Parallelizing LDA using Partially Collapsed Gibbs Sampling

Måns Magnusson* Leif Jonsson† Mattias Villani‡ David Broman§

October 7, 2021

Abstract

Latent dirichlet allocation (LDA) is a model widely used for unsupervised probabilistic modeling of text and images. MCMC sampling from the posterior distribution is typically performed using a collapsed Gibbs sampler that integrates out all model parameters except the topic indicators for each word. The topic indicators are Gibbs sampled iteratively by drawing each topic from its conditional posterior. The popularity of this sampler stems from its balanced combination of simplicity and efficiency, but its inherently sequential nature is an obstacle for parallel implementations. Growing corpus sizes and increasing model complexity are making inference in LDA models computationally infeasible without parallel sampling. We propose a parallel implementation of LDA that only collapses over the topic proportions in each document and therefore allows independent sampling of the topic indicators in different documents. We develop several modifications of the basic algorithm that exploits sparsity and structure to further improve the performance of the partially collapsed sampler. Contrary to other parallel LDA implementations, the partially collapsed sampler guarantees convergence to the true posterior. We show on several well-known corpora that the expected increase in statistical inefficiency from only partial collapsing is smaller than commonly assumed, and can be more than compensated by the speed-up from parallelization for larger corpora.

Keywords: Topic models, bayesian variable selection, parallelism

1 Introduction

Topic modeling or Latent dirichlet allocation (LDA) is an immensely popular way to model text probabilistically. LDA generates documents as probabilistic mixtures of topics. A given document d is assigned a vector θ_d which is a probability distribution over K topics. Each topic is a probability distribution ϕ_k over a dictionary of words. Each word position i in a document d is accompanied with a latent topic indicator z_i generated from θ_d, such that $z_i = k$ means that the word in the ith position is generated from ϕ_k. Let θ denote the set of all θ_d, $\mathbf{z}$ the vector with all z_i in all documents, and let Φ be a $K \times V$ matrix whose kth row holds ϕ_k over a vocabulary of size V. The basic LDA model has been developed further by incorporating author information (Rosen-Zvi et al., 2010), trends (Wang and McCallum, 2006) and supervision (Zhu et al., 2013) to give a few examples.

One of the most popular inferential techniques for topic models is Markov Chain Monte Carlo (MCMC) and the collapsed Gibbs sampler introduced by Griffiths and Steyvers (2004). It is a useful building block to use in other more advanced topic models, but it suffers from its sequential nature, which makes the algorithm practically impossible to parallelize in a way that still generates samples from the correct invariant distribution. This is serious problem as textual

*Department of Computer and Information Science, Linköping University.
†Department of Computer and Information Science, Linköping University and Ericsson AB.
‡Department of Computer and Information Science, Linköping University.
§School of Information and Communication Technlogy, KTH Royal Institute of Technology and the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

data are growing at an increasing rate; some recent applications of topic models are counting the number of documents in the millions and even billions (Yuan et al., 2014). The computational problem is further aggravated since large corpora typically affords more complex models with a larger number of topics.

The response to these computational challenges has been to settle with approximate methods such as variational Bayes. Another solution has been to to use distributed approximations of the collapsed sampler, such as the popular AD-LDA algorithm by Newman et al. (2009). Unfortunately, these approximate methods come without bounds on the total approximation error for the joint posterior (Ihler and Newman, 2012), and the only way to check the accuracy in a given application is to compare the inferences to an MCMC sampler with convergence guarantees.

A natural way to obtain a parallelized Gibbs sampler is to only collapse over the topic proportions $\theta_1, ..., \theta_D$ in each document. The remaining parameters can then be sampled by Gibbs sampling by iterating between the two updates $\mathbf{z}|\Phi$ and $\Phi|\mathbf{z}$, where the topic indicators z_i are conditionally independent between documents and the rows of the topic-word matrix Φ are independent given $\mathbf{z}$. This means that the first step can be parallelized with regard to documents, and the second step can be parallelized with regard to topics, thereby obtaining large speedup over a sequential fully collapsed sampler.

A partially collapsed sampler is noted in Newman et al. (2009), but quickly dismissed because of lower MCMC efficiency than the collapsed sampler. There are well known theoretical results that collapsing generally gives a more efficient MCMC sampler Liu et al. (1995), but how much the efficiency is increased by collapsing parameters is model-specific and must weighed against the benefits of parallelization. One of our contributions is to show empirically that the efficiency loss from using a partially collapsed sampler for LDA compared to a fully collapsed Gibbs sampler is actually small. This result is consistent across different well-known datasets and for different model settings using multicore machines. Furthermore, we show theoretically, under some mild assumptions, that despite the additional sampling of the Φ matrix, the complexity of our sampler is still only $O(\sum_i^N K_{d(i)})$, where N is the total number of tokens in the corpus and $K_{d(i)}$ is the number of nonzero topics in the document of token i.

We also propose several extensions and refinements of the basic PC-LDA sampler to decrease the sampling complexity of the algorithm. First, we propose a Gibbs sampling version of the Walker-Alias tables proposed by Li et al. (2014b) together with binary search proposed by Yu et al. (2014b). We also explore random scan strategies to sample selectively chosen parts of Φ less frequently. Finally, we note that partial collapsing makes it possible to use more elaborate models on Φ for which the fully collapsed sampler can not be applied. As an example, we develop a spike-and-slab prior where we set elements of Φ to zero using ordinary Gibbs sampling, a type of model shown to improve of topic model performance (Chien and Chang, 2014).

2 Related work

The problem of parallelizing topic models has been studied extensively (Ihler and Newman, 2012; Liu et al., 2011; Newman et al., 2009; Qiu et al., 2014; Smola and Narayanamurthy, 2010; Smyth et al., 2009; Yan et al., 2009; Yu et al., 2014b; Ahmed et al., 2012) together with ways of improving the sampling efficiency of the collapsed sampler (Yuan et al., 2014; Porteous et al., 2008; Xiao and Stibor, 2010; Yao et al., 2009; Li et al., 2014a). The standard sampling scheme for the topic indicators is the collapsed Gibbs sampler of Griffiths and Steyvers (2004) where each z_i is sampled from:

$$P(z_i = j|\mathbf{z}_{-i}, \mathbf{w}) \propto \underbrace{\frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta}}_{topic-word} \underbrace{\left(n_{-i,j}^{(d_i)} + \alpha\right)}_{document-topic}, \tag{1}$$

where z_i is the topic indicator of word i and $\mathbf{z}_{-i}$ are all other topic indicators in the corpus. The scalars α and β are prior hyperparameters on θ and Φ: $\theta_d \overset{iid}{\sim} \text{Dirichlet}(\alpha)$ and $\phi_k \overset{iid}{\sim} \text{Dirichlet}(\beta)$. The $n^{(w)}$ are the counts of topic indicators for each topic and word type and $n^{(d)}$ are the topic counts for each topic and document. This sampler is sequential in nature due to the fact that sampling one topic indicator is conditionally dependent on *all other* topic indicators in the whole corpus. This makes it difficult to parallelize it in an efficient way, a problem that is similar for the collapsed variational Bayes algorithm (Asuncion et al., 2009).

The Approximate Distributed LDA (AD-LDA) in Newman et al. (2009) is currently the most common way to parallelize topic models. The idea is that each processor works in parallel with a given set of topic counts in the word-topic count matrix $n^{(w)}$. The word-topic matrices at the different processors are synced after each complete update cycle. This is an approximation of the collapsed sampler since the word-topic matrix available on each local processor is sampled in isolation from all other processors. The resulting algorithm is not guaranteed to converge to the target posterior and will in general not do so. However, Newman et al. (2009) find that this approximation works rather well in practice. A bound for the error of the AD-LDA approximation for the sampling of each topic indicators has been derived by Ihler and Newman (2012). They find that the error of sampling each topic indicator increases with the number of topics and decreases with smaller batch sizes per processor and the total data size (Ihler and Newman, 2012).

Other algorithmic improvements besides parallelization is suggested to improve the efficiency and speed of the collapsed sampler, see the fast-LDA sampler of Porteous et al. (2008) for an early attempt. A sparse LDA sampler is introduced by Yao et al. (2009), making the topic indicator sampling both more memory and time efficient. Sparse LDA reduces the sampling of each topic indicator from $O(K)$ to $O(\max(K_w, K_d))$ where K_w is the number of non-zero topics for the given word type and K_d is the number of non-zero topics in the given document.

