Pitfalls in the use of Parallel Inference for the Dirichlet Process

Yarin Gal YG279@CAM.AC.UK
University of Cambridge

Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
University of Cambridge

Abstract

Recent work done by Lovell, Adams, and Mansingka (2012) and Williamson, Dubey, and Xing (2013) has suggested an alternative parametrisation for the Dirichlet process in order to derive non-approximate parallel MCMC inference for it – work which has been picked-up and implemented in several different fields. In this paper we show that the approach suggested is impractical due to an extremely unbalanced distribution of the data. We characterise the requirements of efficient parallel inference for the Dirichlet process and show that the proposed inference fails most of these requirements (while approximate approaches often satisfy most of them). We present both theoretical and experimental evidence, analysing the load balance for the inference and showing that it is independent of the size of the dataset and the number of nodes available in the parallel implementation. We end with suggestions of alternative paths of research for efficient non-approximate parallel inference for the Dirichlet process.

1. Introduction

The Dirichlet process (Ferguson, 1973) is a distribution that induces clusterings with a varying number of components that can grow with the complexity of the data. It is often used to model the prior over possible clusterings in tasks where the number of clusters is not known in advance; in topic modelling, for example, the number of topics is not usually known, and we would like to be able to capture any arbitrary number of these. This distribution and its derivatives are used in many fields of research including natural language modelling (Teh, 2006), statistical machine translation (Gal & Blunsom, 2013), activity modelling (Fox et al., 2008), word segmentation (Xu et al., 2008), and topic modelling (Teh et al., 2006), to name a few.

Inference for models that use the Dirichlet process can be done using Markov chain Monte Carlo techniques in which a Markov chain is constructed to draw samples from the posterior. These techniques are well known for their long running time since the walk along the chain should in theory converge to its stationary distribution before the samples produced can be used. The convergence process is often slow as it depends on the mixing properties of the sampler – how quickly it “jumps around” in space – while prolonged burn-in time and unbounded variance inhibit running multiple independent chains concurrently in a naive way.

We are thus interested in distributed samplers to answer for the slow sampling from the true posterior of the Dirichlet process. Many have reasoned what requirements such distributed samplers should satisfy (Brockwell, 2006; Wilkinson, 2005; Asuncion et al., 2008; Lovell et al., 2012; Williamson et al., 2013). These samplers should:

1. distribute the computational load evenly across the nodes used in the parallel implementation (processors in a computer, parallel threads, cores in a cluster, computers in a network, and so on),
2. scale favourably with the number of nodes,
3. have low overhead in the global steps,
4. and converge to the true posterior distribution.

Many *approximate* distributed samplers have been suggested over the years – samplers that satisfy some or all of the first 3 conditions but not the last one. Asuncion, Smyth, and Welling (2008) have suggested approximate Gibbs sampling where each of the K nodes handles exactly $1/K$ of the data, locally assigning the data points to clusters according to a stored global state, and occasionally synchronising the global state to keep the clustering from diverging. However, in practice this approach leads to slower

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

convergence (Williamson et al., 2013). In parallel implementations of *variational inference* (Blei & Jordan, 2004; Kurihara et al., 2007), the distribution is approximated using simpler distributions from a parametrised family, and the chosen distribution in the given family is the one minimising the Kullback–Leibler divergence.

Doshi, Knowles, Mohamed, and Ghahramani (2009) have suggested inexact parallel inference in a model closely related to the Dirichlet process called the Indian Buffet Process (Griffiths & Ghahramani, 2006). They presented a way to make the inference exact through the use of Metropolis–Hastings steps (rejecting samples produced by the sampler) but argue that doing so introduced a significant computation overhead which resulted in poor use of computational resources. They presented empirical evidence showing that the approximate sampler behaves in a similar way to the exact sampler with the Metropolis–Hastings corrections. Lastly, the use of Sequential Monte Carlo methods to approximate the distribution using a weighted set of particles has been suggested for the Dirichlet process and its derivatives as well (Fearnhead, 2004; Ülker et al., 2010; Ahmed et al., 2011). In this approach, each particle is independent of other particles and needs to consider only one data point at a time, allowing for efficient parallel implementation. In the global step the whole selection of particles is replaced with new particles sampled from the current posterior estimate to avoid the problems of increasing variance caused by the independent updates.

Recently an attempt has been made to derive a distributed sampler that produces samples from the true posterior of models that use the Dirichlet process. Lovell, Adams, and Mansingka (2012) have suggested an alternative parametrisation for the Dirichlet process in order to derive new parallel MCMC inference for it. They use an auxiliary variable representation of the Dirichlet process and describe a MapReduce algorithm for the implementation of the parallel inference. Independently, Williamson, Dubey, and Xing (2013) have suggested a distributed algorithm that uses a re-parametrisation of the Dirichlet process with the same auxiliary variable scheme. This work was assessed and implemented on a single machine architecture with intentions to extend the implementation into a multi-machine architecture. These approaches were adopted in several different fields (Chahuneau et al., 2013; Deka et al., 2013; Ida et al., 2013) where the parallel inference was implemented as part of the research or intended to be used in future research.

