The report committee for Michael Minyi Zhang certifies that this is the approved version of the following report:

Distributed inference in Bayesian nonparametric models using partially collapsed MCMC

APPROVED BY

SUPERVISING COMMITTEE:

Sinead Williamson, Supervisor

Lizhen Lin

Distributed inference in Bayesian nonparametric models using partially collapsed MCMC

by

Michael Minyi Zhang, B.S.; B.A.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN STATISTICS

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2016

Distributed inference in Bayesian nonparametric models using partially collapsed MCMC

Michael Minyi Zhang, M.S.Stat.
The University of Texas at Austin, 2016

Supervisor: Sinead Williamson

Bayesian nonparametric based models are an elegant way for discovering underlying latent features within a data set, but inference in such models can be slow. Inferring latent components using Markov chain Monte Carlo either relies on an uncollapsed representation, which leads to poor mixing, or on a collapsed representation, which is usually slow. We take advantage of the fact that the latent components are conditionally independent under the given stochastic process (we apply our technique to the Dirichlet process and the Indian buffet process).

Because of this conditional independence, we can partition the latent components into two parts: one part containing only the finitely many instantiated components and the other part containing the infinite tail of uninstantiated components. For the finite partition, parallel inference is simple given the instantiation of components. But for the infinite tail, performing

uncollapsed MCMC leads to poor mixing and thus we collapse out the components. The resulting hybrid sampler, while being parallel, produces samples asymptotically from the true posterior.

Table of Contents

List of Figures

Chapter 1

Introduction

Bayesian nonparametric (BNP) models are a flexible class of models whose complexity adapts to the data under consideration. Typical situations where a nonparametric solution is useful are ones where the complexity of the model must be specified in advance, like in mixture models or in latent feature models. Since, in most real situations, we usually do not know the true number of mixture components or latent features in advance we would then have to fit competing models and select the best model by some model selection metric.

In contrast, BNP models place priors on infinite-dimensional objects, such as partitions with infinitely many blocks; matrices with infinitely many columns; or discrete measures with infinitely many atoms. A finite set of observations is assumed to be generated from a finite—but random—subset of these components, allowing flexibility in the underlying dimensionality and providing the ability to incorporate previously unseen properties as our dataset grows.

While the flexibility and expandability of these models are a good fit for large, complex data sets, distributing existing inference algorithms across multiple machines is challenging. If we explicitly represent subsets of the

underlying infinite-dimensional object – for example using a slice sampler – we can face poor mixing and high memory requirements. Conversely, if we integrate out the infinite-dimensional object, we run into problems due to induced global dependencies.

In this paper, we focus on two popularly used nonparametric Bayesian methods, the Dirichlet process and the Indian buffet process. We choose these because they are the most commonly used nonparametric priors for mixture models and latent feature models, respectively. However, they are both members of a larger class of models based on a Completely Random Measure framework, and the techniques in this paper can easily be extended to apply to several other members of this class.

In the following section we begin by introducing these two distributions, using the common framework of completely random measures, and describe the most techniques for performing approximate inference. We then discuss existing approaches for distributed inference in these models.

Chapter 2

Background and Related Work

Many commonly used nonparametric priors, including the Dirichlet process and the Indian buffet process, can be expressed in terms of a class of distributions known as completely random measures (CRM) [10]. A completely random measure is a random measure μ on some space Ω where the masses $\mu(A_i), \mu(A_j)$ assigned to disjoint subsets $A_i, A_j \subset \Omega$ are independent.

A commonly used example of a completely random measure is the gamma process. If H is some probability measure on Ω, and $\alpha > 0$, then the gamma process assigns a Gamma$(\alpha H(A))$ mass to any subset $A \subset \Omega$. The Dirichlet process is a distribution over random probability measure $D \sim \mathrm{DP}(\alpha, H)$ on Ω that corresponds to the normalization of a gamma process with parameters α and H. Since the gamma process assigns independent gamma-distributed masses to disjoint subsets of Ω, the masses assigned by a Dirichlet process to a finite partition of Ω are necessarily Dirichlet distributed, i.e. if $A_1, \ldots, A_K$ is a partition of Ω, then $(D(A_1), \ldots, D(A_k)) \sim \mathrm{Dirichlet}(\alpha H(A_1), \ldots, \alpha H(A_K))$.

We can use a Dirichlet process-distributed random measure

$$D := \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k} \tag{2.1}$$

to assign parameters to observations:

$$\begin{aligned} D \sim &\mathrm{DP}(\alpha, H) \\ \phi_n \sim &D \end{aligned} \tag{2.2}$$

The discrete nature of D means that a single atom $\pi_k \delta_{\theta_k}$ can be associated with multiple observations, so we will get repeated values of ϕ_n, so that we are partitioning observations into an unbounded number of clusters. If we let z_n be the cluster indicator for the nth data point, we can directly obtain the predictive distribution over $z_{n+1}|z_{1:n}$:

$$P(z_{n+1} = k | z_{1:n}, \alpha) = \begin{cases} \frac{m_k}{n+\alpha} & \text{for } k \leq K \\ \frac{\alpha}{n+\alpha} & \text{for } k = K+1 \end{cases} \tag{2.3}$$

where K is the number of clusters present in the first n observations, and m_k is the number of observations in the kth cluster. To recover a full mixture model from Equation 2.3, we can sample a cluster parameter $\theta_k \sim H$ for each cluster and set $\phi_n := \theta_{z_n}$.

Another commonly used CRM is the beta process [6, 15]. The (homogenous) beta process is a completely random measure where the distribution over atom sizes is given by the limit, as $K \to \infty$, of a Beta $\left(c\frac{\alpha}{K}, c\left(1 - \frac{\alpha}{K}\right)\right)$ distribution, and whose atom locations are i.i.d. according to some probability measure H. The beta process can be used as a prior for latent feature models. If $B := \sum_{k=1}^{\infty} \mu_k \delta_{\theta_k} \sim \mathrm{BP}(\alpha, c, H)$, then we can interpret the μ_k as the probability of an observation containing the kth latent feature θ_k. We can select a finite subset of these latent features for the nth data point by sampling a sequence Z_n of binary indicator variables $z_{n,k} \sim \mathrm{Bernoulli}(\mu_k)$.