Another approach to increase the speedup of the algorithm is by using Walker-Alias tables, proposed by Li et al. (2014a). This method uses similar ideas as in sparse LDA (Yao et al., 2009) but introduce the Walker-Alias method to reduce the complexity of some of the draws to $O(1)$ amortized. Unfortunately, it is not possible to use the Walker-Alias method in a traditional collapsed Gibbs sampler without resorting to the Metropolis-Hastings algorithm. Based on the idea of Walker-Alias tables (Yuan et al., 2014) suggested using a cyclical Metropolis-Hastings algorithm where the proposals for each topic indicator is cycled between a word-topic proposal $p_w(k)$ and a document-topic $p_d(k)$ proposals reducing the sampling complexity to $O(1)$ but introducing inefficiency due to the usage of the Metropolis-Hastings algorithm. Yet another approach is proposed by Yu et al. (2014a) where the problem with updating the cumulative sum for sampling topics is solved using Fenwick Trees. This data structure can update a cumulative sum in $O(\log(K))$ and sample a topic indicator in $O(\log(K))$ reducing the complexity of the sampling but still making use of a Gibbs sampler.

3 Partially Collapsed Gibbs sampling for topic models

3.1 The basic partially collapsed Gibbs sampler

The main problem in parallelizing the sampler for LDA is that it is sequential. Both parameter matrices Θ and Φ from the model have been integrated out, thereby making the sampling of each topic indicator z_i dependent on all the other topic indicators $\mathbf{z}_{-i}$ in the corpus.

The PC-LDA sampler simulates from the joint posterior of $\mathbf{z}$ and Φ by iteratively sampling from the conditional posterior $p(\mathbf{z}|\Phi)$ followed by sampling from $p(\Phi|\mathbf{z})$. Note that the topic proportions $\theta_1, ..., \theta_D$ have been integrated out in both updating steps and that both conditional posteriors can be obtained in analytical form due to conjugacy. The advantage of only collapsing over the θ's is that the update from $p(\mathbf{z}|\Phi)$ can be parallelized over documents and the update from $p(\Phi|\mathbf{z})$ can be parallelized over topics (the rows of Φ are conditionally independent). This gives the following sampler where we first sample the topic indicators for each document in parallel as

$$p(z_i = k|\mathbf{z}_{-i}^{(d)}, \Phi, w_i) \propto \phi_{k,w} \cdot \left(n_{-i,j}^{(d_i)} + \alpha_{d,k}\right) \tag{2}$$

and then sample the rows of Φ in parallel as

$$\phi_k \sim \text{Dir}(n_k^{(\mathbf{w})} + \beta_k^{(\mathbf{w})}). \tag{3}$$

In the following subsection we propose a number of improvements of the basic PC-LDA sampler to reduce the complexity of the algorithm and to speed up computations.

3.2 Sparse Gibbs sampling using the Walker-Alias method on $\mathbf{z}|\Phi$

The sampling of $\mathbf{z}|\Phi$ in the basic PC-LDA sampler is of complexity $O(K)$ per topic indicator, making the sampling time linear in the number of topics. The Walker-Alias method in Li et al. (2014a) exploits local sparsity and can be straightforward extended to the PC-LDA sampler by decomposing

$$\begin{aligned} p(z_i = k|\mathbf{z}_{-i}^{(d)}, \Phi, w_i) &\propto \phi_{k,w} \cdot \left(\alpha_{d,k} + n_{-i,j}^{(d_i)}\right) \\ &= \underbrace{\phi_{k,w} \cdot \alpha_{d,k}}_{a} + \underbrace{\phi_{k,w} \cdot n_{-i,j}^{(d_i)}}_{b}. \end{aligned}$$

To sample a topic indicator for a given token we first need to calculate the normalizing constant

$$q(z) = \sum_{k=1}^{K} \phi_{k,w} \cdot \left(\alpha_{d,k} + n_{-i,j}^{(d_i)}\right) = \sigma_a + \sigma_b$$

where $\sigma_a^w = \sum_{k=1}^{K} \phi_{k,w} \cdot \alpha_{d,k}$ and $\sigma_b^w = \sum_{k=1}^{K} \phi_{k,w} \cdot n_{-i,j}^{(d_i)}$.

With this decomposition we can use the sparsity of the local topic counts $n_{-i,j}^{(d_i)}$ to calculate σ_b by only iterating over the non-zero topic counts. This reduces the complexity from $O(K)$ to $O(K_d)$ where K_d is the number of non-zero topics in a given document.

Following the approaches of sparse-LDA (Yao et al., 2009) and Alias-LDA (Li et al., 2014a) we can now sample z_i as follows. First calculate σ_b and the cumulative sum over non-zero topics in the document. Draw a $U \sim \mathcal{U}(0, \sigma_a + \sigma_b)$. If $U \leq \sigma_a$, we use the Walker-Alias method to sample a topic indicator with $O(1)$. If $U > \sigma_a$, we choose a topic indicator using binary search proposed by Xiao and Stibor (2010) with complexity $O(\log(K_d))$.

Conditioning on Φ give us a couple of advantages compared to the original Walker-Alias method for the collapsed sampler. First, the Alias-Walker method can be used in a Gibbs sampler instead of using it as a proposal in a Metropolis-Hasting algorithm. Conditioning on Φ also makes it possible to create one alias table per word type and per iteration and then reuse these tables in sampling $\mathbf{z}$. As a by-product of calculating the Walker-Alias tables we also calculate the normalizing constant σ_a that can be stored and reused in sampling $\mathbf{z}$. The Alias-LDA sampler by Li et al. (2014a) construct a new alias table every kth iteration, making it amortized $O(1)$ but the number of times the Alias table is created depends on N. Using a partially collapsed version implies that we only need to calculate one Alias table per iteration and this can also easily be done in parallel.

3.3 Computational improvements

3.3.1 Cached Marsaglia sampling for Φ

The Dirichlet can be seen as a normalized sum of gamma distributed variables

$$\phi_i \stackrel{d}{=} \frac{\gamma_i}{\sum_j^W \gamma_j}$$

where $\gamma_i \sim \Gamma(n_k^{(w_i)} + \beta_k^{(w_i)}, 1)$.

To sample the Dirichlet distribution we use the method of Marsaglia and Tsang (2000) to sample the gamma variables efficiently. With this sampler it is also possible to increase the speed further by using the sparsity in the $n^{(w)}$ count matrix. We do this by caching $d = \beta - \frac{1}{3}$ and $c = 1/\sqrt{9d}$ when $n_k^{(w_i)} = 0$ for a given parameter in Φ. In most applications, the $n^{(w)}$ count is very sparse and we can hence increase the speed of sampling each γ_i.

3.3.2 Load balancing through job stealing

A problem with parallelizing topic models taking advantage of sparsity in sampling is that it is difficult to distribute documents evenly over the different cores. Even though it is possible to distribute documents initially using a greedy heuristic, the load will become uneven due to the random document sparsity. The partially collapsed approach allows us to rearrange document between cores *during the sampling* of $\mathbf{z}|\Phi$. This is much more difficult using a collapsed sampling approach since the fully collapsed sampler is conditional on all the other topic indicators. The job stealing approach described in Lea (2000) allows cores that are finished sampling their allocated documents to steal jobs from other cores that are not finished, thus evening out the load over the cores during sampling. We have seen a significant improvement using this approach on large datasets, improving the total running time of the sampler by up to 30% for the PUBMED corpus. On small datasets such as Enron and NIPS the job stealing degrades performance probably by introducing overhead in terms of a larger number of threads used and in the job stealing step which introduces synchronization between threads. On our datasets and with the settings we use, job stealing decreases performance on Enron with 20 topics but increases performance on the Enron dataset with 100 topics.

3.3.3 Improved cache performance through optimal array storage

Different parts of the sampling process accesses the Φ matrix in different ways. In the topic indicator sampling stage the Φ matrix is typically accessed by word type, while in the Φ sampling stage the Φ matrix is accessed in a topic fashion. By transposing the Φ matrix between the two sampling stages, storing it in type order during topic indicator sampling and in topic order during Φ sampling, we can improve the cache performance of the sampler.

3.4 Time complexity of the sampler

Compared to the collapsed Gibbs sampler, the PC-LDA sampler adds the cost of sampling each element in Φ. We argue that this cost is small in comparison to the cost of sampling $\mathbf{z}$ when the number of tokens in the corpus is large.

In the basic collapsed sampler introduced by Griffiths and Steyvers (2004) the sampling complexity per iteration is $O(N \cdot K)$. By taking advantage of sparsity the complexity of sparse-LDA in Yao et al. (2009) is reduced to $O(\sum_i^N \max_i(K_{d(i)}, K_{w(i)}))$ and $\Omega(\sum_i^N \min_i(K_{d(i)}, K_{w(i)}))$ where $K_{d(i)}$ is the number of nonzero topics in the document of token i and $K_{w(i)}$ is the number of nonzero topics in the word type w. But as has been shown in Li et al. (2014a) this complexity will reduce to $O(N \cdot K)$ and $\Omega(\sum_i^N K_{d(i)})$ as the number of documents grows large.

