed; we did so once per run and once per window, respectively. While a window size W of 1 produces an optimal runtime, larger windows might slow the degradation due to repeated rank-one updates.

Our comparison focused on performance relative to other Gibbs samplers. Regarding other IBP inference techniques, we expect it to mix faster than the slice sampler, as the slice sampler mixes more slowly than the collapsed Gibbs sampler (Teh et al., 2007). Well-designed MH-proposals may make a Metropolis sampler efficient in some domains, and other domains may be well-suited for particle filtering, but one advantage of the accelerated Gibbs sampler is its simplicity: neither MH-proposals nor a particle count is needed to perform principled, effective inference. However, an interesting question is if population

[4]Using the semi-collapsed Gibbs sampler instead of the uncollapsed Gibbs sampler will also affect the runtime, but this effect appeared to be small in preliminary experiments.

Monte Carlo techniques can be used to merge the benefits of our accelerated Gibbs sampler and particle filtering.

Finally, we note that the approach described here can be easily converted to provide a greedy, deterministic approximation: instead of sampling, one can simply climb in the direction of increasing likelihood. In practice, this approach gives almost the same quality of solutions as sampling, as posterior modes tend to be highly peaked and with large datasets the sampler never has a chance to explore multiple modes in the posterior.

4.6 Summary

The accelerated Gibbs sampler has the mixing properties of the collapsed Gibbs sampler but the running time of the uncollapsed Gibbs sampler. The key insight was to allow data to share information only through the statistics on the feature posterior. Thus, likelihoods of feature assignments could efficiently be computed using local computations without sacrificing the flexibility of a collapsed sampler. The accelerated Gibbs sampler scaled to inference for several large, real-world datasets. An interesting future direction may be to explore how maintaining distributions over parameters may help accelerate other MCMC inference methods.

Table 4.4: Results on realworld data sets. Dashes indicate values that could not be computed in the time allotted.

Data Set	Collapsed Gibbs	Semi-collapsed Gibbs	Accelerated Gibbs
Iteration Time			
Images	1526.4	38.1	**7.6**
Emoticons	10205.4	32.8	**19.0**
Yale	8759.0	**18.1**	25.5
AR Faces	-	13291.8	**120.9**
EEG	112692.3	**13.0**	21.9
Piano	-	-	**221.5**
Burn-in Count			
Images	12.0	**2.0**	4.0
Emoticons	**35.0**	82.0	42.0
Yale	**45.0**	184.0	133.0
AR Faces	-	-	**31.0**
EEG	-	-	**29.0**
Piano	-	-	**25.0**
L2 Test Error			
Images	0.1066	0.1239	**0.0555**
Emoticons	5905.8	11781.1	**5612.7**
Yale	459.4	**413.0**	435.9
AR Faces	-	3836.4	**919.9**
EEG	267292.3	282108.1	**215174.5**
Piano	-	-	**0.0005**
Test Likelihood			
Images	-2.0	**-1.7**	-2.0
Emoticons	-14.4	**-12.0**	-14.2
Yale	-17.8	**-15.8**	-16.0
AR Faces	-	**-13.3**	-13.7
EEG	-26274.0	**-3621.5**	-14133.3
Piano	-	-	**-7.0**

Chapter 5

Parallel Inference [1]

The accelerated sampler of chapter 4, while orders of magnitude faster than the original collapsed Gibbs sampler, only takes advantage of a single processor. However, from information retrieval and recommender systems, to bioinformatics and financial market analysis, recent years have seen an explosion in the size of datasets. These datasets may be spread across several hard-disks and certainly exceed the memory capacity of a single machine.

While large, these datasets are often sparse and difficult to model. For example, we may have expression levels from thousands of genes from only a few people. A rating database may contain millions of users and thousands of movies, but each user may have only rated a few movies. In such settings, Bayesian methods provide a principled and robust approach to drawing inferences and making predictions from sparse information. However, scaling Bayesian inference, especially for nonparametric models, remains a challenge.

Advances in multicore and distributed computing provide one answer to this challenge. Already techniques exist to efficiently split variables across processors in undirected graphical models (Joseph Gonzalez, 2009) and factor graphs (Stern et al., 2009); specific techniques have been developed data parallelisation for latent Dirichlet allocation (Nallapati et al., 2007; Newman et al., 2008; Asuncion et al., 2008). For more complex models, such as Bayesian factor analysis or the IBP, data parallelisation of inference (with marginalised

[1] Joint work with David Knowles and Shakir Mohammed.

variables) is nontrivial—while simple models might only require pooling a small number of sufficient statistics (Chu et al., 2007), correct inference in more complex models can depend on frequent synchronisation or worse yet, having to communicate entire probability distributions between processors. This chapter describes a message passing framework for data-parallel Bayesian inference for nonparametric models.

To achieve efficient parallelisation, we apply the core idea of chapter 4: given a distribution over certain coupling features A, the distribution over other latent variables Z_n becomes factorised. Coupled with a message passing scheme to maintain the posterior $p(A|Z, X)$, we can distribute inference over many processors while sacrificing little accuracy in performance; we demonstrate our approach on a problem with 100,000 observations. As with the accelerated Gibbs sampler, most elements of our procedure are general to data parallelisation in other models.

5.1 Inference Procedure

In this section, we describe synchronous and asynchronous procedures for approximate, parallel inference in the IBP. Procedures are described for the linear-Gaussian and Bernoulli models; however they should readily extend to other likelihood models. Section 5.2 describes how to make this approximate sampler exact, at the expense of requiring synchronous communications between processors.

The first step is to partition the dataset between the processors. We let X^p and Z^p denote the portions of the data X and feature assignment matrix Z assigned to processor p. In chapter 4, the distribution $P(A|X_{-n}, Z_{-n})$ was used to derive an accelerated sampler for sampling Z_n. Similarly, our parallel inference algorithm will have each processor p to maintain a distribution $P^p(A|X_{-n}, Z_{-n})$, where P^p is an approximation to the true posterior $P(A|X_{-n}, Z_{-n})$. We show that the distributions P^p (as well as counts of Z) will be efficiently updated via message passing between the processors.

The inference alternates between three phases:

- Message passing: processors communicate to compute the exact $P(A|X,Z)$.
- Gibbs sampling: processors sample a new set of Z^p's in parallel.
- Hyperparameter sampling: a designated root processor resamples hyperparameters, which are propagated to other processors

We detail each of these phases below and then describe how they may be implemented in an asynchronous system. Because all of the processors are sampling Z at once, the posteriors $P^p(A|X,Z)$ used by each processor are no longer exact. However, we show empirically in section 5.2 that this approximation has little effect on the inference.

Message Passing The full posterior on the features A is given by $P(A|Z,X)$. Bayes Rule gives us the following factorisation:

$$P(A|Z,X) \propto P(A) \prod_p P(X^p|Z^p,A) \tag{5.1}$$

If the prior $P(A)$ and the likelihoods $P(X^p|Z^p,A)$ are part of conjugate exponential family models, then the product in equation (5.1) is equivalent to summing the sufficient statistics of the likelihoods from all of the processors. In the linear-Gaussian model, these statistics correspond to mean vectors and covariance matrices (handled more readily in information mean and precision form); in the Bernoulli model, the statistics correspond to counts on how often each element A_{kd} is equal to one. The linear-Gaussian messages have size $O(K^2 + KD)$, and the Bernoulli messages $O(KD)$. For nonparametric models such as the IBP, the number of features K grows as $O(\log N)$. This slow growth means that messages remain compact and we can efficiently scale to large datasets.

The most straightforward way to accurately compute the full posterior is to network the processors in a tree architecture. The sufficient statistics for the feature posterior are summed via message passing along the tree, which is an instance of (exact) belief propagation. Specifically, the message s from

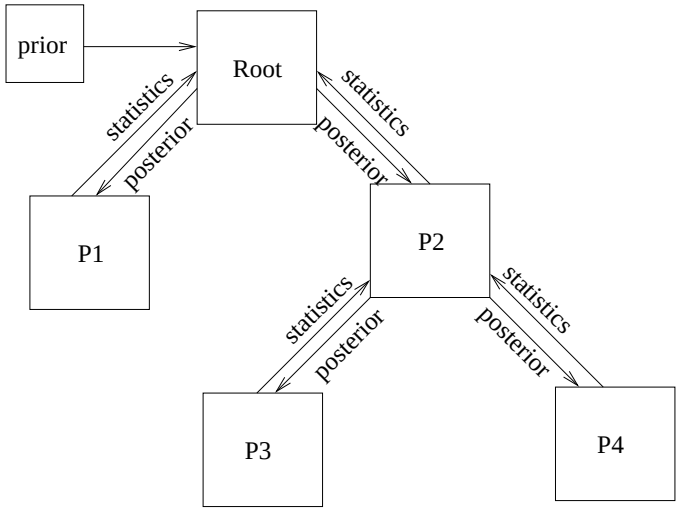


Figure 5.1: Message passing process. The processors send sufficient statistics of the likelihood up to the root, which calculates and sends the full (exact) posterior down to the processors.

processor p to processor q is given by

$$s_{p \to q} = l^p + \sum_{r \in N(p) \backslash q} s_{r \to p}$$

where $N(p)\backslash q$ are the processors attached to p besides q and l^p are the sufficient statistics from processor p. When the summing the statistics, care must be taken to ensure that feature identities stay aligned, that is, ensuring that feature i on processor p is the same as feature i on processor q. Unlike in parallel inference for the HDP (Asuncion et al., 2008), simply matching features by their index, that is, just assuming that the i^{th} feature on processor p was also the i^{th} feature on processor q, often leads to disastrous results. In fact, in the parallel implementation, processors pass around a token, and only a processor possessing a token may add new features. Finally, a dummy neighbour containing the statistics of the prior is connected to the root processor. Also summed are the counts $m_k^p = \sum_{n \in X^p} Z_{nk}^p$, the popularity of feature k within processor p. Figure 5.1 shows a cartoon of the message passing process.

Gibbs Sampling In general, Z_{nk} can be Gibbs-sampled using Bayes rule

$$P(Z_{nk}|Z_{-nk}, X) \propto P(Z_{nk}|Z_{-nk})P(X|Z).$$

The probability $P(Z_{nk}|Z_{-nk})$ depends on the total number of observations and the number of observations m_k for which feature k is active. The message passing provides each processor with an accurate count for m_k, from which it may compute $m_k^{-p} = m_k - m_k^p$ using its current count m_k^p. Each processor updates its own m_k^p as it samples Z^p and assumes that m_k^{-p} stays fixed (a good approximation for popular features).

For conjugate models, we evaluate the likelihood $P(X|Z)$ via the integral

$$P(X|Z) \propto \int_A P(X_n|Z_n, A)P(A|Z_{-n}, X_{-n})dA,$$

where the partial posterior $P(A|Z_{-n}, X_{-n})$ is given by

$$P(A|Z_{-n}, X_{-n}) \propto \frac{P(A|Z, X)}{P(X_n|Z_n, A)}. \tag{5.2}$$

Because the model is conjugate, the partial posterior in (5.2) can be efficiently computed by subtracting observation n's contribution to the sufficient statistics.

For non-conjugate models, we can use a variational distribution $Q(A)$ to approximate $P(A|X, Z)$ during message passing, where $Q(A)$ is a projection of P onto an exponential family (similar to variational message passing (Winn and Bishop, 2005)). Thus, computing the partial posterior $Q^{-p}(A)$ is identical to the conjugate case. Given the partial posterior $Q^{-p}(A)$, the processor initialises an uncollapsed Gibbs sampler with a draw A from $Q^{-p}(A)$. The samples of A from the uncollapsed sampler are used to compute the sufficient statistics for the likelihood $P(X|Z)$.

Hyperparameter Resampling The IBP concentration parameter α and hyperparameters of the likelihood can also be sampled during inference. Resampling α depends only on the total number of active features K and the

number of observations N. Given a vague $\Gamma(1,1)$ prior, the posterior on α is

$$\alpha \sim \Gamma(1+K, 1+H_N)$$

Thus, it can easily be resampled at the root and propagated to the other processors.

In the linear-Gaussian model, the posteriors on the noise and feature variances (starting from inverse gamma priors) depend on various squared-errors, which can also be computed in a distributed fashion:

$$\begin{aligned}
\sigma_n^{-2} &\sim \Gamma(1+\frac{1}{2}ND, 1+\frac{1}{2}\sum_n\sum_d(X_{nd}-\sum_k Z_{nk}A_{kd})^2) \\
\sigma_a^{-2} &\sim \Gamma(1+\frac{1}{2}KD, 1+\frac{1}{2}\sum_k\sum_d A_{kd}^2)
\end{aligned}$$

For more general, non-conjugate models, resampling the hyperparameters requires two steps. In the first step, a hyperparameter value is proposed by the root and propagated to the processors. The processors each compute the likelihood of the current and proposed hyperparameter values and propagate this value back to root. The root evaluates a Metropolis step for the hyperparameters and propagates the decision back to the leaves. The two-step approach introduces a latency in the resampling but does not require any additional message passing rounds.

Asynchronous Operation So far we have discussed message passing, Gibbs sampling, and hyperparameter resampling as if they occur in separate phases. In practice, these phases may occur asynchronously: between its Gibbs sweeps, each processor updates its feature posterior based on the most current messages it has received and sends likelihood messages to its parent. Likewise, the root continuously resamples hyperparameters and propagates the values down through the tree. Asynchronous operation adds an additional layer of approximation in that information about changes to feature posteriors in some processors may be delayed in reaching other processors. As a result, rare features may take longer to be added: a long delay may occur between when one

processor adds a particular feature and a different processor decides to use it, and in that delay the first processor may have deleted the feature. However, asynchronous operation does allow faster processors to share information and perform more inference on their data instead of waiting for slower processors.

Feature Sub-sampling to Scale to Larger Datasets Both likelihood models discussed above require the product ZA to be computed to evaluate each Gibbs update. Normally, this computation requires $O(KD)$ elementary multiplication and addition operations. The overall complexity can be reduced because of the local nature of the Gibbs updates, but it still remains $O(K)$ or $O(D)$, depending on whether the updates are for A or Z, respectively.

If the IBP prior is a reasonable model of the data, we expect the number of features K in a dataset to be $O(\log(N))$. As N grows large, therefore, computations that depend on K are also potentially slow. For larger datasets, we sample only a subset of the K features in each iteration. The features to be sampled is chosen randomly at the start of each Gibbs iteration. As such, over an infinite run of the sampler, each feature will be sampled infinitely often and the sampling method remains exact.

While a promising idea, we found that feature sub-sampling was not useful in our datasets: it largely had only the effect of slowing down the mixing rate of the sampler. However, we believe that smarter choice of which features to sample—we chose them randomly—may make this approach valuable for very large datasets.

5.2 Comparison to Exact Metropolis

Because all Z^p's are sampled at once, the posteriors $P^p(A|X, Z)$ used by each processor in section 5.1 are no longer exact.[2] Below we show how Metropolis–Hastings (MH) steps can make the parallel sampler exact, but the approach introduces significant computational overheads both in computing the transi-

[2] A fully uncollapsed sampler could be parallelised exactly without MH proposals, but collapsed samplers generally have better mixing properties.

tion probabilities and in the message passing. Moreover, we observe empirically that the approximate inference behaves very similarly to the MH sampler.

Exact Parallel Metropolis Sampler. Let $\hat{Z}^p$ be the matrix output after a set of Gibbs sweeps on Z^p. We use all the $\hat{Z}_p$'s to propose a new Z' matrix. The acceptance probability of the proposal is

$$\min(1, \frac{P(X|Z')P(Z')Q(Z' \to Z)}{P(X|Z)P(Z)Q(Z \to Z')}), \tag{5.3}$$

where the likelihood terms $P(X|Z)$ and $P(Z)$ are readily computed in a distributed fashion. For the transition distribution Q, we note that if we set the random seed r, then the matrix $\hat{Z}^p$ from the Gibbs sweeps in the processor is some deterministic function of the input matrix Z_p. The proposal $Z^{p'}$ is a (stochastic) noisy representation of $\hat{Z}^p$ in which for example

$$\begin{aligned} P(Z^{p'}_{nk} = 1) = .99 \quad & \texttt{if} \quad \hat{Z}^p_{nk} = 1, \\ P(Z^{p'}_{nk} = 1) = .01 \quad & \texttt{if} \quad \hat{Z}^p_{nk} = 0 \end{aligned} \tag{5.4}$$

where K should be at least the number of features in $\hat{Z}^p$. We set $Z^{p'}_{nk} = 0$ for $k > K$.

To compute the backward probability, we take $Z^{p'}$ and apply the same number of Gibbs sampling sweeps with the same random seed r. The resulting $\hat{Z}^{p'}$ is a deterministic function of $Z^{p'}$. The backward probability $Q(Z^{p'} \to Z^p) = Q(\hat{Z}^{p'} \to Z^p)$, where $Q(\hat{Z}^{p'} \to Z^p)$ can be computed from (5.4). While the transition probabilities can be computed in a distributed, asynchronous fashion, all of the processors must synchronise when deciding whether to accept the proposal. Figure 5.2 shows a cartoon of the proposal process.

Experimental Comparison To compare the exact Metropolis and approximate inference techniques, we ran both approaches on 1000 block images of Griffiths and Ghahramani (2005) on 5 simulated processors. Each test was repeated 25 times. The approximate inference was run for 500 iterations in which each processor completed 5 Gibbs sweeps per iteration. The exact in-

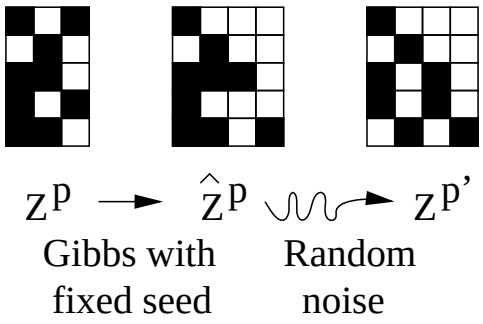


Figure 5.2: Cartoon of the transition distribution for the MH proposal. The first step 'sampling' a deterministic $\hat{Z}^p$ given an input Z^p. Next, $\hat{Z}^p$ is stochastically 'fuzzified' to get the proposal $Z^{p'}$.

ference, due to the MH rejections, required much longer to burn-in, so it was run for 7500 iterations in which each processor computed one Gibbs sweep. To keep the probability of an acceptance reasonable, we allowed each processor to change the feature assignments Z_n for only 1, 5, or 10 observations each per sweep.

In table 5.1, we see that the approximate sampler runs about five times faster than the exact samplers while achieving comparable (or better) predictive likelihoods and reconstruction errors on held-out data. Feature subsampling only sampled a few features per iteration, leading to faster running times but slightly worse performance. Figures 5.3, 5.4, and 5.5 show empirical CDFs, computed using the final half of the 25 trials, for the IBP concentration parameter α, the noise variance σ_n^2, and the feature variance σ_a^2. The approximate sampler (black) produces similar CDFs to the various exact Metropolis samplers (gray). Some of the distributions (especially that on the α parameter) are different. On the exact samplers, we believe that poor mixing is the primary culprit. For the approximate sampler, there is a slight bias toward fewer features due to the latency between when a processor creates a new feature and when other processors have a chance to use it. However, the approximate sampler is close enough to the exact one, so that we use it exclusively in the remaining experiments.

Table 5.1: Comparison of running times and performance between the MH and approximate sampler. Values are averaged over 25 runs.

Method	Time (s)	Test L_2 Error	Test Log Likelihood	MH Accept Proportion
MH, n = 1	717	0.0468	0.1098	0.1106
MH, n = 5	1075	0.0488	0.0893	0.0121
MH, n = 10	1486	0.0555	0.0196	0.0062
Approximate	179	0.0487	0.1292	-
Feature Sub-sampling	120	0.0436	0.1489	-

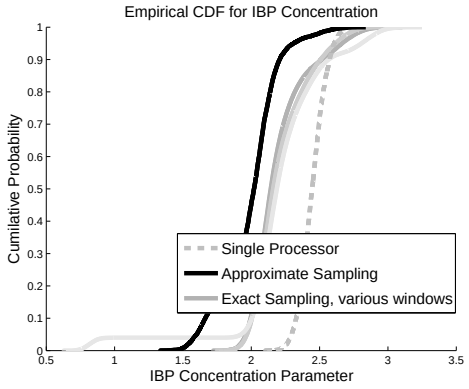


Figure 5.3: Empirical CDF of IBP α. The solid black line is the approximate sampler; the three solid gray lines are the MH samplers with n equal to 1, 5, and 10; lighter shades indicate larger n. The approximate sampler and the MH samplers for smaller n have similar CDFs; the $n = 10$ MH sampler's differing CDF indicates it did not mix in 7500 iterations (reasonable since its acceptance rate was 0.0062).

