A-EPSGLD: An Asynchronous Distributed Framework for Efficient Large-scale LDA Inference

Yuan Yang [1] **Yifan Yang** [1] **Min Lee** [1]

Abstract

Learning large-scale Latent Dirichlet Allocation (LDA) is beneficial in many applications, but is also hard due to the massive amount of data and the lack of efficient inference algorithm. Recently, a novel distributed scheme, i.e. embarrassingly parallel stochastic gradient Langevin dynamics (EPSGLD), was proposed for efficient large-scale LDA inference, which demonstrates four orders of magnitude higher scalability than other existing state-of-the-art methods. However, the success of EPSGLD relies on the constant bulk-synchronizations on the shared variables throughout the training, leading to excessive communication overheads.

In this project, we propose to alleviate this issue by adopting the parameter server paradigm which enables a fully-pipelined update of shared variables with dedicated servers to eliminate the network latency, namely Asynchronous-EPSGLD (A-EPSGLD). We discuss in detail the theoretical and algorithmic aspects of A-EPSGLD. In Experiments, A-EPSGLD demonstrates: 1) a better convergence due to a tighter lower bound obtained from parameter server; and 2) a faster execution speed due the reduction in network latency.

1. Introduction

Latent Dirichlet Allocation (LDA) is widely used in many applications that involve large collections of documents. LDA is a probabilistic generative model which describes the documents and words in them to be generated by sampling from multinomial distributions that are in turn determined by Dirichlet priors. Due to its intractability of exact inference, training LDA model is usually done by utilizing approximate methods such as Markov Chain Monte Carlo (MCMC) and Variational Inference (VI). In particular, MCMC method has become the main workhorse since the Collapsed Gibbs sampling method (Griffiths, 2002) was proposed. For a standard MCMC sampler, the sampling complexity scales linearly with the number of topics and the total number of words in the collection. Apparently, the overhead for such method increases significantly as the model size and/or data size get larger.

There are two approaches to scale up the inference of LDA: stochastic subsampling and distributed computing. Stochastic methods reduce the sampling complexity by estimating the full update of the model by observing only a fraction of the data. For topic models, representative work includes and stochastic gradient MCMC, such as stochastic gradient Riemannian Langevin dynamics (SGRLD) (Patterson & Teh, 2013), The other way is to distribute the computation on the full dataset to multiple compute nodes, with representative work for topic models includes Yahoo!LDA (Ahmed et al., 2012), LightLDA (Yuan et al., 2015). These frameworks follow a common design paradigm of parameter server (Xing et al., 2015), where the global variables are updated and shared through a dedicated server. Another emerging paradigm for parallel computation is Embarrassingly Parallel (EP) (Neiswanger et al., 2013; Wang & Dunson, 2013; Minsker et al., 2014) which, compared to the parameter server approach, requires little (or no) communication between compute nodes during the training process. This property makes it helpful for large-scale LDA inference since the communication cost can be excessive as the model size increases. Recently, Yang et al. (Yang et al., 2016) proposed EPSGLD, an Embarrassingly Parallel scheme dedicated for LDA inference, and was demonstrated to outperform other state-of-the-art schemes at least one order of magnitude. However, due to the very nature of unidentifiability of LDA, EPSGLD still relies heavily on bulk-synchronization in estimating the full-data posterior, imposing a certain amount of communication overhead.

In this project, we propose to reduce this overhead by adopting an asynchronous style for updating the

[*]Equal contribution [1]Carnegie Mellon University, Pittsburgh, US. Correspondence to: Yuan Yang <yuany2@andrew.cmu.edu>, Yifan Yang <yifany1@andrew.cmu.edu>, Min Lee <mhlee@cmu.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

shared parameters, namely Asynchronous-EPSGLD (A-EPSGLD). Specifically, we provide the insight that two distributed paradigms—parameter server and Embarrassingly Parallel—are compatible. To do this, we show that the EP scheme can be interpreted as a classical Expectation-Maximization (EM) process between master and worker nodes. And different ways of updating the shared variables essentially create lower bounds of the true posterior that varies in terms of their tightness. Then we discuss practical aspects of how to combine these two paradigms, and present experiments for further demonstration. The remaining contents are organized as follows: We first review the inference method for LDA, and then discusses in detail the parameter server and EPSGLD; then we introduce how to adopt the parameter server into EPSGLD in both theoretical and algorithmic aspects; in experiment section we compare A-EPSGLD with EPSGLD by measuring their convergence and speed; finally we conclude with a discussion on our future work.