It is easy to see that the complexity of sampling $\mathbf{z}$ in the PC-LDA sampler is $O(\sum_i^N K_{d(i)})$, but unlike fully collapsed samplers, we also need to sample Φ. It is therefore of interest to study how the Φ matrix (i.e. $K \cdot V$) grows with the number of tokens (N), to determine the overall complexity of the PC-LDA sampler. In most languages the number of word types follows quite closely Heaps' law, which models the relationship between word types and tokens as follows

$$V(N) = \xi N^{\varphi},$$

where $\varphi < 1$. Typical values for of φ is within the range of 0.4 to 0.6 and ξ varies between 5 and 50 depending on the corpus (Araujo et al., 1997).

The number of topics, K, in large corpuses is often modeled by a Dirichlet process mixture where the expected number of topics is (Teh, 2010)

$$E(K(N)) = \alpha \log\left(1 + \frac{N}{\alpha}\right).$$

Based on these assumptions we get the following proposition.

Proposition 1 *Assuming a vocabulary size following Heaps' law $V = \xi N^{\varphi}$ with $\xi > 0, 0 < \varphi < 1$ and the number of topics following the mean of a Dirichlet process mixture $K = \alpha \log\left(1 + \frac{N}{\alpha}\right)$ with $\alpha > 0$, the complexity for the PC-LDA sampler is*

$$O\left(\sum_{i=1}^{N} K_{d(i)}\right).$$

Proof. We need to show that there exists a finite $c > 0$ and a real number N_0 such that for all $N \geq N_0$

$$\sum_i^N K_{d(i)} + \alpha\xi \log\left(1 + \frac{N}{\alpha}\right) N^{\varphi} \quad \leq \quad c \sum_i^N K_{d(i)},$$

or equivalently that

$$1 + \alpha\xi \log\left(1 + \frac{N}{\alpha}\right) N^{\varphi} / \sum_i^N K_{d(i)} \leq c.$$

Using the fact that $N \leq \sum_i^N K_{d(i)}$, it is sufficient to prove that for $N \geq N_0 = 1$, there exist a c such that

$$1 + \alpha\xi \log\left(1 + \frac{N}{\alpha}\right) N^{\varphi} / N \leq c. \tag{4}$$

If we show that the result holds for Equation 4 then we also know that it holds for our original problem since $N \leq \sum_i^N K_{d(i)}$ which means that the inequality 4 still holds. Let

$$f(N) = \log\left(1 + \frac{N}{\alpha}\right) / N^{1-\varphi},$$

and note that

$$f(1) = \log(1 + \alpha^{-1}) > 0 \text{ and } \lim_{N \to \infty} f(N) = 0,$$

by the use of the standard limit

$$\lim_{x \to \infty} \left(\frac{\log(x)}{x^b}\right) = 0 \text{ for } b > 0.$$

By the continuity of $f(N)$ there exist an R such that for all $N > R, f(N) < f(1)$. Using the extreme value theorem we know that at the interval $[1, R]$ there exist an $M = \sup f(N)$. Hence, for $N \geq N_0 = 1$, there exists a c such that $c \geq 1 + \alpha\xi M$, which completes the proof.

3.5 Random scan sampling of $\Phi|\mathbf{z}$

The extra computations spent on sampling Φ in the PC-LDA sampler are small compared the cost of sampling $\mathbf{z}$ when the number of topics is moderate, see Section 4.5. For models with a large number of topics, the sampling of Φ starts to becomes more costly. For those situations, we study a random scan sampler that spends relatively more time on sampling elements of Φ for the word types that are the most common in the corpus. A large share of the Φ parameters typically correspond to very rare word types, which therefore contribute very little to the likelihood function. Random scan sampling can exploit this and update those insignificant parameters less frequently than ϕ's for common word types, thereby reducing computing time without sacrificing too much computational efficiency.

In adaptive random scan Gibbs sampling, a given parameter ϕ_j is updated at Gibbs iteration n with a probability a_{jn} (Łatuszyński et al., 2013). Let $\mathbf{a}_n$ be a vector with the selection probability for all parameters at iteration n. In each iteration we start out by first randomly choosing which parameters to update and then updating the selected parameters using ordinary Gibbs sampling. From Theorem 4.2 and Proposition 4.8 in (Łatuszyński et al., 2013) we know that a random scan Gibbs sampler will be ergodic if the systematic Gibbs sampler is ergodic and if the selection probabilities satisfy

$$|\mathbf{a}_{n-1} - \mathbf{a}_n| \xrightarrow{p} 0 \text{ as } n \to \infty,$$

where $\xrightarrow{p}$ denotes convergence in probability. One simple way to ensure this condition is to set $\mathbf{a}_n$ to a vector with fixed selection probabilities.

Let the kth row of Φ be decomposed into elements that have been selected for sampling at a given iteration, $\phi_{k,\mathcal{S}}$, and the remaining non-selected elements $\phi_{k,\mathcal{S}^c}$. It then follows from Ng et al. (2011) that

$$\frac{\phi_{k,\mathcal{S}}}{1 - \sum_{s \in \mathcal{S}^c} \phi_{k,s}} \sim \text{Dir}\left(\phi_{k,\mathcal{S}} | \phi_{k,\mathcal{S}^c}\right). \tag{5}$$

This allows us to update only a part of each ϕ_k without needing to iterate over all elements of the ϕ_k vector. If we sample a given $\phi_k^{(w)}$ we need to recalculate the whole Walker-Alias table for that word type. To reduce redundant computations we therefore suggest to sample all ϕ's for a given word type by first randomly selecting word types and then use the conditional Dirichlet distribution for each topic. The cost of sampling each $\phi_k^{(w)}$ is the same but a larger $n_k^{(w_i)}$ increases the acceptance probability of the gamma sampler.

We would like to sample word types with large $n_k^{(w)}$ to cover as large part of the likelihood as possible while minimizing the number of ϕ's to sample. Our suggestion is to sample word types proportional to the number of tokens with a systematic probability proportional to size approach. We have an array with the cumulative counts of each word type and then choose the word types for each l:th token, giving the sampling threshold. Word types with more than l tokens will then be sampled with probability 1 and the others with probability $n_\cdot^{(w)}/l$.

3.6 Variable selection in Φ

PC-LDA has the additional advantage that more complex models can be used for Φ. As an example we derive a Gibbs sampler for a topic model with a spike-and-slab type prior for Φ that assigns point masses at zero to a subset of the parameters in Φ, thereby providing more clear-cut and interpretable topics. Variable selection is usually associated with increased computation costs. An interesting aspect of variable selection on Φ is that it reduces the computational cost of sampling $\mathbf{z}$. The variable selection approach has previously been proposed by Chien and Chang (2014) using variational Bayes.

The rows of Φ are assumed to be independent a priori, exactly like in the original LDA prior, so let us focus on a given row ϕ_k of Φ. Let $n_k^{(w)}$ be the kth row of $n^{(w)}$, and let $I_k = (I_{k1}, ..., I_{kV})$ be a vector of binary variable selection indicators such that $I_{kv} = 1$ if $\phi_{kv} \neq 0$, and $I_{kv} = 0$ if $\phi_{kv} = 0$. Define I_k^c to be the complement of I_k, i.e. the vector indicating the zeros in ϕ_k. Let ϕ_{k,I_k} be a vector with nonzero elements of ϕ_k. Finally, let $\tilde{n}_k$ be the total number of tokens associated with the kth topic. Our variable selection Dirichlet prior is of the form

$$\begin{aligned} I_{k1}, ..., I_{kV} \mid \pi_k &\overset{iid}{\sim} Bernoulli(\pi_k) \\ \phi_{k,I_k} \mid I_k &\sim Dirichlet(\beta), \end{aligned}$$

and $\phi_{k,I_k^c} = 0$ with probability one. It is important to note that this prior is obtained by imposing the zero restrictions on the general Dirichlet prior in Ng et al. (2011) which makes the prior consistent over all sets of zero restrictions.

The posterior sampling step for ϕ_k is replaced by sampling from $\phi_{k,I_k}|I_k, w$ from the conditional Dirichlet distribution in Ng et al. (2011), followed by sampling from $I_{kv} \mid I_{-kv}, \pi_k, w$, for $v = 1, ..., V$, where I_{-kv} contains all of I_k except I_{kv}. By integrating over ϕ_{k,I_k}, it is straightforward to show that $I_{kv}|I_{-kv}, \pi_k, w$ is Bernoulli with

$$\Pr\left(I_{kv} = j|I_{-kv}, \pi_k, w\right) \propto \frac{\Gamma\left(\beta\right)}{\Gamma\left(\tilde{n}_k + (\sum_{j \neq v} I_{kj} + j)\beta\right)} \pi_k^j (1 - \pi_k)^{1-j}.$$

Note that $I_{kv} = 1$ with certainty if the vth word appears at least once in topic k.