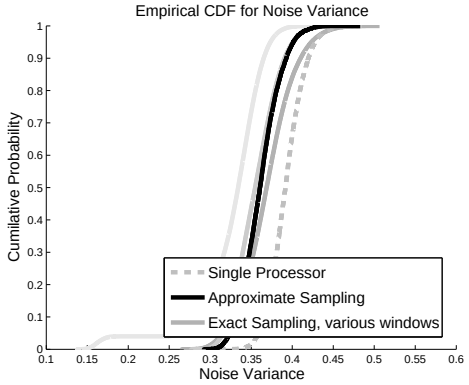


Figure 5.4: Empirical CDF of σ_x^2; same line types as figure 5.3

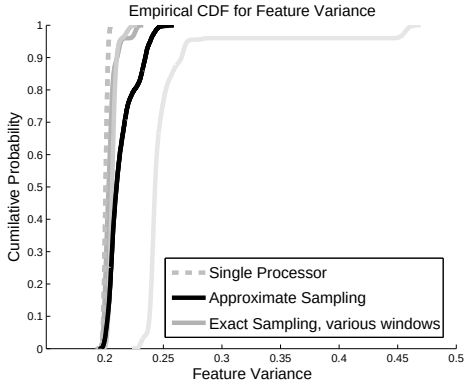


Figure 5.5: Empirical CDF of σ_a^2; same line types as figure 5.3

5.3 Sampler Properties[3]

We ran a series experiments on 10,000 block images of Griffiths and Ghahramani (2005) to study the effects of various sampler configurations on running time and performance, as well as the mixing properties of the sampler. We set 5000 data points as test data and the remainder as training data. Figure 5.6 shows the loglikelihood on the test data using 1, 7, 31 and 127 parallel processors simulated in software, using 1000 parallel (outer) iterations and 5 Gibbs (inner) iterations. The parallel samplers are able to reach the same test likelihood levels as the serial algorithm, but with significant savings in running time. The characteristic shape of the test likelihood, similar across all testing regimes, indicates of the manner in which the features are learnt. Initially, a large number of features are added, which provides improvements in the test likelihood. Further jumps in likelihood occur once the features are refined and excess features are pruned away.

The hairiness index, based on the CUSUM method for monitoring MCMC convergence (Brooks and Roberts, 1998; Robert and Casella, 2004), was used to evaluate the mixing properties of the chain. The method applies a cumulative sum path plot of a single chain of sampler output—here, based on the sampled values of the hyperparameters—to monitor convergence. The intuition is that taking a cumulative sum of the statistics of a well-mixing chain

[3]These tests were run by Shakir Mohammed.

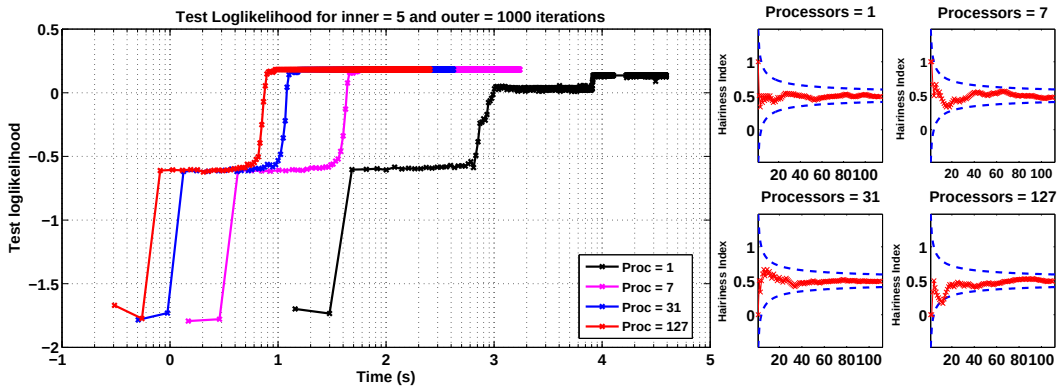


Figure 5.6: Change in likelihood for various numbers of processors over the simulation time. The corresponding hairiness index plots are shown on the left.

should stay near the mean. Large deviations from the mean suggest that the chain is staying in one regime for a long time and then moving to another regime for a long time. Specifically, the hairiness index on statistic s measures how often (on average) the slope

$$\begin{aligned} D_t &= \frac{1}{t}\sum_{\tau} d_\tau \\ d_\tau &= (\mathsf{sign}(s_\tau - s_{\tau-1}) \neq \mathsf{sign}(s_{\tau-1} - s_{\tau-2})) \end{aligned}$$

for t in some range $1 \ldots T$. A fast mixing chain is expected to have many changes of slope, or high "hairiness." By making certain independent-Bernoulli variable assumptions on the probability of the slope changing, one can compute confidence intervals for D_t. Figure 5.6 plots the hairiness index plots and the corresponding 95% confidence bounds after thinning and burnin. The hairiness index lies between the bounds for all the test scenarios, so we have no reason to believe that there are any issues in convergence in either the single-processor or the parallel inference cases.

To gain insight into the tradeoff in choosing between the number of Gibbs (inner) sweeps and parallel (outer) iterations, we show the effective number of samples (Robert and Casella, 2004) for various numbers of inner iterations, after the chain has been thinned and burnt-in, in figure 5.7(a). As expected, the effective number of samples decreases as the number of inner Gibbs sweeps

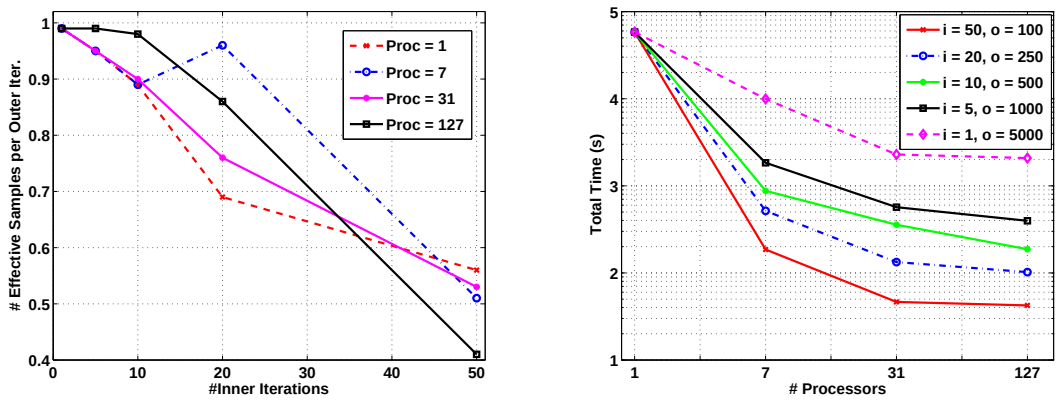


Figure 5.7: Effects of changing the number of inner iterations on: (a) The effective sample size (b) Total running time (Gibbs and Message passing).

increases; changes propagate more slowly because less information is being exchanged between the processors. However, the trade-off for a higher effective number of samples is the need for processors to communicate more often. From the plot, we see that between 10 and 20 Gibbs sweeps between message passing lead to only modest drops in the effective number of samples.

Finally, we also evaluated the running time of the samplers. Each combination of Gibbs sweeps and outer iterations was set so that the total number of sweeps was 5000. During each simulated outer iteration, we set the Gibbs time to the maximum time required by any one of the processors to compute a set of Gibbs sweeps. The message passing time was set to the total time required for all messages (an overestimate; in section 5.4 we see that the message passing times on a real asynchronous system are negligible). In figure 5.7(b), we see that the running time decreased until we reached 31 processors; at this point the overhead of message-passing began to dominate and additional processors did not add speed. Employing a larger number of Gibbs sweeps between outer iterations improves the running time, but at the cost of mixing rates as seen in figure 5.7(a).

Table 5.2: Log likelihoods on test data for real-world datasets for the serial, synchronous and asynchronous inference types.

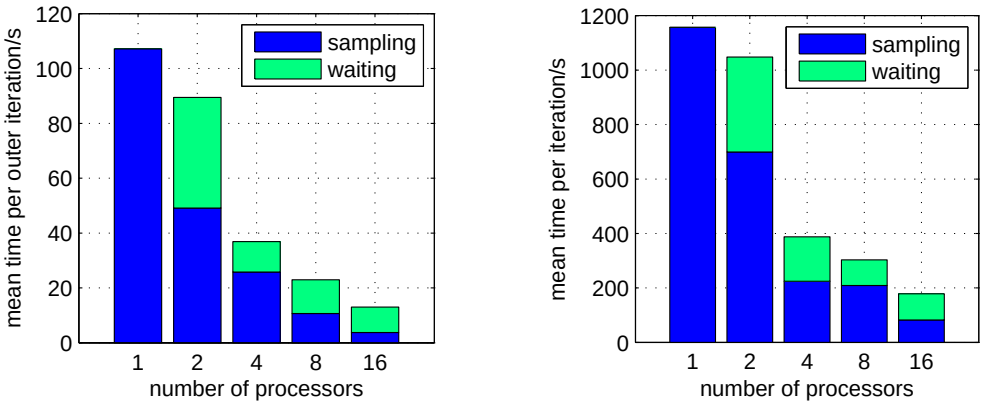
Dataset	N	D	Description	Serial $p = 1$	Sync $p = 16$	Async $p = 16$
AR Faces (Mart'inez and Kak, 2001)	2600	1598	faces with lighting, accessories (real-valued)	-4.74	-4.77	-4.84
Piano (Poliner and Ellis, 2007)	57931	161	STDFT of a piano recording (real-valued)	–	-1.182	-1.228
Flickr (Kollar, 2008)	100000	1000	indicators of image tags (binary-valued)	—	-0.0584	

5.4 Realworld Experiments[4]

We tested our parallel scheme on three real world datasets on a 16 node cluster using Matlab Distributed Computing Engine. The first dataset was a set of 2600 frontal face images with 1598 dimensions (Mart'inez and Kak, 2001). While not extremely large, the high-dimensionality of the dataset makes it challenging for other inference approaches. The piano dataset (Poliner and Ellis, 2007) consisted of 57931 samples from a 161-dimensional short-time discrete Fourier transform of a piano piece. Finally, the binary-valued Flickr dataset (Kollar, 2008) indicated whether each of 1000 popular keywords occurred in the tags of 100000 images from Flickr. Table 5.2 summarises the data and the predictive log-likelihoods obtained on held-out data; we see the performance is comparable across the various methods. Thus, the real question is what computational benefits parallelisation can provide.

Figure 5.8 shows the time processors spent processing and waiting in the synchronous implementation of the parallel inference algorithm. The Gibbs sampling time per iteration decreased almost linearly as the number of processors increased; for example, in the music dataset, using 16 processors sped up the inference by 14 times. The message passing time is negligible: 7% of the

[4]These tests were run by David Knowles.



(a) Timing analysis for faces dataset (b) Timing analysis for music dataset

Figure 5.8: Bar charts comparing sampling time and waiting times for synchronous parallel inference.

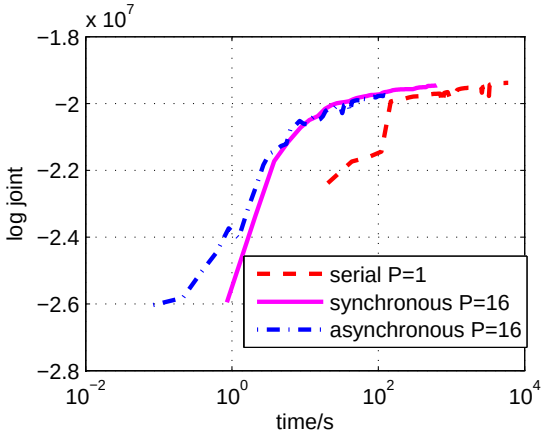


Figure 5.9: Timing comparison for different approaches

Gibbs sampling time for the faces data and 0.1% of the Gibbs sampling time for the music data. However, waiting for synchronisation becomes significant as the number of processors increases. In figure 5.9, we see that with 16 processors, the asynchronous inference is 1.64 times faster than the synchronous case, reducing the computational time from 11.8s per iteration to 7.2s.

5.5 Summary

We have described a parallel inference procedure that allows nonparametric Bayesian models based on the Indian Buffet Process to be applied to large

datasets. The IBP poses specific challenges to data parallelisation in that the dimensionality of the representation changes during inference and that we have an unbounded number of features. Our approach for data parallelisation leverages a compact representation of the feature posterior that approximately decorrelates the data stored on each processor and limits the bandwidth of communications between processors. The approximate sampling empirically behaves similarly to an exact sampler without the additional computational overhead.

The ideas presented here are applicable to a more general problems in unsupervised learning that could benefit from parallel inference. The algorithm can be adapted to scale inference for popular bilinear models ranging from PCA and NMF to ICA. Future directions for this work include heuristics for maintaining mixing as the number of processors increases. For example, in the asynchronous setting, the tree structure can be changed between sampling iterations, moving slower processors towards the leaves of the tree. Changing the structure will not affect the correctness of the inference but will allow faster processors to exchange information more often, allowing for better mixing.

Chapter 6

Variational Methods[1]

The previous chapters have described how to scale Gibbs sampling to large datasets. The advantage of sampling-based approaches to inference is that, run for a long enough time, the samples will be a faithful representation of the true posterior. However, the amount of time required for the Gibbs sampler to fully explore the posterior—a more stringent criterion than mixing—typically grows with the number of observations. In large datasets, it is likely that the Gibbs sampler will only explore one mode of the posterior in a reasonable amount of time.

The IBP is a particularly challenging for sampling-based approaches. Due to its combinatorial nature, even if we limit ourselves to K features for N objects, there exist $O(2^{NK})$ possible feature assignments. In this combinatorial space, sampling-based inference procedures for the IBP often suffer because they assign specific values to the feature-assignment variables. Hard variable assignments give samplers less flexibility to move between optima, and the samplers may need large amounts of time to escape small optima and find regions with high probability mass.

In this chapter, we derive a deterministic approach to inference. Instead of approximating the true posterior over feature assignments $p(Z|X)$ with a series of samples, variational inference approximates $p(Z|X)$ with some simpler

[1]Parts of this chapter were previously published as (Doshi-Velez et al., 2009). Please refer to the technical report accompanying (Doshi-Velez et al., 2009) for more complete derivations and implementation details.

distribution $q(Z)$ in which inference is tractable. More formally, given some family of "nice" distributions Q, we try to find $q \in Q$ such that the KL divergence $D(q||p)$ is minimised.

Inference involves using optimisation techniques to find a good approximate posterior. *Mean-field* variational inference is a special case of variational inference in which $q(Z)$ is assumed to be fully factorable, that is, $q(Z) = \prod_i Z_i$ for all variables Z_i in Z. Our mean field approximation for the IBP maintains a separate probability for each feature-observation assignment. Optimising these probability values is also fraught with local optima, but using the soft variable assignments—that is, using probabilities instead of sampling specific assignments—gives the variational method a flexibility that samplers lack. In the early stages of the inference, this flexibility can help the variational method avoid small local optima.

6.1 Mean Field Approximation

As in previous chapters, we develop our equations for the linear-Gaussian likelihood model. However, the updates derived in this chapter may be easily extended to other exponential family models. Following on variational inference for the Dirichlet Process (Blei and Jordan, 2004), we use a mean-field approximation for p based on the stick-breaking construction to the IBP (Teh et al., 2007). Recall that in the stick-breaking construction to the IBP, the feature probabilities π_k are given by

$$\pi_k = \prod_{m=1}^{k} v_m,$$

where each v_k is drawn independently from a $\mathsf{Beta}(\alpha, 1)$. Let $\boldsymbol{v} = \{v_1 \dots v_K\}$, the collection of these independent Beta variables. The set $\boldsymbol{W} = \{\boldsymbol{v}, A, Z\}$ contains all of the latent variable of interest. We use $\boldsymbol{\theta}$ to denote the set of hyperparameters $\{\alpha, \sigma_a^2, \sigma_n^2\}$.

We approximate the true posterior with a fully factorised distribution of the form

$$q(\boldsymbol{W}) = \prod_{k=1}^{K} q_{\boldsymbol{\tau_k}}(v_k) q_{\boldsymbol{\phi_k}}(\boldsymbol{A_k}) \prod_{n=1}^{N} q_{\boldsymbol{\nu_k}}(\boldsymbol{Z_{nk}}).$$

where q is truncated to contain only K features. Note, however, even though the approximating family q is truncated, we will still be finding the q that minimises the distance to the true, infinite posterior p.[2] Each of the factored distributions has the form:

- $q_{\boldsymbol{\tau}_k}(v_k) = \mathsf{Beta}(v_k; \tau_{k1}, \tau_{k2})$,
- $q_{\boldsymbol{\phi}_k}(A_{k\cdot}) = \mathsf{Normal}(A_{k\cdot}; \bar{\boldsymbol{\phi}}_k, \Phi_k)$,
- $q_{\nu_{nk}}(z_{nk}) = \mathsf{Bernoulli}(Z_{nk}; \nu_{nk})$.

Here, τ_{k1} and τ_{k2} are the variational parameters for the Beta distribution on v_k, $\bar{\boldsymbol{\phi}}_k$ and Φ_k are the variational parameters for the mean and variance of the feature values, and ν_{nk} is the variational parameter for $Pr[Z_{nk} = 1]$.

6.2 Computing the Variational Lower Bound

Minimising the KL divergence is equivalent to maximising a lower bound on the evidence $p(X|\boldsymbol{\theta})$. The evidence may be broken down as

$$\begin{aligned} \log p(X|\boldsymbol{\theta}) &= D(q((W)||p(\boldsymbol{W}|X,\boldsymbol{\theta})) - \int q(\boldsymbol{W}) \log \frac{q(\boldsymbol{W})}{p(\boldsymbol{W}, X|\boldsymbol{\theta})} \\ &= D(q((W)||p(\boldsymbol{W}|X,\boldsymbol{\theta})) + H[q] + \mathbb{E}_q[\log p(\boldsymbol{W}, X|\boldsymbol{\theta})] \end{aligned}$$

Since the evidence has a fixed value with respect to q, finding a distribution q that minimises the KL divergence $D(q((W)||p(\boldsymbol{W}|X,\boldsymbol{\theta}))$ is equivalent to finding a distribution q to maximise the quantity $L(q) \equiv H[q] + \mathbb{E}_q[\log p(\boldsymbol{W}, X|\boldsymbol{\theta})]$.

[2] In our paper (Doshi-Velez et al., 2009), we also derive a 'finite approximation' in which we minimise the KL divergence to a finite approximation of the IBP in which each feature probability π_k has an independent $\mathsf{Beta}(\alpha/K, 1)$ prior. In the limit of $K \to \infty$, the finite approximation becomes the true IBP. Using the finite approximation to the IBP adds another layer of approximation to the inference but results in computationally simpler updates. For details, please refer to the technical report. We also note that the derivations for the finite model were simultaneously derived and published as beta process factor analysis by Paisley and Carin (2009).

Moreover, since the KL divergence is always non-negative, the quantity $L(q)$ also serves as a lower bound on the evidence $p(X|\boldsymbol{\theta})$.