As with the Dirichlet process, it is possible to integrate out the latent measure and work directly with the predictive distribution over the binary sequences. If K is the number of features seen in the first $n-1$ data points, then we can sample the next sequence $Z_n = (z_{n,k})_{k=1}^{\infty}$ as

$$\begin{aligned} z_{n,k} &\sim \text{Bernoulli}\left(\frac{m_k}{n}\right), \qquad k = 1, \ldots, K \\ J &\sim \text{Poisson}\left(\frac{\alpha}{n}\right) \\ z_{n,j} &= 1, \qquad j = K+1, \ldots, K+J \\ z_{n,j} &= 0, \qquad j > K+J \\ K &\leftarrow K + J \end{aligned} \tag{2.4}$$

If we stack the sequences Z_n into a matrix $\mathbf{Z}$, Equation 2.4 describes a distribution over binary matrices known as the Indian buffet process [4].

2.1 Inference approaches

In Section 2, we saw two complementary representations for the cluster/latent variable allocations obtained using a Dirichlet process and an Indian buffet process. We can either explicitly instantiate the latent measure and sample each data point's allocations independently given this measure, or we can integrate out the latent measure and work directly with the predictive distribution.

When designing an MCMC algorithm, these two options lead directly to two different inference approaches, which we will refer to as *collapsed* and *uncollapsed* samplers. In an uncollapsed sampler, we alternate between sam-

pling the random measure given the cluster/latent feature allocations, and sampling the allocations given the random measure. To avoid having to instantiate the infinitely many atoms of the random measure, we can either replace the random measure with a finite-dimensional approximation (see [8] for the Dirichlet process or [12, 18] for the Indian buffet process), or we can construct a slice sampler that employs a random truncation that maintains the correct posterior distribution (see [16, 3] for the Dirichlet process or [14] for the Indian buffet process).

A collapsed sampler (see [7, 11] for the Dirichlet process and [4] for the Indian buffet process) integrates out the random measure, and works only with the cluster allocations (Dirichlet process) or latent feature allocations (Indian buffet process). Since we only observe a finite number of clusters or latent features, we do not need to introduce approximations or slice variables. We make use of the fact that the sequences obtained by integrating out the random variables are exchangeable, meaning we can adapt the predictive distributions in Equations 2.3 and 2.4 to give a conditional distribution $P(Z_n|Z_{-n})$. This can be combined with a likelihood term $P(X_n|Z_n, \Theta)$ to give the full conditional distribution, which can in term be used to construct a Gibbs sampler.

2.2 Distributed inference methods

A number of parallel inference algorithms have been proposed for the Dirichlet process and its variants. [13] proposed an approximate distributed scheme for the hierarchical Dirichlet process, which is easily adaptable to the

Dirichlet process mixture model. On each local step, each node assumes that the sufficient statistics from the other nodes are unchanged from the previous global step. On global steps, the sufficient statistics are updated and new clusters are (approximately) aligned. As [17] show, this approach works well where the clusters are large, but when working with small clusters suffers from alignment issues, due to both problems matching up new clusters, and the possibility of small clusters drifting in location over the course of a local iteration.

[17] proposed a distributed method based on partitioning the Dirichlet process into a mixture of conditionally independent Dirichlet processes. These conditionally independent Dirichlet processes are updated independently on separate processors during local steps. Global moves are used to move data between the conditionally independent Dirichlet processes to ensure correct mixing. While this approach works well for relatively small numbers of processors, its scalability is limited by the fact that each cluster in the overall DPMM must reside on a single processor (as discussed by [17] and [2]). Further, the approach assumes a shared-memory architecture where there is minimal cost to moving data between processors; in a distributed-memory context this would cause significant communication bottlenecks.

A more recent approach, that is explicitly designed for the distributed-memory, low-communication setting, was proposed by [3]. Unlike the previously described approaches, this paper uses an uncollapsed approach, explicitly instantiating the cluster probabilities and parameters. A slice sampler is used

to ensure only a finite number of atoms need to be represented. When new atoms are introduced by the slice sampler, a shared seed is used in the pseudo-random number generators on each node, to ensure that the same atoms are proposed without the need for direct communication.

The downside of this approach is that, by necessity, the new atoms are sampled from the prior, rather than from their conditional distribution given observations. In high dimensions, this means that the proposed atoms are likely to be very far from data, and will therefore tend not to be used, resulting in slow mixing.

Compared with the Dirichlet process, there has been relatively little work on distributed inference for the Indian buffet process. The main contribution in this area is by [1], who deploy an approach similar to that of [13]. Each processor approximates the current feature counts from the other processors with the counts from the previous time step; there is no explicit merging of new features. As we will see in Section 3, this approach will lead to over-estimation of the number of new features, and poor mixing.

2.3 Hybrid collapsed/uncollapsed algorithms for distributed inference

A key goal of a distributed algorithm is to minimize communication between agents. This can be achieved by breaking the algorithm into independent sub-algorithms, which can be run independently on different agents. In practice, we usually cannot split an MCMC sampler on a Bayesian hierarchi-

cal model into entirely independent sub-algorithms, since there are typically some global dependencies implied by the hierarchy. We can either replace these global dependencies with appropriate approximations, or we can make use of conditional independencies to temporarily partition our algorithm. In this paper, we describe methods that take the latter approach.

In Section 2.1, we considered two inference paradigms: collapsed and uncollapsed samplers. Both these approaches lead to difficulties when attempting to parallelize inference. In the collapsed setting, we face the problem that, since the cluster probabilities and parameters are marginalized out, the probability of the nth data point belonging to the kth cluster depends on the cluster allocations of all other data points. In particular, we need up-to-date knowledge of the total number of data points in the kth cluster, and the sufficient statistics associated with that cluster's distribution. If the kth cluster is instantiated on multiple machines, maintaining these statistics requires frequent global communication. Algorithms have been proposed that ensure all data points in a given cluster are associated with the same processor [17]; however this can lead to bottlenecks and limited scalability due to large clusters. Further, we replace near-constant communication about sufficient statistics with less frequent, yet bandwidth-heavy, reallocation of datapoints to different processors. Another approach is to approximate the true counts with "stale" values, in effect assuming counts on other agents have not changed [1]; however this introduces errors and suffers from alignment issues, particularly in the nonparametric section where the number of clusters changes from iteration

to iteration [17].

On the surface, uncollapsed samplers are much more suited to the distributed setting. We can make use of the fact that, conditioned on the latent measure, the cluster/latent feature allocations are conditionally independent. This means that if we split our data between the available agents and send a copy of our latent measure to all these agents, then the agents can independently sample the allocations for their subset of the data. Global communication is then required to sample from the conditional distribution of the random measure given the allocations. Parallelization in an uncollapsed representation for the Dirichlet process has been proposed by [3].