4 Experiments

In the following sections we study the properties of our PC-LDA sampler and the proposed modifications of the basic algorithm. We compare our algorithm with the very efficient sparse LDA implementation of AD-LDA in MALLET (called sparse AD-LDA below). Note that this implementation reduces to an exact Sparse-LDA collapsed sampler when we are only using one core.

4.1 Experiments

4.1.1 Corpuses and settings

We use the same three datasets as in Newman et al. (2009) to evaluate the PC-LDA sampler. Some key summaries of the corpuses can be seen in Table 1.

Corpus	V	D	N
NIPS[1]	12 375	1 499	1 931 939
Enron[2]	28 099	39 860	6 412 150
PubMed[3]	141 043	∼8 200 000	∼785 000 000

Table 1: Corpuses with vocabulary size (V), the number of documents (D), and tokens (N).

In all experiments, we set the prior hyper-parameters β to 0.01 and α to 0.1. As shown in Section 4.5, the sparsity structures from this prior gives an advantage to sparse AD-LDA over the PC-LDA sampler. Following common practice, we remove rare words before fitting the models. We remove the most rare word types that together make up less than 0.1% of the whole

corpus (corresponding to 12 or less occurrences of a word for Nips, 14 for Enron and 150 for PubMed); stop words are removed using a standard stop word list.

We evaluate the convergence of the samplers using the log-likelihood marginalized with respect to Φ and θ, which is comparable across all algorithms. Since the convergence behavior depend on the initialization state of the Gibbs sampler we use the same seed to initialize the different samplers to the exact same starting state.

4.1.2 Hardware and software

The samplers are implemented in Java using version 2.0.7 of the MALLET framework (McCallum, 2002). All compared samplers are also MALLET implementations, with the aim of minimizing differences in results that come from implementations rather than underlying algorithms. The code of our sampler has been released as plug-in to the MALLET framework and can be found at GitHub[1](Jonsson, 2015). The computations have been performed on resources provided by Linkï¿œping University (LiU) at the National Supercomputer Centre (NSC). The speedup experiments were conducted using the HP Cluster Platform with DL170h G6 compute nodes with 4-core Intel Xeon E5520 processors at 2.2GHz. All experiments except the speedup study use 2 sockets (8 cores) with 24 GB memory nodes. In the speedup experiment we used 8 sockets (64 cores) with 1024 GB memory.

4.2 Efficiency of the partially collapsed sampler

The first experiment studies the efficiency of the new sampler compared to the collapsed sampler. This is motivated by the fact that integrating out parameters will always improve mixing (Liu et al., 1995). The question is how much worse the sampling efficiency of the PC-LDA sampler will be compared to a fully (sequential) collapsed sampler. The implementation we are using for this is the standard collapsed sampler included in the MALLET framework.

Each experiment starts with a given random seed and runs for 10 000 iterations (to ensure convergence) using the sequential collapsed Gibbs sampler (the gold standard). The topic indicators $\mathbf{z}$ in the last iteration is then used as initialization point for both a collapsed sampler and a PC-LDA sampler from which we subsequently perform two sub-runs with the collapsed sampler and two with the PC-LDA sampler per experimental setup.

The parameters Θ and Φ are subsequently sampled from $p(\Theta, \Phi|\mathbf{z})$:

$$\begin{aligned} \theta_{d,k}|\mathbf{z} &\sim \text{Dir}(n_k^{(d)} + \alpha) \\ \phi_{k,w}|\mathbf{z} &\sim \text{Dir}(n_k^{(w)} + \beta), \end{aligned}$$

for each of 2 000 iterations for both the collapsed and the PC-LDA. To calculate the total inefficiency of the MCMC sample we calculate the inefficiency for the 1 000 top words for each topic (Φ) and 1 000 random documents (Θ) making the estimate of inefficiency based on $2000 \cdot K$ parameters. To estimate the inefficiency factor for each parameter we compute the spectral density at 0 using the `coda` package (Plummer et al., 2006) in R.

The results from the the first experiment can be seen in Table 2 and 3; the other experiments gave very similar inefficiencies and are not reported. We conclude that the increase in inefficiency of the chains from not collapsing out Θ is unexpectedly low. Comparing the mean efficiency factor between the collapsed and the PC-LDA sampler it is easy to see that the PC-LDA sampler is only slightly less efficient than the collapsed (gold standard) for Θ, and the two samplers have very similar efficiency for the θ parameters (which are integrated out in both samplers).

[1]http://books.nips.cc
[2]http://cs.cmu.edu/ enron
[3]https://archive.ics.uci.edu/ml/datasets/Bag+of+Words
[1]https://github.com/lejon/PartiallyCollapsedLDA/

Data	Topics	Collapsed	PC-LDA	Ratio
Enron	20	3.312 (4.78)	3.535 (6.02)	1.067
Enron	100	2.205 (4.96)	2.287 (5.28)	1.037
NIPS	20	10.767 (32.01)	12.536 (46.56)	1.164
NIPS	100	6.635 (14.18)	7.449 (15.76)	1.123

Table 2: Mean inefficiency factors (standard deviation in parantheses) of Θ..

Data	Topics	Collapsed	PC-LDA	Ratio
Enron	20	5.002 (19.86)	5.031 (14.65)	1.0058
Enron	100	17.843 (49.22)	22.462 (58.00)	1.2588
NIPS	20	28.135 (73.45)	31.474 (81.05)	1.1187
NIPS	100	16.163 (43.12)	23.850 (55.55)	1.4756

Table 3: Mean inefficiency factors standard deviation in parentheses) of Φ.

4.3 Random scan

In most applications of topic models the number of topics are quite modest, but in some situations more extreme number of topics is needed. In those situations the sampling of the Φ matrix will dominate the total sampling time (as shown in Section 4.5). For such situations we propose to use random scan sampling to focus the sampling of Φ to those parts that contribute a lot to the likelihood by using systematic sampling proportional to tokens. To investigate the effect of convergence and computing time we run a total of five seeds for the sampling thresholds $l = 100, 200, 500$ and 1000 in the systematic sampling of word types for the Enron corpus. Since the Φ matrix is relatively cheap to sample for smaller number of topics, we study the effect for more extreme situations with $K = 100$ and $K = 1000$ for the Enron corpus.

We can see the resulting chains in Figure 1 and how the convergence in log-likelihood develops as a function of runtime. Each experiment is run a total of 10 000 iterations.

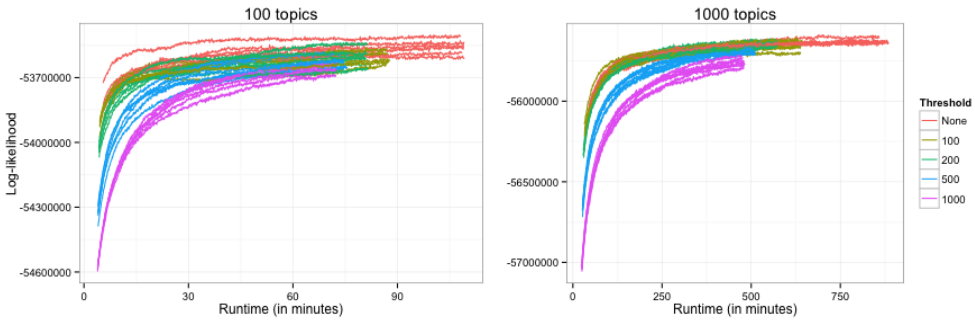

Figure 1: Random scan convergence for the Enron dataset.

It is clear from Figure 1 that the running time decreases with a higher threshold due to the fact that Φ is subsampled (shorter lines). But the efficiency of the samplers is getting worse in an even faster pace. So no one threshold value will improve over the full sampler where the whole Φ is be sampled in each iteration. It also seems like this effect is independent of K.

4.4 Full posterior error with approximate distributed LDA

One of the most popular parallelizations of LDA is using AD-LDA. We know that the approximation will have some effect on sampling each topic indicator, but we have not found any studies on the effect on the joint posterior distribution. We start the sampler with the same initial

state with respect to the topic indicators $\mathbf{z}$ and then run the sampler with different numbers of cores/partitions to see the effect on which mode the sampler converged to. The purpose is to study the effect of the approximation on the full joint posterior distribution $p(\mathbf{z})$. We studied the effect with different cores (partitions) both for the NIPS and Enron dataset for $K = 20$ and $K = 100$. To be sure that the effect of the posterior is not just due to an unlikely partition of the data we randomly reshuffled the documents in the different partition so that each experiment is run with a different random partition of the documents. We then study the effect for 10 different seeds per setup. Due to comptational problems a total of 6 seeds were removed.