For our specific case, we first split the expression into parts the depend on different latent variables:

$$\begin{aligned}\ln p(X|\boldsymbol{\theta}) \quad \geq \quad & \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{v}} \left[\ln p(v_k|\alpha)\right] + \sum_{k=1}^{K}\sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{v},Z} \left[\ln p(Z_{nk}|\boldsymbol{v})\right] \\ & + \sum_{k=1}^{K} \mathbb{E}_{A} \left[\ln p(A_{k\cdot}|\sigma_A^2)\right] + \sum_{n=1}^{N} \mathbb{E}_{Z,A} \left[\ln p(X_{n\cdot}|Z, A, \sigma_n^2)\right] + H[q],\end{aligned}$$

By computing the above expectations, we can keep track of the lower bound on the likelihood. Except for the second term, all of the terms are exponential family calculations (see Appendix B for derivations); they evaluate to

$$\log p(X|\boldsymbol{\theta}) \tag{6.1}$$

$$\begin{aligned}\geq \quad & \sum_{k=1}^{K} \left[\log\alpha + (\alpha-1)\left(\psi(\tau_{k1}) - \psi(\tau_{k1}+\tau_{k2})\right)\right] \\ & + \sum_{k=1}^{K}\sum_{n=1}^{N} \left[\nu_{nk}\left(\sum_{m=1}^{k}\psi(\tau_{k2}) - \psi(\tau_{k1}+\tau_{k2})\right) + (1-\nu_{nk})\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k} v_m\right)\right]\right] \\ & + \sum_{k=1}^{K}\left[\frac{-D}{2}\log(2\pi\sigma_A^2) - \frac{1}{2\sigma_A^2}\left(tr(\boldsymbol{\Phi}_k) + \bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T\right)\right] \\ & + \sum_{n=1}^{N}[-\frac{D}{2}\log(2\pi\sigma_n^2) - \frac{1}{2\sigma_n^2}(X_{n\cdot}X_{n\cdot}^T - 2\sum_{k=1}^{K}\nu_{nk}\bar{\boldsymbol{\phi}}_k X_{n\cdot}^T + 2\sum_{k<k'}\nu_{nk}\nu_{nk'}\bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_{k'}^T \qquad (6.2) \\ & + \sum_{k=1}^{K}\nu_{nk}\left(tr(\boldsymbol{\Phi}_k) + \bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T\right))] \\ & + \sum_{k=1}^{K}[\log\left(\frac{\Gamma(\tau_{k1})\Gamma(\tau_{k2})}{\Gamma(\tau_{k1}+\tau_{k2})}\right) - (\tau_{k1}-1)\psi(\tau_{k1}) - (\tau_{k2}-1)\psi(\tau_{k2}) \qquad (6.3) \\ & + (\tau_{k1}+\tau_{k2}-2)\psi(\tau_{k1}+\tau_{k2})] \\ & + \sum_{k=1}^{K}\frac{1}{2}\log\left((2\pi e)^D|\Phi_k|\right) + \sum_{k=1}^{K}\sum_{n=1}^{N}\left[-\nu_{nk}\log\nu_{nk} - (1-\nu_{nk})\log(1-\nu_{nk})\right]\end{aligned}$$

where $\psi(\cdot)$ is the digamma function, and we have left $\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k} v_m\right)\right]$, a byproduct of the expectation of $\mathbb{E}_{\boldsymbol{v},Z}\left[\log p(Z_{nk}|\boldsymbol{v})\right]$, unevaluated. This expec-

tation has no closed-form solution, so we instead lower bound it (and therefore lower bound the evidence).

The key difficulty lies in computing the expectations for the second expectation: $\mathbb{E}_{\boldsymbol{v},Z}\left[\ln p(Z_{nk}|\boldsymbol{v})\right]$. Following derivations for variational inference in Dirichlet processes (Blei and Jordan, 2004), we break the expectation into several parts:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v},Z}\left[\ln p(Z_{nk}|\boldsymbol{v})\right] &= \mathbb{E}_{\boldsymbol{v},Z}\left[\ln p(Z_{nk}=1|\boldsymbol{v})^{\mathbb{I}(Z_{nk}=1)} p(Z_{nk}=0|\boldsymbol{v})^{\mathbb{I}(Z_{nk}=0)}\right] \\
&= \mathbb{E}_Z\left[\mathbb{I}(Z_{nk}=1)\right]\mathbb{E}_{\boldsymbol{v}}\left[\ln \prod_{m=1}^{k} v_m\right] \\
&\quad + \mathbb{E}_Z\left[\mathbb{I}(Z_{nk}=0)\right]\mathbb{E}_{\boldsymbol{v}}\left[\ln(1-\prod_{m=1}^{k} v_m)\right] \\
&= \nu_{nk}\left(\sum_{m=1}^{k}\Psi(\tau_{k2}) - \Psi(\tau_{k1}+\tau_{k2})\right) + (1-\nu_{nk})\mathbb{E}_{\boldsymbol{v}}\left[\ln(1-\prod_{m=1}^{k} v_m)\right]
\end{aligned}$$

where $\mathbb{I}(\cdot)$ is an indicator function that is 1 if its argument is true and 0 if its argument is false. The first line follows from the definition of $\boldsymbol{v}$, while the second line follows from the properties of Bernoulli and Beta distributions. However, we are still left with the problem of evaluating the expectation $\mathbb{E}_{\boldsymbol{v}}[\ln(1-\prod_{m=1}^{k} v_m)]$, or alternatively, to compute a lower bound for the expression.[3] We describe two approaches for computing this bound.

6.2.1 Taylor Series Bound

A Taylor series approximation can provide an arbitrarily close approximation to $\mathbb{E}_{\boldsymbol{v}}[\ln(1-\prod_{m=1}^{k} v_m)]$. The Taylor series for $\ln(1-x)$ is given by

$$\ln(1-x) = -\sum_{n}^{\infty}\frac{x^n}{n},$$

and it converges for $x \in (-1, 1)$. In our case, x corresponds to the product of probabilities $\prod_{m=1}^{k} v_m$, so the sum will converge unless *all* of the v_m's equal

[3] Jensen's inequality cannot be used to lower bound this expectation; the concavity of the log goes in the wrong direction.

one. Since the distribution over the v_m are continuous densities, the series will almost surely converge.

Applying the Taylor expansion to our formula, we obtain

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v}}[\ln(1-\prod_{m=1}^{k} v_m)] &= \mathbb{E}_{\boldsymbol{v}}[-\sum_{n=1}^{\infty}\frac{1}{n}\prod_{m=1}^{k} v_m^n] \\
&= -\sum_{n=1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\mathbb{E}_{\boldsymbol{v}}[v_m^n] \\
&= -\sum_{n=1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{\Gamma(\tau_{m1}+n)\Gamma(\tau_{m2}+\tau_{m1})}{\Gamma(\tau_{m1})\Gamma(\tau_{m2}+\tau_{m1}+n)} \\
&= -\sum_{n=1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\ldots(\tau_{m1}+n-1)}{(\tau_{m2}+\tau_{m1})\ldots(\tau_{m2}+\tau_{m1}+n-1)}
\end{aligned}$$

where we use the fact that the moments of a Beta distribution are given by

$$\mathbb{E}[x^n] = \frac{\Gamma(\alpha+n)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha+\beta+n)}.$$

As all of the terms in the Taylor series are negative, simply truncating the series will not produce a lower bound. However, note that for integer values of τ_{m2}, most terms in the numerator and denominator will cancel for $n > \tau_{m2}$. Let T be a value greater than $\lfloor \max_{m\in 1\ldots k}(\tau_{m2})\rfloor$; then we can write a lower

bound for the series in the following form:

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{v}}[\ln(1-\prod_{m=1}^{k} v_m)] \\
&\geq -\sum_{n=1}^{T}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\dots(\tau_{m1}+n-1)}{(\tau_{m1}+\tau_{m2})\dots(\tau_{m1}+\tau_{m2}+n-1)} \\
&\quad -\sum_{n=T+1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\dots(\tau_{m1}+n-1)}{(\tau_{m1}+\lfloor\tau_{m2}\rfloor)\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor+n-1)} \\
&= -\sum_{n=1}^{T}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\dots(\tau_{m1}+n-1)}{(\tau_{m1}+\tau_{m2})\dots(\tau_{m1}+\tau_{m2}+n-1)} \\
&\quad -\sum_{n=T+1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor-1)}{(\tau_{m1}+n)\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor+n-1)} \\
&= -\sum_{n=1}^{T}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\dots(\tau_{m1}+n-1)}{(\tau_{m1}+\tau_{m2})\dots(\tau_{m1}+\tau_{m2}+n-1)} \\
&\quad -\prod_{m=1}^{k}((\tau_{m1})\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor-1))\sum_{n=T+1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{1}{(\tau_{m1}+n)\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor+n-1)},
\end{aligned}
$$

where we have factored the final term to make it clear that the second term has n only in the denominator. Now, if T is relatively small, the first sum should not be too expensive to compute. We proceed to find a lower bound for the second term, focusing on the final summation. A fairly trivial lower bound is:

$$
\begin{aligned}
&\sum_{n=T+1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{1}{(\tau_{m1}+n)\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor+n-1)} \\
&\quad\leq \sum_{n=1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{1}{(\tau_{m1}+n)\dots(\tau_{m1}+\lfloor\tau_{m2}\rfloor+n-1)} \\
&\quad\leq \sum_{n=1}^{\infty}\frac{1}{n}\prod_{m=1}^{k}\frac{1}{n^{\lfloor\tau_{m2}\rfloor}} \\
&\quad= \zeta(1+\sum_{m=1}^{k}\lfloor\tau_{m2}\rfloor),
\end{aligned}
$$

where $\zeta(\cdot)$ is the Riemann zeta function.

However, the bound above is a gross underestimate. We can do better by

$$\sum_{n=T+1}^{\infty} \frac{1}{n} \prod_{m=1}^{k} \frac{1}{(\tau_{m1}+n)\ldots(\tau_{m1}+\lfloor\tau_{m2}\rfloor+n-1)} \leq \zeta_H(1+\sum_{m=1}^{k}\lfloor\tau_{m2}\rfloor, T+1),$$

that is, summing from $n = T+1$, not $n = 1$, where $\zeta_H(\cdot,\cdot)$ is the generalized or Hurwitz zeta function. The quality of the bound depends on the choice of T. For larger T, we trade off having to compute more terms in the first summation, but the error introduced by the fact that the denominator of the second term is $(\tau_{m1}+n)$, not n diminishes with large T.

Since all of the terms in the series are negative, we know that the Taylor series reaches the true value from above, and that the value of the zeta function is a bound on the error. Thus, we can place the true expectation in an interval

$$\mathbb{E}_{\boldsymbol{v}}[\ln(1-\prod_{m=1}^{k} v_m)] = -\sum_{n=1}^{T}\frac{1}{n}\prod_{m=1}^{k}\frac{(\tau_{m1})\ldots(\tau_{m1}+n-1)}{(\tau_{m1}+\tau_{m2})\ldots(\tau_{m1}+\tau_{m2}+n-1)} + \epsilon,$$

where

$$\epsilon \in -\prod_{m=1}^{k}((\tau_{m1})\ldots(\tau_{m1}+\lfloor\tau_{m2}\rfloor-1))\cdot[0, \zeta_H(1+\sum_{m=1}^{k}\lfloor\tau_{m2}\rfloor, T+1)].$$

6.2.2 Multinomial Lower Bound

Although the Taylor series approximation can be made arbitrarily accurate, computing the terms in the series can be computationally expensive. Here we present an alternative approach to bounding $\mathbb{E}_{\boldsymbol{v}}[\ln(1-\prod_{m=1}^{k} v_m)]$ that is computationally efficient and has straightforward parameter updates.[4] Empirically we find that the multinomial bound is usually only 2-10% looser than a 50-term Taylor series expansion—and about 30 times faster to compute.

The multinomial approximation to bound $\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k} v_m\right)\right]$ first introduces an auxiliary multinomial distribution $q_k(y)$ and applies Jensen's in-

[4]This approach was suggested by Yee Whye Teh.

equality:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k}v_m\right)\right] &= \mathbb{E}_{\boldsymbol{v}}\left[\log\left(\sum_{y=1}^{k}(1-v_y)\prod_{m=1}^{y-1}v_m\right)\right] \\
&= \mathbb{E}_{\boldsymbol{v}}\left[\log\left(\sum_{y=1}^{k}q_k(y)\frac{(1-v_y)\prod_{m=1}^{y-1}v_m}{q_k(y)}\right)\right] \\
&\geq \mathbb{E}_y\mathbb{E}_{\boldsymbol{v}}\left[\log(1-v_y)+\sum_{m=1}^{y-1}\log v_m\right]+H[q_k] \\
&= \mathbb{E}_y\left[\psi\left(\tau_{y2}\right)+\left(\sum_{m=1}^{y-1}\psi(\tau_{m1})\right)-\left(\sum_{m=1}^{y}\psi(\tau_{m1}+\tau_{m2})\right)\right]+H[q_k].
\end{aligned}$$

If we write the terms in the multinomial $q_k(y)$ as $(q_{k1}, q_{k2}, \ldots, q_{kk})$, we get

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k}v_m\right)\right] \geq\ & \left(\sum_{m=1}^{k}q_{km}\psi(\tau_{m2})\right)+\left(\sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}q_{kn}\right)\psi(\tau_{m1})\right) \\
& -\left(\sum_{m=1}^{k}\left(\sum_{n=m}^{k}q_{kn}\right)\psi(\tau_{m1}+\tau_{m2})\right)-\sum_{m=1}^{k}q_{km}\log q_{km}.
\end{aligned} \tag{6.4}$$

Equation 6.4 holds for any $q_{k1}, \ldots, q_{kk}$ for all $1 \leq k \leq K$.

Next we optimise $q_k(y)$ to maximise the lower bound. Taking derivatives with respect to each q_{ki},

$$0 = \psi(\tau_{i2})+\sum_{m=1}^{i-1}\psi(\tau_{m1})-\sum_{m=1}^{i}\psi(\tau_{m1}+\tau_{m2})-1-\log(q_{ki})-\lambda$$

where λ is the Lagrangian variable to ensure that q is a distribution. Solving for q_{ki}, we find

$$q_{ki} \propto \exp\left(\psi(\tau_{i2})+\sum_{m=1}^{i-1}\psi(\tau_{m1})-\sum_{m=1}^{i}\psi(\tau_{m1}+\tau_{m2})\right) \tag{6.5}$$

where the proportionality ensures that q_k a valid distribution. If we plug this multinomial lower bound back into $\mathbb{E}_{\boldsymbol{v},Z}\left[\log p(z_{nk}|\boldsymbol{v})\right]$, we have a lower bound on $\log p(X|\boldsymbol{\theta})$. We then optimise the remaining parameters to maximise the lower bound.

The auxiliary distribution q_k is largely a computational tool, but it does have the following intuition. Since $\pi_k = \prod_{i=1}^{k} v_i$; we can imagine the event $z_{nk} = 1$ is equivalent to the event that a series of variables $u_i \sim \mathsf{Bernoulli}(v_i)$ all flip to one. If any of the u_i's equal zero, then the feature is off. The multinomial distribution $q_k(y)$ can be thought of as a distribution over the event that the y^{th} variable u_y is the first u_i to equal 0.

6.3 Parameter Updates

Recall that our goal is to find an approximating distribution $q \in Q$ with minimum KL divergence $D(q||p)$ to the true distribution p. We rephrased this optimisation problem in terms of certain expectations and entropies:

$$\arg\min_{\tau,\phi,\nu} D(q||p) = \arg\max_{\tau,\phi,\nu} \mathbb{E}_q[\log(p(X, \boldsymbol{W}|\boldsymbol{\theta})] + H[q]. \tag{6.6}$$

In general, this optimisation can be quite difficult. However, when the conditional distribution and variational distribution are both in the exponential family, each step in the coordinate ascent has a closed form solution (Beal, 2003; Wainwright and Jordan, 2008). If we are updating the variational parameters ξ_i that correspond to W_i, then the optimal ξ_i are the solution to

$$\log q_{\xi_i}(W_i) = \mathbb{E}_{\boldsymbol{W}_{-i}}[\log p(\boldsymbol{W}, X|\boldsymbol{\theta})] + c \tag{6.7}$$

where the expectation is taken over all $\boldsymbol{W}$ except W_i according to the variational distribution.

Using the fixed-point equation 6.7, the updates for most of the variational parameters are straight-forward. These are described in appendix B. The only updates that are difficult are the updates for the τ parameters that describe the variational distribution on each of the stick-breaking variables v. The updates for these parameters depend on whether we use the Taylor or multinomial approximation in the lower bound.[5]

[5]We also tried fitting a Beta distribution to v_k based on its moments, but this approximation produced poor results.

6.3.1 Taylor Update

The Taylor approximation does not lend itself to closed-form updates; instead we optimise the τ_{k1} and τ_{k2} parameters directly with respect to the variational lower bound. To do so, we first note only some terms in equation 6.1 depend on the τ parameters:

$$\begin{aligned} f &= \sum_{k=1}^{K} [\ln \alpha + (\alpha - 1)\left(\Psi(\tau_{k1}) - \Psi(\tau_{k1} + \tau_{k2})\right)] \\ &\quad + \sum_{k=1}^{K}\sum_{n=1}^{N} [\nu_{nk} \left(\sum_{m=1}^{k} \Psi(\tau_{m1}) - \Psi(\tau_{m1} + \tau_{m2}) \right) \\ &\quad -(1 - \nu_{nk}) \sum_{r=1}^{\infty} \frac{1}{r} \prod_{m=1}^{k} \frac{(\tau_{m1}) \ldots (\tau_{m1} + r - 1)}{(\tau_{m1} + \tau_{m2}) \ldots (\tau_{m1} + \tau_{m2} + r - 1)}] \\ &\quad + \sum_{k=1}^{K} [\ln \left(\frac{\Gamma(\tau_{k1})\Gamma(\tau_{k2})}{\Gamma(\tau_{k1} + \tau_{k2})} \right) - (\tau_{k1} - 1)\Psi(\tau_{k1}) \\ &\quad -(\tau_{k2} - 1)\Psi(\tau_{k2}) + (\tau_{k1} + \tau_{k2} - 2)\Psi(\tau_{k1} + \tau_{k2})]. \end{aligned}$$

The derivatives with respect to τ_{k1} are given by

$$\begin{aligned} \frac{\partial f}{\partial \tau_{k1}} &= (\alpha - 1)\left(\Psi'(\tau_{k1}) - \Psi'(\tau_{k1} + \tau_{k2})\right) + \left(\sum_{m=k}^{K}\sum_{n=1}^{N} \nu_{nm} \right) \left(\Psi'(\tau_{k1}) - \Psi'(\tau_{k1} + \tau_{k2})\right) \\ &\quad - \sum_{m=k}^{K} \left(N - \sum_{n=1}^{N} \nu_{nm} \right) \sum_{r=1}^{\infty} \frac{1}{r} \left(\prod_{i=1}^{m} \frac{(\tau_{i1}) \ldots (\tau_{i1} + r - 1)}{(\tau_{i1} + \tau_{i2}) \ldots (\tau_{i1} + \tau_{i2} + r - 1)} \right) \\ &\quad \cdot \sum_{j=1}^{r} \frac{\tau_{k2}}{(\tau_{k1} + j - 1)(\tau_{k1} + \tau_{k2} + j - 1)} \\ &\quad -(\tau_{k1} - 1)\Psi'(\tau_{k1}) + (\tau_{k1} + \tau_{k2} - 2)\Psi'(\tau_{k1} + \tau_{k2}), \end{aligned}$$

where the last term follows if we simplify the initial derivative of the entropy term

$$\frac{\Gamma'(\tau_{k1})}{\Gamma(\tau_{k1})} - \frac{\Gamma'(\tau_{k1} + \tau_{k2})}{\Gamma(\tau_{k1} + \tau_{k2})} - \Psi(\tau_{k1}) - (\tau_{k1} - 1)\Psi'(\tau_{k1}) + \Psi(\tau_{k1} + \tau_{k2}) + (\tau_{k1} + \tau_{k2} - 2)\Psi'(\tau_{k1} + \tau_{k2}),$$

by noting that $\Gamma'(x) = \Gamma(x)\Psi(x)$. Similarly, the derivatives with respect to τ_{k2} are given by

$$\begin{aligned}\frac{\partial f}{\partial \tau_{k2}} &= (\alpha - 1)\left(-\Psi'(\tau_{k1} + \tau_{k2})\right) - \left(\sum_{m=k}^{K}\sum_{n=1}^{N} \nu_{nm}\right)\Psi'(\tau_{k1} + \tau_{k2}) \\ &\quad - \sum_{m=k}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)\sum_{r=1}^{\infty}\frac{1}{r}\left(\prod_{i=1}^{m}\frac{(\tau_{i1})\ldots(\tau_{i1}+r-1)}{(\tau_{i1}+\tau_{i2})\ldots(\tau_{i1}+\tau_{i2}+r-1)}\right) \\ &\quad \cdot\sum_{j=1}^{r}\frac{-1}{\tau_{k1}+\tau_{k2}+j-1} \\ &\quad -(\tau_{k2}-1)\Psi'(\tau_{k2}) + (\tau_{k1}+\tau_{k2}-2)\Psi'(\tau_{k1}+\tau_{k2}).\end{aligned}$$

While messy, they can be computed for particular parameter settings (where we choose some arbitrary truncation level of the infinite sum). Several computations can be reused across k, and others can be computed iteratively across r. We can plug these derivatives into an optimization routine to get updates τ_{k1} and τ_{k2}.