When working with an uncollapsed representation, we need a way of introducing new features. One option is to use a random slice variable, and sample a set of a atoms that are above that slice [16, 3, 9]. Another option is to combine the probabilities for all the uninstantiated clusters into one, and sample a set of auxiliary variables from the prior that act as proposal locations for new clusters [11]. The performance of such algorithms will depend on how close the proposed new atoms or auxiliary variables are to the true cluster parameters. For a low-dimensional parameter space, we are likely to have reasonably good proposals; however as the dimensionality increases our proposals are unlikely to be near the data.

In this section, we propose a hybrid approach that offers the advantages of both a collapsed and an uncollapsed representation. We are able to do this because of the complete randomness of the underlying random measure (beta

process in the case of the Indian buffet process; gamma process in the case of the Dirichlet process). This allows us to split the prior into two independent (in the case of the IBP) or conditionally independent (in the case of the DP) random measures. We choose to split the prior into a finite-dimensional measure corresponding to the currently observed clusters/features, and an infinite-dimensional "tail". We use uncollapsed inference on the finite measure, allowing straightforward parallelization but avoiding ever needing to expand our representation. For the infinite tail, we use a collapsed representation that allows us to efficiently introduce new features even in a high-dimensional setting. We present a hybrid sampler for the Dirichlet process in Section 2.3.1, and for the Indian buffet process in Section 2.3.2.

2.3.1 Distributed hybrid inference in the Dirichlet process

Assume we wish to perform inference in a Dirichlet process mixture model with some arbitrary mixture kernel $f(\theta)$ parametrized by $\theta \in \Omega$. We can write this model as

$$D := \sum_k \pi_k \delta_{\theta_k} \sim \mathrm{DP}(\alpha, H), \qquad \phi_n \sim D, \qquad x_n \sim f(\phi_n). \tag{2.5}$$

Let A be some subset of Ω (where A may have measure zero) and let A^c be its complement in Ω. We can represent this Dirichlet process as the weighted superposition of two independent Dirichlet processes, one on A and one on A^c. Concretely, if $H|_A$ is the restriction of H to A, i.e.

$$H|_A(d\theta) = \begin{cases} H(d\theta) & \theta \in A \\ 0 & \theta \notin A \end{cases} \tag{2.6}$$

then we can represent the $\mathrm{DP}(\alpha, H)$ mixture model described in Equation 2.5 as

$$\begin{aligned} B \sim &\mathrm{Beta}(\alpha H(A), \alpha H(A^c)) & \\ G_1 \sim &\mathrm{DP}(\alpha H|_A) & \phi_n \sim &BG_1 + (1-B)G_2 \\ G_2 \sim &\mathrm{DP}(\alpha H|_{A^c}) & x_n \sim &f(\phi_n) \end{aligned} \tag{2.7}$$

Conditioned on the current cluster counts m_k, the posterior distribution over D is given by

$$D|m_1, \ldots, m_K \sim \mathrm{DP}\left(\alpha + N, \frac{\alpha H + \sum_k m_k \delta_{\theta_k}}{\alpha + N}\right) \tag{2.8}$$

Following from Equation 2.9, we can re-write this as

$$\begin{aligned} B \sim &\mathrm{Beta}(N, \alpha) & & \\ G_1 = \sum_{k=1}^{K} \pi_k \delta_{\theta_k} \sim &\mathrm{DP}\left(N, \frac{\sum_k m_k \delta_{\theta_k}}{N}\right) & \phi_n \sim &BG_1 + (1-B)G_2 \\ G_2 = \sum_{k=K+1}^{\infty} \pi_k \delta_{\theta_k} \sim &\mathrm{DP}(\alpha, H) & x_n \sim &f(\phi_n) \end{aligned} \tag{2.9}$$

We note that $G_1 = \sum_{k=1}^{K} \pi_k \delta_{\theta_k}$, where K is the number of currently occupied clusters, and $(\pi_1, \ldots, \pi_K) \sim \mathrm{Dirichlet}(m_1, \ldots, m_K)$. We have partitioned our Dirichlet process into a weighted combination of a finite-dimensional Dirichlet distribution, with elements corresponding to the currently occupied

clusters, and an infinite-dimensional Dirichlet process, with atoms corresponding to the currently unoccupied clusters.

We can now instantiate the finite measure G_1 on all processors, and integrate out the infinite dimensional tail. We randomly select one out of P processors, by sampling $P^* \sim \text{Uniform}(1, \ldots, P)$. For every other processor, i.e. for each processor $j \neq P^*$, we perform restricted Gibbs sampling [11], only allowing observations to choose between the K clusters in G_1.

On processor P^*, we allow observations to pick a cluster from G_1 with probability B, or a cluster from G_2 with probability $1 - B$. Concretely, the probability a data point x_n on cluster P^* being assigned to cluster k, conditioned on B, G_1 and the other data points on processor P^*, is given by

$$P(\phi_n = \theta_k | -) \propto \begin{cases} B\pi_k f(x_n; \theta_k) & k \leq K \\ \frac{(1-B)m_k}{N} f(x_n; \{x_i : \phi_i = \theta_k, i \neq n\}) & K < k \leq K + J \\ \frac{(1-B)\alpha}{N} & k = K + J + 1 \end{cases} \tag{2.10}$$

where J is the number of atoms in G_2 which are associated with data. Note that the only data points that can be associated with atoms in G_2 are those on processor P^*, so we can evaluate $\frac{m_k}{N} f(x_n; \{x_i : \phi_i = \theta_k, i \neq n\})$ without any knowledge about the other processors.

At each global step, we gather the sufficient statistics from all instantiated clusters – from both G_1 and G_2 – and sample parameters for those clusters. We then create a new partition, redefining the support of G_1 as the

set of instantiated cluster parameters, and resample $B \sim \text{Beta}(N, \alpha)$.

While asymptotically correct, an unfortunate consequence of this sampler is that it is slow to instantiate new clusters. With only $1/P$ of the data points eligible to start a new cluster, the rate at which new clusters are added will decrease with the number of processors. While this is of less concern once we have converged to an appropriate number of clusters, it can lead to slow convergence if we start with too few clusters. To avoid this problem, we initialize our algorithm by allowing *all* processors to instantiate new clusters. At each global step, we decrease the number of randomly selected processors eligible to instantiate new clusters, until we end up with a single processor. This warm start allows us to expand our initial state space with data-driven cluster proposals.

We note that a sampler with multiple processors instantiating new clusters will not converge to the true posterior; instead it will tend to over-estimate the number of clusters. However, the procedure acts in a manner similar to simulated annealing, by encouraging large moves early in the algorithm but gradually decreasing the excess stochasticity until we are sampling from the correct algorithm.