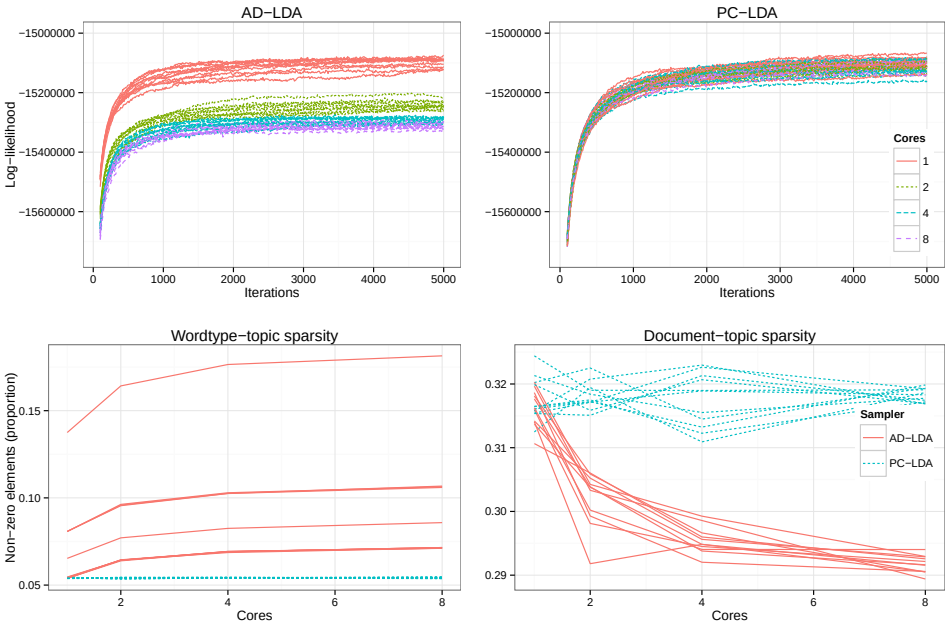

Figure 2: Log-likelihood convergence and sparsity for NIPS $K = 100$.

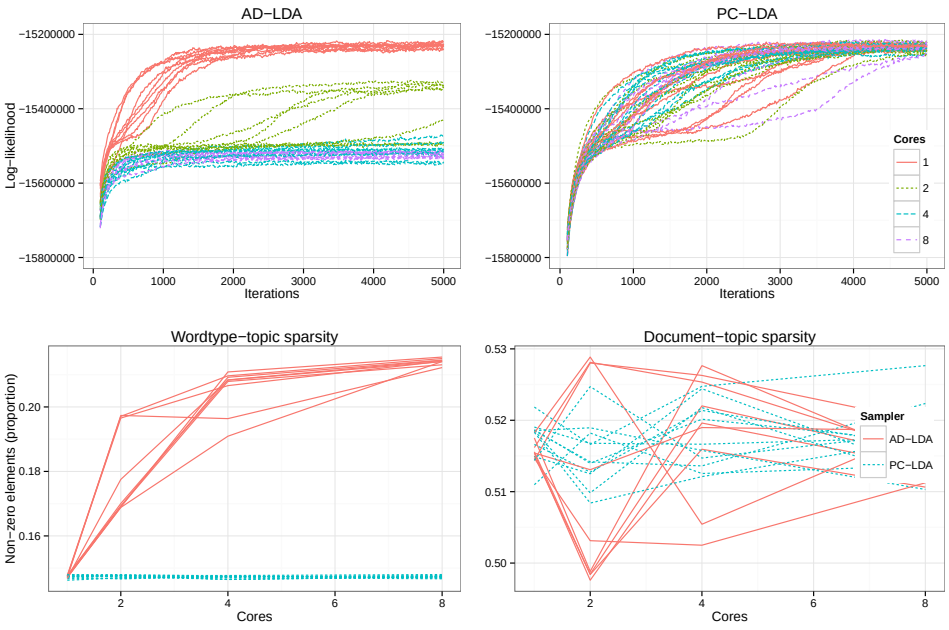

Figure 3: Log-likelihood convergence and sparsity for NIPS $K = 20$.

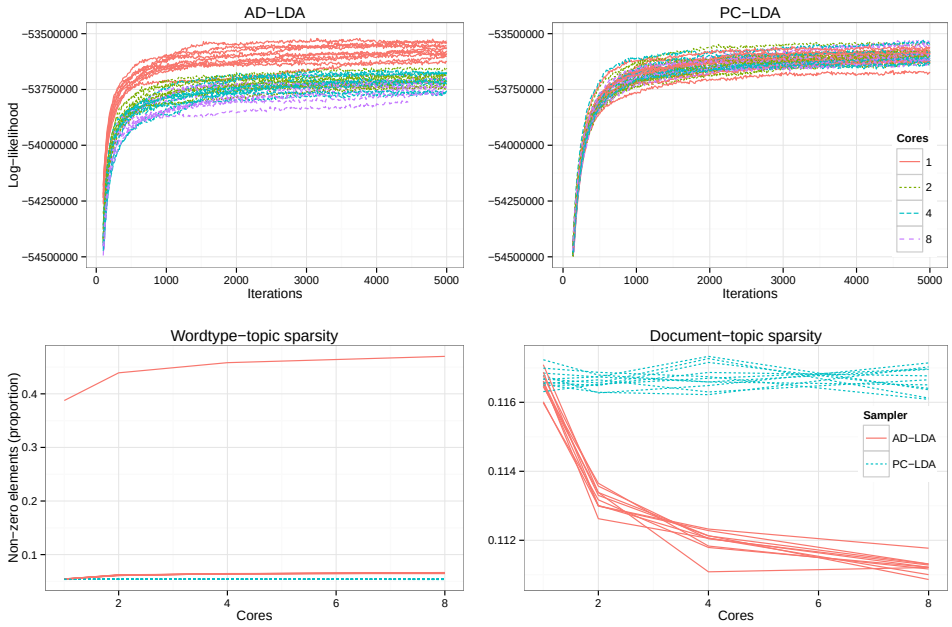

Figure 4: Log-likelihood convergence and sparsity for Enron $K = 100$.

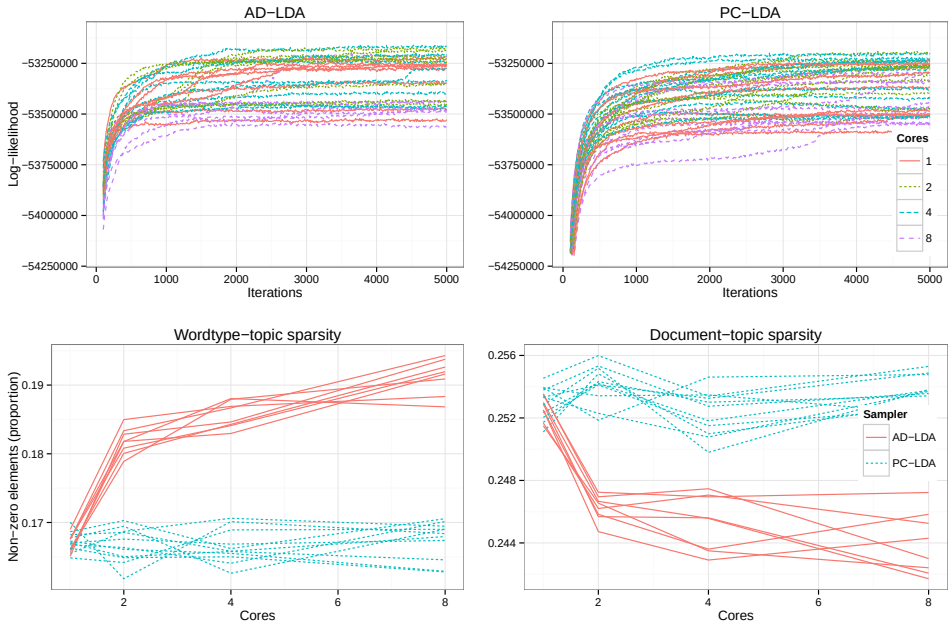

Figure 5: Log-likelihood convergence and sparsity for Enron $K = 20$.