6.3.2 Multinomial Update

The multinomial lower bound allows for far more efficient updates. The terms in the likelihood that contain τ_k are

$$\begin{aligned}l_{\tau_k} &= \left[\alpha + \sum_{m=k}^{K}\sum_{n=1}^{N}\nu_{nm} + \sum_{m=k+1}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)\left(\sum_{i=k+1}^{m} q_{mi}\right) - \tau_{k1}\right]\left(\Psi(\tau_{k1}) - \Psi(\tau_{k1}+\tau_{k2})\right) \\ &\quad + \left[1 + \sum_{m=k}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)q_{mk} - \tau_{k2}\right]\left(\Psi(\tau_{k2}) - \Psi(\tau_{k1}+\tau_{k2})\right) + \ln\left(\frac{\Gamma(\tau_{k1})\Gamma(\tau_{k2})}{\Gamma(\tau_{k1}+\tau_{k2})}\right).\end{aligned}$$

Optimising l_{τ_k} with respect to τ_k (holding q_k fixed) is a standard exponential family variational update. The optimal values of τ_{k1} and τ_{k2} are

$$\begin{aligned}\tau_{k1} &= \alpha + \sum_{m=k}^{K}\sum_{n=1}^{N}\nu_{nm} + \sum_{m=k+1}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)\left(\sum_{i=k+1}^{m} q_{mi}\right) \\ \tau_{k2} &= 1 + \sum_{m=k}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)q_{mk}.\end{aligned}$$

6.4 Truncation Bounds

The previous sections described how to perform variational inference in the IBP. One of the key elements in our approach was truncating the variational distribution to K features. In this section, we explore how the choice of a truncation level K affects the quality of our approximation. Specifically, given a number of observations N and a concentration parameter α, we consider the difference between the IBP prior and the prior truncated at level K. Intuitively, the difference amounts to how often we expect to see features beyond K in a dataset of size N—if it is unlikely that the dataset contains more than K features, then truncating the prior should have little effect.

Our development parallels a bound for the Dirichlet Process by Ishwaran and James (2001) and presents the first such truncation bound for the IBP. Let us denote the marginal distribution of observation X by $m_\infty(X)$ when we integrate $\boldsymbol{W}$ with respect to the true IBP stick-breaking prior $p(\boldsymbol{W}|\boldsymbol{\theta})$. Let $m_K(X)$ be the marginal distribution when $\boldsymbol{W}$ are integrated out with respect to the truncated stick-breaking prior with truncation level K, $p_K(\boldsymbol{W}|\boldsymbol{\theta})$. For consistency, we continue to use the notation from the linear-Gaussian model, but the derivation that follows is independent of the likelihood model.

We can show that difference between the IBP prior and the prior truncated at level K is at most

$$\begin{aligned} \frac{1}{4}\int |m_K(X) - m_\infty(X)| dX &\leq \Pr(\text{any } z_{ik} = 1,\ k > K) \\ &= 1 - \Pr\left(\text{all } z_{1k} = 0,\ i \in \{1, \ldots, N\}, k > K\right) \\ &= 1 - \mathbb{E}\left(\left[\prod_{i=K+1}^{\infty} (1 - \pi_i)\right]^n\right) \end{aligned} \tag{6.8}$$

Our goal is therefore to either find an upper bound of

$$1 - \mathbb{E}\left(\left[\prod_{i=K+1}^{\infty} (1 - \pi_i)\right]^n\right)$$

or to approximate it so that we can determine the rate of convergence. We describe two approaches to this bound below.

6.4.1 Log Bound

We need to evaluate the following using the expectation of the r^{th} moment of independent Beta$(\alpha, 1)$ random variables:

$$\begin{aligned}\sum_{i=K}^{\infty} \mathbb{E} \log\left(1 - \prod_{j=1}^{i} v_j\right) &= -\sum_{i=K}^{\infty}\sum_{r=1}^{\infty} \frac{1}{r} \prod_{j=1}^{i} \frac{\alpha}{\alpha + r} \\ &= -\sum_{r=1}^{\infty} \frac{1}{r} \sum_{i=K}^{\infty} \left(\frac{\alpha}{\alpha + r}\right)^i \\ &= -\sum_{r=1}^{\infty} \frac{1}{r^2} \frac{\alpha^K}{(\alpha + r)^{K-1}}.\end{aligned}$$

If we abandon having a strict bound and say that $\frac{\alpha^K}{r^2(\alpha+r)^{K-1}} \approx \frac{\alpha^K}{(\alpha+r)^{K+1}}$, then we can substitute in the Zeta function and get the following expression:

$$\begin{aligned}1 - \mathbb{E}\left(\left[\prod_{i=K}^{\infty}(1-\pi_i)\right]^n\right) &\leq 1 - \exp\left(-n\sum_{r=1}^{\infty} \frac{1}{r^2} \frac{\alpha^K}{(\alpha + r)^{K-1}}\right) \\ &\approx 1 - \exp\left(-n\alpha^K \zeta(K+1, \alpha)\right).\end{aligned}$$

In practice, we find this heuristic bound to be very close to the true value.

More formally[6], we can write

$$\begin{aligned}\sum_{r=1}^{\infty} \frac{1}{r^2} \frac{\alpha^K}{(\alpha + r)^{K-1}} &\leq \frac{\alpha^K}{(\alpha+1)^{K-1}} + \int_1^{\infty} \frac{1}{r^2} \frac{\alpha^K}{(\alpha + r)^{K-1}} \\ &= \frac{\alpha^K}{(\alpha+1)^{K-1}} + \int_0^1 \frac{\alpha^K}{(\alpha + \frac{1}{t})^{K-1}} \\ &= \frac{\alpha^K}{(\alpha+1)^{K-1}} + \frac{\alpha^K}{K} F(K-1, K; K+1; -a)\end{aligned}$$

where $F(a, b; c; x)$ is Gauss's hypergeometric function. The first line applies the integral inequality, where we have included the first term to ensure that we have an upper bound. Next, we substitute $t = \frac{1}{r}$ into the integral and evaluate. Next, we apply the reflection law of hypergeometric functions. The

[6]We thank Professor John Lewis from MIT for his insights for manipulating hypergeometric functions regarding this approach.

reflection law states

$$\frac{1}{(1-z)^a}F(a,b;c;\frac{-z}{1-z}) = F(a,c-b;c;z)$$

and allows us to simplify the expression to the following hypergeometric function, which we expand out into its sum.

$$\begin{aligned}
\sum_{r=1}^{\infty}\frac{1}{r^2}\frac{\alpha^K}{(\alpha+r)^{K-1}} &\leq \frac{\alpha^K}{(\alpha+1)^{K-1}} + \frac{\alpha^K}{K(\alpha+1)^K}F(2,K;K+1;\frac{a}{a+1}) \\
&= \frac{\alpha^K}{(\alpha+1)^{K-1}} + \frac{\alpha^K}{(\alpha+1)^K}\sum_{j=0}^{\infty}\left(\frac{\alpha}{\alpha+1}\right)^j\frac{1+j}{K+j} \\
&\leq \frac{\alpha^K}{(\alpha+1)^{K-1}} + \frac{\alpha^K}{(\alpha+1)^K}\sum_{j=0}^{\infty}\left(\frac{\alpha}{\alpha+1}\right)^j \\
&= 2(\alpha+1)\left(\frac{\alpha}{\alpha+1}\right)^K
\end{aligned}$$

which we can plug into our original expression to get

$$1-\exp\left(-N\sum_{r=1}^{\infty}\frac{1}{r^2}\frac{\alpha^K}{(\alpha+r)^{K-1}}\right) \leq 1-\exp\left(-2N(\alpha+1)\left(\frac{\alpha}{\alpha+1}\right)^K\right)$$

6.4.2 Levy-Kintchine Approach

A very similar bound can be derived using the Levy-Kintchine formula.[7] We begin the derivation of the formal truncation bound by noting that beta-Bernoulli process construction for the IBP (Thibaux and Jordan, 2007) implies that the sequence of $\pi_1, \pi_2, \ldots$ may be modelled as a Poisson process on the unit interval $[0,1]$ with rate $\mu(x) = \alpha x^{-1}dx$. It follows that the sequence of $\pi_{K+1}, \pi_{K+2}, \ldots$ may be modelled as a Poisson process on the interval $[0, \pi_K]$ with the same rate. The Levy-Khintchine formula (Applebaum, 2004) states that the moment generating function of a Poisson process X with rate μ can be written as

$$\mathbb{E}[\exp(tf(X))] = \exp\left(\int\left(\exp(tf(y))-1\right)\mu(y)dy\right).$$

[7] Thanks to Yee Whye Teh for deriving this bound.

where we use $f(X)$ to denote $\sum_{x \in X} f(x)$.

Returning to Equation 6.8, if we rewrite the final expectation as

$$\mathbb{E}\left[\left(\prod_{i=K+1}^{\infty}(1-\pi_i)\right)\right] = \mathbb{E}\left[\exp\left(\sum_{i=K+1}^{\infty}\log(1-\pi_i)\right)\right],$$

then we can apply the Levy-Khintchine formula to get

$$\begin{aligned}\mathbb{E}\left[\exp\left(\sum_{i=K+1}^{\infty}\log(1-\pi_i)\right)\right] &= \mathbb{E}_{\pi_K}\left[\exp\left(\int_0^{\pi_K}\left(\exp(\log(1-x))-1\right)\mu(x)dx\right)\right] \\ &= \mathbb{E}_{\pi_K}[\exp(-\alpha\pi_K)].\end{aligned}$$

Finally, we apply Jensen's inequality, using the fact that π_K is the product of independent $\mathsf{Beta}(\alpha, 1)$ variables:

$$\begin{aligned}\mathbb{E}_{\pi_K}[\exp(-\alpha\pi_K)] &\geq \exp\left(\mathbb{E}_{\pi_k}[-\alpha\pi_K]\right) \\ &= \exp\left(-\alpha\left(\frac{\alpha}{1+\alpha}\right)^K\right).\end{aligned}$$

Substituting this expression back into Equation (6.8) gives us the bound

$$\frac{1}{4}\int |m_K(X) - m_\infty(X)|dX \leq 1 - \exp\left(-N\alpha\left(\frac{\alpha}{1+\alpha}\right)^K\right). \tag{6.9}$$

Similar to truncation bound for the Dirichlet Process, the expected error increases as N and α, the factors that increase the expected number of features, increase. However, the bound decreases exponentially quickly as truncation level K is increased.

Figure 6.1 shows our truncation bound and the true L_1 distance based on 1000 Monte Carlo simulations of an IBP matrix with $N = 30$ observations and $\alpha = 5$. As expected, the bound decreases exponentially fast with the truncation level K. The bound is loose, however; in practice, we find that the heuristic bound is nearly equal to the true bound.

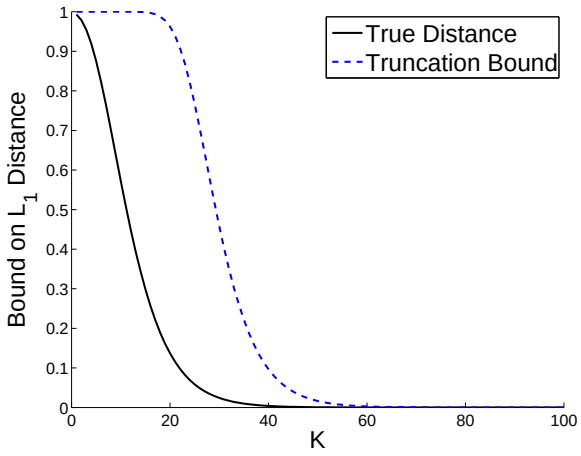


Figure 6.1: Truncation bound and true L_1 distance.

6.5 Experiments and Discussion

We compared our variational approaches with both Gibbs sampling (Griffiths and Ghahramani, 2005) and particle filtering (Wood and Griffiths, 2007). As variational algorithms are only guaranteed to converge to a *local* optima, we applied standard optimisation tricks to avoid small minima. Each run was given a number of random restarts and the hyperparameters for the noise and feature variance were tempered to smooth the posterior. We also experimented with several other techniques such as gradually introducing data and merging correlated features. The latter techniques proved less useful as the size and dimensionality of the datasets increased; they were not included in the final experiments.

The sampling methods we compared against were the collapsed Gibbs sampler of Griffiths and Ghahramani (2005) and a partially-uncollapsed alternative in which instantiated features are explicitly represented and new features are integrated out. In contrast to the variational methods, the number of features present in the IBP matrix will adaptively grow or shrink in the samplers. To provide a fair comparison with the variational approaches, we also tested finite variants of the collapsed and uncollapsed Gibbs samplers. We also tested against the particle filter of Wood and Griffiths (2007). All sampling methods were tempered and given an equal number of restarts as the variational methods.

Both the variational and Gibbs sampling algorithms were heavily optimised for efficient matrix computation so we could evaluate the algorithms both on their running times and the quality of the inference. For the particle filter, we used the implementation provided by Wood and Griffiths (2007). To measure the quality of these methods, we held out one third of the observations on the last half of the dataset. Once the inference was complete, we computed the predictive likelihood of the held out data (averaged over restarts).

6.5.1 Synthetic Data

The synthetic datasets consisted of Z and A matrices randomly generated from the truncated stick-breaking prior. Figure 6.2 shows the evolution of the test-likelihood over a thirty minute interval for a dataset with 500 observations of 500 dimensions and with 20 latent features. The error bars indicate the variation over the 5 random starts. [8] The finite uncollapsed Gibbs sampler (dotted green) rises quickly but consistently gets caught in a lower optima and has higher variance. Examining the individual runs, we found the higher variance was not due to the Gibbs sampler mixing but due to each run getting stuck in widely varying local optima. The variational methods were slightly slower per iteration but soon found regions of higher predictive likelihoods. The remaining samplers were much slower per iteration, often failing to mix within the allotted interval.

Figure 6.3 shows a similar plot for a smaller dataset with $N = 100$. Here, the variational approaches do less well at finding regions of large probability mass than the Gibbs samplers. We believe this is because in a smaller dataset, the Gibbs samplers mix quickly and explore the posterior for regions of high probability mass. However, the variational approach is still limited by performing gradient ascent to one optima.

Figures 6.4 and 6.5 show results from a systematic series of tests in which we tested all combinations of observation counts $N = \{5, 10, 50, 100, 500, 1000\}$,

[8] The particle filter must be run to completion before making prediction, so we cannot test its predictive performance over time. We instead plot the test likelihood only at the end of the inference for particle filters with 10 and 50 particles (the two magenta points).

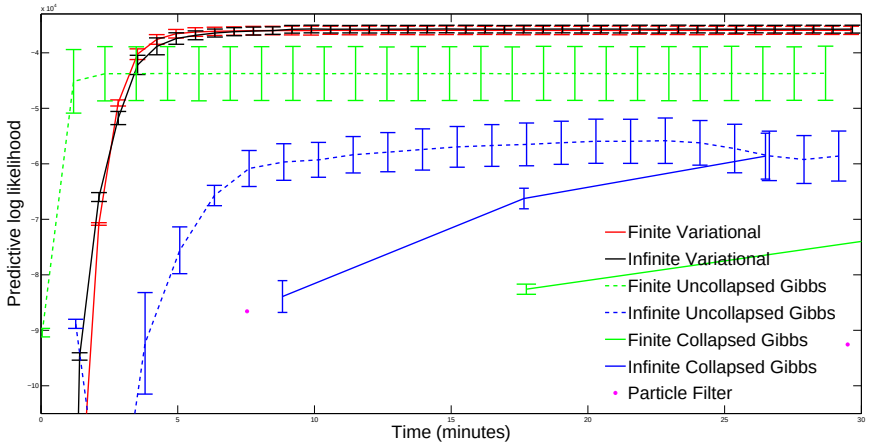


Figure 6.2: Evolution of test log-likelihoods over a thirty-minute interval for $N = 500$, $D = 500$, and $K = 20$. The finite uncollapsed Gibbs sampler has the fastest rise but gets caught in a lower optima than the variational approach.

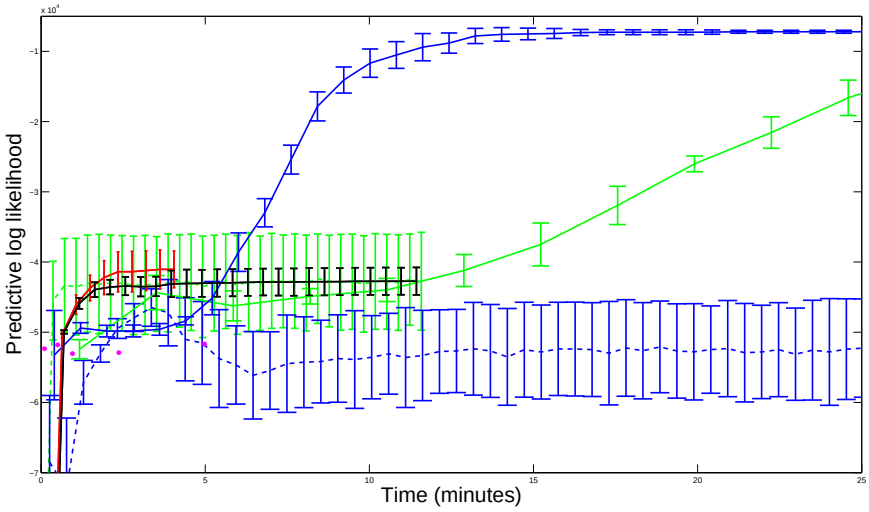


Figure 6.3: Evolution of test log-likelihoods over a thirty-minute interval for $N = 100$, $D = 500$, and $K = 25$. For smaller N, the Gibbs sampler does better at finding an optima of high probability mass.

dimensionalities $D = \{5, 10, 50, 100, 500, 1000\}$, and truncation levels $K = \{5, 10, 15, 20, 25\}$. Each of the samplers was run for 1000 iterations on three chains and the particle filter was run with 500 particles. For the variational methods, we used a stopping criterion that halted the optimisation when the variational lower bound between the current and previous iterations changed by a multiplicative factor of less than 10^{-4} and the tempering process had completed.

Figure 6.4 shows how the computation time scales with the truncation level. The variational approaches and the uncollapsed Gibbs are consistently an order of magnitude faster than other algorithms. Figure 6.5 shows the interplay between dimensionality, computation time, and test log-likelihood for datasets of size $N = 5$ and $N = 1000$ respectively. For $N = 1000$, the collapsed Gibbs samplers and particle filter did not finish, so they do not appear on the plot. We chose $K = 20$ as a representative truncation level. Each line represents increasing dimensionality for a particular method (the large dot indicates $D = 5$, the subsequent dots correspond to $D = 10$, 50, etc.). The nearly vertical lines of the variational methods show that they are quite robust to increasing dimension. Moreover, as dimensionality and dataset size increase, the variational methods become increasingly faster than the samplers. By comparing the lines across the likelihood dimension, we see that for the very small dataset, the variational method often has a lower test log-likelihood than the samplers. In this regime, the samplers are fast to mix and explore the posterior. However, the test log-likelihoods are comparable for the $N = 1000$ dataset.

6.5.2 Real Data

We applied our variational method to two real-world datasets to test how it would fare with complex, noisy data not drawn from the IBP prior[9]. The Yale Faces (Georghiades et al., 2001) dataset consisted of 721 32x32 pixel frontal-face images of 14 people with varying expressions and lighting conditions. We set σ_a and σ_n based on the variance of the data. The speech dataset consisted of

[9]Note that our objective was not to demonstrate low-rank approximations.

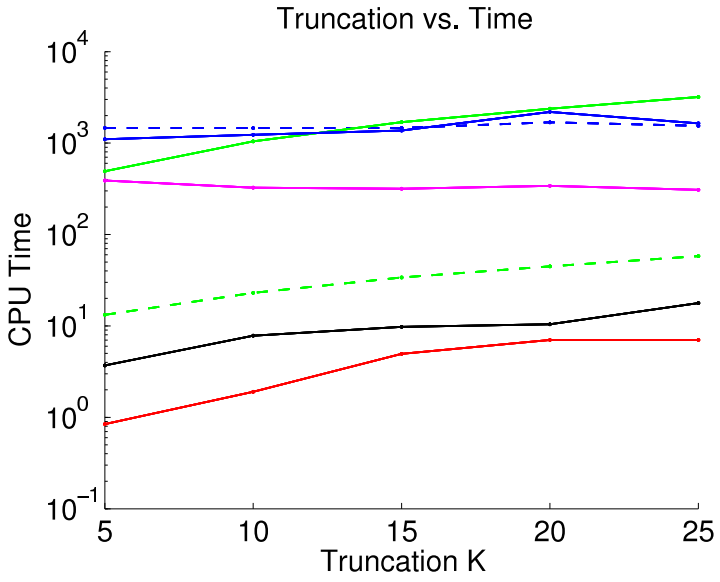


Figure 6.4: Time versus truncation (K). The variational approaches are generally orders of magnitude faster than the samplers (note log scale on the time axis).