However, in this paper we show that the auxiliary variable approach suggested by Lovell et al. (2012) and Williamson et al. (2013) for the implementation of the inference in a parallel way results in an extremely unbalanced distribution of the data to the different nodes in the parallel implementation – violating the conditions stated above. This follows from the sparseness properties of the Dirichlet distribution used for the re-parametrisation which suggest that the number of nodes used in the parallel implementation is independent of the number of available nodes for computation, and depends only on n, the size of the dataset, and α, the parameter used for the distribution – violating the second condition stated above. Thus, even if a large number of machines is available for the inference, only a small subset of it would actually be in use (logarithmic in the size of the dataset: $\alpha \log(n)$). Moreover, because of the exponential decay in the size of the clusters of the Dirichlet process, most of the work will be sent to a small number of nodes independently of the number of available nodes *or* the number of data points (95% of the data points will be sent to roughly $\approx \frac{1.3}{\log(\alpha+1)-\log(\alpha)}$ nodes) – violating the first condition stated above. This renders the inference impractical for large networks and many real-world problems. For example, in natural language modelling the value of the parameter α would usually be chosen (or inferred, in the case of Lovell et al. (2012)) to be a small number of the order 0.1. This means that **1 machine** would handle 95% of the data, with another machine handling the rest of the data.

The main contribution of this paper is an analysis of the parallel inference introduced by Lovell, Adams, and Mansingka (2012) and Williamson, Dubey, and Xing (2013), demonstrating its impracticality as follows from the requirements of a parallel inference mentioned above, as well as the proposal of directions for future research for non-approximate parallel inference for the Dirichlet process to answer for that.

2. The Dirichlet process

We now review some of the properties of the Dirichlet process that are important for our further analysis. The Dirichlet process (DP) is a distribution that generates finite point measures given some base distribution H and a concentration parameter α. These point measures can be used to induce clusterings over data points using a procedure known as the “Paintbox construction” (Kingman, 1978) where for each data point we sample from the point measure, obtain some $\theta \in H$, and group all data points assigned to the same θ together. The Dirichlet process with parameter α can be seen as the infinite dimensional generalisation of the Dirichlet distribution with K components and parameter α/K when the number of components of the Dirichlet distribution tends towards infinity.

We can generate clusterings from the distribution by marginalising over the point measures sampled from the Dirichlet process using a construction called the *Chinese Restaurant Process* (CRP). This process can be described using the metaphor of a restaurant with customers entering and sitting next to tables with different probabilities depending on the tables' relative sizes (being partitioned by their choice of a table). More formally though, one defines the CRP as a distribution over partitions of the naturals such that:

$$P(z_i = k | z_1, ..., z_{i-1}) = \begin{cases} \frac{n_k}{i-1+\alpha} & \text{if } n_k > 0 \\ \frac{\alpha}{i-1+\alpha} & k \text{ is a new class} \end{cases}$$

where z_i is a random variable denoting the class allocation of the i'th data point, n_k is the number of data points in the class k, and $\alpha > 0$. This distribution is exchangeable, as the probability of the allocation of points to specific classes doesn't change when the points are permuted.

We will also make use of the residual allocation model where samples from the Dirichlet process are generated using the following "stick-breaking" construction given some concentration parameter $\alpha \in (0, \infty)$ and underlying measure H (Sethuraman, 1994):

$$\beta'_i \overset{iid}{\sim} \text{Beta}(1, \alpha), \ i \geq 1$$
$$\beta_i = \beta'_i \prod_{k=1}^{i-1} (1 - \beta'_k)$$
$$f_j = \sum_{i=1}^{\infty} \beta_i \delta_{\theta_i}, \ \theta_i \sim H$$

with the property that a.s. $\sum_{i=1}^{\infty} \beta_i = 1$. This construction can be interpreted intuitively as the breaking of a stick of unit length (at a point sampled from a Beta distribution), taking the length of one part to be the probability of some $\theta \in H$, and breaking the rest of the stick in a recursive manner.

The Dirichlet process induces clusterings with an exponential decay in the size of the clusters: clusterings with a large number of customers sitting next to a single table, with the next largest table having a much smaller number of customers, and so on (in more detail, the relative customer frequency of the k'th largest table being a/s^k for some constants a, s). In this setting, one would usually choose the parameter α of the Dirichlet process to be of small magnitude (a value around 0.1 is common when the parameter is not marginalised out; in such cases, a vague gamma prior is often used (Teh et al., 2006) which in practice puts an upper bound on the value the concentration can take). The choice of small values for the concentration parameter arises from the "rich get richer" property observed in many real world problems and captured by the Dirichlet process. The concentration parameter controls this behaviour – for small values one would observe a relatively small number of large tables with many customers sitting around each one and many small tables with a small number of customers, whereas with large values for the concentration one would observe a large number of large tables, with fewer customers sitting around each table for the same size dataset.