With the exception of the Enron corpus with 20 topics, there is a clear tendency for sparse AD-LDA to converge to a lower mode as more cores are used to parallelize the sampler. To get some more insights into this convergence behavior of AD-LDA the figures also displays the sparsity of the $n^{(w)}$ and $n^{(d)}$ matrices (fraction of zero elements) as a function of the sampling iterations. The figures show that the sparsity of $n^{(w)}$ decreases while the sparsity in $n^{(d)}$ increases as more cores are added. We interpret theses results as the AD-LDA posterior drifting towards finding a better local model on each core (a more sparse $n^{(d)}$ matrix) and a less good global model (a more dense $n^{(w)}$ matrix). These empirical results should be followed up by a careful theoretical analysis, but that is beyond the scope of this paper.

We can also see that there can be quite different levels that the sparsity ends up into using the collapsed sampler. It is not uncommon that the collapsed sampler ends up in a less spare mode, most clearly this can bee seen in Figure 2 where the global matrix $n^{(w)}$ ends up in four different modes, while the partially collapsed sampler in all runs and examples end up in a more sparse mode this is indifferent to the number of cores.

In Ihler and Newman (2012) the error of sampling individual topic indicators is studied in detail and they are able to bound the error introduced by the approximation. They reason that it is not tractable to analyze the full posterior but assume that "the joint error will be decreased at each step by the progress of Gibbs sampling towards the stationary distribution" (Ihler and Newman, 2012). One of the results of Ihler and Newman (2012) is that the the error of sampling each topic indicator using AD-LDA is approximately $4\sqrt{T/NP}$, where T is the number of topics, N is the number of tokens and P is the number of partitions (cores). We can see a similar result for the joint posterior. When the number of topics increases in relation to the number of tokens, the sampler converges to a lower mode. But unlike the result in Ihler and Newman (2012) we also see a lower mode as the number of partitions increases. This effect can only be seen when the sampler is initialized from the same state for each run with different number of cores, which is probably why this problems has gone unnoticed in the literature. We have not found any large difference in the top words in the different modes on the Enron and NIPS data, but the results in Figure 2, 3, 4 and 5 clearly raise concerns about the effect of the AD-LDA approximation on the joint posterior in large parameter spaces. Especially in situations like those in Yuan et al. (2014) where the size of $n^{(w)}$ is larger than the number of tokens.

4.5 Prior effects and sampling time of Φ

Our choice of prior hyper-parameters $\alpha = 0.1$ and $\beta = 0.01$ in all experiments influences the sparsity of $n^{(w)}$ and $n^{(d)}$, and hence also the speed of the different samplers. Sparse LDA is mainly fast for sparse $n^{(w)}$ while PC-LDA and Alias LDA uses the sparsity of $n^{(d)}$. Figures 6 contrast the computing time per iteration of the different samplers for different values of α (0.1 and 0.01) and β (0.1 and 0.01) for the Enron corpus with $K = 100$ and $K = 1000$. It is clear from these figures that our benchmark prior with $\alpha = 0.1$ and $\beta = 0.01$ implies a sparsity that is actually disadvantageous for our PC-LDA sampler; PC-LDA gains in speed relative to AD-LDA if we switch to any of the three alternative prior settings. We can also see that the sampling of Φ does not dominate total computing time in a model with 100 topics on a medium sized corpus, but that Φ takes 2-3 times longer to sample than $\mathbf{z}$ when $K = 1000$ for the Enron corpus.

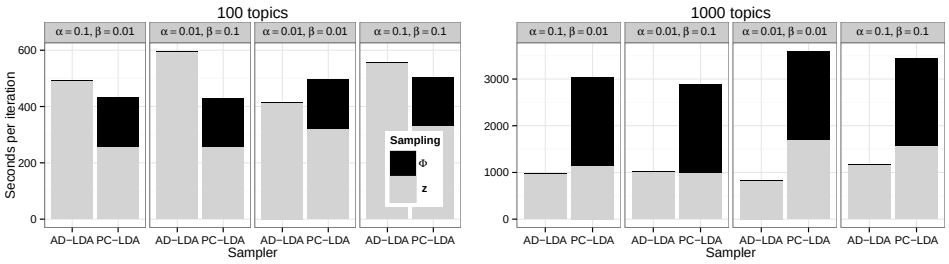

Figure 6: Time per iteration for $K = (100, 1000)$ for the Enron corpus.

4.6 Speedup and parallelism

We perform a set of performance measurements to determine the runtime efficiency and parallel scaling characteristics of our sampler compared to reference samplers. Table 4 summarizes the setup of the measurements. We study the speedup behavior in the two most interesting aspects of the samplers' runtime characteristics, time until convergence and running time.

First, we want the sampler to reach convergence as fast as possible (burn-in) so we measure the running time (rather than the number of iterations) for the samplers to *converge*. We define convergence as having occurred when a sampler has reached within 1% of the sparse LDA sampler's maximum log-likelihood value. We choose this definition because the sequential collapsed sampler is recognized to be the gold standard in terms of convergence behavior. Similar approaches have been used previously to assess convergence performance in mixture models Villani et al. (2009).

The second aspect is the total running time of the sampler, i.e. time to burn-in plus the time for the sampler to explore the posterior distribution of the parameters. Ideally we want both these parts to be done as fast and efficiently as possible. We initialize $\mathbf{z}$ at the same state for each seed and run a total of 20 000 iterations.

Dataset	Sampler	Topics	No. Cores	Priors
NIPS, Enron	Sparse LDA	20, 100	1	$\alpha = 0.1$, $\beta = 0.01$
NIPS, Enron	PC-LDA	20, 100	1,4,8,16,32,64	$\alpha = 0.1$, $\beta = 0.01$
NIPS, Enron	Sparse AD-LDA	20, 100	4,8,16,32,64	$\alpha = 0.1$, $\beta = 0.01$

Table 4: Summary of the speedup experiments.

Real speedup (Sahni and Thanvantri, 1996) can be defined as:

$$RealSpeedup(I, P) = \frac{Time(I, Q_b, 1)}{Time(I, Q_p, P)}, \tag{6}$$

where $Time(I, Q_b, 1)$ is the time it takes to solve the task I using the best sequential program. Q_b on 1 core, and $Time(I, Q_p, P)$ is the time it takes to solve task I using program Q_p on P processors. In our measurements, sparse LDA is the fastest sampler on one core for all configurations, so we will take sparse LDA to be the "best sequential program" Q_b and compare it with the PC-LDA sampler and AD-LDA as Q_p in our measurements for real speedup.

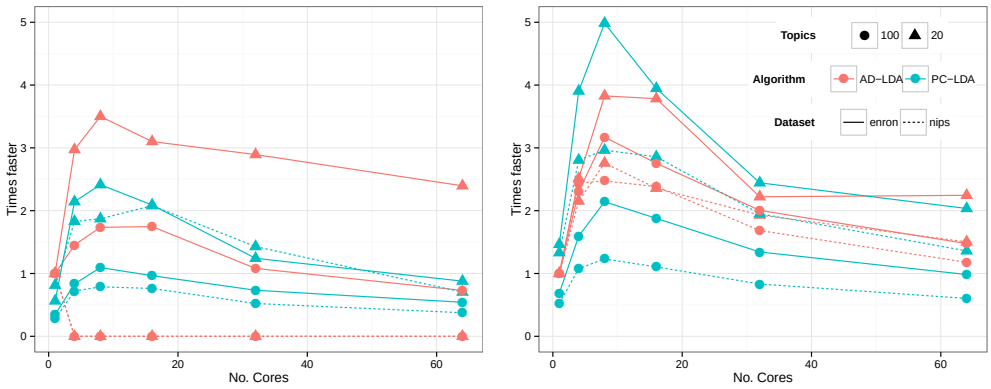

Figure 7: Real speedup to reach convergence (left) and speedup measured in total running time (right) of PC-LDA averaged over five seeds compared with the fastest implementation on one core (Sparse AD-LDA).

Figure 7 shows that we obtain a reasonable speedup to convergence up to eight cores on the Enron and NIPS datasets with 20 topics. The maximum speedup with respect to time to convergence we get on 20 topics is roughly 2.5 times on the Enron dataset with eight cores and roughly two times on the NIPS dataset with 16 cores. With relatively few topics compared to the size of the dataset the cost of sampling Φ is more than offset by the gains from parallelism.

When we increase the number of topics from 20 to 100 on these datasets, we do not get any speedup compared to sparse LDA on one core no matter how many cores we use. This is because

the number of topics is relatively large compared to the size of the dataset, and hence sparse LDA gets a maximum effect of the sparsity of $n^{(w)}$. But as shown below this effect diminish for larger corpora such as the PUBMED corpora. The sampling of Φ in the PC-LDA sampler takes three to five times longer when going from 20 to 100 topics, and is not offset by a large amount of topic indicator sampling, thus eating up the gains we get from the parallelization.