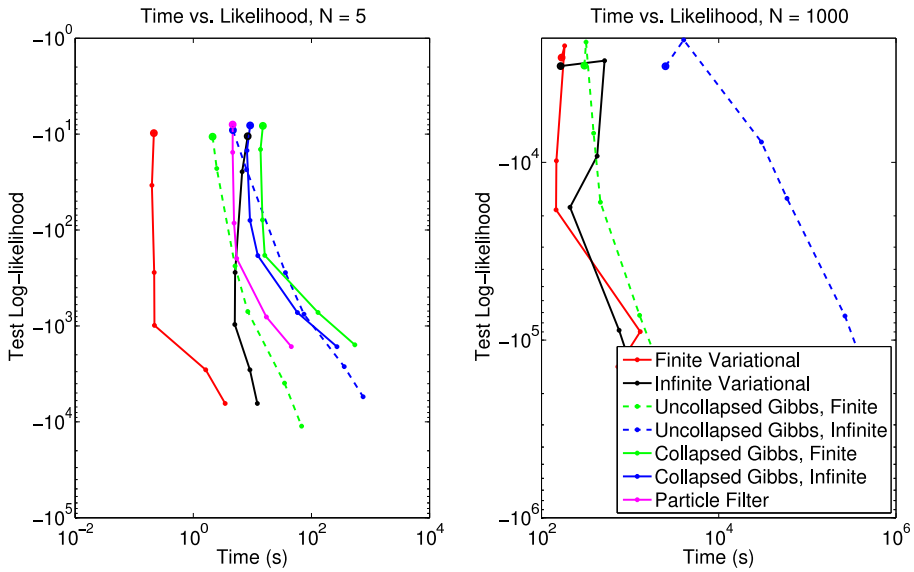


Figure 6.5: Time versus log-likelihood plot for $K = 20$. The larger dots correspond to $D = 5$ the smaller dots to $D = 10, 50, 100, 500, 1000$.

245 observations sampled from a 10-microphone audio recording of 5 different speakers. We applied the ICA version of our inference algorithm, where the mixing matrix S modulated the effect of each speaker on the audio signals. The feature and noise variances were taken from an initial run of the Gibbs sampler where σ_n and σ_a were also sampled.

Tables 6.1 and 6.2 show the results for each of the datasets. All Gibbs samplers were uncollapsed and run for 200 iterations.[10] In the higher dimensional Yale dataset, the variational methods outperformed the uncollapsed Gibbs sampler. When started from a random position, the uncollapsed Gibbs sampler quickly became stuck in a local optima. The variational method was able to find better local optima because it was initially very uncertain about which features were present in which data points; expressing this uncertainty explicitly through the variational parameters (instead of through a sequence of samples) allowed it the flexibility to improve upon its bad initial starting point.

Table 6.1: Running times in seconds and test log-likelihoods for the Yale Faces dataset.

Algorithm	**K**	**Time**	**Test Log-Likelihood** ($\times 10^6$)
Finite Gibbs	5	464.19	-2.250
	10	940.47	-2.246
	25	2973.7	-2.247
Finite Variational	5	163.24	-1.066
	10	767.1	-0.908
	25	10072	-0.746
Infinite Variational	5	176.62	-1.051
	10	632.53	-0.914
	25	19061	-0.750

The story for the speech dataset, however, is quite different. Here, the variational methods were not only slower than the samplers, but they also achieved lower test-likelihoods. The evaluation on the synthetic datasets points

[10] On the Yale dataset, we did not test the collapsed samplers because the finite collapsed Gibbs sampler required one hour per iteration with $K = 5$ and the infinite collapsed Gibbs sampler generated one sample every 50 hours. In the iICA model, the collapsed Gibbs sampler could not be run because the features A cannot be marginalised.

to a potential reason for the difference: the speech dataset is much simpler than the Yale dataset, consisting of 10 dimensions (vs. 1032 in the Yale dataset). In this regime, the Gibbs samplers perform well and the approximations made by the variational method become apparent. As the dimensionality grows, the samplers have more trouble mixing, but the variational methods are still able to find regions of high probability mass.

Table 6.2: Running times in seconds and test log-likelihoods for the speech dataset.

Algorithm	**K**	**Time**	**Test Log-Likelihood**
Finite Gibbs	2	56	-0.7444
	5	120	-0.4220
	9	201	-0.4205
Infinite Gibbs	na	186	-0.4257
Finite Variational	2	2477	-0.8455
	5	8129	-0.5082
	9	8539	-0.4551
Infinite Variational	2	2702	-0.8810
	5	6065	-0.5000
	9	8491	-0.5486

6.6 Summary

We observed empirically that the variances reported by our variational method were quite small, suggesting that the method had found one, highly-peaked mode. While one might argue that a sampler would provide a better representation of the posterior, we show that in these situations samplers rarely explore multiple modes, and therefore are effectively performing stochastic gradient ascent to a MAP estimate.[11] The variational method achieves these results faster by using deterministic optimisation techniques.

The soft assignments in the variational method are another key advantage that sets it apart from sampling-based methods. The combinatorial nature

[11]Moreover, in many cases, the machine learning community only reports predictive likelihoods or reconstruction errors with respect to a known model for performance. If these are truly the measures of interest, then sampling is clearly the wrong approach, as it attempts to explore the full posterior, not find the most likely mode.

of the Indian Buffet Process poses specific challenges for sampling-based inference procedures. Whereas sampling methods work in the discrete space of binary matrices, the variational method allows for soft assignments of features because it approaches the inference problem as a continuous optimisation. Especially for high dimensional problems, the soft assignments seem to allow the variational methods to explore the posterior space faster than sampling-based approaches.

Chapter 7

Correlated Non-Parametric Latent Feature Models[1]

In previous chapters, we discussed several methods for scaling inference in the IBP (and similar conjugate models). Perhaps the simplest nonparametric latent feature model, the IBP is an attractive starting point for nonparametric modelling of multiple-membership data. In many real situations, however, observations may not be exchangeable or features may not be independent. For example, suppose observations correspond to image pixels, and latent features correspond to objects in the scene, such as a pen or lamp. Sets of objects—that is, certain latent features—may tend to occur together: an image with a desk is likely to contain a pen; an image with a knife is likely to contain a fork. The IBP is unable to discover or leverage such correlations. In this chapter, we lay out a general framework for nonparametric correlated feature models.[2]

We first motivate why learning correlations between features may be useful—after all, one could always group co-occurring features into a single feature and thus avoid the need to model correlations between features. However, ignoring the underlying structure could result in less robust inference if the set of features does not always occur as a set: for example, if a pen is missing from a particular desk. Moreover, such an approach would have to learn separate

[1] Parts of this chapter was previously published in (Doshi-Velez and Ghahramni, 2009).

[2] For a review of work on other types of extensions to the IBP (such as introducing correlations between observations), refer to appendix C

models for a pen on a desk and a pen in a store and thus lose to opportunity to leverage all pen instances to refine its object model.

One approach to modelling correlations, the correlated topic model of Blei and Lafferty (2006), directly learns parameters for a joint distribution. Our approach, however, draws on the hierarchical structures of sigmoid (Neal, 1992) and deep belief (Hinton et al., 2006a; Bengio, 2007a) networks, where correlations reflect a higher layer of structure. For example, in the image scenario, being at a table may explain correlations between knives and forks. In the nonparametric setting, topic models such as Pachinko allocation (Li et al., 2007) also use hierarchies to model correlations. Closest to our interest in nonparametric feature models is the infinite hierarchical factorial regression (Rai and Daume, 2009). IHFR is a partially-conditional model that uses an IBP to determine what features are present and then applies Kingman's coalescent to model correlations given the active features. Drawing on the use of hierarchies in topic models and deep belief nets, we develop a more general framework for unconditional nonparametric correlated feature models and demonstrate applications to several real-world datasets.

7.1 General Framework

Our nonparametric correlated featured model places a prior over a structure describing the correlations and cooccurrences among an infinite number of features and observations. Inference on this infinite structure is tractable if the prior ensures that a finite set of observations affects only a finite part of the structure. More generally, the following properties would be desirable in a nonparametric correlated feature model:

- A finite dataset should be generated by a finite number of latent features with probability one.
- Features and data should remain exchangeable.
- Correlations should capture motifs, or commonly occuring sets of features.

The first desideratum requires particular attention if the hidden features are correlated. The model must ensure that the correlations do not cause an infinite number of features to be expressed in any observation.

Let the feature assignment matrix Z be a binary matrix where $z_{nk} = 1$ if feature k is present in observation n. In our model, Z depends on a set of category assignments C and a set of category-feature relations M (see graphical model in figure 7.1). The binary category-assignment matrix C contains $c_{nl} = 1$ if observation n is a member of category l. Similarly, $m_{lk} = 1$ if feature k is associated with category l. Features become correlated because observations choose features only through categories, and categories are associated with sets of features (see figure 7.2 for a more explicit illustration). Finally, the data X are produced by the feature assignments Z and some parameters A.

Formal Description. For each observation, the generative model first draws one or more categories via a non-parametric process with hyper-parameter α_C.

$$C \sim \mathsf{NP1}(\alpha_C) \tag{7.1}$$

where c_{nl} indicates whether category l is active in observation n. A second nonparametric process with parameter α_M associates categories with features:

$$M \sim \mathsf{NP2}(\alpha_M) \tag{7.2}$$

where m_{lk} indicates whether category l chose feature k. The processes NP1 and NP2 should ensure that, with probability one, each observation is associated

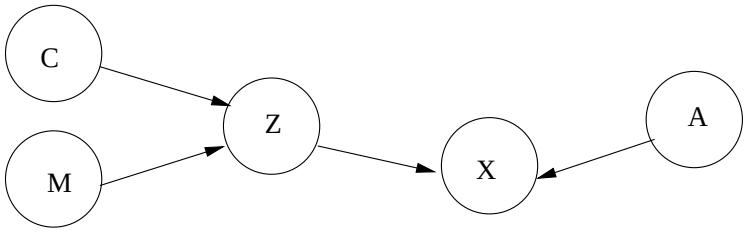


Figure 7.1: Graphical model. Feature assignments Z depend on category assignments C and category-feature relations M. Data X depend on Z and parameters A (all hyperparameters omitted for clarity).

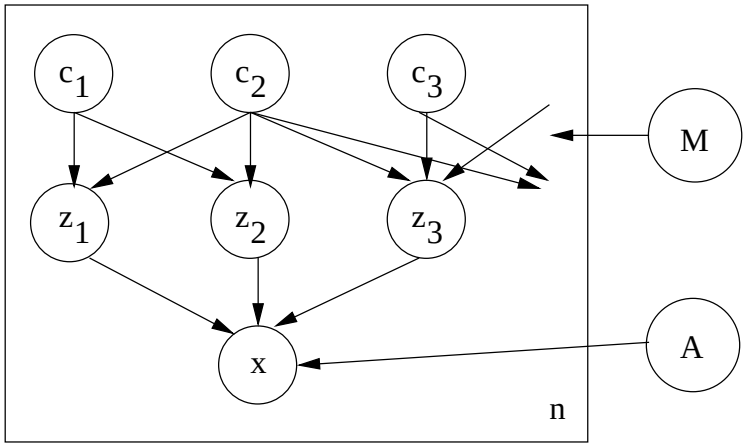


Figure 7.2: Plate for one observation. Observation n is generated from a set of categories c_{nl} which in turn select features z_{nj}. The connection matrix M describes the links between features and categories.

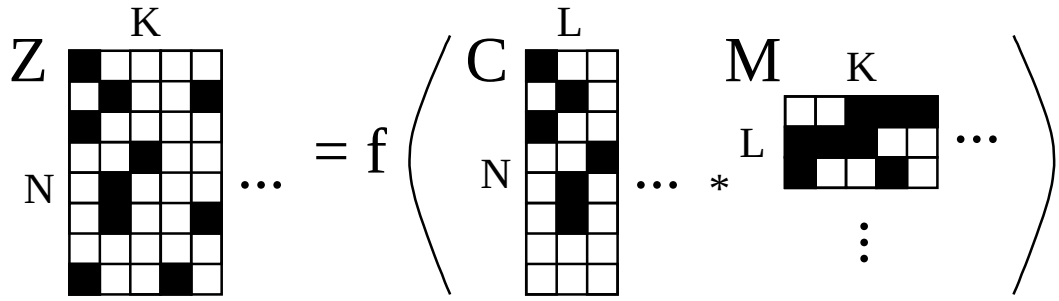


Figure 7.3: Cartoon of the matrix product. The function f describes how feature assignments are derived from the category matrix C and the connection matrix M.

with a finite number of categories and each category is associated with a finite number of features. Finally, the feature-assignment matrix $Z = f(CM)$, where f is some (possibly stochastic) function that converts the matrix product CM element-wise into a set of binary values. Figure 7.3 shows a cartoon of this process.[3]

We summarise sufficient conditions on the binary-valued function f to ensure a finite set of observations will contain a finite number of active features below:

[3]It is possible to add more layers to the hierarchy, but we believe two layers should suffice for most applications. Inference also becomes more complex with more layers.

Proposition 7.1.1. *If* $z_{nk} = f(c_n^\top m_k)$ *and* $c_n^\top m_k = 0$ *implies* $f(c_n^\top m_k) = 0$*, then the number of features in a finite dataset will be finite with probability one.*

Proof. Let L be the number of active categories in N observations, and let K be the number of active features in the L categories. The conditions on NP1 ensure L is bounded with probability one. Since L is bounded, the conditions on NP2 ensure K is also bounded with probability one. Let $C_{1:N}$ denote the first N rows of C. Thus the product $C_{1:N}M$ contains at most NK nonzero values. The second condition on f ensures that $Z_{1:N}$ has a finite number of non-zero values with probability one. □

Intuitively, the sufficient conditions imply (1) only the presence—not the absence–of a category can cause features to be present in the data and (2) categories can only cause the presence of the features associated with them. These implications are similar to those of the standard IBP model, where we require only the presence of a feature has an effect on the observations.

The previous discussion was limited to situations where C, M, and Z are binary. However, other sparse processes may be used for C and M, and the output of $f(\cdot)$ need not be binary as long as $f(0) = 0$ with probability one. For example, the infinite gamma-poisson process (Titsias, 2007) creates a sparse integer-valued matrix; such a prior may be appropriate if categories are associated with multiple copies of a feature.

7.2 Specific Models

Many choices exist for the nonparametric processes NP1 and NP2 and the function f. Here we describe nested models that use the Dirichlet Process (DP) and the Indian Buffet Process as base processes. However, other models such as the Pitman-Yor Process could also be used. The DP-IBP model is a factorial approach to clustering where we expect clusters to share features. The IBP-IBP models add an additional layer of sophistication: an observation may be associated with multiple feature sets, and sets may share features.

The DP and the associated Chinese restaurant process (CRP) are a distribution on discrete distributions which can be used for clustering (see Ferguson, 1973). We represent the CRP in matrix form by setting $c_{nl} = 1$ if observation n belongs to cluster l. The IBP (Griffiths and Ghahramani, 2005) is a feature model in which each observation is associated with $\mathsf{Poisson}(\alpha)$ features. Similar to the DP, a few popular features are present in most of the observations. Specifically, given N total observations, the probability that observation n contains an active feature k is r_k/N, where r_k is the number of observations currently using feature k. Both the CRP and the IBP are exchangeable in the observations and features.

7.2.1 DP-IBP Model

The DP-IBP model draws C from a CRP and M from an Indian Buffet Process. We let $f(c_n^\top m_k) = c_n^\top m_k$ and thus $Z = CM$:

$$\begin{aligned} C &\sim \mathsf{CRP}(\alpha_C) \\ M &\sim \mathsf{IBP}(\alpha_M) \\ z_{nk} &= c_n^\top m_k \end{aligned} \tag{7.3}$$

In the context of the Chinese restaurant analogy for the DP, the DP-IBP corresponds to customers (observations) sitting at tables associated with combo meals (categories) instead of single dishes, and different combo meals may share specific dishes (features). As in the DP, the dishes themselves are drawn from some continuous or discrete base distribution.

Properties. The properties of the DP and IBP ensure the DP-IBP will be exchangeable over features and the observations. The distribution over the number of features has no closed form, but we can bound its expectation. The expected number of categories N_C in a DP with N observations is $O(\alpha_C \log(N))$. Given N_C, the number of features N_f is distributed as $\mathsf{Poisson}(\alpha_M H_{N_C})$, where H_{N_C} is the harmonic number corresponding to N_C. We apply Jensen's inequality to the iterated expectations expression for $E[N_f]$ to bound the expected

number of features:

$$\begin{aligned} E[N_f] &= E_c[E_m[N_f|N_c]] \\ &= E_c[\alpha_M H_{N_c}] \\ &= E_c[\alpha_M \log(N_c) + O(1)] \\ &\leq \alpha_M \log(E_C[N_c]) + O(1) \\ &= \alpha_M \log(O(\alpha_C \log(N))) + O(1) \\ &= O(\log\log(N)) \end{aligned} \tag{7.4}$$

Inference. We apply the partial Gibbs sampling scheme described in Neal (2000) to resample the DP category matrix C. The IBP matrix M can be resampled using the standard equations described in Griffiths and Ghahramani (2005). In both cases, the sampling equations have the same general form:

$$\begin{aligned} &P(m_{lk}|X, Z, C, M_{-lk}, A) \\ &\propto \; P(m_{lk}|M_{-lk})P(X|Z, A)P(Z|C, M) \end{aligned} \tag{7.5}$$

where M_{-lk} denotes the elements of M excluding m_{lk}. An attractive feature of the DP-IBP model is that because Z is a deterministic function of C and M, the likelihood term $P(X|Z, A)P(Z|C, M)$ reduces to $P(X|C, M, A)$. Because the data is directly considered when sampling categories and connections, without Z as an intermediary, the sampler tends to mix quickly.

Demonstration. We applied the Gibbs sampler to a synthetic dataset of 700 block images from Griffiths and Ghahramani (2005). The 6x6 pixel images contained four types of blocks, shown in the lower left quadrant of figure 7.4, which always cooccurred in specific combinations (lower right quadrant). We ran 3 chains for 1000 iterations with the DP-IBP model, using a likelihood model of the form $X = ZA + \epsilon$. The features A had an exponential prior and ϵ was Gaussian white noise uncorrelated across observations and dimensions. All hyperparameters were sampled using vague Gamma priors.

The top half of figure 7.4 shows a representative sample from the inference. The DP-IBP recovers that the images contain four types of blocks cooccurring in nine combinations. In particular, the DP-IBP hierarchy allows the inference to naturally discover the null-cluster, corresponding to no features being present, without additional parameters (as required for IHFR (Rai and Daume, 2009)). The sampler quickly converges near the true number of features and clusters (figure 7.5).

Figure 7.4: Sample showing structure found by the DP-IBP. Both the features and clusters (top row) closely match the underlying structure (bottom row).

7.2.2 IBP-IBP Model [4].

The DP-IBP associates each observation with only one cluster. However, some situations may naturally contain observations with memberships in multiple categories. For example, a image of a picnic may contain typical outdoor elements, such as trees and sky, as well as food-related objects. A multiple membership model at the category level would allow an observation to be part of multiple sets. In the IBP-IBP model, we place IBP priors on both C and

[4] A similar model was simultaneously derived as the infinite factor model hierarchy (Courville, 2009)

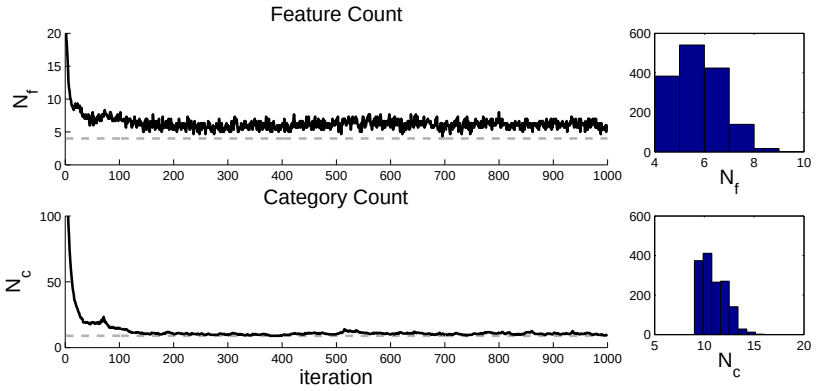


Figure 7.5: Evolution of the number of clusters and features (dashed lines show true values). The histograms of posterior over feature and cluster counts were computed from the final 500 samples. The hyperparameters converged in a similar fashion.

