Distributed Online Learning for Latent Dirichlet Allocation

JinYeong Bak, Dongwoo Kim and Alice Oh
Department of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
{jy.bak, dw.kim}@kaist.ac.kr, alice.oh@kaist.edu

Abstract

A major obstacle in using Latent Dirichlet Allocation (LDA) is the amount of time it takes for inference, especially for a dataset that starts out large and expands quickly, such as a corpus of blog posts or online news articles. Recent developments in distributed inference algorithms for LDA, as well as minibatch-based online learning algorithms have offered partial solutions for problem. In this paper, we propose a distributed online learning algorithm for LDA for dealing with both aspects of this problem at once. We apply our learning algorithm to a corpus of Twitter conversations and show that it achieves the same model fit within a much shorter learning time.

1 Introduction

Probabilistic topic models such as LDA [2] are useful for various types of unstructured data. They are especially convenient for Web documents, such as online news articles, blogs, and microblogs, where topics and other latent dimensions are not known a priori. Two practical problems arise when applying topic models to Web documents. First, a corpus of Web documents is often very large, consisting of hundreds of thousands of documents, which makes inference quite slow. Second, a corpus of Web documents expands rapidly, so the model parameters must be updated continuously to deal with new documents.

Modeling a large dataset with LDA is difficult because of two practical reasons. First, two popular inference methods, Gibbs sampling and variational inference, both take up much time and memory. To make the inference practical for a large dataset, researchers proposed distributed algorithms for Gibbs sampling [9, 10, 13, 16] and variational inference [1, 8, 17]. Second, many Web documents are generated in real time. Continuously re-estimating the model parameters from the updated dataset would be redundant and costly. Online algorithms for LDA based on Gibbs sampling [3] and variational inference [6] offer solutions for this problem. We simply propose to combine distributed inference and online learning for LDA, such that the resulting learning algorithm would be able to model a large and continuously expanding dataset.

2 Distributed Online Learning for LDA

We describe our online learning algorithm for LDA in a distributed system. Among the two major inference algorithms, we chose to use variational inference.

2.1 Online Learning of LDA with Variational Inference

Latent Dirichlet Allocation (LDA) is a Bayesian probabilistic topic model. LDA assumes that the documents in a corpus are generated from a set of K topics. Each topic k is represented by a multinomial distribution over the vocabulary and is drawn from a Dirichlet, $\beta_k \sim Dir(\eta)$. Each document d has a topic proportion that is drawn from a Dirichlet, $\theta_d \sim Dir(\alpha)$. Then, to generate each word n in document d, first draw a topic index z_{dn} from θ_d, $z_{dn} \sim Mult(\theta_d)$ and draw the observed word w_{dn} from selected topic, $w_{dn} \sim Mult(\beta_{z_{dn}})$. Two major algorithms for inferring the model parameters are variational inference [2] and gibbs sampling [4].

We based our algorithm on online variational Bayes algorithm [6] because diagnosing convergence of Gibbs sampling is difficult [11] and dependencies between topic indices z_{dn} and $z_{d'n'}$ are broken when sampling is distributed [10].

In variational inference for LDA, the original LDA model is simplified by removing edges between the hidden variables, and adding a free variational parameter for each hidden variable. Topic index variables z_{dn} are drawn from $z_{dn} \sim Mult(\phi_{dw_{dn}})$, topic proportions θ_d from $\theta_d \sim Dir(\gamma_d)$, and word distributions β_k from $\beta_k \sim Dir(\lambda_k)$. To infer the variational parameters, we use the variational EM algorithm. In the E step, we compute ϕ_{dw} and γ_d for all documents, holding $\boldsymbol{\lambda}$ fixed. In the M step, we update $\boldsymbol{\lambda}$ given new values of ϕ_{dw} [2, 6].

Hoffman, et al. [6] proposed an online variational Bayes algorithm for LDA. This algorithm updates $\boldsymbol{\lambda}$ by stochastic learning. It updates $\boldsymbol{\lambda}$ with the weighted average of current $\boldsymbol{\lambda}$ and new $\boldsymbol{\lambda}_{new}$ which is computed over a mini-batch of new documents with a learning rate $\rho \equiv (\tau_0 + t)^{-\kappa}$, where $\kappa \in (0.5, 1]$ and t is the number of mini-batches seen so far.

2.2 Distributed Online Learning for LDA

Zhai, et al. [17] proposed a distributed variational inference algorithm using the MapReduce framework. The E step is done in the Mapper, and the M step in the Reducer.

Similarly, we carry out the E step in the Mapper, and to parallelize the E step, we use the mini-batch technique in stochastic learning. That is, we look at more than one document at each update [7] by specifying a size of a mini-batch, and the documents in the mini-batch are distributed to several Mappers where each Mapper carries out an E step.

