Parallel Markov Chain Monte Carlo for Nonparametric Mixture Models

Sinead A. Williamson SINEAD@CS.CMU.EDU
Avinava Dubey AKDUBEY@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15201, USA

Abstract

Nonparametric mixture models based on the Dirichlet process are an elegant alternative to finite models when the number of underlying components is unknown, but inference in such models can be slow. Existing attempts to parallelize inference in such models have relied on introducing approximations, which can lead to inaccuracies in the posterior estimate. In this paper, we describe auxiliary variable representations for the Dirichlet process and the hierarchical Dirichlet process that allow us to perform MCMC using the correct equilibrium distribution, in a distributed manner. We show that our approach allows scalable inference without the deterioration in estimate quality that accompanies existing methods.

1. Introduction

Models based on the Dirichlet process (DP, Ferguson, 1973) and its extension the hierarchical Dirichlet process (HDP, Teh et al., 2006) have a number of appealing properties. They allow a countably infinite number of mixture components *a priori*, meaning that a finite dataset will be modeled using a finite, but random, number of parameters. If we observe more data, the model can grow in a consistent manner. Unfortunately, while this means that such models can theoretically cope with data sets of arbitrary size and latent dimensionality, in practice inference can be slow, and the memory requirements are high.

Parallelization is a technique often used to speed up computation, by splitting the computational and memory requirements of an algorithm onto multiple machines. Parallelization of an algorithm involves exploitation of (conditional) independencies. If we can update one part of a model independently of another part, we can split the corresponding sections of code onto separate processors or cores. Unfortunately, many models do not have appropriate independence structure, making parallelization difficult. For example, in the Chinese restaurant process representation of a Dirichlet process mixture model, the conditional distribution over the cluster allocation of a single data point depends on the allocations of *all* the other data points.

In such cases, a typical approach is to apply approximations that break some of the long-range dependencies. While this can allow us to parallelize inference in the approximate model, the posterior estimate will, in general, be less accurate. Another option is to use a sequential Monte Carlo approach, where the posterior is approximated with a swarm of independent particles. In its simplest form, this approach is inherently parallelizable, but such a naive implementation will run into problems of variance overestimation. We can improve the estimate quality by introducing global steps such as particle resampling and Gibbs steps, but these steps cannot easily be parallelized.

In this paper, we show how the introduction of auxiliary variables into the DP and HDP can create the conditional independence structure required to obtain a parallel Gibbs sampler, without introducing approximations. As a result, we can make use of the powerful and elegant representations provided by the DP and HDP, while maintaining manageable computational requirements. We show that the resulting samplers are able to achieve significant speed-ups over existing inference schemes for the "exact" models, with no loss in quality. By performing inference in the true model, we are able to achieve better results than those obtained using approximate models.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

2. Background

The Dirichlet process is a distribution over discrete probability measures $D = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}$ with countably infinite support, where the finite-dimensional marginals are distributed according to a finite Dirichlet distribution. It is parametrized by a base probability measure H, which determines the distribution of the atom locations, and a concentration parameter $\alpha > 0$, which acts like an inverse variance. The DP can be used as the distribution over mixing measures in a nonparametric mixture model. In the Dirichlet process mixture model (DPMM, Antoniak, 1974), data $\{x_i\}_{i=1}^n$ are assumed to be generated according to

$$D \sim \mathrm{DP}(\alpha, H)\,, \qquad \theta_i \sim D\,, \qquad x_i \sim f(\theta_i)\,. \tag{1}$$

While the DP allows an infinite number of clusters *a priori*, any finite dataset will be modeled using a finite, but random, number of clusters.

Hierarchical Dirichlet processes extend the DP to model grouped data. The HDP is a distribution over probability distributions $D_m, m = 1, \ldots, M$, each of which is conditionally distributed according to a DP. These distributions are coupled using a discrete common base-measure, itself distributed according to a DP. Each distribution D_m can be used to model a collection of observations $\mathbf{x}_m := \{x_{mi}\}_{i=1}^{N_m}$, where

$$\begin{array}{cc} D_0 \sim \mathrm{DP}(\alpha, H)\,, & D_m \sim \mathrm{DP}(\gamma, D_0)\,, \\ \theta_{mi} \sim D_m\,, & x_{mi} \sim f(\theta_{mi})\,, \end{array} \tag{2}$$

for $m = 1, \ldots, M$ and $i = 1, \ldots, N_m$. HDPs have been used to model data including text corpora (Teh et al., 2006), images (Sudderth et al., 2005), time series data (Fox et al., 2008), and genetic variation (Sohn & Xing, 2009).