3. Parallel inference for the Dirichlet process

The parallel inference suggested can be presented by the following formulation of the Dirichlet process given $\alpha > 0$ and base distribution H (Lovell, Adams, and Mansingka, 2012):

$$\gamma \sim \text{Dir}_K(\alpha\mu_1, ..., \alpha\mu_K)$$
$$G_k \sim \text{DP}(\alpha\mu_k, H)$$
$$G = \sum_{k=1}^{K} \gamma_k G_k \qquad (1)$$

for some given $(\mu_k)_1^K$ where $\sum_{k=1}^K \mu_k = 1$ and $\mu_k \geq 0$. The resulting G has the same distribution as $\text{DP}(\alpha, H)$ as proved in (Williamson et al., 2013), and is actually represented as a mixture of Dirichlet processes with smaller concentration values. Intuitively, given a network with K nodes to carry out the parallel inference, one samples from a Dirichlet distribution with K components and parameter α/K (corresponding to $\mu_k = \frac{1}{K}$ for all k). The sample produced determines the distribution of the load in the network: each component corresponds to one node in the network, and the value for each component in the vector γ determines the relative amount of data to be sent to this node.

Williamson, Dubey, and Xing (2013) give a mixture model instead of G, and introduce an additional step that samples the node assignment $\pi_i \sim \gamma$ and only then samples the cluster assignment $\theta_i \sim G_{\pi_i}$ to obtain $x_i \sim f(\theta_i)$. Williamson et al. (2013) give an intuitive interpretation to this construction: for K nodes the data is split to groups according to the probability induced by γ. Then, conditioned on the node allocation of the data, the data points are clustered in independent Dirichlet processes.

4. Impracticality of the inference – an asymptotic analysis

The main problem with this approach is that as the number of nodes K increases, the samples γ from the Dirichlet distribution become sparser. Even for the optimal choice of parameters $\mu_1 = \frac{1}{K}, ..., \mu_K = \frac{1}{K}$ (every other choice will skew the distribution even further) one obtains a sample from $\text{Dir}_K(\alpha/K)$, where a fixed α and large K would produce samples with most of the mass concentrated in the corners of the simplex, resulting in an unbalanced distribution of the data.

The exact number of nodes used in the parallel implementation of the inference can be derived from the asymptotic properties of the Dirichlet process. This is because of the special structure of the Dirichlet distribution used in the reparametrisation. For the optimal case of $\mu_i = 1/K$ one obtains a sample from a Dirichlet distribution $\text{Dir}_K(\alpha/K)$, which in the limit of $K \to \infty$ gives us a sample from a Dirichlet process (with parameter α and a base distribution $\mathcal{P}$ over the infinite set of possible nodes). The asymptotic number of component locations (corresponding to nodes) drawn from a sample from this high-dimensional Dirichlet distribution converges to the asymptotic number of unique clusters induced by a Dirichlet process with the same parameter. We thus get that the number of nodes used in the parallel inference is not dependent on the number of available nodes: sampling the allocation of n data points from a Dirichlet process with parameter α gives on average $\alpha \log(n)$ unique clusters (Arratia et al., 2003). For a *mega-set* (a dataset with a size of the order of a million data points) only $\approx 14\alpha$ nodes would be in use – for $\alpha = 0.1$ this amounts to 2 nodes used and for $\alpha = 10$ this amounts to 140 nodes in use, no matter how many nodes are available. For a *giga-set* (a dataset with 10^9 data points), one would get $\approx 21\alpha$ nodes in use, and for a *tera-set* (10^{12} data points) one would get $\approx 28\alpha$ nodes in use (only 3 nodes used for the case $\alpha = 0.1$). From this it might look like the solution to the problem would be to only use the inference for very large datasets.

However, because of the exponential decay in the size of the clusters of the Dirichlet process, the number of data points sent to each cluster is unbalanced. We now show that for all $0 < p < 1$, a fraction p of the data is sent to a small number of nodes independently of the number of available nodes, and independently of the number of data points as well. This follows from the stick-breaking construction of the Dirichlet process brought above.

Following the stick-breaking construction, we get that the expected value of β_1 (the length of the first stick) is given by $\frac{1}{1+\alpha}$. The expected value of β_2 is given by

$$E[\beta_2] = E[(1-\beta_1')\beta_2'] = E[(1-\beta_1')]E[\beta_2'] = \frac{\alpha}{(1+\alpha)^2}$$

where the second transition follows from the independence of β_1' and β_2'. Similarly, the expectation of β_3 is given by

$$\frac{\alpha^2}{(1+\alpha)^3},$$

and so on.