Figure 7 shows that the general speedup characteristics of the PC-LDA sampler is similar to those of the sparse AD-LDA sampler. The negative speedups after eight cores are probably due to cache effects and communication costs. Here we also see the effect of the approximation of the AD-LDA approach described in Section 4.4, in that it never reaches within 1% of the max log-likelihood value of the collapsed sampler on two configurations (this is indicated with a speedup of zero on the NIPS dataset with 20 and 100 topics in the left part of Figure 7) as the number of cores increase.

When it comes to the total time speedup we see how the PC-LDA sampler is penalized on small datasets with many topics, such as the NIPS dataset with 100 topics. In this case we get a slight speedup (roughly 20%) with four, eight and 16 cores compared to sparse LDA with one core. In contrast, for larger dataset with fewer topics, such as the Enron dataset with 20 topics, we get the maximum speedup (roughly five times) with eight cores.

To study the weak scaling characteristics of the samplers (i.e., to increase the problem size when increasing the number of CPU cores) we run them for 1 000 iterations on the PUBMED corpus. Since running a sampler using only one core on this large dataset would take too long, we cannot calculate the traditional real speedup as defined according to Eqns. 6 on this dataset. In this experiment we therefore compare our PC-LDA sampler with sparse AD-LDA relative to the execution time on 16 cores. We ran the samplers on one seed for 100 topics as well as for 1000 topics. The results are summarized in Table 5 and Figure 8.

Sampler	No. Topics	Total Runtime	Speedup Relative 16 Cores	Speedup Relative 32 Cores
PC-LDA	100	3.59 hours	1.99	1.39
Sparse AD-LDA	100	6.70 hours	1.44	1.09
PC-LDA	1000	13.61 hours	1.85	1.50
Sparse AD-LDA	1000	10.24 hours	1.65	1.24

Table 5: Speedup results on PUBMED (64 cores).

In this setting the PC-LDA sampler is faster for the smaller number of topics but slower on 1000 topics. Figure 8 on the other hand shows that the PC-LDA is very competitive when it come to speed to convergence. In some situations PC-LDA is actually faster to convergence, making it an attractive alternative for larger corpuses.

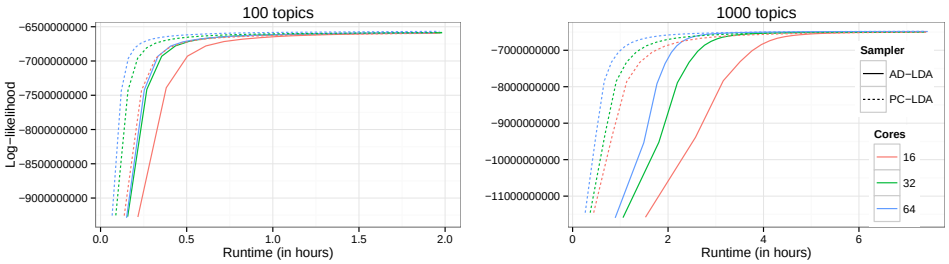

Figure 8: Convergence for the PUBMED corpus for 100 topics (left) and 1000 topics (right).

A conclusion from our speedup experiments is that PC-LDA have much the same general characteristics as sparse AD-LDA in terms of speedup through parallelization but with a slightly

higher overhead from the additional sampling of Φ. This effect is particularly notable with large number of topics on relatively small datasets. Several characteristic of the data and the models affect the relative performance of the algorithms. The largest effect is in the number of selected topics, which affects the sparsity of the $n^{(w)}$ and $n^{(d)}$ matrices and therefore heavily affects the speed of the samplers. Another important factor is the choice of prior, and we document in Section 4.5 that our speedup experiments are performed with a generally unfavorable prior setting ($\alpha = 0.1$ and $\beta = 0.01$) for the PC-LDA sampler.

The effects in terms of strong scaling diminishes after eight cores, but we are able to handle substantially larger datasets such as PUBMED that could not be practically handled using only one core (weak scaling). In this case we also see continued improved performance of 85-99% when going from 16 to 64 cores. The general conclusion is that the PC-LDA sampler is faster than the sparse AD-LDA approach when the number of topics is small to medium while the sparse AD-LDA is faster when the number of topics is large. It should again be emphasized however that PC-LDA is guaranteed to converge to the correct posterior whereas AD-LDA only approximates the posterior with no known bounds on the approximation error of the joint posterior.

4.7 Variable selection

Earlier research has already concluded that introducing variable selection for Φ in LDA can reduce the perplexity and increase sparsity Chien and Chang (2014). Here we study the effect of variable selection for the Enron and NIPS corpus with $\pi = 0.5, 0.1, 0.001, 0.00001$ and $K = 20, 100$. It is clear from Figure 4.7 and 4.7 that the variable selection sampling can be used to force sparsity into the Φ matrix by setting a rather strong prior of $\pi = 0.0001$. In this situation the sparsity of Φ is quite large, but on the other hand the document topic distribution of $\mathbf{z}$ is becoming much less sparse as a consequence.

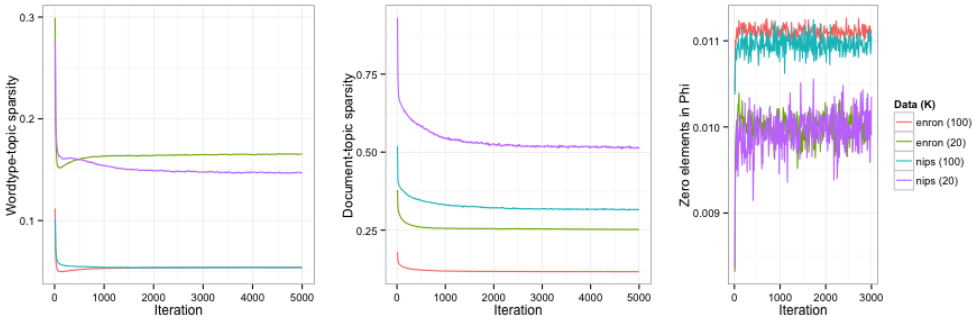

Figure 9: Document sparsity, word-type sparsity and proportion of zeroes in Φ for $\pi = 0.5$.

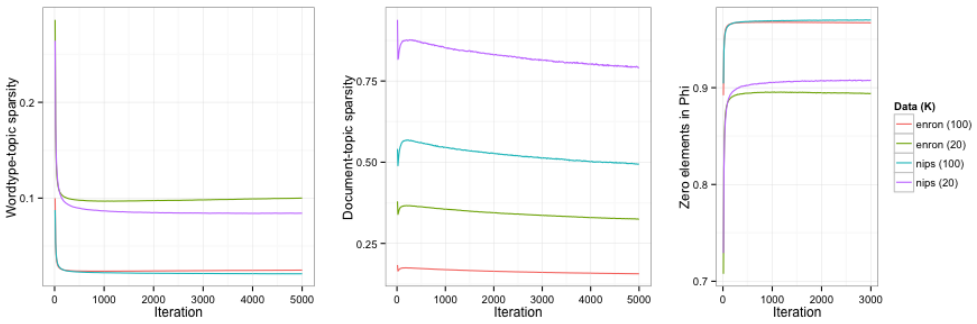

Figure 10: Document sparsity, word-type sparsity and proportion of zeroes in Φ for $\pi = 0.0001$.

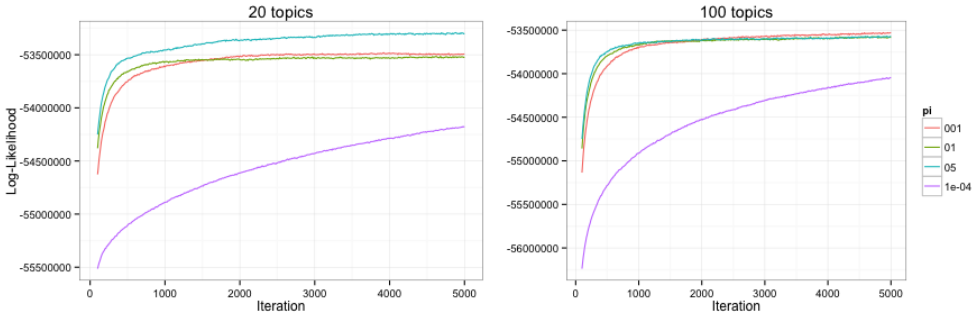

Figure 11: Log-Likelihood values for the Enron corpus.