A number of inference schemes have been developed for the DP and the HDP. The most commonly used class of inference methods is based on the Chinese restaurant process (CRP, see for example Aldous (1985)). Such schemes integrate out the random measures (D in Eq. 1; D_0 and $\{D_m\}_{m=1}^M$ in Eq. 2) to obtain the conditional distribution for the cluster allocation of a single data point, conditioned on the allocations of all the other data points. A variety of Gibbs samplers based on the CRP can be constructed for the DP (Neal, 1998) and the HDP (Teh et al., 2006). Unfortunately, because the conditional distribution for the cluster allocation of a single data point depends on all the data, this step cannot be parallelized without introducing approximations.

An alternative class of inference schemes involve explicitly instantiating the random measures (Ishwaran & James, 2001). In this case, the observations are i.i.d. given the random measures, and can be sampled in parallel. However, since the random measure depends on the cluster allocations of all the data points, sampling the random measure cannot be parallelized.

Among existing practical schemes for parallelizable inference in DP and HDP, the following three are the most popular:

2.1. Sequential Monte Carlo methods

Sequential Monte Carlo (SMC) methods approximate a distribution of interest using a swarm of weighted, sequentially updated, particles. SMC methods have been used to perform inference in a number of models based on the DP (Fearnhead, 2004; Ulker et al., 2010; Rodriguez, 2011; Ahmed et al., 2011). Such methods are appealing when considering parallelization, because each particle (and its weight) are updated independently of the other particles, and need consider only one data point at a time. However, a naive implementation where each particle is propagated in isolation leads to very high variance in the resulting estimate. To avoid this, it is typical to introduce *resampling* steps, where the current swarm of particles is replaced by a new swarm sampled from the current posterior estimate. This avoids an explosion in the variance of our estimate, but the resampling cannot be performed in parallel.

2.2. Variational inference

Variational Bayesian inference algorithms have been developed for both the DP (Blei & Jordan, 2004; Kurihara et al., 2007) and the HDP (Teh et al., 2007; Wang et al., 2011). Variational methods approximate a posterior distribution $p(\theta|X)$ with a distribution $q(\theta)$ belonging to a more manageable family of distributions and try to find the "best" member of this family, typically by minimizing the Kullback-Leibler divergence between $p(\theta|X)$ and $q(\theta)$. This also gives us a lower bound on the log likelihood, $\log p(X)$. A typical approach to selecting the family of approximating distributions is to assume independencies that may not be present in the true posterior. This means that variational algorithms are often easy to parallelize. However, by searching only within a restricted class of models we lose some of the expressiveness of the model, and will typically obtain less accurate results than MCMC methods that asymptotically sample from the true posterior.

2.3. Approximate parallel Gibbs sampling

An approximate distributed Gibbs sampler for the HDP is described by Asuncion et al. (2008). The ba-

sic sampler alternates distributed Gibbs steps with a global synchronization step. If there are P processors, in the distributed Gibbs steps, each processor updates $1/P$ of the cluster allocations. To sample the cluster allocation for a given observation, rather than use the current allocations of *all* the other data, the sampler uses the current cluster allocations for the observations stored on the same processor, and for all other observations it uses the allocations after the last synchronization step. In the synchronization step, the cluster counts are amalgamated. This can lead to problems with cluster alignment. In particular, there is no clear way to decide whether to merge a new cluster on processor p with a new cluster on processor p'. An asynchronous version of the algorithm avoids the bottleneck of a global synchronization step; however in practice it obtains slower convergence.

3. Introducing auxiliary variables to obtain conditional independence

The key to developing a parallel inference algorithm is to exploit or introduce independencies. In the sequential Monte Carlo samplers, this independence can lead to high variance in the resulting estimate. In the other algorithms described, the independence is only achieved by introducing approximations.

If observations are modeled using a mixture model, then conditioned on the cluster allocations the observations are independent. The key idea that allows us to introduce conditional independence is the fact that, for appropriate parameter settings, *Dirichlet mixtures of Dirichlet processes are Dirichlet processes*. In Theorems 1 and 2, we demonstrate situations where this result holds, and develop mixtures of nonparametric models with the appropriate marginal distributions and conditional independence structures. The resulting models exhibit conditional independence between parameters that are coupled in Eq. 1 and Eq. 2, allowing us to perform parallel inference in Section 4 without resorting to approximations.

Theorem 1 (Auxiliary variable representation for the DPMM)**.** *We can re-write the generative process for a DPMM (given in Eq. 1) as*

$$D_j \sim \mathrm{DP}\left(\frac{\alpha}{P}, H\right), \quad \phi \sim \mathrm{Dirichlet}\left(\frac{\alpha}{P}, \ldots, \frac{\alpha}{P}\right), \quad \pi_i \sim \phi, \quad \theta_i \sim D_{\pi_i}, \quad x_i \sim f(\theta_i), \tag{3}$$

for $j = 1, \ldots, P$ and $i = 1, \ldots, N$. The marginal distribution over the x_i remains the same.