2. Background & Related Work

2.1. LDA & sampling method

LDA is a probabilistic graphical model for topic discovery in text documents. It describes how the words in documents are explained by a set of K topics in a generative procedure. Each of the topics is represented as a V-dimensional multinomial distribution φ_k, where V is the vocabulary size, and is referred to a topic-word distribution. The topics are often assumed to follow a conjugate Dirichlet prior, that is, $\varphi_k \sim Dir(\beta)$, with hyperparameter β. For each document $\mathbf{w}_d$ that contains N_d tokens, where each token in it is denoted as w_{dn}, a K-dimensional topic mixing distribution γ_d is sampled from a Dirichlet prior $Dir(\alpha)$. Then, for each token, a topic assignment z_{dn} is sampled from a multinomial distribution $z_{dn} \sim \text{Multi}(\gamma_d)$, followed by sampling the token itself again from a multinomial distribution $w_{dn} \sim \text{Multi}(\varphi_{z_{dn}})$. The matrix that contains all φ_k is denoted as Φ, and the one that contains all γ_d is Γ.

Inference over LDA is to determine the posterior distribution of all topic assignment $P(\mathbf{z}|\mathbf{w}, \alpha, \beta)$, where the Φ and Γ are integrated out. A classical method for performing MCMC sampling on this marginalized posterior is collapsed Gibbs sampling (Griffiths, 2002). For each z_{dn} we sample a new topic using

$$P(z_{dn} = k|rest) \propto \frac{(\alpha + n_{dk}^{-z_{dn}})(\beta + n_{kw}^{-z_{dn}})}{\sum_k \beta + n_{kw}^{-z_{dn}}} \quad (1)$$

where n_{dk} denotes the number of times that topic k is assigned to document d, n_{kw} denotes the number of times that topic k is assigned to word w, z_{dn} the topic assignments, superscript $-z_{dn}$ denotes the counts matrix without z_{dn}, and we omit the condition $\mathbf{w}$ for simplicity. Operationally, we sample using Eq.(1) for all assignments for a few iterations, and the last sample of $\bar{\mathbf{z}}$ is used to estimate Φ and Γ.

One can notice that sampling z_{dn} grows at the speed of $\mathcal{O}(K)$, which can be inefficient as the K becomes large. One variants of this method, namely LightLDA (Yuan et al., 2015) proposes to use alias table and successfully reduces the complexity into $\mathcal{O}(1)$. Specifically, it decomposes Eq.(1) into

$$P(z_{dn} = k|rest) \propto (\alpha + n_{dk}^{-z_{dn}}) \times \frac{(\beta + n_{kw})}{\sum_k \beta + n_{kw}}.$$

For each time it samples from the first term and the second term alternatively. The first term can be sampled in $\mathcal{O}(1)$ time since $\mathbf{z}_d$ is already the alias table of it; and the second term uses a stale copy of n_{kw}, out of which one can build an alias table and sample from it with $\mathcal{O}(1)$ time. Since samples are drawn from an incorrect distribution, an extra Metropolis step is used to correct the bias in the samples.

2.2. Parameter server

Parameter server paradigm is widely used to maintain the globally shared variables in distributed optimization or inference tasks, such as generalized linear models, graphical models and deep learning models (Li et al., 2014). For LDA, representative work includes Yahoo!LDA (Ahmed et al., 2012) and Petuum (Xing et al., 2015) on which the state-of-the-art LightLDA scheme is built. The key idea is exploiting the error-tolerant nature of machine learning algorithms: both of them, especially Petuum, introduce the idea of bounded staleness and synchronize the shared parameters in a delayed but efficient way. Formally, for LDA inference, the global shared matrix n_{kw} is stored in the server, while each client node i also maintains a local copy of it $\tilde{n}_{kw,i}$. And $\tilde{n}_{kw,i}$ is allowed to be different from n_{kw} due to the delayed update. If the staleness threshold is reached, global matrix n_{kw} is updated using

$$n_{kw} \leftarrow n_{kw} + \sum_i (\tilde{n}_{kw,i} - n_{kw,i}). \quad (2)$$

2.3. EPSGLD

Another paradigm for distributed inference is known as Embarrassingly parallel (EP), based on which Yang et al. (Yang et al., 2016) proposed a novel distributed scheme, namely EPSGLD, for LDA sampling. This paradigm allows worker nodes to draw samples only from their sub-posteriors, and then approximate the full-data posterior (true posterior) samples in master node using some aggregation methods, such as weighted average (Neiswanger et al., 2013), kernel density estimators (Wang & Dunson, 2013), and posterior median (Minsker et al., 2014). One of the central features for this paradigm is that individual workers need no/few communications when sampling from

sub-posteriors, which significantly reduces the network I/O and latency.

However, sampling from the posterior of shared variables requires to average the same local variables (e.g. all variables that correspond to the same topic). But due to the unidentifiability of LDA, it becomes infeasible as the number of topics increases. Thus EPSGLD makes use of an EM-style mechanism to synchronize between the true posterior stored in the master node and the sub-posterior in worker nodes. This leads a certain amount of communication overheads. And since this process is implemented with bulk-synchronization, local sampling process would be suspended until the update runs to a full completion, which further introduces potential network latency into the scheme.