For each β_i, the average number of points sent to node θ_i is given by $E[\beta_i]$, and since the values of β_i partition the unit interval and sum to one, each data point sampled from the sample from the Dirichlet process corresponds to one and only one β_i corresponding to the node θ_i. When summing the first n expected values of β_i, one gets the relative number of data points belonging to one of the first n nodes (disregarding the identifiers θ_i's of the nodes):

$$\begin{aligned} E\left[\sum_{i=1}^{K}\beta_i\right] &= \sum_{i=1}^{K} E[\beta_i] = \sum_{i=1}^{K}\frac{\alpha^{i-1}}{(1+\alpha)^i} \\ &= \frac{1}{1+\alpha}\sum_{i=0}^{K-1}\frac{\alpha^i}{(1+\alpha)^i} = \frac{1}{1+\alpha}\,\frac{1-\left(\frac{\alpha}{1+\alpha}\right)^K}{1-\frac{\alpha}{1+\alpha}} \\ &= 1-\left(\frac{\alpha}{1+\alpha}\right)^K. \end{aligned}$$

To see how many nodes handle a fraction $0 < p < 1$ of the data, we solve the following equation

$$1-\left(\frac{\alpha}{1+\alpha}\right)^K = p$$

which amounts to solving (taking log in natural base)

$$K \log\left(\frac{\alpha}{1+\alpha}\right) = \log(1-p)$$

which results in

$$K = \frac{\log(1-p)}{\log(\alpha) - \log(\alpha+1)}, \qquad (2)$$

i.e. the number of nodes handling any fixed percentage of the data is independent of the number of data points and depends only on the concentration parameter.

From this we get that even for the tera-set analysed above (and for that matter for peta-sets and exa-sets as well), the number of nodes handling 95% of the data for $\alpha = 0.1$ would be 1, and for $\alpha = 10$ would be 31 (out of which one node would handle 10% of the tera-set, the next would handle 8%, the next 7%, and so on). In this view, using the inference for large datasets would penalise us by sending large amounts of data to single nodes. It is interesting to note that these results were not observed in the experimental work of Lovell, Adams, and Mansingka (2012) and Williamson, Dubey, and Xing (2013), as in these the number of nodes was restricted to small values (in Williamson et al. (2013) for example it was restricted to 4) and the choice of the concentration parameter for the synthetic datasets was large (Lovell et al. (2012) actually inferred the parameter's value but the value itself was not reported). An experimental analysis of this distribution of the load is presented in the next section.

5. Impracticality of the inference – an empirical analysis

The problem becomes even worse in practical implementations in many real-world applications, as one would often choose the parameter α to be of small magnitude which results in a very small parameter value for the Dirichlet distribution, and thus in sparse samples as most of the mass is concentrated in the corners of the simplex.

We can see this behaviour more clearly by looking at the two-staged Chinese restaurant process introduced in Lovell, Adams, and Mansingka (2012), where each customer chooses one of the K restaurants according to its popularity:

$$P(\text{data point } n \text{ chooses node } k \mid \alpha) = \frac{\alpha\mu_k + \sum_{i=1}^{n-1} \mathbb{I}(s_{z_i} = k)}{\alpha + n - 1}$$

For small values of $\alpha\mu_k$ we get that most customers go to the same restaurants, as the first assignment will give a much higher weight to one restaurants in the next assignment, and the next assignment has high probability of increasing this weight even further.

As a concrete example, we will simulate a sample from a DP (0.1) with 4 nodes in the distributed implementation. A selection of samples from a Dir$_4$ $(0.1/4)$ would be (sorting the nodes by decreasing load):

$$\begin{gathered}(1.0000, 0.0000, 0.0000, 0.0000)\\(0.5658, 0.3985, 0.0357, 0.0000)\\(0.9999, 0.0001, 0.0000, 0.0000)\\(1.0000, 0.0000, 0.0000, 0.0000)\\(1.0000, 0.0000, 0.0000, 0.0000)\\(0.9867, 0.0092, 0.0042, 0.0000)\end{gathered}$$

Which means that in 5 out of the 6 runs 99% of the data would be sent to a single node.

Sampling $10,000$ samples from Dir$_K$ (α/K) and averaging, for different real-world values of α and K, we obtain the loads displayed in figure 1 showing that for $\alpha = 0.1$ all of the data is sent to only two nodes, with 94% of it sent to one of the two, and for $\alpha = 2$ (figure 2) we have that 92% of the data is sent to only 5 nodes (out of the $K = 10, ..., 10000$ nodes), where two of these handle more than two thirds of the load.

# of nodes / param.	$\alpha = 0.1$
$K = 10^1$	94%, 6%, 0%, 0%, 0%, ...
$K = 10^2$	94%, 6%, 0%, 0%, 0%, ...
$K = 10^3$	94%, 6%, 0%, 0%, 0%, ...
$K = 10^4$	94%, 6%, 0%, 0%, 0%, ...
$K = 10^5$	94%, 6%, 0%, 0%, 0%, ...

Figure 1. Average load on each node in decreasing order for $\alpha = 0.1$

# of nodes / param.	$\alpha = 2$
$K = 10^1$	54%, 23%, 12%, 6%, 3%, ...
$K = 10^2$	48%, 22%, 12%, 7%, 4%, ...
$K = 10^3$	48%, 21%, 12%, 7%, 4%, ...
$K = 10^4$	48%, 21%, 12%, 7%, 4%, ...
$K = 10^5$	48%, 21%, 12%, 7%, 4%, ...