Proof. We prove a more general result, that if $\phi \sim \mathrm{Dirichlet}(\alpha_1, \ldots, \alpha_P)$ and $D_j \sim \mathrm{DP}(\alpha_j, H_j)$, then $D := \sum_j \phi_j D_j \sim \mathrm{DP}(\sum_j \alpha_j, \frac{\sum_j \alpha_j H_j}{\sum_j \alpha_j})$.

A Dirichlet process with concentration parameter α and base probability measure H can be obtained by normalizing a gamma process with base measure αH. Gamma processes are examples of completely random measures (Kingman, 1967), and the superposition of P completely random measures is again a completely random measure. In particular, if $G_j \sim \mathrm{GaP}(\alpha_j H_j)$, $j = 1, \ldots, P$, then $G := \sum_j G_j \sim \mathrm{GaP}(\sum_j \alpha_j H_j)$.

Note that the total masses of the G_j are gamma-distributed, and therefore the vector of normalized masses is Dirichlet-distributed. It follows that, if $\phi \sim \mathrm{Dirichlet}(\alpha_1, \ldots, \alpha_P)$ and $D_j \sim \mathrm{DP}(\alpha_j, H_j)$, then $D := \sum_j \phi_j D_j \sim \mathrm{DP}(\sum_j \alpha_j, \frac{\sum_j \alpha_j H_k}{\sum_j \alpha_j})$. This result, which is well known in the nonparametric Bayes community, is explored in more depth in Chapter 3 of Ghosh & Ramamoorthi (2003). An alternative proof is given in the supplementary material. □

The auxiliary variables π_i introduced in Eq. 3 introduce conditional independence, which we exploit to develop a distributed inference scheme. If we have P processors, then we can split our data among them according to the π_i. Conditioned on their processor allocations, the data points are distributed according to independent Dirichlet processes. In Section 4, we will see how we can combine local inference in these independent Dirichlet processes with global steps to move data between processors.

We can follow a similar approach with the HDP, assigning each observation x_{mi} in each collection $\mathbf{x}_m$ to one of P groups corresponding to P processors. However, to ensure the higher level DP can be split in a manner consistent with the lower level DP, we must impose some additional structure into the generative process described by Eq. 2 – specifically, that $\gamma \sim \mathrm{Gamma}(\alpha)$.[1] We can then introduce auxiliary variables as follows:

Theorem 2 (Auxiliary variable representation for the HDP)**.** *If we incorporate the requirement that the concentration parameter γ for the bottom level DPs $\{D_j\}_{j=1}^M$ depends on the concentration parameter α for the top level DP D_0 as $\gamma \sim \mathrm{Gamma}(\alpha)$, then we can rewrite the generative process for the HDP as:*

$$\begin{aligned} \zeta_j &\sim \mathrm{Gamma}(\alpha/P), & \pi_{mi} &\sim \nu_m, \\ D_{0j} &\sim \mathrm{DP}(\alpha/P, H), & \theta_{mi} &\sim D_{m\pi_{mi}}, \\ \nu_m &\sim \mathrm{Dirichlet}(\zeta_1, \ldots, \zeta_P), & x_{mi} &\sim f(\theta_{mi}), \\ D_{mj} &\sim \mathrm{DP}(\zeta_j, D_{0j}), \end{aligned} \tag{4}$$

[1] More generally, we can allow separate concentration parameters γ_m for each Dirichlet process, provided $\gamma_m \sim \mathrm{Gamma}(\alpha)$.

for $j = 1, \ldots, P$, $m = 1, \ldots, M$, and $i = 1, \ldots, N_m$. The marginal distribution over the $\mathbf{x}_{mi}$ is the same in Eq.s 2 and 4.

Proof. If $\zeta_j \sim \text{Gamma}(\alpha/P)$, then $\gamma := \sum_j \zeta_j \sim \text{Gamma}(\alpha)$. Since $D_{0j} \sim \text{DP}(\alpha/P, H)$, it follows that $G_{0j} := \zeta_j D_{0j} \sim \text{GaP}\left(\frac{\alpha}{P} H\right)$ and $G_0 := \sum_j G_{0j} \sim \text{GaP}(\alpha H)$. If we normalize G_0, we find that $D_0 := \sum_j \frac{\zeta_j}{\gamma} D_{0j} \sim \text{DP}(\alpha, H)$, as required by the HDP.

If we write $G_{mj} \sim \text{GaP}(G_{0j}) = \text{GaP}(\zeta_j D_{0j})$, then we can see that $G_{0j} = \eta_{mj} D_{mj}$, where $\eta_{mj} \sim \text{Gamma}(\zeta_j)$, and $D_{mj} \sim \text{DP}(\zeta_j, D_{0j})$.