2.3.2 Distributed hybrid inference in the Indian buffet process

If $B \sim \text{BP}(\alpha, c, H)$ and $Z_n \sim \text{BeP}(B)$, then the posterior distribution over B is given by

$$B|Z_1,\ldots,Z_n \sim \mathrm{BP}\left(\frac{c\alpha+\sum_k m_k}{c+n}, c+n, \frac{c\alpha H+\sum_k m_k\delta_{\theta_k}}{c\alpha+\sum_k m_k}\right) \tag{2.11}$$

Since the beta process is a completely random measure, we can partition this into the superposition of two independent completely random measures, so that

$$\begin{aligned} B_1 &:= \sum_{k=1}^{K}\mu_k\delta_{\theta_k} \sim \mathrm{BP}\left(\frac{\sum_k m_k}{c+n}, c+n, \frac{\sum_k m_k\delta_{\theta_k}}{\sum_k m_k}\right) \\ B_2 &:= \sum_{k=K+1}^{\infty}\mu_k\delta_{\theta_k} \sim \mathrm{BP}\left(\frac{c\alpha}{c+n}, c+n, H\right) \\ B &= B_1 + B_2 \end{aligned} \tag{2.12}$$

We note that the distribution over the atom sizes $\mu_1,\ldots,\mu_k$ of B_1 is simply a sequence of K Beta$(N-m_k, m_k+c)$ random variables. This allows us to split the IBP into two independent feature selection mechanisms: one (controlled by B_1) with a finite number of currently instantiated features, and one (controlled by B_1) with an unbounded number of currently uninstantiated features.

This observation allows us to construct a distributed MCMC sampler where, at any given time, only data on a single processor is allowed to sample from the full conditional distribution over features. On all the other processors, data points can only use features from B_1. Ergodicity is ensured by periodically re-defining B_1 to include all instantiated features, and randomly resampling the single processor that is able to instantiate new features.

Concretely, at each global iteration, we randomly select one processor with indicator $P^* \sim \text{Uniform}(1, \ldots, P)$. For every other processor, i.e. for each processor $j \neq P^*$, we perform restricted Gibbs sampling [11], only allowing data points to select subsets of the K features in B_1. The probability that $z_{nk} = 1$ for such a data point is given by:

$$P(z_{nk} = z|-) \propto \begin{cases} \mu_k f(x_n|z_{nk} = 1, z_{n,-k}, \theta_1, \ldots, \theta_k) & z = 1 \\ (1 - \mu_k) f(x_n|z_{nk} = 0, z_{n,-k}, \theta_1, \ldots, \theta_k) & z = 0. \end{cases} \tag{2.13}$$

On processor P^*, data points can select features from B_1 or B_2. Let K be the number of atoms in B_1, and let J be the number of instantiated features in B_2. The first K features are selected according to Equation **??**. If we are able to marginalize over the feature locations θ_k, the next J features are selected according to

$$P(z_{nk} = z|Z_{-nk}, x_n) \propto \begin{cases} m_k f(x_n|z_{nk} = 1, Z_{-nk}) & z = 1 \\ (N - m_k) f(x_n|z_{nk} = 0, Z_{-nk}) & z = 0. \end{cases} \tag{2.14}$$

If we are unable to marginalize over the θ_k, we can instantiate them as described in [1] and include them in the appropriate likelihood term. Finally, we propose adding Poisson(α/N) new features, using a Metropolis-Hastings step.

At each global step, we gather the sufficient statistics from each instantiated feature, and sample a feature value θ_k for each one conditioned on the data points exhibiting that feature. We redefine B_1 and B_2 so that B_1 contains all (and only) instantiated features, and sample $\mu_k \sim \text{Beta}(m_k, N - m_k + c)$

for each feature in B_1. We then sample a new processor indicator $P^* \sim \text{Uniform}(1, \ldots, P)$ and repeat.

As with the Dirichlet process sampler described in Section 2.1, the distributed sampler will be slow to add features, since to ensure correctness of the transition distribution, only one processor can add features at a time. We can dramatically improve the time to convergence by using a warm-start procedure as described in Section 2.3.1, where we initially allow all processors to instantiate new features, and gradually reduce the number of processors adding new features until we have the correct sampling distribution.

Chapter 3

Experimental evaluation

We begin by showing that, while parallelizable, an entirely uncollapsed sampler is a poor choice when dimensionality increases. We go on to compare our distributed hybrid samplers with a range of other parallel methods for the DP and the IBP.

3.1 Limitations of an entirely uncollapsed approach

In an entirely uncollapsed sampler, we must ensure that a global set of atom sizes and locations is shared across all processors. This means that we must sample new parameters from the prior. One method of doing so is obtained by modifying Algorithm 8 of [11]. Algorithm 8 describes a scheme for Gibbs sampling a Dirichlet process mixture with a non-conjugate likelihood, where we can integrate out the atom sizes but must sample the atom locations. We can modify this to give a fully uncollapsed algorithm, where both atom sizes and atom locations are instantiated.

At each global step, let K be the total number of occupied clusters, and let $\mathbf{m} = m_1, \ldots, m_K$ be the cluster counts. We can proceed as follows:

- Discard any atoms that are not associated with any data points, leaving

only K instantiated atoms.

- Sample new atom locations $\theta_1, \ldots, \theta_K$ for the K atoms, using the conditional distribution given the associated data points.

- Sample J new atom locations $\theta_{K+1}, \ldots, \theta_J$ from the base measure, where $J \geq 1$.

- Sample atom weights for all $K + J$ atoms

$$\boldsymbol{\pi} := (\pi_1, \ldots, \pi_K, \ldots, \pi_{K+J}) \sim \text{Dirichlet}(m_1, \ldots, m_k, \alpha/J, \ldots, \alpha/J)$$

- For each data point x_n, sample a cluster indicator c_n according to

$$P(c_n = k | \boldsymbol{\pi}, \boldsymbol{\theta}) \propto \pi_k f(x_n; \theta_k)$$

  where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K, \ldots, \theta_{K+J})$.