Figure 2. Average load on each node in decreasing order for $\alpha = 2$

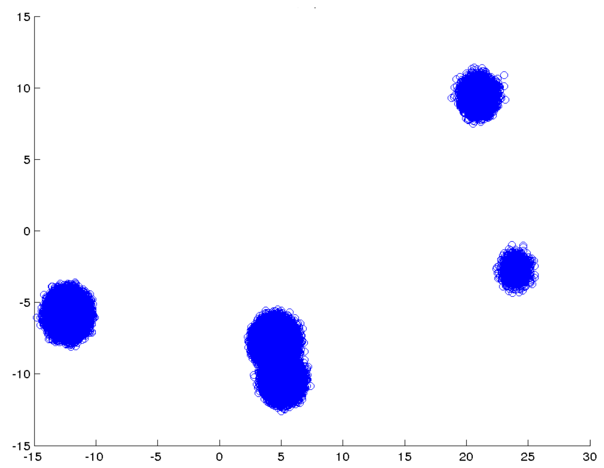
Sampling from the prior of a Dirichlet process mixture model with different values for the parameter α, and running the proposed inference procedure as implemented in (Chang & Fisher III, 2013) with minor bug corrections on a 12 cores machine, we can analyse the average time spent in each node in a real-world scenario. In this experiment we sampled a mega-set (1M points) from a DP mixture of Gaussians with a Gaussian-Wishart prior for parameter values $\alpha = 0.5$ and $\alpha = 2$ (a value of 0.5 was chosen instead of 0.1 as in the previous experiment for practical reasons, since the number of samples required to obtain several clusters with $\alpha = 0.1$ is much larger). The sampled datasets can be seen in figure 3. We then ran the inference procedure for 250 iterations initialising the cluster assignments to a single cluster and the parameter α to the true concentration, and evaluated the relative time spent in each node (clock cycles per thread to be exact) using $K = 2, 4, 6, 8$ nodes. The average time spent in each node is shown in figure 4, demonstrating the unbalanced distribution of the load among the different nodes.

6. Optimality cases and suggestions for future research

An interesting question to ask is whether there exists a setting of the inference which would give a better load balance. In this section we study several different optimal settings showing that a small improvement can be achieved but that the balance remains skewed. We then explore the strengths and weaknesses of alternative approaches to non-approximate parallel inference and discuss possible directions for future research.

6.1. Optimal number of nodes

Since one cannot usually control the concentration parameter value as it is determined by the model to be captured, as well as the number of data points to be processed, the only “tunable parameter” for the inference scheme at hand is the number of network nodes to be used in the parallel inference implementation. Given the computational resources, one would want to utilise these in the best possible way. However, the use of a large number of nodes in the inference would entail, apart from the unbalanced distribu-



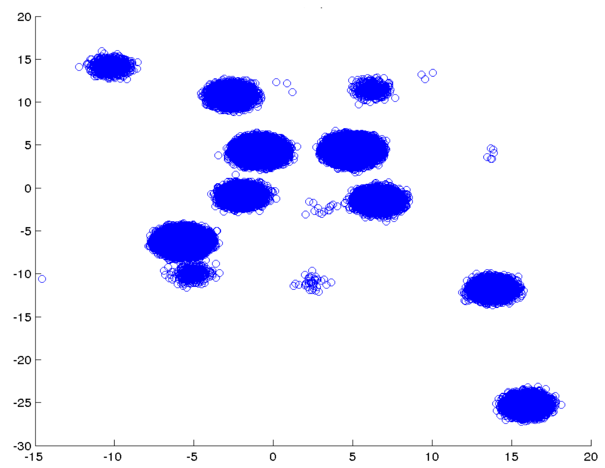
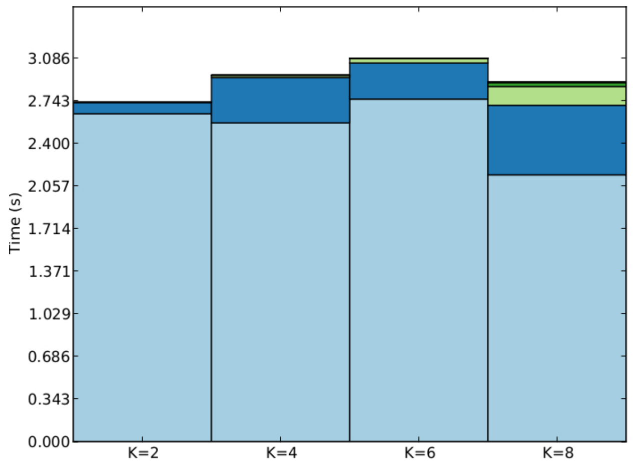


Figure 3. Samples of 1M points from a DP mixture of Gaussians with a Gaussian-Wishart prior for parameter values $\alpha = 0.5$ (left) and $\alpha = 2$ (right)