If we normalize the G_{mj}, we find that

$$\begin{aligned} D_m :=& \sum_j \frac{\eta_{mj}}{\sum_k \eta_{mk}} D_{mj} \\ \sim& \text{DP}\left(\sum_j \zeta_j, \frac{\sum_j \zeta_j D_{0j}}{\sum_j \zeta_j} \right) = \text{DP}(\gamma, D_0), \end{aligned}$$

as required by the HDP. Since the η_{mj} only appear as a normalized vector, we can write $\nu_m = (\eta_{m1}, \ldots, \eta_{mP}) / \sum_{j=1}^{P} \eta_{mj} \sim \text{Dirichlet}(\zeta_1, \ldots, \zeta_P)$.

A more in-depth version of this proof is given in the supplementary material. □

Again, the application to distributed inference is clear: Conditioned on the π_{mi} we can split our data into P independent HDPs.

4. Inference

The auxiliary variable representation for the DP introduced in Theorem 1 makes the cluster allocations for data points where $\pi_i = j$ conditionally independent of the cluster allocations for data points where $\pi_i \neq j$. We can therefore split the data onto P parallel processors or cores, based on the values of π_i. We will henceforth call π_i the "processor indicator" for the ith data point. We can Gibbs sample the cluster allocations on each processor independently, intermittently moving data between clusters to ensure mixing of the sampler.

4.1. Parallel inference in the Dirichlet process

We consider first the Dirichlet process. Under the construction described in Eq. 3, each data point x_i is associated with a processor indicator π_i and a cluster indicator z_i. All data points associated with a single cluster will have the same processor indicator, meaning that we can assign each cluster to one of the P processors (i.e., all data points in a single cluster are assigned to the same processor). Note that the jth processor will typically be associated with multiple clusters, corresponding to the local Dirichlet process D_j. Conditioned on the assignments of the auxiliary variables π_i, the data points x_i in Equation 3 depend only on the local Dirichlet process D_j and the associated parameters.

We can easily marginalize out the D_j and ϕ. Assume that each data point x_i is assigned to a processor $\pi_i \in \{1, \ldots, P\}$, and a cluster z_i residing on that processor. We will perform *local* inference on the cluster assignments z_i, and intermittently we will perform *global* inference on the π_i.

4.1.1. Local inference: Sampling the z_i

Conditioned on the processor assignments, sampling the cluster assignments proceeds exactly as in a normal Dirichlet process with concentration parameter α/P. A number of approaches for Gibbs sampling in the DPMM are described by Neal (1998).

4.1.2. Global inference: Sampling the π_i

Under the auxiliary variable scheme, each cluster is associated with a single processor. We jointly resample the processor allocations of all data points within a given cluster, allowing us to move an entire cluster from one processor to another. We use a Metropolis Hastings step with a proposal distribution that independently assigns cluster k to processor j with probability $1/P$. This means our accept/reject probability depends only on the ratio of the likelihoods of the two assignments.

The likelihood ratio is given by:

$$\begin{aligned} \frac{p(\{\pi_i^*\})}{p(\{\pi_i\})} =& \frac{p(\{x_i\}|\pi_i^*) p(\{\pi_i^*\}|\alpha, P)}{p(\{x_i\}|\pi_i) p(\{\pi_i\}|\alpha, P)} \\ =& \frac{p(\{z_i\}|\pi_i^*) p(\{\pi_i^*\}|\alpha, P)}{p(\{z_i\}|\pi_i) p(\{\pi_i\}|\alpha, P)} \\ =& \prod_{j=1}^{P} \prod_{i=1}^{\max(N_j, N_j^*)} \frac{a_{ij}!}{a_{ij}^*!}, \end{aligned} \tag{5}$$

where N_j is the number of data points on processor j, and a_{ij} is the number of clusters of size i on processor j. In fact, we can simplify Eq. 5 further, since many of the terms in the ratio of factorials will cancel. A derivation of Eq. 5 is given in the supplementary material.

The reassignment of clusters can be implemented in a number of different manners. Actually transferring data from one processor to another will lead to bottlenecks, but may be appropriate if the entire data set

is too large to be stored in memory on a single machine. If we can store a copy of the dataset on each machine, or we are using multiple cores on a single machine, we can simply transfer updates to lists of which data points belong to which cluster on which machine. We note that the reassignments need not occur at the same time, reducing the bandwidth required.