3. Towards Asynchronous-EPSGLD

We begin our discussion by first take insights into the scheme of EPSGLD. We first introduce Θ, an unnormalized version of Φ whose each element can be recovered by $\varphi_{kw} = \frac{\theta_{kw}}{\sum_w \theta_{kw}}$, as the globally shared variable in EPSGLD. And the unnormalized model maintained locally in each computational node i is denoted as T_i. Then for each topic k, EP scheme (specifically Weierstrass transform (Wang & Dunson, 2013)) claims the following probabilistic locality

$$P(\theta_k, t_{k,1} \cdots, t_{k,n}|\mathbf{w}) \propto \prod_i K(\theta_k, t_{k,i}) P(t_{k,i}|\mathbf{w}_i), \quad (3)$$

from which we can further derive a block-wise Gibbs sampling procedure to estimate these variable

$$\begin{aligned} P(t_{k,i}|\theta_k, \mathbf{w}_i) &\propto K(\theta_k, t_{k,i}) P(t_{k,i}|\mathbf{w}_i) \\ P(\theta_k|t_{k,1}, \cdots, t_{k,n}) &\propto \mathcal{N}(\bar{t}_k, H_0), \end{aligned} \quad (4)$$

where H_0 is a constant, and $\bar{t}_k = \frac{1}{n}\sum_i t_{k,i}$. The first equation in Eq.(4) indicates that the posterior distribution of $t_{k,i}$ given the evidence of global estimation can be seen as the product of the local sub-posterior in node i and a kernel function w.r.t $t_{k,i}$ and θ_k. The second term suggests that the full conditional distribution of θ_k follows a Gaussian distribution with the mean as the mean local evidences. This implies a point-estimate of θ_k can be as simple as $\hat{\theta}_k = \frac{1}{n}\sum_i t_{k,i}$. Operationally, Eq.(4) suggests in order to sample from the joint, one first comes up with an initial global estimation and then sample $t_{k,i}$ independently in each node, and finally collect and aggregate these samples to give a hopefully better estimation of θ_k for another round.

We now provide a few insights about EPSGLD that are not (or not fully) explored in Yang et al's discussion (Yang et al., 2016), which will not only help us to better understand the nature of EPSGLD, but also serve as the theoretical foundations for our later effort of bridging between two paradigms of distributed computation: EP and parameter server.

3.1. Inference on sub-posterior

First, we notice that EP scheme itself does not specify any method to perform inference over $P(t_{k,i}|\theta_k, \mathbf{w}_i)$ which we refer to as the corrected sub-posterior. And indeed, in (Wang & Dunson, 2013) Wang et al. demonstrate this scheme with models such as logistic regression with only vanilla MCMC as its local inference device. EPSGLD, however, opts to apply one of the members of Hybrid Monte Carlo methods (HMC) family as its inference device, i.e. Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011). As its name suggests, SGLD owes its origin to the joint effort from both optimization community and Bayesian community. It combines Robbins-Monro type algorithms which stochastically optimize a likelihood, with Langevin dynamics which injects noise into the parameter updates in such a way that the trajectory of the parameters will converge to the full posterior distribution rather than just the maximum a posteriori mode. The resulting algorithm starts off being similar to stochastic optimization, then automatically transitions to one that simulates samples from the posterior using Langevin dynamics (Welling & Teh, 2011). Formally, in EPSGLD a new sample of $t_{kw,i}$ in time $(s+1)$ is generated using

$$\begin{aligned} t_{kw,i}^{(s+1)} \leftarrow |t_{kw,i}^{(s)} + \frac{\epsilon^{(s)}}{2}(-h(t_{kw,i}^{(s)} - \theta_{kw}^{(s)}) + \\ \beta - t_{kw,i}^{(s)} + \mathbb{E}) + (t_{kw,i}^{(s)})^{\frac{1}{2}} \eta_{kw}|, \end{aligned} \quad (5)$$

where $\epsilon^{(s)}$ denotes the step size at the sth iteration and n_{dkw} denotes the number of times topic k is assigned to w in document d. η_{kw} is obtained from $\eta_{kw} \sim \mathcal{N}(0, \epsilon^{(s)})$, h is a constant and $\mathbb{E}$ is the shorthand of expectation over distribution over all z_{dn} in a minibatch

$$\frac{D}{D^{(s)}} \sum_{d:\ \mathbf{w}_d \in \mathbf{w}^{(s)}} \mathbb{E}_{z_{dn}|rest,\Phi}[n_{dkw} - \varphi_{kw} n_{dk}], \quad (6)$$

where $d:\ \mathbf{w}_d \in \mathbf{w}^{(s)}$ denotes each document in the minibatch and the numbers of documents in $\mathbf{w}^{(s)}$ and $\mathbf{w}$ are denoted as $D^{(s)}$ and D respectively. For the rest of our discussion we assume this method as the local inference device for consistency, but we will see later this implicitly couples the update of global and local variables and thus making it difficult for adopting the delayed (asynchronous) update. And we will discuss how to overcome this issue in later contents.