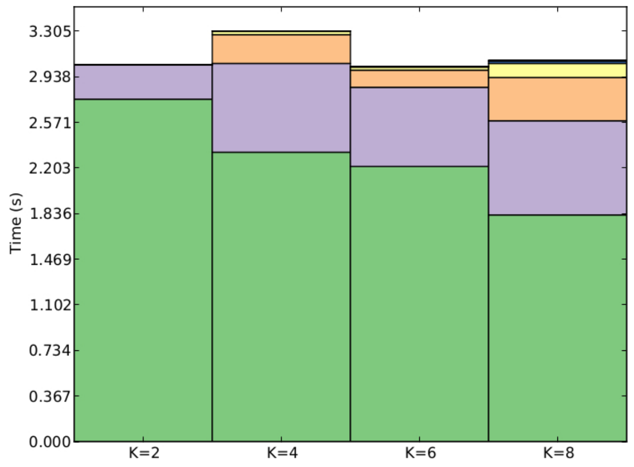


Figure 4. **Average relative time spent in each node** (clock cycles per thread) using $K = 2, 4, 6, 8$ nodes on the datasets produced above. Cluster assignments were initialised using a single cluster and α was set to the true concentration. The average time spent in each node as a function of the number of nodes is shown where the time intervals are stacked on top of each other for comparison. Shown $\alpha = 0.5$ (left) and $\alpha = 2$ (right).

tion of the data where a small number of nodes carries out most of the work, that each node performs inference over a Dirichlet process with parameter nearing zero (DP (α/K) for large K) – which in practice would cause the local sampler to assign all data points to a single cluster and all of the work to be done in the global steps.

Thus, to avoid the waste of computational resources, the suggested parallel implementation should be used when *a small number of nodes is available*. As the extreme case of using one single node would move all of the work to the local steps, it would seem that there exists an "optimal value" for the number of nodes to be used. This number happens to be α itself – the use of $K = \lceil \alpha \rceil$ nodes in the network would mean that the distribution of the balance would be sampled from a Dirichlet distribution with parameter 1, thus the load would be distributed uniformly. However, this solution forces the distributed nature of the inference to depend on the model to be captured – which renders the optimal choice of the number of nodes impractical for many real-world problems as explained earlier.

6.2. Optimal initialisation

Another possible optimality case is when the posterior is balanced. The analysis above of the distribution of the data to the different nodes looks at the a-priori cluster sizes of the Dirichlet process. For some problems the a-posteriori allocation of the data is balanced – i.e. the data is composed of evenly distributed classes. In such cases, when the sam-

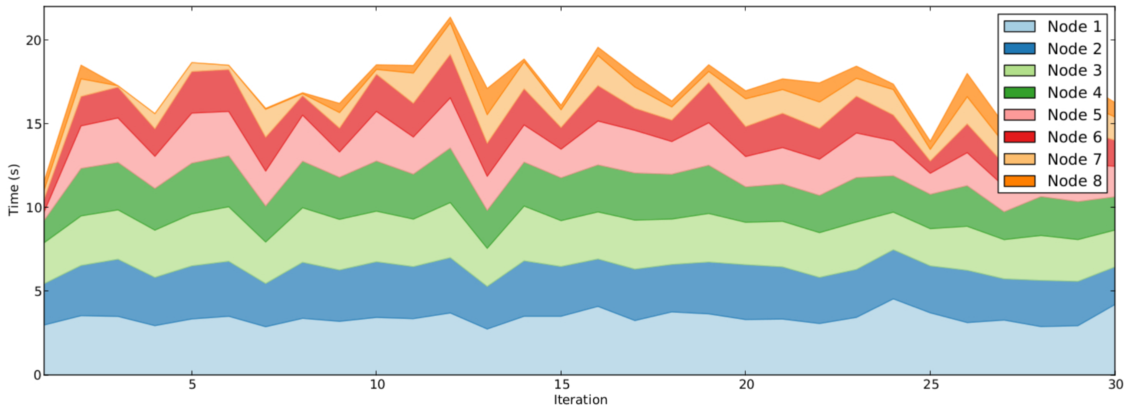


Figure 5. **Accumulative plot of time spent in each node (for optimal initialisation).** Inference was done on a sample from the prior with $\alpha = 10$ with 138 clusters and 8 nodes, where the number of clusters was initialised to 100. The time spent in each node as a function of the iteration is shown where the time intervals are stacked on top of each other for comparison. 4 out of the 8 nodes are balanced, while nodes 5 and 6 carry 80% of the work of the others, and *nodes 7 and 8 do almost no work.*

pler is close to convergence after it had "moved" the data points between the different clusters from the initial unbalanced state, we will converge towards the true posterior with the data points distributed evenly among the different clusters. However, the distribution of the different clusters themselves among the nodes in the parallel implementation still follows an exponential decay (as can be observed from the two-staged Chinese restaurant process brought above). In fact, the exact update for cluster assignment in the different nodes is brought in (Lovell et al., 2012) as:

$$P(\text{cluster } j \text{ chooses node } k \mid \alpha, \{J_{k'/j}\}_{k'=1}^{K}) = \frac{\alpha\mu_k + J_{k/j}}{\alpha + \sum_{k'=1}^{K} J_{k'/j}} \tag{3}$$

for $J_{k/j}$ the number of clusters in node k not including cluster j. Even if the clusters happen to divide the data equally under the posterior, the allocation of clusters to nodes according to equation 1 would distribute the load following the same extremely unbalanced exponential decay, since the updates in equation 3 distribute the clusters to nodes independently of the data. In such a case one could *initialise the sampler close to the posterior* if it is known in advance to have many evenly balanced clusters, and get a less distorted distribution of the load. We performed an additional experiment using a sample from the prior with $\alpha = 10$ with 138 clusters and 8 nodes, where we initialised the number of clusters to be 100, randomly assigning data points to the different clusters. The time spent in each node as a function of the iteration can be seen in figure 5. As we can see, 4 out of the 8 nodes are balanced, while nodes 5 and 6 do about 80% of the work of the others, and nodes 7 and 8 do almost no work.