4.2. Parallel inference in the HDP

Again, we can assign tokens x_{mi} to one of P processors according to π_{mi}. Conditioned on the processor assignment and the values of ζ_j, the data on each processor is distributed according to an HDP. We instantiate the processor allocations π_{mi} and the bottom-level DP parameters, plus sufficient representation to perform inference in the processor-specific HDPs. We assume a Chinese restaurant franchise representation (Teh et al., 2006) – data points in the lower-level Dirichlet processes are clustered into "tables", and in the upper-level Dirichlet process, these "tables" are clustered and each cluster is assigned a "dish".

4.2.1. Local inference: Sampling the table and dish allocations

Conditioned on the processor assignments, we simply have P independent HDPs, and can use any existing inference algorithm for the HDP. In our experiments, we used the Chinese restaurant franchise sampling scheme (Teh et al., 2006); other representations could also be used.

4.2.2. Global inference: Sampling the π_{mj} and the ζ_j

We can represent the ζ_j as $\zeta_j := \gamma\xi_j$, where $\gamma \sim \text{Gamma}(\alpha, 1)$ and $\boldsymbol{\xi} := (\xi_1, \ldots, \xi_P) \sim \text{Dirichlet}(\alpha/P, \ldots, \alpha/P)$. We sample the π_{mj} and the ξ_j jointly, and then sample γ, in order to improve the acceptance ratio of our Metropolis-Hastings steps.

Again, we want to reallocate whole clusters rather than independently reallocate individual tokens. So, our proposal distribution again assigns cluster k to processor j with probability $1/P$. Note that this means that a single data point does not necessarily reside on a single processor – its tokens may be split among multiple processors. We also propose $\boldsymbol{\xi}^* \sim \text{Dirichlet}(\alpha/P, \ldots, \alpha/P)$, and accept the resulting state with probability $\min(1, r)$, where

$$\begin{aligned} r =& \frac{p(\{x_{mi}\}|\{\pi^*_{mi}\}, \gamma, \boldsymbol{\xi}^*, \alpha, P)}{p(\{x_{mi}\}|\{\pi_{mi}\}, \gamma, \boldsymbol{\xi}, \alpha, P)} \\ & \cdot \frac{p(\{\pi^*_{mi}\}|\gamma, \boldsymbol{\xi}^*)}{p(\{\pi_{mi}\}|\gamma, \boldsymbol{\xi})} \frac{p(\boldsymbol{\xi}^*|\alpha, P)}{p(\boldsymbol{\xi}|\alpha, P)} \\ =& \prod_{j=1}^{P} \frac{(\xi^*_j)^{T^*_{\cdot j}+\alpha/P}}{(\xi_j)^{T_{\cdot j}+\alpha/P}} \frac{T^*_{\cdot j}!}{T_{\cdot j}!} \frac{\Gamma(\alpha/P + T_{\cdot j})}{\Gamma(\alpha/P + T^*_{\cdot j})} \\ & \cdot \prod_{i=1}^{n_{\cdots}} \frac{b_{ji}!}{b^*_{ji}!} \prod_{m=1}^{M} \frac{a_{jmi}!}{a^*_{jmi}!}. \end{aligned} \tag{6}$$

A derivation of Eq. 6 is given in the supplementary material. As before, many of the ratios can be simplified further, reducing computational costs.

As with the Dirichlet process sampler, we can either transfer the data between machines, or simply update lists of which data points are "active" on each machine. We can resample γ after sampling the ξ_j and π_{mj} using a standard Metropolis Hastings step.

5. Experimental evaluation

Our goal in this paper is to employ parallelization to speed up inference in the DP and HDP, without introducing approximations that might compromise the quality of our model estimate. To establish whether we have achieved that goal, we perform inference in both the DP and HDP using a number of inference methods, and look at appropriate measures of model quality as a function of inference time. This captures both the speed of the algorithms and the quality of the approximations obtained.

5.1. Dirichlet process mixture of Gaussians

We begin by evaluating the performance of the Dirichlet process sampler described in Section 4.1. We generated a synthetic data set of one million data points from a mixture of univariate Gaussians. We used 50 components, each with mean distributed according to $\text{Uniform}(0, 10)$ and fixed variance of 0.01. A synthetic data set was chosen because it allows us to compare performance with "ground truth". We compared four inference algorithms:

- **Auxiliary variable parallel Gibbs sampler (AVparallel)** – the model proposed in this paper, implemented in Java.
- **Sequential Monte Carlo (SMC)** – a basic sequential importance resampling algorithm, implemented in Java.

- **Variational Bayes (VB)** – the collapsed variational Bayesian algorithm described in Kurihara et al. (2007). We used an existing Matlab implementation[2].

- **Synchronous approximate parallel DP (Synch)** – we modified the synchronous approximate sampler for the HDP (Asuncion et al., 2008) to be applicable to the DP. We implemented the sampler in Java, using the settings described in Asuncion et al. (2008).