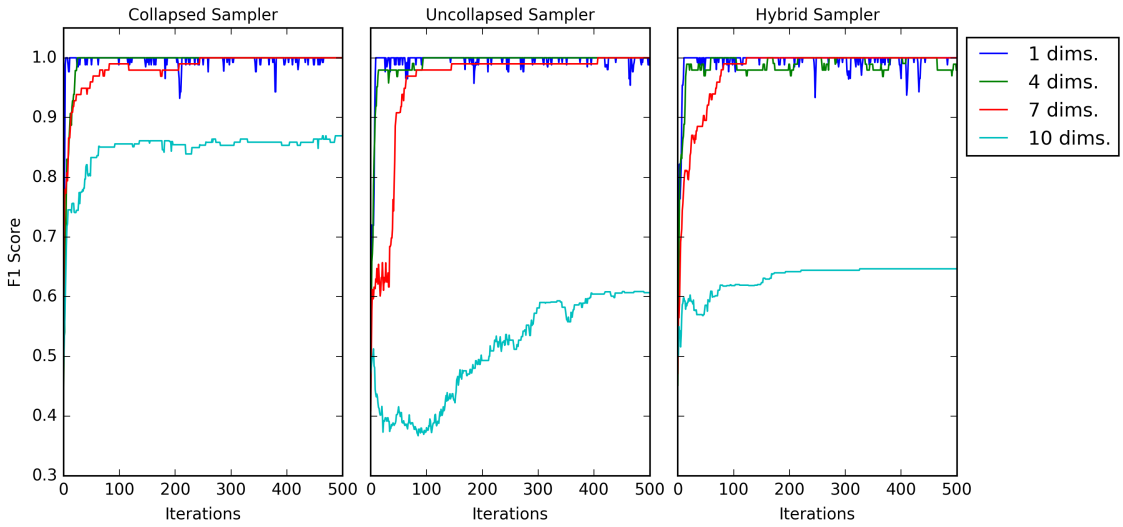
The final, time-consuming step, where we sample the cluster indicators, can be parallelized.

Unfortunately, we can run into problems when it comes to proposing new parameter values $\theta_{K+1}, \ldots, \theta_{K+J}$. In high dimensions, it is unlikely that a proposed parameter will be near our data, so the associated likelihood of any given data point will be low. This is in contrast to the collapsed setting, where we integrate over all possible locations.

Figure 3.1 shows convergence plots for three algorithms: The uncollapsed algorithm described above; a standard collapsed Gibbs sampler; and

the single-processor version of our hybrid algorithm. The data set is a D dimensional synthetic data set consisting of 100 observations of Gaussian mixtures with 2 true mixture components centered at $5 \times \{1\}^D$ and $-5 \times \{1\}^D$ with an identity covariance matrix. We can clearly see that the collapsed sampler performs better than the uncollapsed sampler for 10 dimensional data. Since the hybrid sampler only uses collapsed sampling for newly introduced features, the performance of the hybrid sampler in this situation is expected to be similar to the uncollapsed sampler although the hybrid sampler reaches its maximum F1 score faster than the uncollapsed sampler.

Figure 3.1: Comparison of F1 scores over iteration for the collapsed, uncollapsed and hybrid samplers



3.2 Experimental evaluation: Dirichlet Process

In this section, we show how our inference algorithm can speed up inference in a Dirichlet process mixture of normals. To evaluate our algorithm, we generated parameter weights given the α parameter from the stick breaking Dirichlet process prior. Then, we sample locations for the clusters from a Normal-Inverse Wishart prior and for n observations we sample a cluster indicator from the cluster weights and then sample from the cluster parameter. Our experiment shows the F1 score of test set data as the number of processors increases. As we can see in Figure 3.2, our hybrid method is capable of achieving a higher F1 score faster than the lower processor runs.

Next, we evaluate the performance of our DPMM sampler by adjusting the separability of the true cluster locations. Intuitively, we observe that our algorithm performs poorly when there is little separation between the clusters (Figure 3.3) and performs well when there is large separation between clusters (Figure 3.4).

3.3 Experimental evaluation: Indian Buffet Process

Next, we show how our inference algorithm can speed up inference in an Indian buffet process latent feature model. We use a linear Gaussian likelihood, modeling the data as

Figure 3.2: F1 score for test set synthetic data.

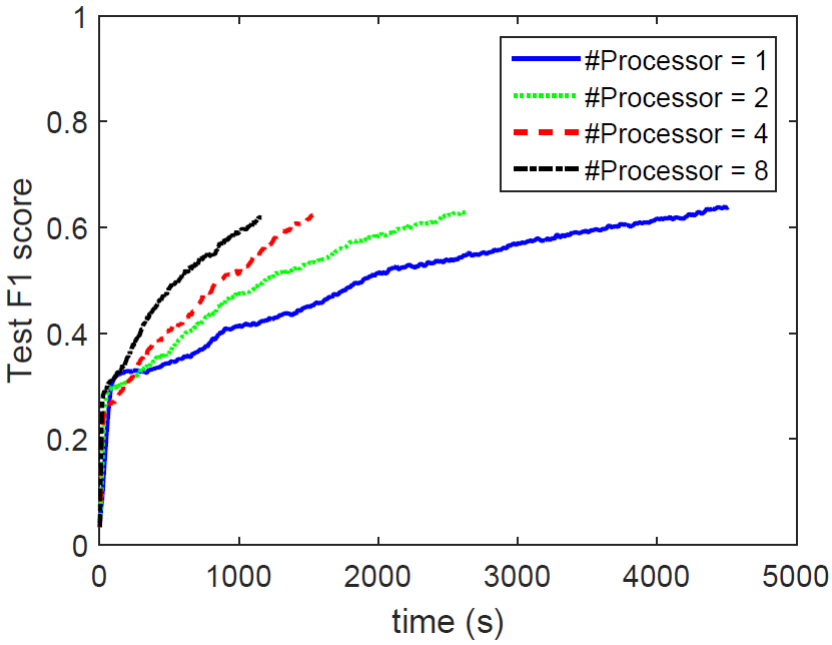


Figure 3.3: F1 score over iterations for synthetic data set with small separation