M and set the link function f to the 'or' of the product $c_n^\top m_k$:

$$\begin{aligned} C &\sim \mathsf{IBP}(\alpha_C) \\ M &\sim \mathsf{IBP}(\alpha_M) \\ z_{nk} &= (c_n^\top m_k) > 0 \end{aligned} \tag{7.6}$$

Properties. The expected number of features in the IBP-IBP can be bounded similarly to the DP-IBP. The number of active categories N_C in N observations is distributed as $\mathsf{Poisson}(\alpha_C H_N)$, so the expected number of categories is still $O(\log(N))$. Given a number of categories, the number of features is distributed as $\mathsf{Poisson}(\alpha_M H_{N_C})$. Thus, by equation 7.4, the expected number of features is bounded by $O(\log\log(N))$. The distribution of Z is exchangeable in both the features and observations from the properties of the IBP.

Inference. To Gibbs sample in the IBP-IBP model, we use the equations of Griffiths and Ghahramani (2005) to sample both C and M.

7.2.3 Noisy-Or IBP-IBP Model

The 'or' in the IBP-IBP model implies that a feature is present in an observation if any of its parent categories are present. This hard constraint may be

unrealistic: for example, kitchen scenes may often contain refrigerators, but not always. The noisy-or IBP-IBP uses a stochastic link function in which

$$P(z_{nk} = 1|C, M) = 1 - q^{c_n^\top m_k} \tag{7.7}$$

where $q \in [0, 1]$ is the noise parameter (Pearl, 1988).[5] An attractive feature of the noisy-or formulation is that the probability of a feature being present increases as more of its parent categories become active. For example, we might expect a scene tagged with both kitchen and dining categories may be more likely to contain a table than a scene tagged as only a kitchen.

The noisy-or IBP-IBP is summarised by:

$$\begin{aligned} C &\sim \mathsf{IBP}(\alpha_C) \\ M &\sim \mathsf{IBP}(\alpha_M) \\ z_{nk} &\sim \mathsf{Bernoulli}(1 - q^{c_n^\top m_k}) \end{aligned} \tag{7.8}$$

Properties. The noisy-or IBP-IBP inherits its exchangeability properties and feature distribution from the IBP-IBP (the parameter q only scales the expected number of features by a multiplicative constant). As $q \to 0$, the noisy-or IBP-IBP reduces to the IBP-IBP.

Inference. Because f is stochastic, the feature assignments Z must also be sampled. Given M, C, and X, the probability that a feature $z_{nk} = 1$ is given by

$$P(z_{nk} = 1|C, M, X, A) \propto (1 - q^{c_n^\top m_k}) P(x_n|z_n, A). \tag{7.9}$$

Gibbs sampling C and M is identical to the IBP-IBP case except the likelihood terms now depend on Z and are generally independent of the data.[6] For

[5] Proposition 7.1.1 requires that the noisy-or does not also leak, that is, $P(z_{nk} = 1)$ must be 0 if all the parents c_n of z_{nk} are zero.

[6] The exception is when new features are being sampled.

example, when resampling m_{lk},

$$\begin{aligned} & P(m_{lk}|X, Z, C, M_{-lk}, A) \\ & \quad \propto \quad P(m_{lk}|M_{-lk})P(X|Z, A)P(Z|C, M) \\ & \quad \propto \quad P(m_{lk}|M_{-lk})P(Z|C, M) \end{aligned} \tag{7.10}$$

The constraints of the noisy-or model pose problems when naively sampling a single element of C or M. For example, suppose n is the only observation using category l. According to the IBP prior, $Pr[c_{nl} = 1] = 0$. However, if c_{nl} is the only active parent of feature z_{nk}, and $z_{nk} = 1$, then according to the likelihood $P(c_{nl} = 1) = 1$. In such situations, c_{nl} and its children from z_n should be sampled jointly.

Another problem arises when all the parents of z_{nk} are inactive but the likelihood $P(x_n|z_n, A)$ prefers $z_{nk} = 1$. To set $z_{nk} = 1$, one of z_{nk}'s parents must become active. However, if $z_{nk} = 0$, z_{nk}'s parents are unlikely to turn on. Jointly sampling z_{nk} with its parents resolves these issues.

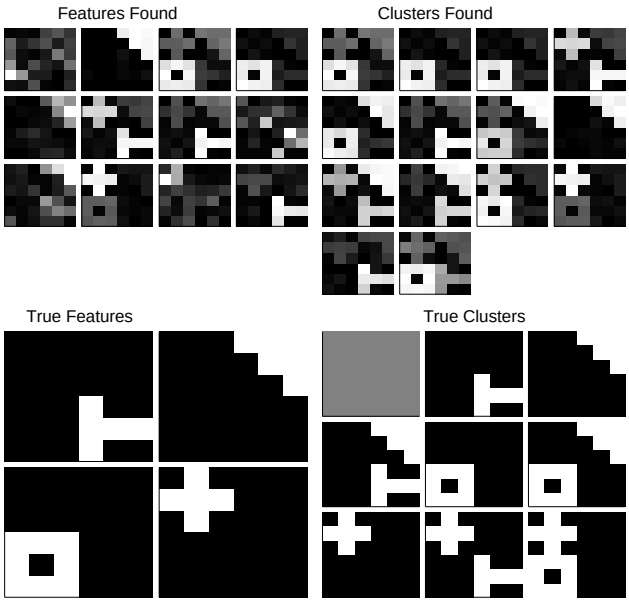


Figure 7.6: Sample showing structure found by the Noisy-Or IBP-IBP. Both the features and clusters (top row) reflect the underlying structure (bottom row), but often contain replicas.

Demonstration. We return to the blocks example of section 7.2.1, using the same clusters as before. However, unlike in section 7.2.1, we allow multiple (often overlapping) clusters to be present in an observation when generating the observations. Thus, the observations used to demonstrate the noisy-or IBP are *not* the same as the observations used in section 7.2.1; they have significantly more complex structure.

Figure 7.6 shows the inferred features and clusters for a typical sample from the inference. The inferred features largely match the true features, but they are more noisy, and features are sometimes repeated. Similarly, the inferred clusters contain the true clusters and some replicas. The ghosted features and replicas are a common occurrence when sampling in IBP-like models; they occur when multiple observations propose similar features. Over time they tend to disappear, but this time can be exponential in the size of the dataset.[7]

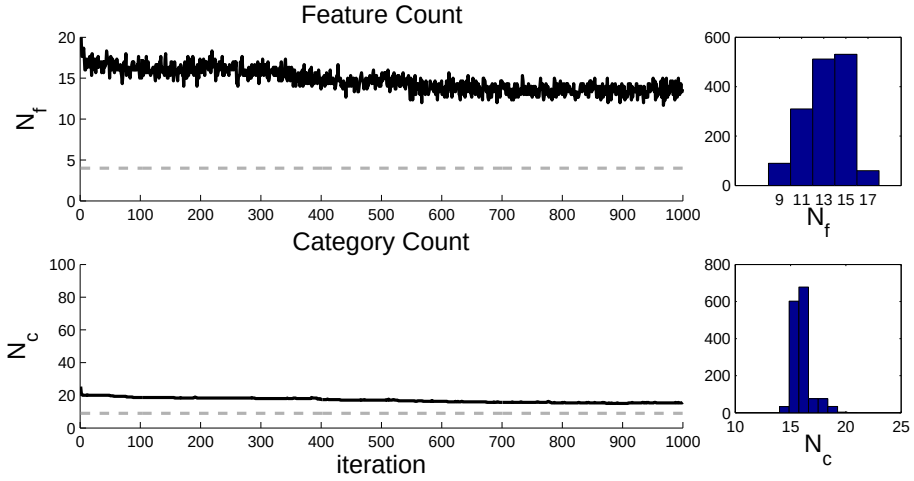


Figure 7.7: Evolution of the number of clusters and features. The dashed lines show the true values; the histograms show the posterior over the number of features and clusters. As before, the hyperparameters, given vague priors, converged in a similar fashion.

Figure 7.7 confirms that the number of features and categories is often overestimated. The posterior of the flexible noisy-or has many local optima; even after the sampler has mixed, many modes must be explored. However,

[7] One might introduce Metropolis moves to merge features or clusters. However, we find that these moves, while effective on small problems, have little effect in more complex, higher-dimensional real-world models.

Table 7.1: Descriptions of data sets.

Dataset	N	D	Description
UN	155	15	Human development statistics for 155 countries
Joke	500	30	User ratings (continuous) of 30 jokes
Gene	251	226	Expression levels for 226 genes
Robot	750	23	Visual object detections made by a mobile robot
India	398	14	Socioeconomic statistics for 398 Indian households

the replicas do not prevent the noisy-or IBP-IBP from producing good reconstructions of the data.

7.3 Experiments

We applied the three models of section 7.2 to five real-world datasets (table 7.1). The gene data consisted of expression levels for 226 genes from 251 subjects (Carvalho et al., 2008). The UN data consisted of a dense subset of global development indicators, such as GDP and literacy rates, from the UN Human Development Statistics database (UN, 2008). Similarly, the India dataset consisted of development statistics from Indian households (Desai et al., 2005). The joke data consisted of a dense subset of continuous-valued ratings of various jokes (Goldberg et al., 2001). Finally, the robot data consisted of hand-annotated image tags of whether certain objects occurred in images from a robot-mounted camera (Kollar, 2008).

Inference was performed using uncollapsed Gibbs sampling on 3 chains for 1000 iterations. Chains for more complex models were initialised with the final sample outputted by simpler models: the IBP and the DP-IBP were initialised with the output of the DP, the IBP-IBP was initialised from the DP-IBP, and the noisy-or IBP-IBP was initialised from the IBP-IBP. The likelihood model $x_n = z_n A + \epsilon$, where $a_{kd} \sim \mathsf{Exponential}(\lambda)$ and the noise $\epsilon \sim \mathsf{Normal}(0, \sigma_n^2)$ was used for the continuous-valued datasets. Under this model, the conditional posterior on a_{kd} was a truncated Gaussian. For the binary robot data, the

likelihood was given by

$$P(x_{nd} = 1|z_{nd} = 1) = 1 - m_d$$
$$P(x_{nd} = 1|z_{nd} = 0) = f_d$$

where f_d was the probability of a false detection, and m_d was the probability of a missed detection. These simple likelihood models are not necessarily the best match for complex, real-world data. However, even these simple models allowed us to find shared structure in the observations. (For a real application, of course, one would use an appropriately designed likelihood model.) Finally, all hyperparameters were given vague priors and sampled during the inference.

Quantitative Evaluation We evaluated inference quality by holding out approximately $10D$ X_{nd} values during the initial inference. No observation had all of its dimensions missing, and no dimension had all of its observations missing. Because models had different priors, inference quality was measured by evaluating the test log-likelihood and the L2 reconstruction error of the missing elements (a complete Bayesian model comparison would require an additional prior over the model classes). The evaluation metrics were averaged from the final 50 samples of the 3 chains.

Tables 7.2 and 7.3 and figure 7.8 compare the reconstruction errors and predictive likelihoods of the three variants with the standard IBP, which does not model correlations between features, the DP, a flat clustering model. As we are most interested in feature-based models (IBP, DP-IBP, IBP-IBP, or noisy-or IBP-IBP), both the best-performing feature-based model, as well as the best overall model, are highlighted in bold. Plots for the hyperparameters are not shown, but the posteriors seemed to converge during the inference.

The structured variants outperformed the standard IBP in almost all cases. In particular, the DP-IBP usually had the best performance among the feature-based models. Its performance was on par with the DP—a simpler model with a much more robust inference procedure. Indeed, we believe that the robustness of inference in the DP—and the difficulty of inference in the complex

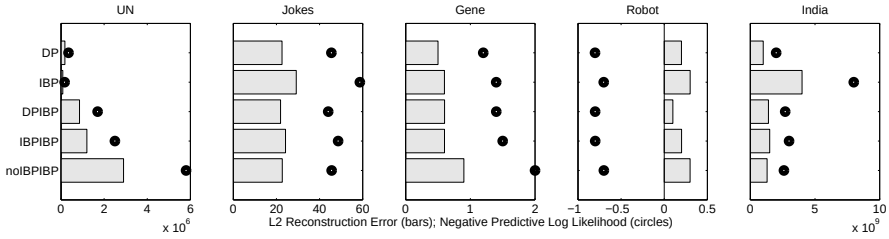


Figure 7.8: Negative predictive log-likelihoods (circles) and L2 reconstruction errors (bars) for held-out data from the real-world datasets. Both metrics have the same scale. Smaller values indicate better performance; note that models have consistent performance with respect to both predictive log-likelihoods and L2 reconstruction errors across all datasets.

Table 7.2: Predictive likelihoods on held-out data (higher is better). Bold figures indicate the best feature-based and best overall models for each dataset.

Model	UN	Jokes	Gene	Robot	India
DP	-3.5e5	-45.5	**-1.2**	0.8	**-2.0e9**
IBP	**-1.7e5**	-58.7	-1.5	0.7	-8.0e9
DPIBP	-17.0e5	**-44.0**	**-1.4**	**0.8**	-2.7e9
IBPIBP	-25.0e5	-48.6	-1.6	0.8	-3.0e9
noIBPIBP	-58.0e5	-45.6	-2.0	0.7	**-2.6e9**

posterior landscapes of the structured models—is the primary reason for the difference in the models' performance: the tight coupling between parameters in the structured models make them incredibly slow to mix. Better inference procedures for the structured models will certainly improve their performance (since the structured models have DPs and IBPs as special cases, they should be able to perform at least as well). However, unlike the flat clusters provided by the DP, the DP-IBP can provide a structured representation of the data

Table 7.3: L2 reconstruction errors on held-out data (lower is better). Bold figures indicate the best feature-based and best overall models for each dataset.

Model	UN	Jokes	Gene	Robot	India
DP	18.0e4	22.6	**0.5**	0.2	**1.0e9**
IBP	**8.5e4**	29.2	0.6	0.3	4.0e9
DPIBP	86.0e4	**21.9**	**0.6**	**0.1**	1.4e9
IBPIBP	120.0e4	24.2	0.6	0.2	1.5e9
NoisyOr IBPIBP	290.0e4	22.7	0.9	0.3	**1.3e9**

alongside good quantitative performance. These more qualitative benefits are explored in the next section.

Qualitative Examples In table 7.2, we saw DP clustering had lower error rates than any of the feature-based models on the UN dataset. However, even in this case, the structured representation of the correlated feature models can provide some explanatory power.[8] The image in figure 7.9 shows a representative feature matrix A from the DP-IBP. Each row in the matrix corresponds to a development statistic; for visualisation the values in each row have been scaled to $[0, 1]$, each column corresponds to a feature.

We see that the first column, with high GDP and healthcare scores, is what we would expect to find in highly developed countries. The third column is representative of countries with an intermediate level of development, and the final column, with low GDP and high tuberculosis rates, has a pattern common in developing countries. The other columns highlight specific sets of statistics: column two augments technology and education, while column four corresponds to higher private healthcare spending and higher prison populations (and always occurs in conjunction with some other features).

What distinguishes the DP-IBP from simple clustering is that the features are shared among clusters. The circles on top of the feature matrix in figure 7.9 represent categories or clusters (only 5 of the 15 are shown). The M matrix is visualised in the links between the categories and the features. Each column represents a feature, where the rows are the dimensions of the data. As with flat clustering, some categories ($C2, C5$) only use one feature (that is, connect to only one column). For example, certain developed nations such as Canada and Sweden are well characterised by only the first feature, and developing nations such as Mali and Niger are characterised by only the final feature.

However, feature sharing allows other categories to use multiple features. For example, the United States has high development statistics, like Canada and Sweden, but it also has, added on, relatively high tuberculosis rates, private healthcare spending, and prison populations. Showing what character-

[8] We stress that as we are using unsupervised methods, the structures found cannot be considered to be discovering the 'true' or 'real' structure of dataset. We can only say that the structures found explain the data.

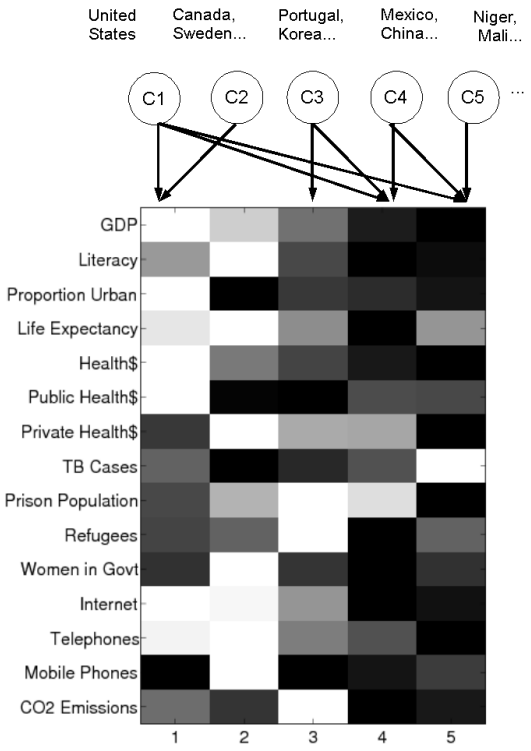


Figure 7.9: Part of a DP-IBP sample from the UN data. The rows in the image correspond to the development statistics, and columns represent feature vectors. The top row of circles, with links to various features, are some of the categories in this sample (there were 15 categories in total). Representative countries are listed above each category.

istics the United States shares with other countries is more informative than simply placing it in its own cluster.

The DP-IBP also found informative clusters in the robot data. As with the UN dataset, the DP-IBP was able to do a more sophisticated clustering that reflected shared elements in the data. Some of the categories of image tags discovered are listed below:

```
C1: hallway, door, trash can, chair, desk,
    office, computer, whiteboard
C2: door, trash can, robot, bike,
    printer, couch
C3: trash can, monitor, keyboard, book,
```

```
    robot, pen, plant
C4: hallway, door, trash can
```

Categories C1 and C4 often occurred singly in the data, in the form of simple clusters that corresponded to hallway and office scenes. Categories C2 and C3 often augmented category C1, reflecting office scenes that also included parts of a printing area (C2) and close-up views of desk spaces (C3).

7.4 Discussion

We find in particular that the DP-IBP model, combining the unbounded number of clusters of a DP mixture with the nonparametric shared features of an IBP, provides a promising method for hierarchical hidden representations of data. Occupying a regime between pure clustering and pure feature-based modelling, the DP-IBP can capture dominant categorical qualities of the real-world datasets but still discover shared structure between clusters. It outperforms the standard IBP because allowing categories to share features lets each feature use raw evidence from more observations and thus grow more refined. At the same time, forcing observations to be associated with only one category limits the model's flexibility. Thus, the DP-IBP has fewer of the identifiability issues common to feature-based models and produces more relevant categories.

In some situations, the more complex structured models may have been better matches for the data—for example, the robot data almost surely contained situations where multiple categories of noisy features were present. Here, better inference techniques could have proved beneficial. Split-merge moves may help accelerate mixing, but from a limited set of experiments we found that the benefits were highly data dependent: such moves provided some benefit in the toy images data set, where replica features tended to be a problem, but proved to be less useful in the more complex local optima of the real world data. Inference that uses "soft" assignments—such as variational techniques—may prove to be more robust to these local optima.

We see an important trade-off when choosing what kind of nonparametric model to apply. Our work was initially motivated to create a model for sce-

narios like the robot data, and thus we desired a generative process that would explain noisy, multiple-membership correlated feature models. An interesting question is how one may perform model selection across different choices of non-parametric priors within this general framework: while these models perform well when the prior reflects the data—such as in the toy blocks examples—the structure appropriate for real-world data is much more difficult to ascertain.

7.5 Summary

Our probabilistic setting generates models with an unbounded number of features and categories and provides a general framework for modelling a variety of different types of correlations. The framework can also model other useful properties such as feature sparsity (that is, many observations being featureless). Interesting extensions might include incorporating aspects of the phylogenetic IBP (Miller et al., 2008b) or infinite factorial HMM (Van Gael et al., 2009) to create models that consider correlations both between features and also between observations, as well as exploring methods to model negative correlations. The work in this paper also provides avenues which could be used to develop "deep" nonparametric models with multiple unbounded layers of hidden variables.