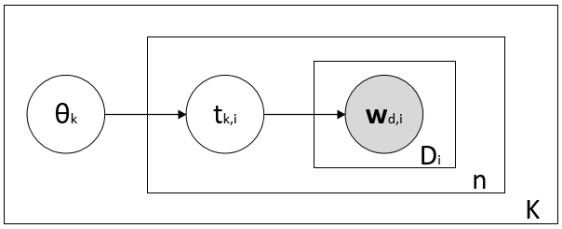


Figure 1. A graphical model representation of the underlying probabilistic dependencies among three variables that are implied by the EP scheme. Shaded circle means the variable is observed, otherwise unobserved.

3.2. EP as an EM process

The update equation Eq.(4) suggests an alternative way of estimation for interdependent variables. In (Yang et al., 2016) Yang realizes this as a certain form of classical Expectation-Maximization (EM) algorithm, but then fails to fully exploit this idea. On the other hand, Eq.(4) also suggests that θ_k needs to be updated for each round of the update, which, again, couples the updates of global and local variables, and eventually leads to potential latency and excessive communication cause. Yang proposes to allow for using a stale copy of θ_k in each node when sampling from the corrected sub-posterior, and the update of θ_k is not performed until node being notified by a scheduler that consists of a heuristic function determining the timing of update. Yang realizes that this can introduce error into the procedure but failed to characterize this error with respect to its EM algorithm nature. In this section we explore in detail the connection between EP and EM algorithm and provide theoretical insights about the error introduced by the delayed update.

Through inspection, we notice that Eq.(4) essentially represents an EM algorithm procedure over an implicit graphical model with unobserved variables θ_k. This is illustrated in Fig.1. Formally, for each computational node i it "maximizes" [1] the marginal posterior distribution $P(t_{k,i}|\mathbf{w}_i)$ where θ_k was integrated out. It is then equivalent to maximize the log posterior, and thus the lower bound of the target is give as:

$$P(t_{k,i}|\mathbf{w}_i) \propto log \int P(t_{k,i}, \theta_k|\mathbf{w}_i)d\theta_k$$
$$\geq \int Q(\theta_k) log \frac{P(t_{k,i}, \theta_k|\mathbf{w}_i)}{Q(\theta_k)} d\theta_k,$$

where $Q(\theta_k)$ is an arbitrary distribution of θ_k and is usually set to its posterior distribution $P(\theta_k|t_{k,1}, \cdots, t_{k,n})$—exactly the same as the second term in Eq.(4). Thus the EM process is formulated below

E-step:
$$E_{\theta_k|t_{k,1},\cdots,t_{k,n}}[\theta_k] = \frac{1}{n}\sum_i t_{k,i} \tag{7}$$

M-step:
$$\arg\max_{t_{k,i}} P(t_{k,i}|\theta_k, \mathbf{w}_i). \tag{8}$$

Now we justify the delayed update of θ_k.

Proposition 3.2. *For any time step* $(s) \in \{(1), ..., (s), ...\}$ *in EM update, using* $\theta_k^{(t)}$ *with* $(t) < (s)$ *does not change the monotonicity of the EM algorithm, given that* $\theta_k^{(x)} \in \{\theta_k^{(y)}|(y) \leq (t)\}$, *for* $\forall(x) \leq (s)$.

This is straightforward, since Eq.(7) constructs a tight lower bound of the sub-posterior, so if the E-step is delayed, it essentially means Eq.(8) still maximizes the old lower bound instead of the one that is up-to-date, which indeed does no harm to the EM process. However the value will not be further improved if maximum is reached. But we notice that, since the realization of Eq.(8) is done iteratively using MCMC which usually takes many iterations to iterations to converge, thus a reasonable delay in E-step is not likely to bound the M-step, leading a waste of computational resources.

3.3. Identifiability

Those readers familiar with Bayesian mixture model may worry about the correctness of Eq.(7). Indeed, like Gaussian mixture model (GMM), LDA has the issue of "unidentifiability": The complete posterior $P(T_i|\Theta_k, \mathbf{w}_i)$ are essentially multimodal and there are $K!$ equivalent permutations to realize the the same posterior probability. Thus the same index k in different models may indicates different topics. This is also referred as the label switching effect (Stephens, 2000). Methods dealing with this effect usually requires polynomial time that explores every K labels, and thus assuming K to be small. However, in large-scale LDA K can be the order of 10^4 or even larger. Such explicit methods are, therefore, practically infeasible.

A heuristic approach is to use the same initialization copy for each node and then synchronize (i.e. perform Eq.(7)) as frequently as possible to prevent the topics from drifting apart (Newman et al., 2009). This approach is followed in (Yang et al., 2016), and Yang further found that insufficient synchronization leads to convergence on a sub-optimum, and this optimum gets worse as even less synchronizations are performed.

This error, again, can be qualitatively characterized from EM perspective. Since EM algorithm always proposes the lower bound of target distribution, and tightens this bound by setting $Q(\theta_k)$ to its posterior distribution. Due to the

[1] The term "maximize" is not a strictly accurate description since it implies a frequentist treatment. However, since we notice that the estimation of $t_{k,i}$ is done deterministically and can be viewed as the result of maximum a posterior (MAP) approach which is readily compatible with maximum likelihood approach, we still use this term to fit the EP process into EM framework.