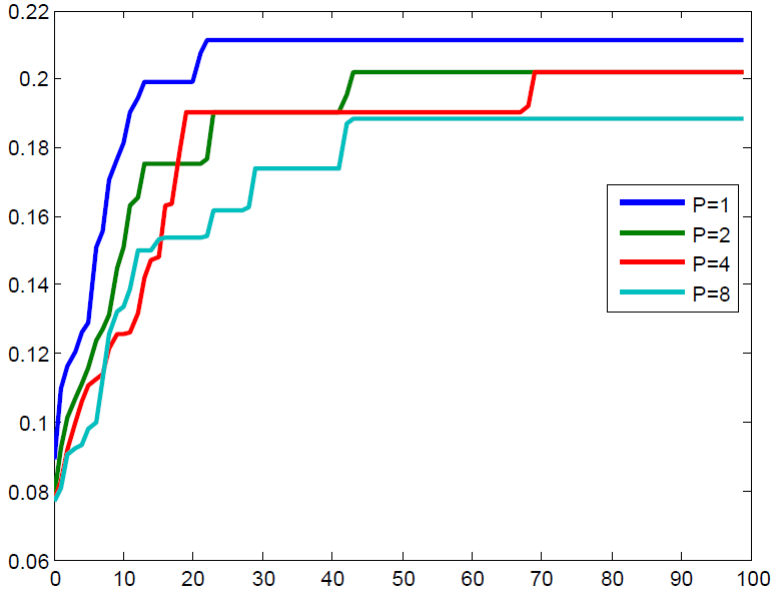
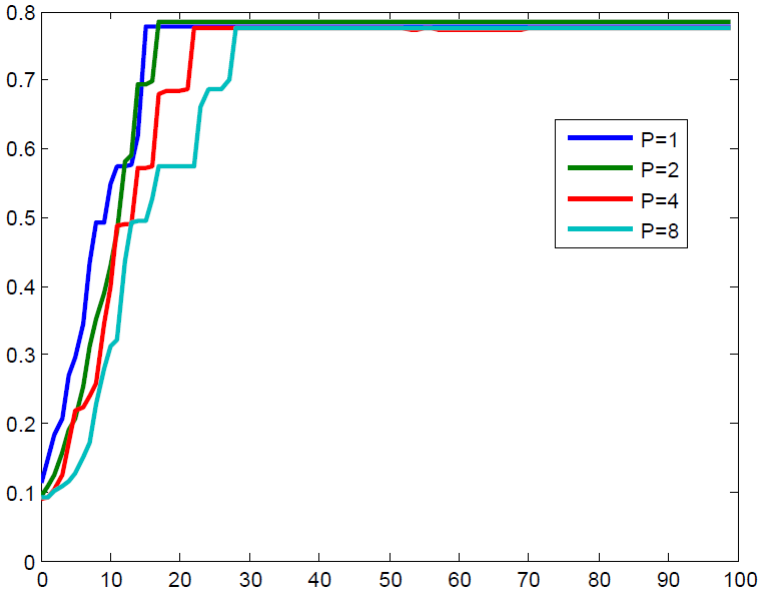
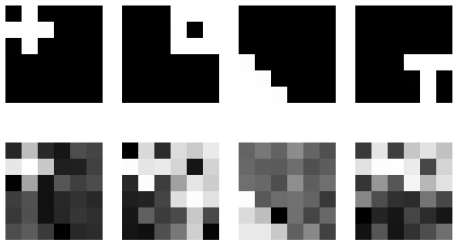


Figure 3.4: F1 score over iterations for synthetic data set with large separation



$$
\begin{aligned}
Z \sim & \text{IBP}(\alpha, 1) \\
A_k \sim & \text{Normal}(0, \sigma_A^2 \mathbf{I}) \\
X_n \sim & \text{Normal}\left(\sum_k z_{nk} A_k, \sigma_X^2 \mathbf{I}\right)
\end{aligned}
\tag{3.1}
$$

Figure 3.5: Top: The true features present in the synthetic data set. Bottom: Examples of observations in the synthetic data set.

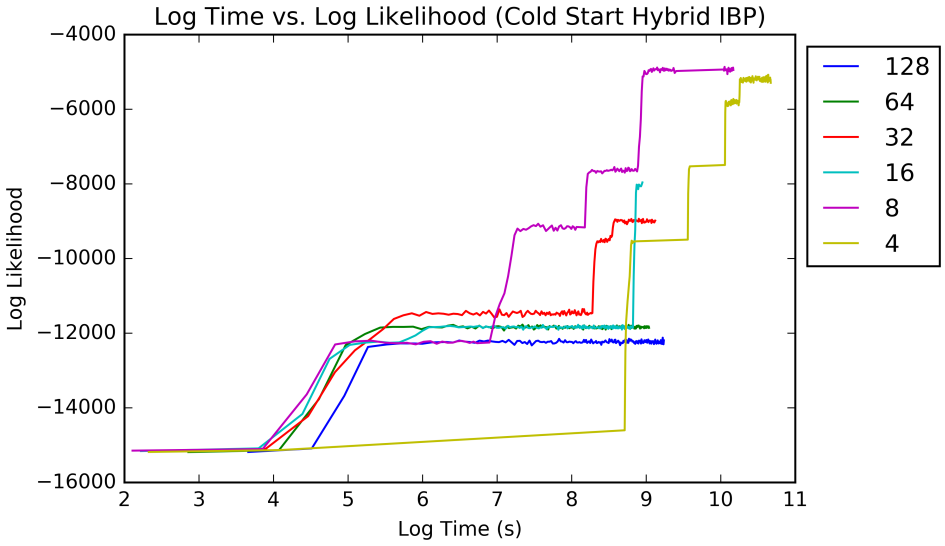


We evaluated the model on a synthetic data set consisting of 10,000 observations. This dataset was an extension of the "Cambridge" dataset, used in the original IBP paper [5], where each data point contains a randomly selected subset of 4 binary features, plus Gaussian noise ($\sigma_X = 0.5$). In the IBP experiments, we ran the hybrid sampler for 1000 total observations with a synchronization step every 5 iterations and we run the hybrid sampler for 4, 8, 16, 32, 64 and 128 processors.

We run the Hybrid IBP sampler under a "cold start", where only one processor is allowed to introduce new features for the entire duration of the sampler. We can see that the cold start results in the test set log likelihood of the higher processors failing to converge properly (Fig. 3.6). Since the number

of features in each experiment is fixed at 2, we observe that the sampler, over 8 processors, accepts few features than in subsequent examples (Fig. 3.7).