In particular, the DP-IBP model, combining nonparametric clustering with the shared features of an IBP, is a promising method for layered representations of latent variables. However, given the complexities of performing inference in these models, more analysis is needed to study the behaviours these models in real-world applications and to determine the sensitivities of the models to various hyperpriors. The work presented here is only one step toward achieving more structured nonparametric models.

Chapter 8

Conclusions and Future Work

In this thesis, we presented several new inference algorithms for the Indian Buffet Process, enabling the use of the IBP in datasets with up to 100,000 observations. We also described a framework for extending the standard IBP model to incorporate correlations among the latent features. We conclude with some final comparisons among the different inference techniques, and a discussion of future work.

Comparing Sampling and Variational Inference

In chapter 6, we concluded that variational inference was a sensible option when collapsed Gibbs sampling was not possible. However, the accelerated sampler of chapter 4 allows for efficient sampling in datasets where Gibbs sampling was previously computationally intractable. Figure 8.1 shows the evolution of the training log likelihood for collapsed Gibbs sampling, accelerated Gibbs sampling, and variational inference on the Yale dataset. The Gibbs samplers were run for 250 iterations each; the variational inference was run until the variational lower bound converged up to a stopping threshold.

Consistent with the results of chapter 6, variational inference finds a reasonable mode faster than collapsed Gibbs sampling. However, the accelerated sampler finds a better mode faster than the variational inference (note log-scale on the plot). Thus, we reiterate that variational inference should be used only

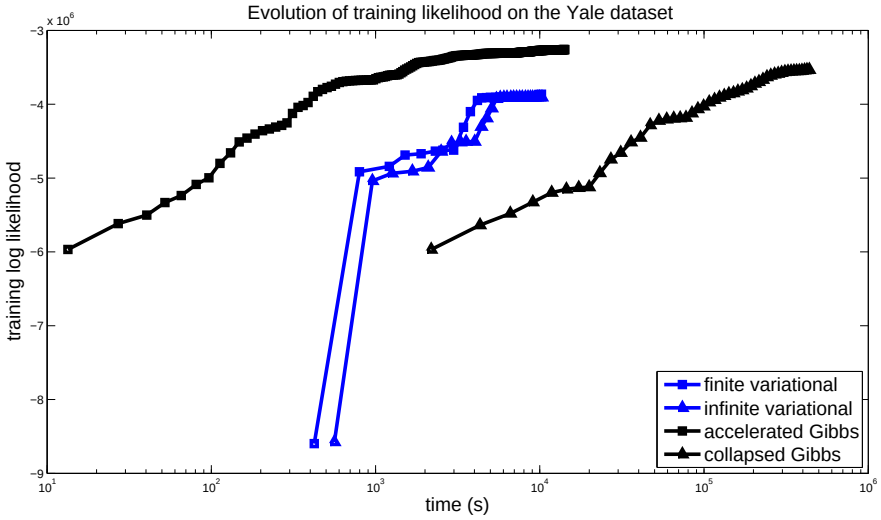


Figure 8.1: Evolution of training log likelihood on the Yale dataset. The accelerated Gibbs sampler finds a better mode faster than variational inference.

when collapsed Gibbs sampling is not possible—either in very large datasets or in non-conjugate exponential family models.

Comparison to other Deep Models

There exists a vast literature on other 'deep' models (PDP Research Group, 1986; Hinton et al., 2006b; Hinton and Salakhutdinov, 2006; Bengio, 2007b), that is, models with one or more layers of latent structure. One of the most common is the restricted Boltzmann machine (PDP Research Group, 1986). A Boltzmann machine is a network of binary nodes. The energy of a node i is given by

$$E_i = \sum_j W_{ij} s_j - \theta_i$$

where W is a symmetric weight matrix, s_i is the value of node i, and θ_i is a threshold parameter. Given the energy, the probability that node i is active is

$$p(s_i = 1) = \frac{1}{1 + \exp(-E_i/T)}$$

for some temperature parameter T. If the matching implied by W is bipartite, the model is called a restricted Boltzmann machine (RBM).[1]

We compared L2 reconstruction errors on held-out data on the IBP to the an RBM with 1000 binary hidden units [2]. The data were normalised to 0-1 for processing, and then scaled back to produce the original images. Under these settings, the RBM had a running time comparable to the accelerated Gibbs sampler for the Yale, AR Faces, and EEG datasets; it was 10 times fast for the piano dataset. In all cases, the reconstruction errors were comparable (see table 8.1; more analysis is needed as to what model is more appropriate for specific situations.

Table 8.1: L2 reconstruction errors per pixel on held-out data for the RBM (1000 nodes) and the IBP.

Dataset	RBM (1000 nodes)	IBP (accelerated sampler)	IBP (uncollapsed sampler)
Yale	459.3	459.4	413.0
AR Faces	2128.5	919.9	3836.4
EEG	259230	282108	215174
Piano	0.0009	0.0005	-

Future Work

The IBP has a philosophical elegance that many other latent feature models lack: the feature-assignment matrix Z has a clear interpretation as which features are present in which observations, and the directed model clearly shows how features and feature assignments generate the observations. Furthermore, the IBP does not bound the number of active features K; a posterior over the number of active features is computed as part of the inference process.

[1] RBMs are closely related to deep belief nets, which have the following form: there are visible units v and hidden units h. The probability of a vector v is given by

$$p(v) = \sum_h p(h|W)p(v|h, W)$$

for some symmetric weight matrix W.

[2] Using code from Hinton and Salakhutdinov (2006)

For real-world applications, philosophical elegance is only part of the package. If IBPs are to be accepted outside the small community of nonparametric Bayes practitioners, rigorous tests must compare the IBPs performance, both in terms of runtime and solution quality, to other models (deep belief nets/restricted Boltzmann machines, factor analysis, iterative generalised least squares) from the literature. We must also test where the truly nonparametric is useful: in chapter 6, we saw that truncated models often perform as well as infinite ones, with better computational properties. An interesting inference problem might be developing a more efficient RJMCMC procedure for determining the more likely numbers of features quickly.

Regarding the inference itself, this thesis makes important contributions to scaling IBP inference, especially in conjugate models. However, many realworld applications do not have conjugate models, and making Gaussianity assumptions about the data can lead to poor results. In addition to nonconjugate likelihoods, deeper hierarchical structures, such as those presented in chapter 7, generally have nonconjugate forms between the layers. Moreover, these more complex models often suffer from many local optima. Another area for future work is to improve inference in nonconjugate and hierarchical models.

More generally, from genetics to social choice, from dynamical systems to computer vision, researchers have read about IBPs and found its story compelling. What is needed from the nonparametric Bayes community are the methods and the rigorous evaluations to fulfil this promise.

Appendix A

Likelihood Models

The IBP provides a prior over feature assignment matrices Z; however, to complete the generative model we also require a likelihood function $P(X|Z, A)$ that describes how the feature assignments produce the data, where A represents a set of parameters associated with the likelihood function. Combined with the IBP prior $P(Z)$ and a prior $P(A)$, the joint probability distribution $P(X, Z, A)$ is

$$P(X, Z, A) = P(X|Z, A)P(Z)P(A). \tag{A.1}$$

Choosing an appropriate likelihood function usually involves a tradeoff between expressing the data well (application dependent) and computational tractability. This thesis uses three general-purpose functions in which the likelihood $P(X|Z, A)$ is some function of the product ZA. Thus, each row of A can be interpreted as a vector characterising some latent feature in the environment. We emphasise that practical applications will likely require more sophisticated likelihood models.

A.1 Linear-Gaussian

In the linear-Gaussian feature model, we model the data X as a product $ZA+\epsilon$, where ϵ is some observation noise in our data. Here, X is a $N \times D$ matrix of observations, Z is the $N \times K$ binary matrix of feature assignments, and A is a $K \times D$ matrix of features. We assume that the noise w is independent of Z

and A and uncorrelated across observations. Figure A.1 shows a cartoon of the model.

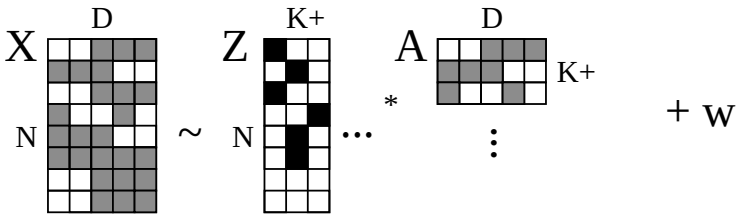


Figure A.1: The linear feature model proposes the data X is the product of Z and A with some noise.

Each element of ϵ is assumed to be an independent, zero-mean Gaussian with some noise variance σ_n^2. Specifically, we assume that the noise is uncorrelated across observations and dimensions. We also place an independent zero-mean Gaussian prior on each element of A with variance σ_a^2. (If the data is not zero-mean, it can always be centred first.)

A key advantage of the linear-Gaussian model is that it is fully-conjugate. As a result, we have an analytic expression for $P(X|Z)$:

$$\begin{aligned} & p(X|Z,\sigma_n,\sigma_a) \\ = \; & \frac{\exp(-\frac{1}{2\sigma_n^2}(X^T(I - Z(Z^TZ + \frac{\sigma_n^2}{\sigma_a^2}I)^{-1}Z^T)X))}{(2\pi)^{\frac{ND}{2}}\sigma_n^{(N-K)D}\sigma_a^{KD}|Z^TZ + \frac{\sigma_n^2}{\sigma_a^2}I|^{\frac{D}{2}}} \end{aligned} \tag{A.2}$$

where σ_n^2 is the noise variance and σ_a^2 is the feature variance. As seen in equation A.2, integrating out the features A when computing $p(X|Z)$ correlates the observations. A naive computation of equation A.2 requires $O(N^3)$ operations; Griffiths and Ghahramani (2005) describe a set of rank-one updates to reduce this computational cost to $O(N^2)$ per call.

The posterior on A given Z and X also has an analytic form:

$$\begin{aligned} \mu^A &= (Z^TZ + \frac{\sigma_n^2}{\sigma_a^2}I)^{-1}Z^TX \\ \Sigma^A &= \sigma_n^2(Z^TZ + \frac{\sigma_n^2}{\sigma_a^2}I)^{-1} \end{aligned} \tag{A.3}$$

where the mean μ^A has the dimensionality of A (KxD). All dimensions are independent with identical variances, so the joint covariance is block diagonal with D identical KxK blocks. We let Σ^A be the KxK covariance for $A_{\cdot,d}$.

When doing inference in the IBP model, the linear-Gaussian likelihood allows us to integrate out A and sample only Z. A disadvantage of the linear-Gaussian likelihood model is that it is not invariant to rotations. As features can contain negative and positive values, parts of features can be used to cancel parts of other features. Thus, posterior space has many more modes and local optima.

A.2 Exponential-Gaussian

The exponential-Gaussian model has the form as the linear-Gaussian model, where the data is modelled as $ZA + \epsilon$. However, instead of a Gaussian prior on the features A, we apply an exponential prior in which each element of A is drawn from an independent exponential distribution with rate λ.

Unlike the linear-Gaussian model, the exponential model is not fully conjugate. Inference in this model requires that A be explicitly represented. The probability of a single element of A_{kd} (given all of the other elements) is

$$P(A_{kd}|Z, A_{-kd}, X) \propto P(X|Z, A)P(A_{kd})$$

where $P(X|Z, A)$ is a Gaussian and $P(A_{kd})$ is an exponential. Thus, the product is a truncated Gaussian. While we can no longer integrate out A, we can at least sample new values of A_{kd} one element at a time from this truncated Gaussian.

Although the model can no longer be collapsed—that is, we do not have an expression for $P(X|Z)$—a key advantage of the exponential-Gaussian model is that the posterior space has fewer local optima. All features are non-negative, so features can no longer cancel out other features.

A.3 Binary-Bernoulli

The previous two models are for real-valued data. For binary data, we model the probability $P(X_{nd} = 1|Z, A)$ by a leaky, noisy-or:

$$P(X_{nd} = 1|Z, A) = 1 - \epsilon\lambda^{Z_n A_d} \tag{A.4}$$

where A is also a binary matrix in which each element A_{kd} has an independent probability of p_A to be one.

As with the exponential-Gaussian model, we cannot integrate out A to find an expression for $P(X|Z)$. However, since each element of A is binary, we can compute the likelihood with $A_{kd} = 1$ and $A_{kd} = 0$ when evaluating $P(A_{kd}|Z, A_{-kd}, X)$

Appendix B

Derivations for Variational Inference in the IBP

Chapter 6 described a variational inference method for the IBP. This appendix derives the variational lower bound and the variational updates for terms in (6.1) that belong to the exponential family.

B.1 Variational Lower Bound

We derive expressions for each expectations in Equation (6.1):

1. Each stick is independent, and substituting the form of the beta prior we get

$$\begin{aligned}\mathbb{E}_{\boldsymbol{v}}\left[\log p(v_k|\alpha)\right] &= \mathbb{E}_{\boldsymbol{v}}\left[\log\left(\alpha v_k^{\alpha-1}\right)\right], \\ &= \log\alpha + (\alpha-1)\,\mathbb{E}_{\boldsymbol{v}}\log(v_k), \\ &= \log\alpha + (\alpha-1)\left(\psi(\tau_{k1}) - \psi(\tau_{k1}+\tau_{k2})\right),\end{aligned}$$

   where $\psi(\cdot)$ is the digamma function.

2. For the feature assignments, which are Bernoulli-distributed given the feature probabilities, we first break the expectation into the following

parts

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v},Z}\left[\log p(z_{nk}|\boldsymbol{v})\right] &= \mathbb{E}_{\boldsymbol{v},Z}\left[\log p(z_{nk}=1|\boldsymbol{v})^{z_{nk}} p(z_{nk}=0|\boldsymbol{v})^{1-z_{nk}}\right] \\
&= \mathbb{E}_Z\left[z_{nk}\right]\mathbb{E}_{\boldsymbol{v}}\left[\log\prod_{m=1}^{k} v_m\right] + \mathbb{E}_Z\left[1-z_{nk}\right]\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k} v_m\right)\right] \\
&= \nu_{nk}\left(\sum_{m=1}^{k}\psi(\tau_{k2}) - \psi(\tau_{k1}+\tau_{k2})\right) \\
&\quad +(1-\nu_{nk})\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k} v_m\right)\right]
\end{aligned}$$

The second line follows from the definition of $\boldsymbol{v}$, while the third line follows from the properties of Bernoulli and beta distributions.

3. For the feature distribution, we simply apply the properties of expectations of Gaussians to get

$$\begin{aligned}
\mathbb{E}_A\left[\log p(A_{k\cdot}|\sigma_A^2 I)\right] &= \mathbb{E}_A\left[\log\left(\frac{1}{(2\pi\sigma_A^2)^{D/2}}\exp\left(-\frac{1}{2\sigma_A^2}A_{k\cdot}^T A_{k\cdot}\right)\right)\right], \\
&= \mathbb{E}_A\left[\frac{-D}{2}\log(2\pi\sigma_A^2) - \frac{1}{2\sigma_A^2}A_{k\cdot}^T A_{k\cdot}\right], \\
&= \frac{-D}{2}\log(2\pi\sigma_A^2) - \frac{1}{2\sigma_A^2}\left(tr(\boldsymbol{\Phi}_k) + \bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T\right).
\end{aligned}$$

4. The likelihood for a particular observation is identical to the finite model, so we again have

$$\begin{aligned}
&\mathbb{E}_{Z,A}\left[\log p(X_{n\cdot}|Z_{n\cdot}, A, \sigma_n^2 I)\right] \\
&= -\frac{D}{2}\log(2\pi\sigma_n^2) \\
&\quad -\frac{1}{2\sigma_n^2}\left(X_{n\cdot}X_{n\cdot}^T - 2\sum_{k=1}^{K}\nu_{nk}\bar{\boldsymbol{\phi}}_k X_{n\cdot}^T + 2\sum_{k<k'}\nu_{nk}\nu_{nk'}\bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_{k'}^T + \sum_{k=1}^{K}\nu_{nk}\left(tr(\boldsymbol{\Phi}_k) + \bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T\right)\right).
\end{aligned}$$

5. The entropy can also be easily computed, since we have chosen exponential family distributions for our variational approximation:

$$\begin{aligned} H[q] &= -\mathbb{E}_q \log \left[\prod_{k=1}^{K} q_{\boldsymbol{\tau}_k}(v_k) \prod_{k=1}^{K} q_{\boldsymbol{\phi}_k}(A_{k\cdot}) \prod_{k=1}^{K} \prod_{n=1}^{N} q_{\nu_{nk}}(z_{nk}) \right], \\ &= \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{v}}(-\log q_{\boldsymbol{\tau}_k}(v_k)) + \sum_{k=1}^{K} \mathbb{E}_A(-\log q_{\boldsymbol{\phi}_k}(A_{k\cdot})) + \sum_{k=1}^{K} \sum_{n=1}^{N} \mathbb{E}_Z(-\log q_{\nu_{nk}}(z_{nk})), \end{aligned}$$

where

$$\begin{aligned} \mathbb{E}_{\boldsymbol{v}}(-\log q_{\boldsymbol{\tau}_k}(v_k)) &= \log \left(\frac{\Gamma(\tau_{k1})\Gamma(\tau_{k2})}{\Gamma(\tau_{k1}+\tau_{k2})} \right) - (\tau_{k1}-1)\psi(\tau_{k1}) \\ &\quad -(\tau_{k2}-1)\psi(\tau_{k2}) + (\tau_{k1}+\tau_{k2}-2)\psi(\tau_{k1}+\tau_{k2}). \\ \mathbb{E}_A(-\log q_{\boldsymbol{\phi}_k}(A_{k\cdot})) &= \frac{1}{2} \log \left((2\pi e)^D |\Phi_k| \right). \\ \mathbb{E}_Z(-\log q_{\nu_{nk}}(z_{nk})) &= -\nu_{nk} \log \nu_{nk} - (1-\nu_{nk}) \log(1-\nu_{nk}). \end{aligned}$$

Putting all the terms together gives us the variational lower bound in Equation (6.1).

B.2 Parameter Updates

To optimise the parameters, we can directly optimise Equation (6.1). However, since the conditional updates for the features A and feature assignments Z remain within the exponential family, so we can update the parameters more efficiently using equation (6.7).