6.3. Metropolis–Hastings corrections

Since whole clusters are sent to single nodes in the proposed inference, datasets that have very large clusters would distribute the load in an unbalanced and inefficient way. An implementation of non-approximate parallel inference for the Dirichlet process should therefore *split the cluster representation* among different nodes. Such implementations need take care of the overhead from cluster maintenance communication, as they would need to constantly transmit information about their relative sizes across all nodes. Following Doshi et al. (2009) we would like to take an efficient approximate parallel inference and make it exact (or at least non-approximate) by Metropolis–Hastings corrections.

A recent attempt influenced by (Jain & Neal, 2004) at doing so is presented in (Chang & Fisher III, 2013), where the data is decoupled for each finite K (the number of components of the DP) conditioned on the probability of each component (out of K). This then gives approximate inference where the approximating distribution is a finite mixture model (a non-ergodic Markov chain which is referred to as a "restricted chain" in the paper; this chain is ergodic though over the subspace of finite mixture models with K components). The sampler then transitions between different subspaces of the possible distributions (each subspace corresponding to a finite mixture model with different K) via merge-split Metropolis–Hastings proposals. The split proposals, however, depend linearly on α, while the merge proposals depend linearly on α^{-1} (note that this is true for both random and non-random split-merge moves as referred to in the paper). This means that for large values of α almost all merge proposals would be rejected, while for small values of α almost all split proposals would be rejected (even though for small values of α the number of cluster increases as the dataset size increases). Thus, the inference becomes very susceptible to initialisation – initialising the sampler with the data points randomly allocated to a large number of clusters would lead the sampler to merge the clusters quickly for small α but very slowly for large α, while initialising the sampler with the data points randomly allocated to a small number of clusters would lead the sampler to split the clusters quickly for large α but very slowly for small α.

We ran an experiment demonstrating this for $\alpha = 0.2, 1, 5$ using samples from the prior produced in the same way as in the previous section, and evaluated the average number of splits and merges (both random and non-random) per iteration. The results are shown in table 6. This dependence on α in the split-merge proposals makes (Chang & Fisher III, 2013)'s inference suitable for the case when *the posterior is known in advance* and the initialisation can reflect that. However we suspect that introducing additional random moves that depend on α in an inverse way might remove that limitation. We were not able to compare the overhead created by time spent in the cluster maintenance communication between the nodes as the implementation was done for a single-machine architecture.

	100 initial clusters		1 initial cluster	
	splits	merges	splits	merges
$\alpha = 0.2$	0.00	1.48	0.03	0.00
$\alpha = 1$	0.01	1.29	0.03	0.00
$\alpha = 5$	0.32	0.16	0.15	0.00

Figure 6. Average splits and merges (both random and non-random) per iteration in (Chang & Fisher III, 2013)'s inference. Samples from the prior were used with concentration values $\alpha = 0.2, 1, 5$ and different numbers of initial clusters. For large α most merges are rejected, and for small α most split are rejected. Note that for 1 cluster initialisation all merges are rejected as expected.

6.4. Directions for future research

An alternative to the inference procedures above might be the development of better approximate parallel inference. The current approach uses Gibbs sampling after distributing the data evenly across the different nodes (Asuncion et al., 2008). We synchronise the state of the nodes only in the global step, which means that the distribution would diverge from the true posterior. Williamson et al. (2013) reported this inference to have slow convergence in practice, which raises the question of whether this approximate parallel inference can be adjusted to have better mixing. For many problems an approximate posterior would suffice, and this inference might be a suitable alternative to non-approximate inference.

Lastly, one can also use distributions alternative to the Dirichlet process for clustering. Miller & Harrison (2013) have recently shown that the Dirichlet process posterior is inconsistent in the number of cluster and suggested an alternative distribution for clustering: the use of a Poisson mixture of Dirichlet distributions. The use of this alternative distribution might open the door for more efficient parallel inference. Furthermore, one might want to use a partly-parametric partly-nonparametric mixture of the Dirichlet-K distribution and Dirichlet process for clustering. This would allow us to use an unbounded number of clusters with at least K clusters, where the distribution of the load would be partly balanced.

7. Conclusions

In this paper we presented an asymptotic analysis as well as an empirical analysis of the parallel inference introduced by Lovell, Adams, and Mansingka (2012) and Williamson, Dubey, and Xing (2013). We showed that the inference doesn't satisfy the conditions one would expect of a distributed sampler, suggesting that it would lead to a waste of computational resources.