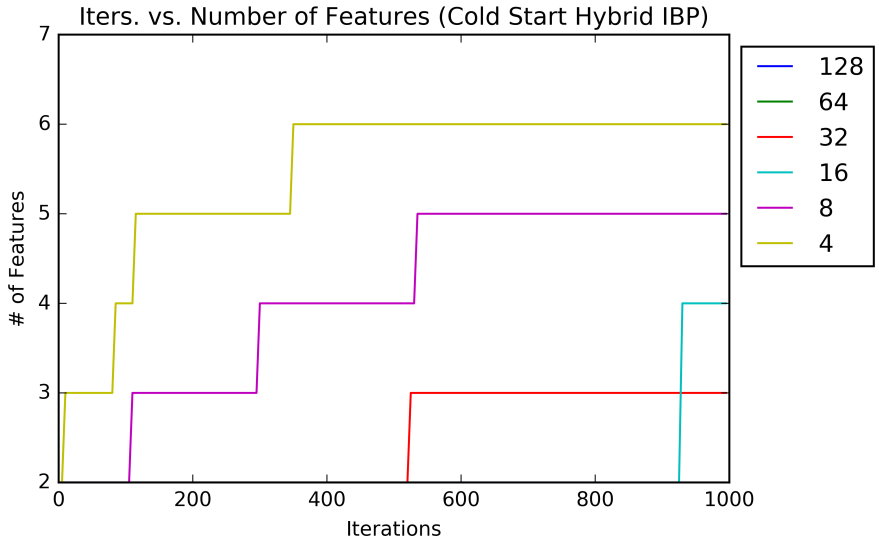
Figure 3.6: Test set log likelihood on synthetic data without warm-start initialization.



Next, we evaluated the effect of warm-start initialization, where initally all processors could propose new features; we reduced the number of processors able to add new features by 0.99 at each global step. Figure 3.8 shows the predictive log likelihood over time (shown on a log scale), for 4, 8, 16, 32, 64, and 128 processors. Clearly, the convergence rate for high processor trials is better than in the cold-start trials and the number of features is generally close to the true number of features, 4. Additionally, we can see that the 128 processor run converges the fastest and all the other processor runs converge in descending order of number of processors.

Finally, we allowed all the processors to propose new features for the

Figure 3.7: Number of features over iterations for synthetic data without warm-start initialization.



entire duration of the sampler ("always-hot"). Using the same experimental synthetic data scenario described earlier, we can see that all the processor runs roughly converge to the same test log likelihood. However, the number of features introduced is much greater than the warm start experiment, and the number of features introduces as the number of processors increase too. Moreover, the difference in convergence rates between processors is not as dramatic as in the warm-start trials. The results of the always-hot trials approximately replicate the behavior of the parallel IBP sampler in [1].

Figure 3.8: Test set log likelihood on synthetic data with warm-start initialization.

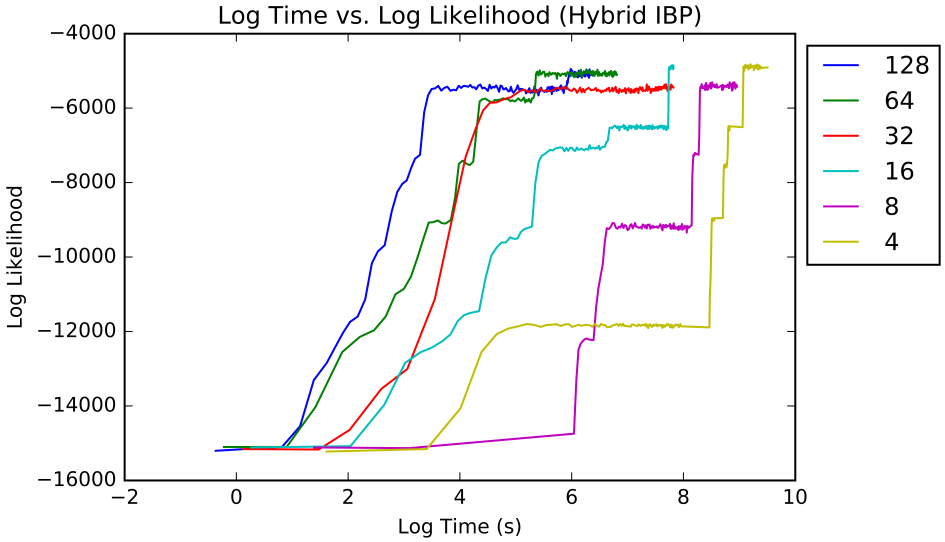


Figure 3.9: Number of features over iterations for synthetic data with warm-start initialization.

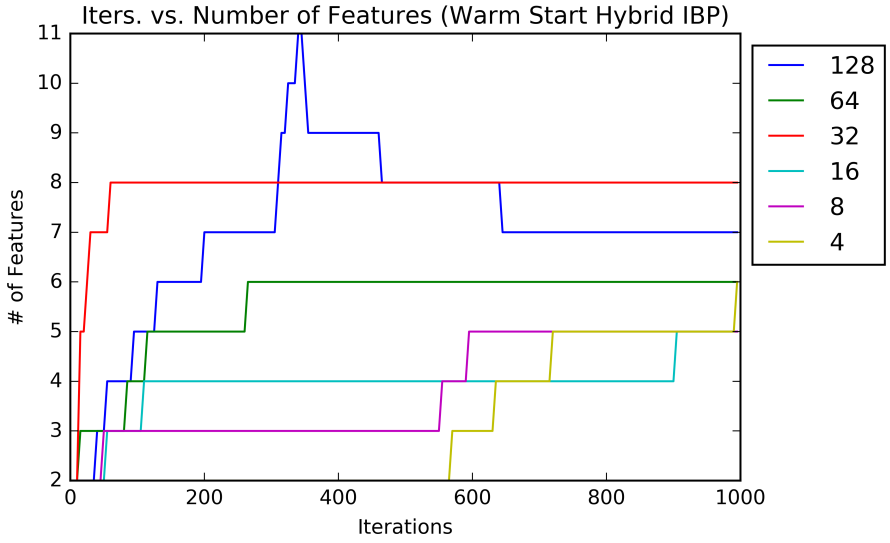


Figure 3.10: Test set log likelihood on synthetic data with all processors introducing features.