1. For the feature distribution at the optimal $\bar{\boldsymbol{\phi}}_k$ and Φ_k

$$
\begin{aligned}
&\log q_{\boldsymbol{\phi}_k}(A_{k\cdot}) \\
&= \mathbb{E}_{A_{-k},Z}\left[\log p_K(\boldsymbol{W}, X|\boldsymbol{\theta})\right] + c, \\
&= \mathbb{E}_{A_{-k},Z}\left[\log p_K(A_{k\cdot}|\sigma_A^2) + \sum_{n=1}^{N} \log p_K(X_{n\cdot}|Z_{n\cdot}, A, \sigma_n^2)\right] + c, \\
&= -\frac{1}{2\sigma_A^2}\left(A_{k\cdot}A_{k\cdot}^T\right) - \frac{1}{2\sigma_n^2}\sum_{n=1}^{N}\mathbb{E}_{A_{-k},Z}\left[\left(X_{n\cdot} - Z_{n\cdot}A\right)\left(X_{n\cdot} - Z_{n\cdot}A\right)^T\right] + c, \\
&= -\frac{1}{2}\left[A_{k\cdot}\left(\frac{1}{\sigma_A^2} + \frac{\sum_{n=1}^{N}\nu_{nk}}{\sigma_n^2}\right)A_{k\cdot}^T - 2A_{k\cdot}\left(\frac{1}{\sigma_n^2}\sum_{n=1}^{N}\nu_{nk}\left(X_{n\cdot} - \left(\sum_{l:l\neq k}\nu_{nl}\bar{\boldsymbol{\phi}}_l\right)\right)\right)^T\right] + c.
\end{aligned}
$$

   Completing the squares and using Equation (6.7) gives us that for the optimal $\bar{\phi}_k$ and $\boldsymbol{\Phi}_k$, we must have

$$
\log q_{\boldsymbol{\phi}_k}(A_{k\cdot}) = -\frac{1}{2}\left(A_{k\cdot}\boldsymbol{\Phi}_k^{-1}A_{k\cdot}^T - 2A_{k\cdot}\boldsymbol{\Phi}_k^{-1}\bar{\boldsymbol{\phi}}_k^T\right) + c,
$$

   which gives us that the updates

$$
\begin{aligned}
\bar{\boldsymbol{\phi}}_k &= \left[\frac{1}{\sigma_n^2}\sum_{n=1}^{N}\nu_{nk}\left(X_{n\cdot} - \left(\sum_{l:l\neq k}\nu_{nl}\bar{\boldsymbol{\phi}}_l\right)\right)\right]\left(\frac{1}{\sigma_A^2} + \frac{\sum_{n=1}^{N}\nu_{nk}}{\sigma_n^2}\right)^{-1}, \\
\boldsymbol{\Phi}_k &= \left(\frac{1}{\sigma_A^2} + \frac{\sum_{n=1}^{N}\nu_{nk}}{\sigma_n^2}\right)^{-1} I.
\end{aligned}
$$

2. The updates for the variational distribution on Z are slightly different. For $\boldsymbol{\nu}$ parameters,

$$
\begin{aligned}
\log q_{\nu_{nk}}(z_{nk}) &= \mathbb{E}_{\boldsymbol{v},A,Z_{-nk}}\left[\log p(\boldsymbol{W}, X|\boldsymbol{\theta})\right] + c, \\
&= \mathbb{E}_{\boldsymbol{v},A,Z_{-nk}}\left[\log p(z_{nk}|\boldsymbol{v}) + \log p(X_{n\cdot}|Z_{n\cdot}, A, \sigma_n^2)\right] + c,
\end{aligned}
$$

   where

$$
\mathbb{E}_{\boldsymbol{v},Z_{-nk}}\left[\log p(z_{nk}|\boldsymbol{v})\right] = z_{nk}\sum_{i=1}^{k}\left(\psi(\tau_{i1}) - \psi(\tau_{i1}+\tau_{i2})\right) + (1-z_{nk})\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1 - \prod_{i=1}^{k} v_i\right)\right]
$$

and

$$\begin{aligned}
&\mathbb{E}_{A,Z_{-nk}}\left[\log p(X_{n\cdot}|Z_{n\cdot},A,\sigma_n^2)\right] \\
&= -\frac{1}{2\sigma_n^2}\left[-2z_{nk}\bar{\boldsymbol{\phi}}_k X_{n\cdot}^T + z_{nk}\left(tr(\boldsymbol{\Phi}_k)+\bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T\right)+2z_{nk}\bar{\boldsymbol{\phi}}_k\left(\sum_{l:l\neq k}\nu_{nl}\bar{\boldsymbol{\phi}}_l^T\right)\right]+c.
\end{aligned}$$

Therefore

$$\begin{aligned}
\log q_{\nu_{nk}}(z_{nk}) &= z_{nk}\left[\sum_{i=1}^{k}\left(\psi(\tau_{i1})-\psi(\tau_{i1}+\tau_{i2})\right)-\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{i=1}^{k}v_i\right)\right]\right. \\
&\qquad \left.-\frac{1}{2\sigma_n^2}\left(tr(\boldsymbol{\Phi}_k)+\bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T-2\bar{\boldsymbol{\phi}}_k X_{n\cdot}^T+2\bar{\boldsymbol{\phi}}_k\left(\sum_{l:l\neq k}\nu_{nl}\bar{\boldsymbol{\phi}}_l^T\right)\right)\right]+c.
\end{aligned}$$

From the canonical parameterisation of the Bernoulli distribution, we get that

$$\begin{aligned}
\log\frac{\nu_{nk}}{1-\nu_{nk}} &= \sum_{i=1}^{k}\left(\psi(\tau_{i1})-\psi(\tau_{i1}+\tau_{i2})\right)-\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{i=1}^{k}v_i\right)\right] \\
&\quad -\frac{1}{2\sigma_n^2}\left(tr(\boldsymbol{\Phi}_k)+\bar{\boldsymbol{\phi}}_k\bar{\boldsymbol{\phi}}_k^T\right)+\frac{1}{\sigma_n^2}\bar{\boldsymbol{\phi}}_k\left(X_{n\cdot}^T-\left(\sum_{l:l\neq k}\nu_{nl}\bar{\boldsymbol{\phi}}_l^T\right)\right) \\
&\equiv \vartheta,
\end{aligned}$$

where the remaining expectation can be computed using either the multinomial approximation or the Taylor series (see chapter 6). This gives us the update

$$\nu_{nk} = \frac{1}{1+e^{-\vartheta}}.$$

3. The updates for $\boldsymbol{\tau}$ depend on how we deal with the term $\mathbb{E}_{\boldsymbol{v}}\left[\log\left(1-\prod_{m=1}^{k}v_m\right)\right]$; these are described in detail in chapter 6.

Appendix C

Structured Nonparametric Latent Feature Models

The IBP is the most basic nonparametric latent feature model: observations are exchangeable, and features are independent. Recent years have seen many extensions to create more structured nonparametric latent feature models. One form of structure, discussed in chapter 7, involves incorporating correlations between the features. Other extensions have considered how to adjust the sparsity of the IBP and how to incorporate various forms of correlations between the observations.

C.1 Adjusting Sparsity Properties

In the standard construction of the IBP, the concentration parameter α governs both the number of features that we expect to see in each observation (α) and the expected total number of features ($\alpha \log(N)$). Ghahramani et al. (2007) derive a two-parameter extension to introduce additional flexibility. The expected number of features per observation remains α, however, a new 'stickiness' parameter β describes to what extent features are likely to be shared across observations (and thus governs the total number of features we expect to see).

More formally, in the buffet construction for the generative process, customer n serves himself dish k with probability

$$p(Z_{nk} = 1|Z_{1:n-1,k}) \propto \frac{m_k}{\beta + n - 1}.$$

where m_k is the number of customers who have previously tried dish k. Note that for $\beta = 1$, this probability is equivalent to the one-parameter IBP, but larger values of β will cause customers to choose a broader range of dishes.[1] The expected total number of dishes is given by

$$E[K|N] = \alpha \sum_{n=1}^{N} \frac{\beta}{\beta + n - 1},$$

and the probability of a particular left-ordered matrix is given by

$$P([Z]) = \frac{(\alpha\beta)^{K_+}}{\prod_{h=1}^{2^N-1} K_h!} \exp\{-\alpha \sum_{n=1}^{N} \frac{\beta}{\beta + n - 1}\} \prod_{k=1}^{K_+} B(b_k, N - b_k + \beta),$$

where $B(\cdot, \cdot)$ is the Beta-function. Observations in the two-parameter IBP are still exchangeable, and features remain independent. Finally, very recent work Teh (2009) has investigated a three-parameters extension of the IBP to allow for power-law behaviour in the feature popularities (both the one and two parameter constructions have an exponential drop-off in feature popularities).

C.2 Temporal Correlations

When observations come from a time-sequence, it is natural to expect that features present at time n will influence the features present at the next time $n + 1$. The Markov IBP (mIBP) or infinite factorial HMM (Van Gael et al., 2009) introduces the simplest kind of temporal correlation in which features present at time n are likely to persist to time $n + 1$. The infinite latent events

[1] A stick-breaking construction for the two parameter IBP has also been derived (John Paisley, personal communication June 2009), based on the Beta-Bernoulli process interpretation of the IBP (recall from chapter 2 that the Beta process had an additional parameter c that was set to 1 to produce the IBP).

model (ILEM) of Wingate et al. (2009) is a more general model that allows the presence of a feature at time $n+1$ to depend on all the features present at time n.

The mIBP model has separate parameters p_{k1} and p_{k0} for each feature, where p_{k1} is the probability $p(Z(n,k) = 1|Z(n-1,k) = 1)$, and p_{k0} is the probability $p(Z(n,k) = 1|Z(n-1,k) = 0)$. While p_{k1} can be sampled from an arbitrary $\mathsf{Beta}(\gamma, \delta)$ distribution, p_{k0} must be generated from $\mathsf{Beta}(\alpha/k, 1)$ to ensure that, for large k, an inactive feature is unlikely to become active. The process is initialised with $Z(0,k) = 0$ for all k.

The ILEM is more general model that takes inspiration from the HDP-HMM (Teh et al., 2006). While the details are somewhat involved, the core of the generative model is keeping track of C_{ij}, the number of times feature i caused feature j, and B_j, the number of times any feature caused feature j. To determine what features will be active at time $n+1$, we loop through each feature i active at time n and apply the following procedure:

- Sample $N \sim \mathsf{Poisson}(\lambda)$. This is the number of 'points' feature i has to cause events.
- Sample N times from a Chinese restaurant process in which feature j is chosen with probability $\frac{C_{ij}}{C_i+\alpha_T}$ and 'new' is chosen with probability $\frac{\alpha_T}{C_i+\alpha_T}$. Features j chosen by feature i will be active at time $n+1$.
- For every instance 'new' in the previous sampling step, choose a previously initialised feature j with probability $\frac{B_j}{B+\alpha_B}$. With probability $\frac{\alpha_B}{B+\alpha_B}$ choose a feature j that has never been active before. All of these features j become active at time $n+1$.

In this way, the ILEM encourages features to activate features that it has activated in the past and, if it chooses to activate something new, activate features that other features have activated in the past.

C.3 Correlated Observations

Temporal correlations are only one form of correlations one might expect between observations. For example, if the observations represent user preferences, prior knowledge about types of users may suggest groups of people whose preferences may have tighter correlations. The Phylogenetic Indian Buffet Process (pIBP) (Miller et al., 2008b) and its variants (Miller et al., 2008a) use a tree structure to describe correlations between observations, where nodes that are closer together in the tree are more likely to share active features.

As with the ILEM, the generative process is somewhat involved, but the core idea is that a customer chooses dishes not based simply on their popularity, but based on their popularity among customers that are similar to him. When sampling a new dish, a customer leaves an annotation so that other customers know from where in the tree the dish was created. Customers who share tight similarities with other customers are likely to sample what their friends sampled and rarely propose new features; in contrast, customers who are highly dissimilar to their neighbours are likely to propose many new features. In this way, all customers still have α dishes in expectation, but the rare or unpopular dishes are spread mostly among the customers who are most unlike the others. With a proper choice of parameters, the process remains exchangeable among families (observations that split from the same level of the tree).

C.4 Correlated Features

The previous sections described how to introduce correlations between the observations. However, in many applications, we may also expect features to be correlated: for example, notes might often occur together in chords; sets of objects may tend to cooccur in a scene. Here we review two works (also referenced in chapter 7) that introduce correlations among the features. Infinite hierarchical factorial regression (Rai and Daume, 2009) uses an IBP to model what features are present in an observation and a Kingman's coalescent to model correlations. The infinite factor model hierarchy (IFMH) (Courville,

2009) is similar to the IBP-IBP model in chapter 7, creating correlations in the feature assignments through a matrix product. Finally, recent work has considered unbounded levels of hierarchies, allowing for very deep structured correlations (Adams, 2009).

One of the earliest steps to correlated nonparametric feature models, the IHFR models correlations using a two part process. First, a standard IBP is used to determine what features are active in which observations. Given a sampled feature assignment matrix Z with K active features, a second N by K matrix V is fitted using a Kingman's coalescent to model correlations between the K columns. For example, if two features i and j are tightly correlated, then V_{ni} and V_{nj} will be tightly correlated. The matrices Z and V are multiplied element-wise to create the final feature-assignment loadings; the data is modelled as $X = (Z \odot V)A$. Inference in IHFR is efficient, leading to good performance on several real datasets. However, since the coalescent on V is defined only after fixing the number of active features K, IHFR is not fully nonparametric; we cannot think of observations as a few nodes from some infinite model.

Unlike IHFR, the IFMH is a fully nonparametric model. In the language of the correlated nonparametric latent feature framework in chapter 7, observations in the IFMH choose a set of categories C based on the IBP. Next, a connection vector M is sampled as follows: first, a set of IBP stick-breaking probabilities π_k are drawn, and from these a random variable M_k is drawn from $\mathsf{Beta}(c\pi_k, c(1-\pi_k)+1)$ (in contrast, the IBP-IBP model of chapter 7 draws M_k from $\mathsf{Bernoulli}(\pi_k)$). The features Z_{nk} are drawn from a noisy-or: $\mathsf{Bernoulli}(1 - \prod_j (1 - C_{nj}M_k)$. Since the M_k gets very small for large k, there are still only a finite number of $Z_{nk} = 1$ for fixed N. One of the key benefits of the IFMH over the IBP-IBP is that M_k is continuous-valued, rather than binary, leading to better behaved inference.

Bibliography

Ryan Adams. personal communication, June 2009.

David Andrzejewski, Anne Mulhern, Ben Liblit, and Xiaojin Zhu. Statistical debugging using latent topic models. 2007.

D. Applebaum. *Lévy Processes and Stochastic Calculus*. Cambridge University Press, 2004.

Arthur Asuncion, Padhraic Smyth, and Max Welling. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems 21*, 2008.

M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, UCL, 2003.

Yoshua Bengio. Learning deep architectures for ai. In *Dept. IRO, Universite de Montreal*, number 1312. 2007a.

Yoshua Bengio. Learning deep architectures for ai. Technical report, Dept. IRO, Universite de Montreal, 2007b.

D. Blei and M. Jordan. Variational methods for the Dirichlet process. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

D. Blei and J. Lafferty. Correlated topic models. In *Advances in Neural Information Processing Systems*, 2006.

S. P. Brooks and G. O. Roberts. Convergence assessment techniques for Markov Chain Monte Carlo. *Statistics and Computing*, 8:319–335, 1998.

Carlos M. Carvalho, Jeffrey Chang, Joseph E. Lucas, Joseph R. Nevins, Quanli Wang, and Mike West. High-dimensional sparse factor modelling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484), 2008.

C.T. Chu, S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems*, page 281. MIT Press, 2007.

Wei Chu, Zoubin Ghahramani, Roland Krause, and David L. Wild. Identifying protein complexes in high-throughput protein interaction screens using an infinite latent feature model. In *Pacific Symposium on Biocomputing*, pages 231–242, 2006.

Aaron Courville. personal communication, June 2009.

Aaron Courville. The hierarchical indian buffet process. Nonparametric Bayes Workshop at ICML/UAI/COLT, 2008.

Desai, Solande, Reeve, and Vanneman. India human development survey. In *United Nations Development Programme*, number ICPSR 22626. 2005.

F. Doshi and J. Van Gael. Nonparametric bayesian methods for finding software bugs. In *CRISM Workshop for High Dimensional Data*, 2008.

F. Doshi-Velez and Z. Ghahramani. Accelerated inference for the Indian buffet process. In *International Conference on Machine Learning*, 2009.

F. Doshi-Velez, K. T. Miller, J. Van Gael, and Y. W. Teh. Variational inference for the indian buffet process. In *Proc. of the Conference on Artificial Intelligence and Statistics*, 2009.

Finale Doshi-Velez and Zoubin Ghahramni. Correlated nonparametric latent feature models. In *Conference on Uncertainty in Artificial Intelligence*, 2009.

David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives, 2005.

Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

J. Van Gael, Y. W. Teh, and Z. Ghahramani. The infinite factorial hidden markov model. Nonparametric Bayes Workshop at ICML/UAI/COLT, 2008.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2003.

A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

Z. Ghahramani, T. Griffiths, and P. Sollich. Bayesian nonparametric latent feature models. In *Bayesian Statistics*, volume 8, 2007.

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4 (2), 2001.

Dilan Görür, Frank Jäkel, and Carl Edward Rasmussen. A choice model with infinitely many latent features. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 361–368, New York, NY, USA, 2006. ACM.

Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

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

G. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. In *Neural Computation*. 2006a.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comp.*, 18(7):1527–1554, July 2006b.

Ulrich Hoffmann, Jean-Marc Vesin, Touradj Ebrahimi, and Karin Diserens. An efficient P300-based brain-computer interface for disabled subjects. *Journal of Neuroscience Methods*, 167(1):115–125, 2008.

Hemant Ishwaran and Lancelot F. James. Gibbs sampling methods for stick breaking priors. *Journal of the American Statistical Association*, 96(453): 161–173, 2001.

Sonia Jain and Radford M. Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182, 2000.

I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.

Carlos Guestrin Joseph Gonzalez, Yucheng Low. Residual splash for optimally parallelizing belief propagation. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 177–184. JMLR, 2009.

D. Knowles and Z. Ghahramani. Infinite Sparse Factor Analysis and Infinite Independent Components Analysis. *Lecture Notes in Computer Science*, 4666:381, 2007.

Tom Kollar. personal communication, 2008.

Wei Li, David Blei, and Andrew McCallum. Nonparametric bayes pachinko allocation. In *UAI 07*, 2007.

Aleix M. Mart'inez and Avinash C. Kak. PCA versus lDA. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23:228–233, 2001.

Edward Meeds, Zoubin Ghahramani, Radford M. Neal, and Sam T. Roweis. Modeling dyadic data with binary latent factors. In *Advances in Neural Information Processing Systems 19*. 2007.

K. Miller, T. Griffiths, and M. Jordan. Variations on non-exchangeable nonparametric priors for latent feature models. Nonparametric Bayes Workshop at ICML/UAI/COLT, 2008a.

K. Miller, T. Griffiths, and M. Jordan. The phylogenetic indian buffet process: A non-exchangeable nonparametric prior for latent features. UAI, 2008b.

Ramesh Nallapati, William Cohen, and John Lafferty. Parallelized variational EM for Latent Dirichlet Allocation: An experimental evaluation of speed and scalability. In *ICDMW '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 349–354, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3033-8. doi: http://dx.doi.org/10.1109/ICDMW.2007.70.

R. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

Radford Neal. Connectionist learning of belief networks. In *Journal of Artificial Intelligence*, volume 56. 1992.

Radford M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, 1993.

Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–741, 2003.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed inference for Latent Dirichlet Allocation. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1081–1088. MIT Press, Cambridge, MA, 2008.

John Paisley and Lawrence Carin. Nonparametric factor analysis with beta process priors. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 777–784, New York, NY, USA, 2009. ACM.

CORPORATE PDP Research Group. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. MIT Press, Cambridge, MA, USA, 1986.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, 1988.

Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP J. Appl. Signal Process.*, 2007(1): 154–154, 2007.

Piyush Rai and Hal Daume. The infinite hierarchical factor regression model. In *NIPS*, 2009.

C. R. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, second edition, 2004.

R. N. Shepard and P. Arabie. Additive clustering - representation of similarities as combinations of discrete overlapping properties. *Psychol. Rev.*, 86(2):87–123, 1979.

Charles Spearman. "general intelligence," objectively determined and measured. *American Journal of Psychology*, 15:201–293, 1904.

David Stern, Ralf Herbrich, and Thore Graepel. Matchbox: Large scale online Bayesian recommendations. In *18th International World Wide Web Conference (WWW2009)*, April 2009.

Y. W. Teh, D. Gorur, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *Proceedings of the 11th Conference on Artificial Intelligence and Statistic*, 2007.

Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476), December 2006.

Yee Whye Teh. personal communication, August 2009.

R. Thibaux and M. Jordan. Hierarchical beta processes and the indian buffet process. AISTATS, 2007.

M. Titsias. The infinite gamma-poisson feature model. In *Advances in Neural Information Processing Systems 19*. 2007.

Michalis Titsias. The infinite gamma-poisson feature model. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1513–1520. MIT Press, Cambridge, MA, 2008.

Amos Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79(4):281–299, 1972.

N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. 2003.

UN. Human development report. In *United Nations Development Programme*. 2008.

Jurgen Van Gael, Yee W. Teh, and Zoubin Ghahramani. The infinite factorial hidden markov model. In *Advances in Neural Information Processing Systems 21*. 2009.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1 (1-2):1–305, 2008.

Max Welling and Kenichi Kurihara. Bayesian K-means as a "maximization-expectation" algorithm. In *Sixth SIAM International Conference on Data Mining*, 2006.

David Wingate, Noah D. Goodman, Daniel M. Roy, and Joshua B. Tenenbaum. The infinite latent events model. In *Conference on Uncertainty in Artificial Intelligence*, 2009.

John Winn and Christopher M. Bishop. Variational message passing. *J. Mach. Learn. Res.*, 6:661–694, 2005. ISSN 1533-7928.

Frank Wood and Thomas L. Griffiths. Particle filtering for nonparametric Bayesian matrix factorization. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

Frank Wood, Zoubin Ghahramani, and Tom Griffiths. A non-parametric bayesian method for inferring hidden causes. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI*, pages 536–543, 2006.

Richard S. Zemel and Geoffrey E. Hinton. Developing population codes by minimizing description length. *Neural Computation*, 7:11–18, 1994.