We continued to present experimental support of this pathology where we evaluated the proposed inference procedure analysing the average time spent in each node for different initialisations and datasets. Finally, we assessed the best case scenarios arising from different initialisations and number of nodes, showing that a small improvement can be achieved but that the balance is still skewed. We further explored the strengths and weaknesses of other approaches to parallel inference and proposed new possible approaches to research.

Acknowledgments

The authors would like to thank Dr Christian Steinruecken, Dr Daniel Roy, and Dr Jose Miguel Hernandez Lobato for reviewing the paper and their helpful comments. Yarin Gal is supported by the Google European Fellowship in Machine Learning.

References

Ahmed, Amr, Ho, Qirong, Teo, Choon H, Eisenstein, Jacob, Xing, Eric P, and Smola, Alex J. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *International Conference on Artificial Intelligence and Statistics*, pp. 101–109, 2011.

Arratia, Richard, Barbour, Andrew D, and Tavaré, Simon. *Logarithmic Combinatorial Strucutures: A Probabilistic Approach*. European Mathematical Society, 2003.

Asuncion, Arthur U, Smyth, Padhraic, and Welling, Max. Asynchronous distributed learning of topic models. In

Advances in Neural Information Processing Systems, pp. 81–88, 2008.

Blei, David M and Jordan, Michael I. Variational methods for the Dirichlet process. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 12. ACM, 2004.

Brockwell, A. E. Parallel Markov Chain Monte Carlo simulation by Pre-Fetching. *Journal of Computational and Graphical Statistics*, 15(1):pp. 246–261, 2006. ISSN 10618600.

Chahuneau, Victor, Schulam, Peter, and Gadde, Phani. Faster unsupervised morphology induction. Technical report, School of Computer Science, Carnegie-Mellon University, 2013.

Chang, Jason and Fisher III, John W. Parallel sampling of DP mixture models using sub-cluster splits. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 620–628. 2013.

Deka, Biplab, Birklykke, Alex A, Duwe, Henry, Mansinghka, Vikash K, and Kumar, Rakesh. Markov chain algorithms: A template for building future robust low power systems. 2013.

Doshi, Finale, Knowles, David, Mohamed, Shakir, and Ghahramani, Zoubin. Large scale non-parametric Bayesian inference: Data parallelisation in the Indian Buffet Process. In *Proceedings of NIPS*, 2009.

Fearnhead, Paul. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

Ferguson, Thomas S. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pp. 209–230, 1973.

Fox, Emily B, Sudderth, Erik B, Jordan, Michael I, and Willsky, Alan S. Nonparametric Bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems*, pp. 457–464, 2008.

Gal, Yarin and Blunsom, Phil. A systematic Bayesian treatment of the IBM alignment models. In *Proceedings of NAACL-HLT*, pp. 969–977, 2013.

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

Ida, Yasutoshi, Nakamura, Takuma, and Matsumoto, Takashi. Domain-dependent/independent topic switching model for online reviews with numerical ratings. In *Proceedings of the 22Nd ACM International Conference on Conference on Information #38; Knowledge Management*, CIKM '13, pp. 229–238, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2263-8.

Jain, Sonia and Neal, Radford M. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):pp. 158–182, 2004. ISSN 10618600.

Kingman, JFC. The representation of partition structures. *Journal of the London Mathematical Society*, 2(2):374–380, 1978.

Kurihara, Kenichi, Welling, Max, and Teh, Yee Whye. Collapsed variational Dirichlet process mixture models. In *IJCAI*, volume 7, pp. 2796–2801, 2007.

Lovell, Dan, Adams, Ryan P, and Mansingka, VK. Parallel Markov chain Monte Carlo for Dirichlet process mixtures. In *Workshop on Big Learning, NIPS*, 2012.

Miller, Jeffrey W and Harrison, Matthew T. A simple example of Dirichlet process mixture inconsistency for the number of components. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 199–206. 2013.

Sethuraman, J. A constructive denition of Dirichlet priors. *Statistica Sinica*, 1994.

Teh, Yee Whye. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 985–992. Association for Computational Linguistics, 2006.

Teh, Yee Whye, Jordan, Michael I, Beal, Matthew J, and Blei, David M. Hierarchical Dirichlet processes. *Journal of the american statistical association*, 101(476), 2006.

Ülker, Yener, Günsel, Bilge, and Cemgil, Ali T. Sequential Monte Carlo samplers for Dirichlet process mixtures. In *International Conference on Artificial Intelligence and Statistics*, pp. 876–883, 2010.

Wilkinson, Darren J. Parallel Bayesian computation. In Kontoghiorghes, Erricos John (ed.), *Handbook of Parallel Computing and Statistics*, volume 184, pp. 477–508. Chapman and Hall/CRC, Boca Raton, FL, USA, 2005.

Williamson, Sinead, Dubey, Avinava, and Xing, Eric P. Parallel Markov Chain Monte Carlo for nonparametric mixture models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 98–106, 2013.

Xu, Jia, Gao, Jianfeng, Toutanova, Kristina, and Ney, Hermann. Bayesian semi-supervised Chinese word segmentation for statistical machine translation. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pp. 1017–1024, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6.