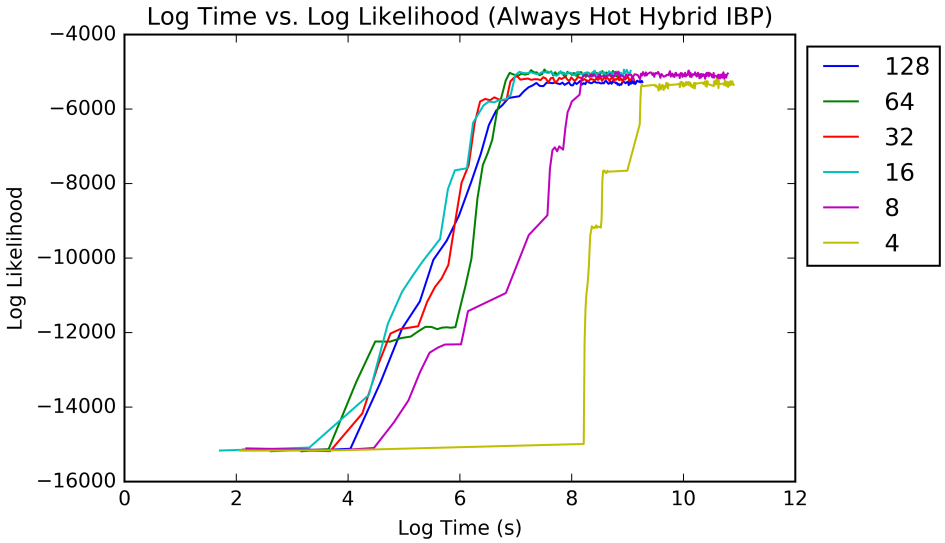
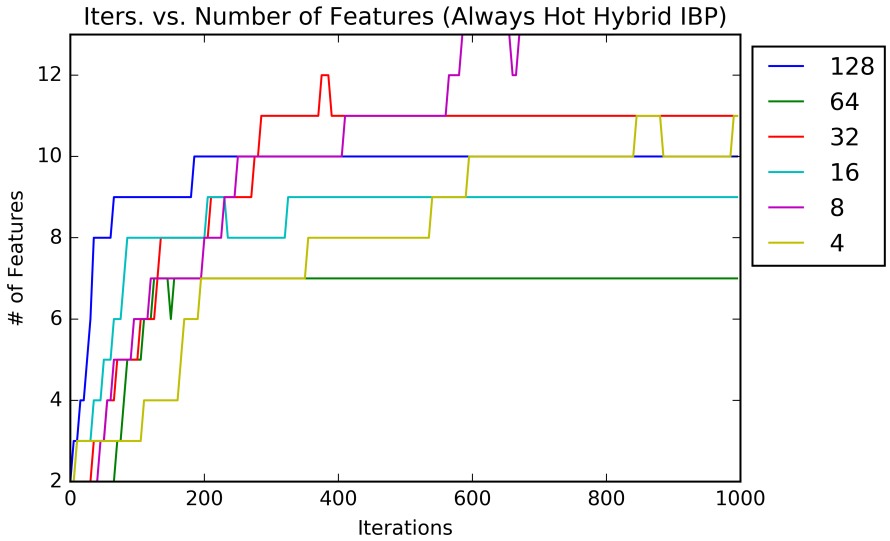


Figure 3.11: Number of features over iterations for synthetic data with all processors introducing features.



Chapter 4

Conclusion

As seen in the previous experiments, we now have a general strategy of parallelizing inference for a potentially wide class of Bayesian nonparametric models. Due to the conditional independence between the infinite dimensional latent components in the BNP models considered in this paper, we can partition the latent components into the finite-sized instantiated partition and the infinite-sized uninstantiated partition. We can take advantage of the inherent parallelizability of the uncollapsed sampler on the finite partition, which performs adequately for popular features. But collapsed sampling will perform better for proposing new components and allocating observations to newly added components. Thus, we restrict collapsed sampling only to the infinite dimensional partition of the latent components. After partitioning the data across P processors, each processor will independently sample the allocation of the latent components to observations of the data and on a global step, a master processor will gather summary statistics from each machine and send new features and posterior values for parameters to all other machines.

In a distributed setting, we must restrict collapsed sampling to only one processor to have a valid MCMC algorithm. But we have seen that con-

vergence will generally be poor under a large number of processors because the sampler can only propose new features on $1/P$-th of the data. To overcome this issue, we suggest using a "warm-start" procedure where all processors may introduce new features and we gradually reduce the number of processors introducing features each global step until only one processor may perform collapsed sampling.

One of the major issues regarding MCMC inference methods is that they are generally slow, especially as the size of the data increases. Big data is an increasingly important concern for machine learning tasks because the nature of the data available now has grown to such a massive size that the scalability of an algorithm needs to be a primary concern in developing machine learning tools. Inference in the Bayesian nonparametrics, especially for the IBP, has generally been difficult but we have developed an inference algorithm that has made BNP models amenable for huge data sets.

Chapter 5

Bibliography

[1] Finale Doshi-Velez, Shakir Mohamed, Zoubin Ghahramani, and David A. Knowles. Large scale nonparametric Bayesian inference: Data parallelisation in the Indian buffet process. In *Advances in Neural Information Processing Systems*, pages 1294–1302, 2009.

[2] Yarin Gal and Zoubin Ghahramani. Pitfalls in the use of parallel inference for the Dirichlet process. In *Proceedings of the 31st International Conference on Machine Learning*, pages 208–216, 2014.

[3] Hong Ge, Yutian Chen, Moquan Wan, and Zoubin Ghahramani. Distributed inference for Dirichlet process mixture models. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2276–2284, 2015.

[4] Zoubin Ghahramani and Thomas L. Griffiths. Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems*, pages 475–482, 2005.

[5] Thomas L. Griffiths and Zoubin Ghahramani. The Indian buffet process:

An introduction and review. *The Journal of Machine Learning Research*, 12:1185–1224, 2011.

[6] Nils L. Hjort. Nonparametric Bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, pages 1259–1294, 1990.

[7] Hemant Ishwaran and Lancelot F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 2011.

[8] Hemant Ishwaran and Mahmoud Zarepour. Exact and approximate sum representations for the Dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283, 2002.

[9] Maria Kalli, Jim E. Griffin, and Stephen G. Walker. Slice sampling mixture models. *Statistics and computing*, 21(1):93–105, 2011.

[10] John F. C. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

[11] Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[12] John Paisley and Lawrence Carin. Nonparametric factor analysis with beta process priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 777–784. ACM, 2009.

[13] Padhraic Smyth, Max Welling, and Arthur U. Asuncion. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems*, pages 81–88, 2009.

[14] Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. Stick-breaking construction for the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, pages 556–563, 2007.

[15] Romain Thibaux and Michael I. Jordan. Hierarchical beta processes and the Indian buffet process. In *International conference on artificial intelligence and statistics*, pages 564–571, 2007.

[16] Stephen G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in StatisticsSimulation and Computation*, 36(1):45–54, 2007.

[17] Sinead Williamson, Avinava Dubey, and Eric Xing. Parallel Markov chain Monte Carlo for nonparametric mixture models. In *Proceedings of the 30th International Conference on Machine Learning*, pages 98–106, 2013.

[18] Mingyuan Zhou, Haojun Chen, Lu Ren, Guillermo Sapiro, Lawrence Carin, and John W. Paisley. Non-parametric bayesian dictionary learning for sparse image representations. In *Advances in neural information processing systems*, pages 2295–2303, 2009.