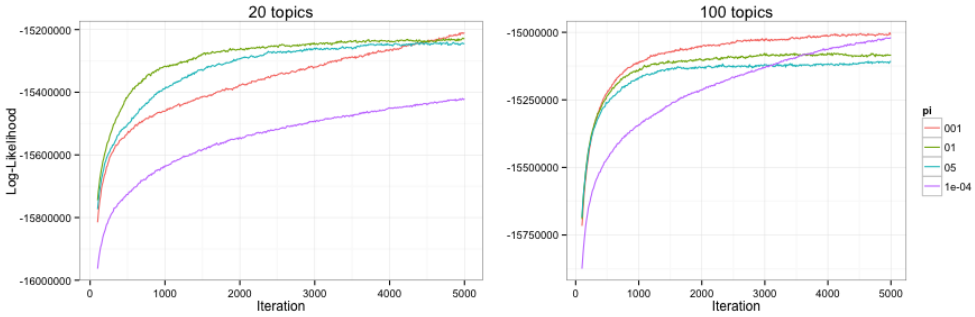

Figure 12: Log-Likelihood values for the NIPS corpus.

With a quite large $n^{(w)}$, as in the case of NIPS with $K = 100$, the introduced sparsity in Φ seems to produce better performance of the sampler with a better log-likelihood. This can be expected since the matrix $n^{(w)}$ in these situations are quite sparse, making a model with some elements of Φ set to 0 a reasonable model. But introducing a too strong prior towards sparsity seems to reduce the speed of convergence.

The improved model performance from a spike-and-slab prior should be weighed against the increased cost in sampling Φ. When only a small subset of elements in Φ are zero we need to perform a Bernoulli draw for all topic-word combination with zero counts (using a costly log-gamma function) in addition to the gamma draws for the elements in Φ. On the other hand, variable selection in Φ may lead to a better model and the obtained sparsity in Φ may be used to speed up the $\mathbf{z}$-sampling with ideas from sparse LDA.

5 Discussion and conclusions

We propose PC-LDA, an enhanced partially collapsed Gibbs sampler for LDA. Contrary to state-of-the-art parallel samplers such as AD-LDA, our sampler is guaranteed to converge to the true posterior. This is an important property as our experiments indicate that AD-LDA can converge to a suboptimal posterior mode which also decreases with the number of cores used for parallelization.

Our PC-LDA sampler is shown under reasonable assumptions to have the same complexity $O\left(\sum_{i=1}^{N} K_{d(i)}\right)$ as other efficient collapsed sparse samplers such as Alias-LDA. Moreover, contrary to conventional wisdom, we demonstrate on several commonly used corpora that a partially collapsed sampler has nearly the same MCMC efficiency as the gold standard sequential collapsed sampler, but enjoys the advantage of straightforward parallelization. Our results show significant speedups on up to 8 cores on the moderately sized NIPS and Enron corpora, and up to at least 64 cores on the relatively large PubMed corpus.

An important advantage of PC-LDA is that it allows for more interesting non-conjugate models for Φ, such as regularized topic models Newman et al. (2011). As an example, we show how a spike-and-slab prior for Φ can be easily implemented with PC-LDA. Our results demonstrate and the spike-and-slab prior can give a dramatic increase in sparsity, and thereby possibly also make the solution more interpretable, without severely degrading model fit.

In summary, we propose and evaluate a new partially collapsed Gibbs sampler for LDA with several algorithmic improvements. PC-LDA is fast, efficient and correct, and is applicable in a larger class of extended LDA models than the currently popular collapsed and AD-LDA samplers.

Acknowledgments

This work was supported in part by the Swedish Research Council (#623-2013-8591), the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies) and Ericsson AB.

References

Amr Ahmed, Moahmed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J Smola. Scalable inference in latent variable models. pages 123–132, 2012.

Márcio Drumond Araujo, Gonzalo Navarro, and Nivio Ziviani. Large text searching allowing errors. 1997.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. pages 27–34, 2009.

Jen-Tzung Chien and Ying-Lan Chang. Bayesian sparse topic model. *Journal of Signal Processing Systems*, 74(3):375–389, 2014.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

Alexander Ihler and David Newman. Understanding errors in approximate distributed latent dirichlet allocation. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5):952–960, 2012.

Leif Jonsson. Partiallycollapsedlda: First public release. may 2015. URL `http://dx.doi.org/10.5281/zenodo.18102`.

Krzysztof Łatuszyński, Gareth O. Roberts, and Jeffrey S. Rosenthal. Adaptive Gibbs samplers and related MCMC methods. *The Annals of Applied Probability*, 23(1):66–98, February 2013. ISSN 1050-5164. doi: 10.1214/11-AAP806. URL `http://projecteuclid.org/euclid.aoap/1359124382`.

Doug Lea. A java fork/join framework. In *Proceedings of the ACM 2000 Conference on Java Grande*, JAVA '00, pages 36–43, New York, NY, USA, 2000. ACM. ISBN 1-58113-288-3. doi: 10.1145/337449.337465. URL `http://doi.acm.org/10.1145/337449.337465`.

Aaron Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. Reducing the sampling complexity of topic models. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2014a. URL `http://www.sravi.org/pubs/fastlda-kdd2014.pdf`.

Aaron Q Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 891–900. ACM, 2014b.

Jun S Liu, Wing H Wong, and Augustine Kong. Covariance structure and convergence rate of the gibbs sampler with various scans. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–169, 1995.

Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. Plda+. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–18, April 2011. ISSN 21576904. doi: 10.1145/1961189.1961198. URL `http://dl.acm.org/citation.cfm?doid=1961189.1961198`.

George Marsaglia and Wai Wan Tsang. A simple method for generating gamma variables. *ACM Transactions on Mathematical Software (TOMS)*, 26(3):363–372, 2000.

Andrew K McCallum. {MALLET: A Machine Learning for Language Toolkit}. 2002.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.

David Newman, Edwin V Bonilla, and Wray Buntine. Improving topic coherence with regularized topic models. In *Advances in neural information processing systems*, pages 496–504, 2011.

Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. *Dirichlet and Related Distributions: Theory, Methods and Applications*, volume 888. John Wiley & Sons, 2011.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. Coda: Convergence diagnosis and output analysis for mcmc. *R News*, 6(1):7–11, 2006. URL `http://CRAN.R-project.org/doc/Rnews/`.

Ian Porteous, David Newman, Arthur Asuncion, and Max Welling. Fast Collapsed Gibbs Sampling For Latent Dirichlet Allocation. 2008.

Zhuolin Qiu, Bin Wu, B Wang, and C Shi. Collapsed Gibbs Sampling for Latent Dirichlet Allocation on Spark. *jmlr.org*, (2004):17–28, 2014. URL `http://jmlr.org/proceedings/papers/v36/qiu14.pdf`.

Michal Rosen-Zvi, Chaitanya Chemudugunta, Thomas Griffiths, Padhraic Smyth, and Mark Steyvers. Learning author-topic models from text corpora. *ACM Transactions on Information Systems (TOIS)*, 28(1):4, 2010.

Sartaj Sahni and Venkat Thanvantri. Parallel computing: Performance metrics and models. *IEEE Parallel and Distributed Technology*, 4(1):43–56, 1996.

Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, September 2010. ISSN 21508097. doi: 10.14778/1920841.1920931. URL `http://dl.acm.org/citation.cfm?doid=1920841.1920931`.

Padhraic Smyth, Max Welling, and Arthur U Asuncion. Asynchronous distributed learning of topic models. pages 81–88, 2009.

Yee Whye Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer, 2010.

Mattias Villani, Robert Kohn, and Paolo Giordani. Regression density estimation using smooth adaptive gaussian mixtures. *Journal of Econometrics*, 153(2):155–173, 2009.

Xuerui Wang and Andrew McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 424–433, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150450. URL `http://doi.acm.org/10.1145/1150402.1150450`.

Han Xiao and Thomas Stibor. Efficient Collapsed Gibbs Sampling For Latent Dirichlet. pages 1–16, 2010.

Feng Yan, Ningyi Xu, and Yuan Qi. Parallel inference for latent dirichlet allocation on graphics processing units. pages 2134–2142, 2009.

Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, page 937, 2009. doi: 10.1145/1557019.1557121. URL `http://portal.acm.org/citation.cfm?doid=1557019.1557121`.

Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, S. V. N. Vishwanathan, and Inderjit S. Dhillon. A scalable asynchronous distributed algorithm for topic modeling. *CoRR*, abs/1412.4986, 2014a. URL `http://arxiv.org/abs/1412.4986`.

Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, SV Vishwanathan, and Inderjit S Dhillon. A scalable asynchronous distributed algorithm for topic modeling. *arXiv preprint arXiv:1412.4986*, 2014b.

J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, and W.-Y. Ma. LightLDA: Big Topic Models on Modest Compute Clusters. *ArXiv e-prints*, December 2014.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. Gibbs max-margin topic models with fast sampling algorithms. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 124–132, 2013.