In each case, we ran the algorithms on one, two, four and eight processors on a single multi-core machine[3], with one processor for the AVparallel method corresponding to the regular Gibbs sampler. We initialized each algorithm by clustering the data into 80 clusters using K-means clustering, and split the resulting clusters among the processors.

We consider the F1 score between the clusterings obtained by each algorithm and the ground truth, as a function of time. Let $\mathcal{P}^{(g)}$ be the set of pairs of observations that are in the same cluster under ground truth, and let $\mathcal{P}^{(m)}$ be the set of pairs of observations that are in the same cluster in the inferred model. Then we define the F1 score of a model as the harmonic mean between the *precision* – the proportion $|\mathcal{P}^{(g)} \cap \mathcal{P}^{(m)}|/|\mathcal{P}^{(m)}|$ of pairs that are co-clustered by the model that also co-occur in the true partition – and the *recall* – the proportion $|\mathcal{P}^{(g)} \cap \mathcal{P}^{(m)}|/|\mathcal{P}^{(g)}|$ of true co-occurrences that are co-clustered by the model. This definition of F1 is invariant to permutation, and so is appropriate in an unsupervised setting (Xing et al., 2002).

Figure 1(a) shows the F1 scores for our auxiliary variable method over time, using one, two, four and eight processors. As we can see, increasing the number of processors increases convergence speed without decreasing performance. Figure 1(b) shows the F1 scores over time for the four methods, each using eight cores. While we can get very fast results using variational Bayesian inference, the quality of the estimate is poor. Conversely, we achieve better performance (as measured by F1 score) than competing MCMC algorithms, with faster convergence. Figure 1(c) shows the time taken by each algorithm to reach convergence, for varying numbers of processors. AV parallel and Synch perform similarly. Figure 1(d) shows the relative time spent sampling the processor allocations (the global step) and sampling the cluster allocations (the local step), over 500 iterations. This explains the similar scaleability of AVparallel and Synch: In AVparallel the majority of time is spent on local Gibbs sampling, which is implemented identically in both models.

5.2. HDP topic model

Next, we evaluate the performance of the HDP sampler on a topic modeling task as described by Teh et al. (2006). Our dataset was a corpus of NIPS papers[4], consisting of 2470 documents, containing 14300 unique words and around 3 million total words. We split the dataset into a training set of 2220 documents and a test set of 250 documents, and evaluated performance in terms of test set perplexity. We compared three inference methods:

- **Auxiliary variable parallel Gibbs sampler (AVparallel)** – the model proposed in this paper, implemented in Java.

- **Variational Bayes (VB)** – the collapsed variational Bayesian algorithm described in Teh et al. (2007). We used an existing Java implementation.[5]

- **Synchronous approximate parallel HDP (Synch)** – the synchronous sampler for the HDP (Asuncion et al., 2008). We implemented the sampler in Java, using the settings described in the original paper.

Again, we ran each method on one, two, four and eight processors, and initialized each document to one of 80 clusters using K-means.

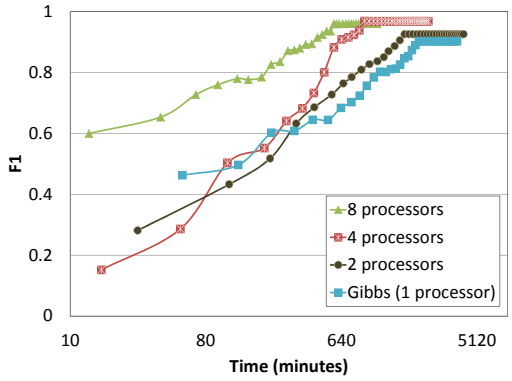
Figure 2(a) shows the perplexity obtained using our auxiliary variable method over time, using one, two, four and eight processors, and Figures 2(b) and 2(c) compare the performance of the three inference methods. As with the DPMM, while the variational approach is able to obtain results very quickly, the quality is much lower than that obtained using MCMC methods. The AVparallel method achieves much better perplexity than the approximate Synch method – the difference is much more striking than that seen in the DPMM. Note that, in the synthetic data used for the DPMM model, the true clusters are of similar size, while in the real-world data used for the HDP experiment there are likely to be many small clusters. We hypothesise that while the errors introduced in the synchronous approximate method have little effect if the clusters are large, they become more significant if

[2]Code obtained from https://sites.google.com/site/kenichikurihara/academic-software/variational-dirichlet-process-gaussian-mixture-model

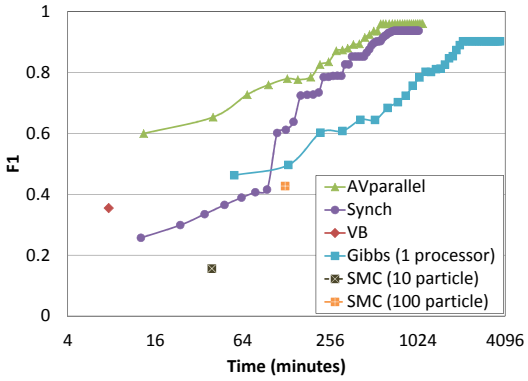
[3]An AMD FX 8150 3.6 GHz (8 core) with 16 gig ram.