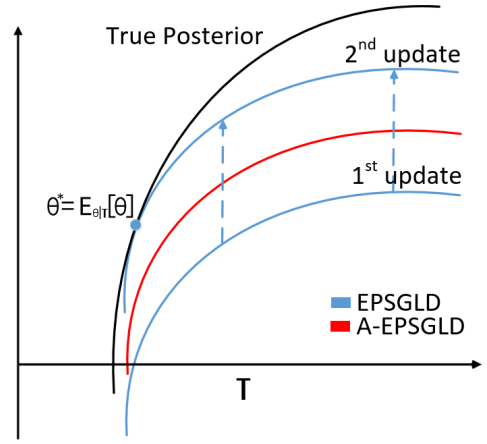


Figure 2. A hypothetical example showing how A-EPSGLD creates a lower bound of the true posterior compared with those created by EPSGLD in two updates.

switching effect, the resulting $E[\theta_k]$ does not come from the posterior but from some unknown distribution, which leads to a subsequent optimization actually performed on a "loose" lower bound. And this eventually leads to a sub-optimum.

Algorithm 1 A-EPSGLD for client node i

Local tables for ith worker: $t_{k,i}$
Global PS table: θ_k
Initialization;
for batch of documents in the collection **do**
  $g = \nabla P(t_{k,i}|\text{PS.Get}(\theta_k), \mathbf{w}_i)$
  $t_{k,i} = |t_{k,i} + \epsilon \cdot g|$
  // ϵ: the step size for gradient descent
  PS.Inc($\theta_k, \frac{g}{n}$)
  // n: the number of client nodes
  PS.Clock()
end for

4. Asynchronous-EPSGLD with Parameter Server

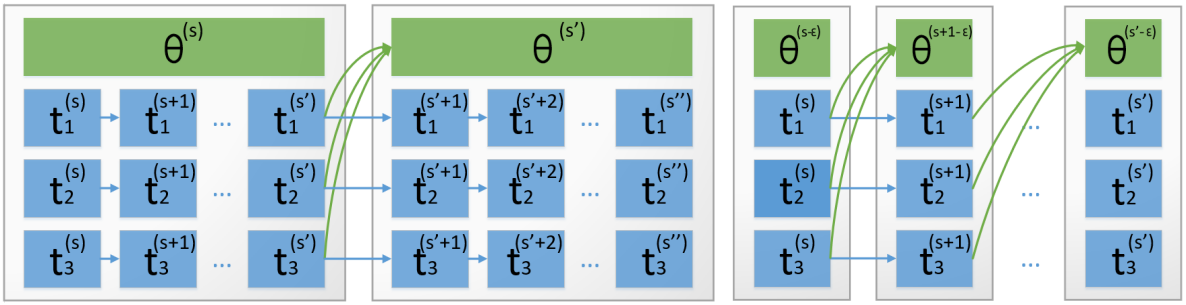
In previous section, we have shown that delaying Eq.(7) is not likely to hamper the EM process (label switching effect is discussed in later contents), and is computationally beneficial since it significantly reduces the communication overhead. The update scheme is illustrated in Fig.3(a). Intuitively, it's even more beneficial if we can do Eq.(7) asynchronously. Compared to EPSGLD, where E-step is performed as an atomic operation, this approach completely decouples the E-step from M-step, which eliminates completely the network latency. It is also theoretically correct, since it creates an automatically tightened lower bound as the training proceeds. Fig.2 illustrates this point with a hypothetical example. It suggests that lower bound obtained in A-EPSGLD is always tighter than that obtained using bulk-synchronization, since the shared variables ares updated the same time when workers perform local sampling. And a tighter lower bound helps to reduce the approximation error due to the unidentifiability property.

We now see that the EM interpretation eventually leads us to a more general update schedule. And this idea turns out to fit perfectly with the model-parallelism paradigm. Specifically, in parameter server (PS) community, the idea of asynchronous update is studied extensively (Ahmed et al., 2012; Li et al., 2014; Xing et al., 2015). Data and model parameters are distributed into several client nodes. In the meantime, a server node has the global model parameters. client nodes use CLOCK operator to record the timestamps. PS synchronizes parameters between clients and server via staleness value. It requires that the difference of timestamps between server and client nodes is less than staleness value.

Treating Θ the global word-topic distribution as the shared variables. The A-EPSGLD scheme can be readily implemented with common PS toolkits. And in this project, we use Petuum (Xing et al., 2015), a distributed ML platform with built-in high performance PS and communication protocols. Procedures in worker node are straightforward to design. We present its pseudo-code in Algorithm.1. We calculate gradient for a batch of documents using GET operator to retrieve stale global θ_k. Then the gradient is applied to the local $t_{k,i}$ and global θ_k via INC operator. It is worth noting that we have to make sure the updated $t_{k,i}$ is positive. If $t_{k,i} + \epsilon \cdot g$ is a negative value, we use $\frac{-2t_{k,i}-\epsilon \cdot g}{\epsilon}$ as our gradient, which has the equivalent result as taking absolute value. The update scheme is illustrated in Fig.3(b).