For processing new documents in our distributed online LDA (DoLDA), Driver should get new documents from the source, and set hyperparameters α, η, learning rate ρ and variational parameter $\boldsymbol{\lambda}$ to distributed cache. Those are loaded in Mapper and Reducer to update $\phi, \gamma, \boldsymbol{\lambda}_{new}$. Then, Driver runs the MapReduce job.

In the Map task, Mapper loads α, $\boldsymbol{\lambda}$ and number of topics K and words W from distributed cache and carries out the E step in online LDA [6]. After that, Mapper outputs variational parameters for document d's topic proportion γ_d and sufficient statistics for updating $\boldsymbol{\lambda}$, ϕ_d.

In the Reduce task, Reducer loads η, learning rate ρ, $\boldsymbol{\lambda}$, mini-batch size S, number of topics K, and documents D from distributed cache and carries out the M step in online LDA [6]. After that, Reducer outputs new $\boldsymbol{\lambda}_{new}$ updated by the new documents. Finally Driver updates $\alpha, \eta, \boldsymbol{\lambda}$ from the MapReduce output and repeats the same jobs until there are no more new documents [1].

To improve efficiency, we combine the sufficient statistics $n_{dw}\phi_{dwk}$ in Combiner task. Generally, the number of observed words in a document is less than the size of the vocabulary, such that most entries of $n_{dw}\phi_{dwk}$ are 0. To calculate $\tilde{\lambda}_{kw}$, we sum all $n_{dw}\phi_{dwk}$, but we can ignore $n_{dw}\phi_{dwk}$ for which $n_{dw} = 0$ and sum the rest. This decreases network I/O cost.

[1] For a more detailed explanation of the algorithm, see the supplementary material.

Table 1: Comparison of running time (seconds) between Online LDA (oLDA) and Distributed Online LDA (DoLDA). We used a corpus of Twitter conversations, varying the size of mini-batch and number of topics.

Mini-batch	oLDA	DoLDA	oLDA/DoLDA
1024	15494.14	33066.70	0.47
4096	10239.85	10220.95	1.00
8192	8952.56	6269.11	1.43
16384	8740.42	4390.66	1.99

# topics	oLDA	DoLDA	oLDA/DoLDA
25	5100.42	3169.82	1.61
50	7062.51	3839.34	1.84
100	8740.42	4390.66	1.99
200	13187.21	6646.85	1.98
500	27987.05	12346.10	2.27

3 Experiment

We implemented DoLDA using Python with Hadoop 0.20.2. The implementation is based on the open-source code of online LDA [2]. We first built a Hadoop system with three computers, each one with four 2.33GHz cores. We also used a 60-node Hadoop system, each with eight 2.40GHz cores. We ran DoLDA and online LDA experiments on these to compare learning time and model fit.

To show the practical advantage of DoLDA, we chose Twitter data to test and compare the performance against online LDA. We crawled Twitter for conversations, where a conversation is defined as a sequence of reply tweets by two Twitter users. We defined a document as a conversation. The number of documents is 973,266, and the average document length is 7.54 tweets. We removed words with document frequency of 30 or less.

We fixed $\tau_0 = 1024$ and $\kappa = 0.7$. We also fixed α and η to $\frac{1}{K}$, and $K = 100$.

Running Time We ran online LDA and DoLDA with the Twitter conversation corpus, and compared the running time. Table 1 shows the results.

When mini-batch size is 1024, online LDA is faster than DoLDA. The reason for this and a drawback of DoLDA is the network I/O cost. Each machine is connected by the network which is slower than accessing main memory. However, when mini-batch size is increased, DoLDA performs faster than online LDA. When mini-batch size is 16,384, DoLDA is almost twice as fast as online LDA.

Next, we compare running time with various values of K, fixing the mini-batch size as 16,384. As expected, when we increase the number of topics, running time also increases. But with large K, performance advantage of DoLDA increases. This is a practical advantage of DoLDA because in many applications of LDA, it is important to try increasing K to fit the needs of the application.

Perplexity Another important aspect of a topic model is the model fit, measured often by perplexity per word on held-out data, estimated by a lower bound on perplexity [6].

We compared perplexity using various mini-batch sizes for online LDA and DoLDA. Figure 1 (a) shows the results. Online LDA and DoLDA show a similar pattern for the same mini-batch size. As the number of seen documents increases, perplexity converges for both algorithms and all mini-batch sizes. This shows that DoLDA performs as well as online LDA in terms of model fit.

Scalability One primary aspect of a distributed algorithm is scalability. We ran DoLDA on a 60-node Hadoop system. We fixed the mini-batch size at 16,384 and assigned various numbers of mappers for each job. Figure 1 (b) shows the results. As expected, when we increase the number of mappers, performance also improves. However, beyond eight mappers, performance does not improve anymore; in fact, performance slightly worsens because of Amdahl's law [5, 12].