[4]http://ai.stanford.edu/~gal/data.html

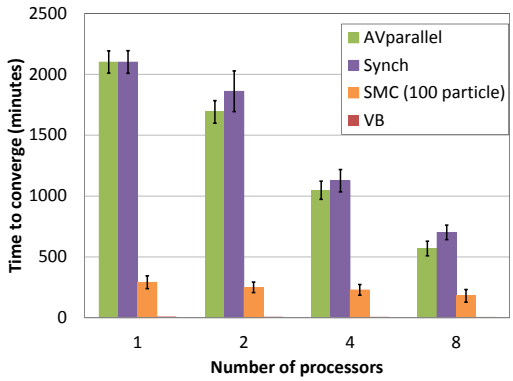
[5]Code obtained from http://www.bradblock.com/tm-0.1.tar.gz



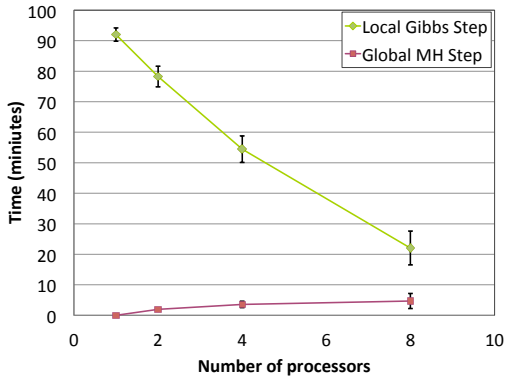
(a) F1 score against run time for AVparallel.



(b) F1 score against run time for various algorithms. Unless otherwise specified, eight processors are used.



(c) Time taken to reach convergence ($< 0.1\%$ change in F1).



(d) Time spent in global and local steps for AVparallel, over 500 iterations.

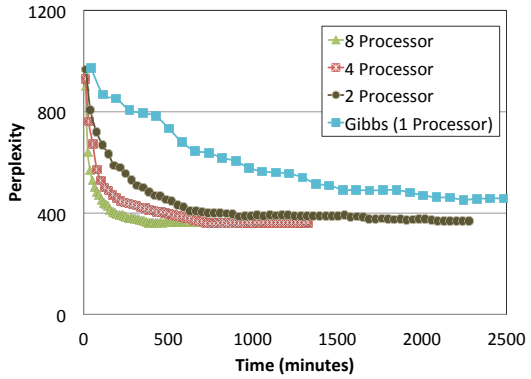
Figure 1. Synthetic data modeled using a DPMM.

we have small clusters. Again, we find (Figure 2(d)) that the majority of time is spent in the local Gibbs sampler, meaning we can obtain a good rate of increase of speed by increasing the number of processors (Figures 2(a) and 2(c)).

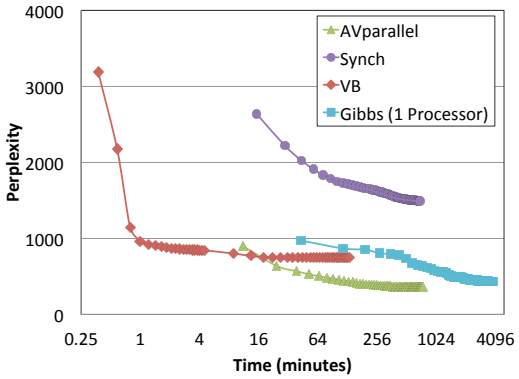
6. Discussion and future work

We have shown how alternative formulations for the DP and HDP can yield parallelizable Gibbs samplers that allow scalable and accurate inference. Our experiments show that the resulting algorithms offer better performance and scalability than existing parallel inference methods.