Note that A-EPSGLD is even more flexible than other distributed LDA implementations that are based on bounded staleness, such as LightLDA. In LightLDA, bounded staleness is applied on the model block in each computational node, which will potentially suspend the local sampling process if the bound is reached. But in A-EPSGLD, the bound is applied on θ_k instead of $t_{k,i}$ and as already indicated in Proposition 3.2, a reasonably stale θ_k is not likely to hamper the local M-step. Therefore, this completely decouples the local process from the distribution framework with theoretical guarantees of correctness.

To address the label switching effect, we still lack a method that is both theoretically sound and computationally cheap. And a full discussion on this topic lies beyond the scope of this paper. Here, as a compromise, we follow the same heuristics used in (Newman et al., 2009; Yang et al., 2016). We control the synchronization frequency by adding an additional staleness bound on each θ_{kw} entry. The logical clock for this bound is (s_i): the number of iterations



(a) (b)

Figure 3. Overview of update strategies of: (a) EPSGLD and (b) A-EPSGLD, with 3 worker nodes. Global variables have green shade, while local have blue shade. Arrows with different colors denote update operation of corresponding variables. The box with gray shade indicates the life span of a certain instant of global variable and ϵ denotes the potential staleness.

elapsed for a node generating the sample $t_{kw,i}^{(s_i)}$, this time is compared to the time label of last synchronization and then tested if it lies within the bound. As suggested in (Yang et al., 2016), this bound should be initially small since topics are likely to vary frequently; and then gradually increases as the model tends to converge.

5. Experiments

To demonstrates the benefits of A-EPSGLD over the original version we conduct experiments for two frameworks together with a serial SGLD. Results are compared in terms of their (1) convergence: for the same amount of data, which one converges faster; and (2) performance: for the same amount of time, which one leads to a higher log-likelihood. All frameworks are implemented on distributed machine learning framework Bosen[2]. And experiments are done in Amazon Web Service with C4.4xlarge instance. It has 16 cores and 30GB memory. Both A-EPSGLD and EPSGLD are run with 8 workers.

Dataset: We use 20 Newsgroups[3] and NYTimes[4] datasets. 20 Newsgroups contains 18846 passages with vocabulary size of 61188. NYTimes contains 299752 financial documents with vocabulary size of 101636 and average length of a document of 332. When processing the dataset, 476 documents for 20 Newsgroups and 1000 documents for NYTimes are extracted in order to form the held-out test datasets for evaluation.

Hyperparameters: Both models require hyperparameters in order to determine the trade-off point between convergence and execution speed. In EPSGLD a scheduler needs to be specified by a polynomial function such as N^2, and N^3; while in A-EPSGLD one needs to specify the staleness bound with an integer. For 20 Newsgroup dataset we conduct a comprehensive comparison, where different settings of two frameworks are tested: EPSGLD with schedule $2N$, N^2, and N^3 and A-EPSGLD with staleness 0, 2, 5 and 10. For a larger dataset (i.e. NYTimes) the best settings found in 20 Newsgroup are used.

[2]https://github.com/sailing-pmls/bosen

[3]https://archive.ics.uci.edu/ml/machine-learning-databases/20newsgroups-mld/

[4]https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/

5.1. Convergence

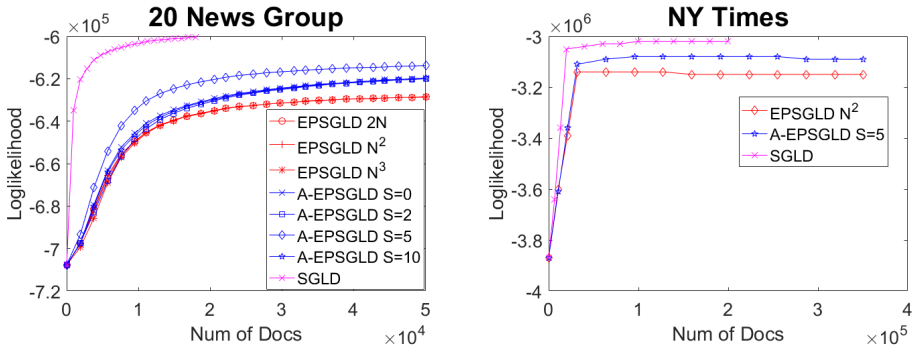
We first show A-EPSGLD indeed yields a tighter lower bound than previous version , which should eventually lead to a better convergence. To show this we compare log-likelihood of held-out test datasets against number of document visited, using

$$l(\mathbf{w}) = \sum_{w} \hat{n}_{d\cdot} \hat{n}_{\cdot w},$$

as specified in (Wallach et al., 2009), where $\hat{n}_{dk}$ and $\hat{n}_{kw}$ are obtained by running Gibbs sampling on test set with Φ and Γ specified by the model. We compare these two model in two datasets 20 Newsgroup and NYTimes with 8 threads in a single machine.