[2] http://www.cs.princeton.edu/∼mdhoffma/

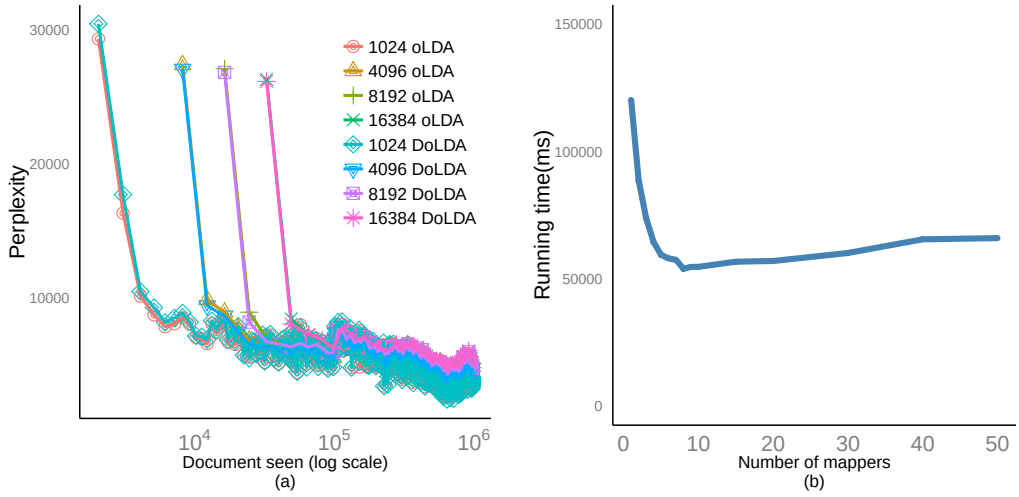

Figure 1: (a) Held-out perplexity between Online LDA (oLDA) and Distributed online LDA (DoLDA) with Twitter conversation corpus. As the number of seen documents increases, perplexity converges for all mini-batch sizes. (b) Running time of one MapReduce job for various numbers of mappers. As the number of mappers increase, running time decreases up to eight mappers but increases slightly beyond eight mappers.

4 Conclusion and Future Work

We introduced DoLDA, an online learning algorithm for LDA in a distributed system. We implemented it based on the open-source online LDA code [6], running on Hadoop system. Source code and data are available for research purposes [3]. We showed that the running time of DoLDA is as much as twice faster than online LDA, and the perplexity of DoLDA is comparable to that of online LDA.

This is ongoing work, and we are looking to improve the algorithm and conduct more experiments. In the current version of DoLDA, Driver gets new documents and sets up the MapReduce job. Mappers get documents from Driver and run the E step. However, in a more realistic setting, crawling the Web is distributed to many machines, so we can complete the E step on those machines with the documents that already reside locally. This will reduce the network I/O cost of communicating the documents and thereby improve efficiency. We also plan to develop a distributed online learning algorithm for hierarchical Dirichlet processes (HDP) [14, 15].

We ran the experiments in this paper with seven million tweets. However, real data such as Twitter data and user comments on news article or online forum is potentially much larger and generated in real time. For example, the Twitter public stream[4] generates more than 200,000 tweets per hour. Therefore, we would need to implement and experiment with DoLDA on a large cluster of machines.

Acknowledgments

We acknowledge the Department of Web Science & Technology, KAIST for providing the 60-node Hadoop system. This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0025706).

[3] http://uilab.kaist.ac.kr/research/NIPSBL12

[4] https://dev.twitter.com/docs/streaming-apis/streams/public

References

[1] Apache Mahout, http://mahout.apache.org.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[3] K. Canini, L. Shi, and T. Griffiths. Online inference of topics with latent dirichlet allocation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 65–72, 2009.

[4] T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[5] J. Hennessy and D. Patterson. *Computer architecture: a quantitative approach*. Morgan Kaufmann, 2011.

[6] M. Hoffman, D. Blei, and F. Bach. Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864, 2010.

[7] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[8] L. Mariote, C. Medeiros, and R. da Torres. Parallelized variational em for latent dirichlet allocation: An experimental evaluation of speed and scalability. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 349–354. IEEE, 2007.

[9] A. K. McCallum. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet, 2002.

[10] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.

[11] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[12] M. Schatz. Cloudburst: highly sensitive read mapping with mapreduce. *Bioinformatics*, 25(11):1363–1369, 2009.

[13] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.

[14] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[15] C. Wang, J. Paisley, and D. Blei. Online variational inference for the hierarchical dirichlet process. In *Proc. AISTATS*, 2011.

[16] Y. Wang, H. Bai, M. Stanton, W. Chen, and E. Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. *Algorithmic Aspects in Information and Management*, pages 301–314, 2009.

[17] K. Zhai, J. Boyd-Graber, N. Asadi, and M. L. Alkhouja. Mr. lda: a flexible large scale topic modeling package using variational inference in mapreduce. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 879–888, New York, NY, USA, 2012. ACM.