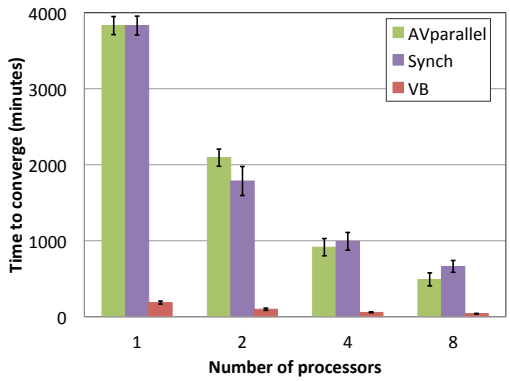
Since the assignments of clusters to processors is random, and since clusters vary in sizes, the loads assigned to each processor will vary. In addition, since each cluster must reside on a single processor, the scalability of our algorithms is limited by the size of the largest cluster. An interesting avenue for future development is to investigate approximate methods for splitting large clusters onto multiple processors, to en-



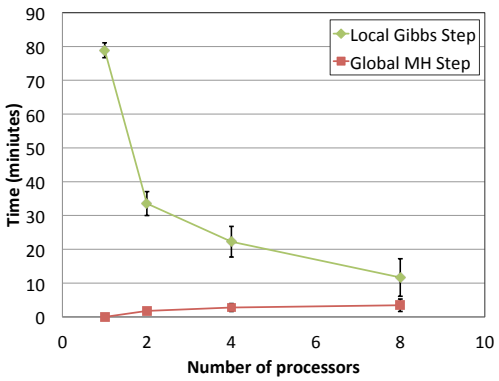
(a) Test set perplexity against run time for AVparallel.



(b) Test set perplexity against run time for various algorithms. Unless otherwise specified, eight processors are used.



(c) Time taken to reach convergence ($< 0.1\%$ change in perplexity).



(d) Time spent in global and local steps for AVparallel, over 20 iterations.

Figure 2. NIPS corpus, modeled using an HDP.

able better allocation of resources.

We note that a related parallel MCMC method for the Dirichlet process has been developed concurrently and independently by Lovell et al. (2012). This work uses the same auxiliary variable representation of the Dirichlet process, and describes a MapReduce algorithm; the hierarchical Dirichlet process is not considered. Their preliminary results using the MapReduce framework suggest that the speed-ups obtained in this paper in a multi-core environment will carry over to multi-machine architectures. Our next goal is to develop and publish code appropriate for multi-machine architectures, and to extend our approach to other nonparametric models.

Acknowledgements

This research was supported by AFOSR FA9550010247, NIH R01GM087694 and DARPA XDATA FA87501220324. We would like to thank Iain Murray for helpful comments on a draft.

References

Ahmed, A., Ho, Q., Teo, C. H., Eisenstein, J., Smola, A. J., and Xing, E. P. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *AISTATS*, 2011.

Aldous, D. J. Exchangeability and related topics. In *École d'Été de probabilités de Saint-Flour XIII*. 1985.

Antoniak, C. E. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Ann. Statist.*, 2(6):1152–1174, 1974.

Asuncion, A., Smyth, P., and Welling, M. Asynchronous distributed learning of topic models. In *NIPS*, 2008.

Blei, D. M. and Jordan, M. I. Variational methods for the Dirichlet process. In *ICML*, 2004.

Fearnhead, P. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14:11–21, 2004.

Ferguson, T. S. A Bayesian analysis of some nonparametric problems. *Ann. Statist.*, 1(2):209–230, 1973.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. An HDP-HMM for systems with state persistence. In *ICML*, 2008.

Ghosh, J. K. and Ramamoorthi, R. V. *Bayesian Nonparametrics*. Springer, 2003.

Ishwaran, H. and James, L. F. Gibbs sampling methods for stick-breaking priors. *JASA*, 96(453):161–173, 2001.

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

Kurihara, K., Welling, M., and Teh, Y.-W. Collapsed variational Dirichlet process mixture models. In *IJCAI*, 2007.

Lovell, D., Adams, R. P., and Mansingka, V. K. Parallel Markov chain Monte Carlo for Dirichlet process mixtures. In *Workshop on Big Learning, NIPS*, 2012.

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

Rodriguez, A. On-line learning for the infinite hidden Markov model. *Communications in Statistics - Simulation and Computation*, 40(6):879–893, 2011.

Sohn, K.-A. and Xing, E. P. A hierarchical Dirichlet process mixture model for haplotype reconstruction from multi-population data. *Ann. Appl. Stat.*, 3(2): 791–821, 2009.

Sudderth, E. B., Torralba, A., Freeman, W. T., and Willsky, A. S. Describing visual scenes using transformed Dirichlet processes. In *NIPS*, 2005.

Teh, Y.-W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476): 1566–1581, 2006.

Teh, Y.-W., Kurihara, K., and Welling, M. Collapsed variational inference for HDP. In *NIPS*, 2007.

Ulker, Y., Gunsel, B., and Cemgil, A. T. Sequential Monte Carlo samplers for Dirichlet process mixtures. In *AISTATS*, 2010.

Wang, C., Paisley, J., and Blei, D. M. Online variational inference for the hierarchical Dirichlet process. In *AISTATS*, 2011.

Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. Distance metric learning, with application to clustering with side-information. In *NIPS*, 2002.