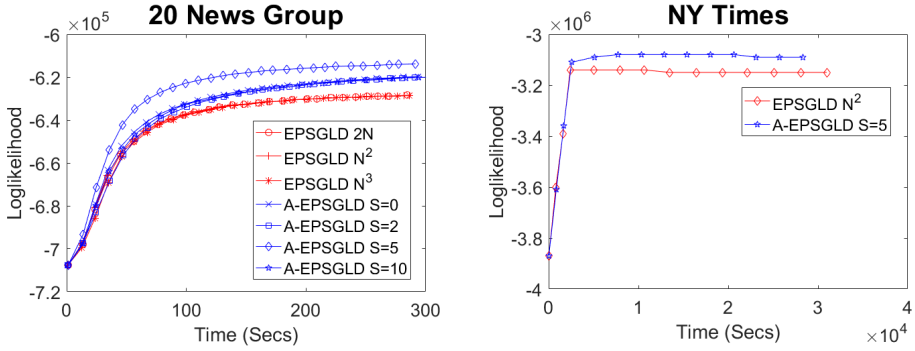
As demonstrated in Fig.4(a), A-EPSGLD with blue curves has overall higher log-likelihood value and faster convergence than EPSGLD with red curves. Since experiments are explored in a small dataset, some comparisons are hard to distinguish. Staleness 5 is the best instead of staleness 0. This result is out of our expectation. One assumption is that the Petuum performs unstably in a small dataset. Trying a larger dataset will alleviate this problem. We follow the best parameter settings of $S = 5$ for A-EPSGLD and N^2 for EPSGLD. Using this parameters, the same faster convergence of A-EPSGLD is shown in Fig.4(b) for NYTimes. This outcome demonstrates our previous explanation on the intuition of our approach. Unlike EPSGLD that relies on bulk synchronization in estimating posterior, A-EPSGLD generates a tighter lower bound of posterior and converges faster.

The curve of single thread SGLD serves as the upper bound



(a) (b)

Figure 4. Log-likelihood against the number of visited documents for SGLD, EPSGLD, and A-EPSGLD. (a) Comparing different synchronization schedulers for EPSGLD and different staleness values for A-EPSGLD on 20 Newsgroup; (b) SGLD, EPSGLD, and A-EPSGLD with the best setting on NYTimes.



(a) (b)

Figure 5. Log-likelihood against training time for EPSGLD and A-EPSGLD. (a) Comparing different synchronization schedulers for EPSGLD and different staleness values for A-EPSGLD on 20 Newsgroup; (b) EPSGLD an A-EPSGLD with the best setting on NYTimes.

of both of the distributed schemes, since the approximation error is zero. We can see in smaller dataset, i.e. 20 Newsgroup SGLD performs significantly better, while in NY Times, the differences are not obvious. This is a sensible results since partitioning a small dataset to many workers is highly likely to violate the i.i.d assumption, leading to a larger approximation error. And as the size of dataset grows, this error is compensated by the corpus redundancy leading to a smaller error.

5.2. Performance

We compared the log-likelihood of A-EPSGLD with that of EPSGLD in terms of time. In Fig.5(a), A-EPSGLD with blue curves has overall higher log-likelihood value and have better performance than EPSGLD with red curves. Tuning staleness value is a trade off between network latency and convergence. Small staleness value will spend more time on synchronization, but it will converge a better result. We found the best parameter settings of $S = 5$ for A-EPSGLD and N^2 for EPSGLD. Using this parameters, we confirmed the same trend that A-EPSGLD has better performance than EPSGLD in Fig.5(b) for NYTimes. Given the same amount of time, we expect A-EPSGLD has less network latency due to its asynchronous style of updating the shared parameters. Thus, it has better performance than EPSGLD over the time.

5.3. Network Latency

To validate our previous statement that A-EPSGLD has the smaller latency than EPSGLD. We computed the network latency that indicates the total time of synchronization for each model using the best parameter settings for the two dataset. In Fig.6, A-EPSGLD with blue bar graphs have much less network latency than EPSGLD with red bar graphs.

5.4. Scalability of A-EPSGLD

In this experiments, we test scalability of A-EPSGLD framework by setting different number of worker nodes and record the wall-time usage for completing a epoch of update on a dataset. We run A-EPSGLD on a single machine with 16 cores. Time usage for 1, 2, 4 and 8 threads is shown

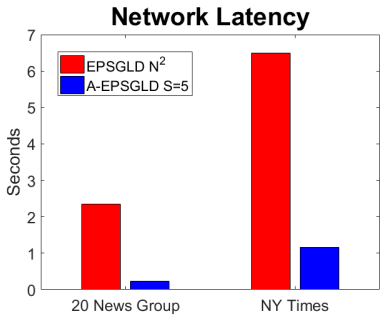


Figure 6. Network Latency of A-EPSGLD and EPSGLD using the best parameter settings.

in Fig.7 together with ideal scaling curve. We can see that A-EPSGLD scales closely to the ideal curve, demonstrating a low network latency.

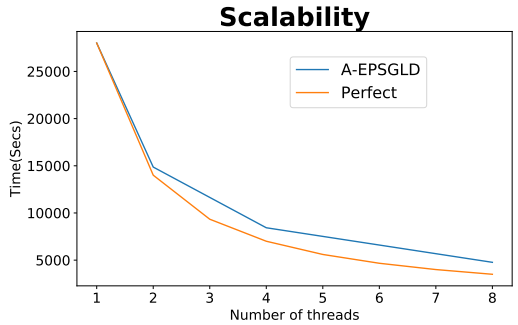


Figure 7. Scalability of A-EPSGLD with 1, 2, 4 and 8 threads on a single machine.

6. Discussion

In this paper, we have discussed how to adopt the model-parallelism setting into a data-parallelism paradigm (i.e. EP) for a particular way of training certain type of topic models. However, this method is not restricted to any specific optimization/sampling method or model, but applicable to general distributed ML tasks.

One can view this as a generic strategy of trading-off between system overhead and algorithm accuracy with theoretical guarantees of convergence. For example, any application with PS can be seen as a EP paradigm with only one set of variables, i.e. global variables. If each worker is allowed to explicitly maintain a local copy of the shared variables, then it becomes an standard EP scheme. Applying this strategy, essentially leads to a complete decoupling between the synchronization of shared variables and the local working flow, leading to zero network latency. This is comparable to a complete asynchronous update without staleness bound. However, the latter lacks a theoretical guarantee of convergence, and is often found hard to use and fine-tune in practice.

We expect to continue to explore this idea in the future, as well as better methods for solving the unidentifiability issue with LDA model.

7. Conclusion

We have presented A-EPSGLD, an effort of integrating two distributed computation paradigm, i.e. parameter server (PS) and embarrassingly parallel (EP), for large-scale LDA inference. We first show how EP paradigm can be abstracted from the local inference process and then explained within the EM framework, which provides theoretical support for adopting PS paradigm. Then we show how a specific implementation of PS, i.e. Petuum, can fit into this EM framework. Using the implementation of A-EPSGLD and EPSGLD on the Pentuum system, we demonstrated that A-EPSGLD outperforms EPSGLD. Specifically, A-EPSGLD has tighter lower bound in estimating posterior and faster convergence than EPSGLD given the same amount of data. A-EPSGLD has performance better in time than EPSGLD due to the smaller network latency.

References

Ahmed, Amr, Aly, Moahmed, Gonzalez, Joseph, Narayanamurthy, Shravan, and Smola, Alexander J. Scalable inference in latent variable models. In *Proceedings of the fifth ACM International Conference on Web Search and Data Mining*, pp. 123–132, 2012.

Griffiths, Tom. Gibbs sampling in the generative model of latent dirichlet allocation. 2002.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, Mu, Andersen, David G, Park, Jun Woo, Smola, Alexander J, Ahmed, Amr, Josifovski, Vanja, Long, James, Shekita, Eugene J, and Su, Bor-Yiing. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pp. 583–598, 2014.

Minsker, Stanislav, Srivastava, Sanvesh, Lin, Lizhen, and Dunson, David B. Robust and scalable bayes via a median of subset posterior measures. *arXiv preprint arXiv:1403.2660*, 2014.

Neiswanger, Willie, Wang, Chong, and Xing, Eric. Asymptotically exact, embarrassingly parallel mcmc. *arXiv preprint arXiv:1311.4780*, 2013.

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

Patterson, Sam and Teh, Yee Whye. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 3102–3110, 2013.

Stephens, Matthew. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.

Wallach, Hanna M, Murray, Iain, Salakhutdinov, Ruslan, and Mimno, David. Evaluation methods for topic models. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1105–1112. ACM, 2009.

Wang, Xiangyu and Dunson, David B. Parallel mcmc via weierstrass sampler. *arXiv preprint arXiv:1312.4605*, 2013.

Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 681–688, 2011.

Xing, Eric P, Ho, Qirong, Dai, Wei, Kim, Jin Kyu, Wei, Jinliang, Lee, Seunghak, Zheng, Xun, Xie, Pengtao, Kumar, Abhimanu, and Yu, Yaoliang. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

Yang, Yuan, Chen, Jianfei, and Zhu, Jun. Distributing the stochastic gradient sampler for large-scale lda. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1975–1984. ACM, 2016.

Yuan, Jinhui, Gao, Fei, Ho, Qirong, Dai, Wei, Wei, Jinliang, Zheng, Xun, Xing, Eric Po, Liu, Tie-Yan, and Ma, Wei-Ying. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1351–1361, 2015.