Asynchronous Stochastic Variational Inference

Saad Mohamad[1] Abdelhamid Bouchachia[1] and Moamar Sayed-Mouchaweh[2] *

1- Department of Computing - Bournemouth University
Poole, UK

2- Department of Informatics and Automatics - Ecole des Mines
Douai, France

Abstract. Stochastic variational inference (SVI) employs stochastic optimization to scale up Bayesian computation to massive data. Since SVI is at its core a stochastic gradient-based algorithm, horizontal parallelism can be harnessed to allow larger scale inference. We propose a lock-free parallel implementation for SVI which allows distributed computations over multiple slaves in an asynchronous style. We show that our implementation leads to linear speed-up while guaranteeing an asymptotic ergodic convergence rate $O(1/\sqrt{T})$ while the number of slaves is bounded by $\sqrt{T}$ (T is the total number of iterations). The implementation is done in a high-performance computing environment using message passing interface for python (MPI4py). The empirical evaluation shows that our parallel SVI is lossless, performing comparably well to its counterpart serial SVI with linear speed-up.

1 Introduction

Probabilistic models with latent variables have grown into a backbone in many modern machine learning applications such as text analysis, computer vision, time series analysis, network modelling, and others. The main challenge in such models is to compute the posterior distribution over some hidden variables encoding hidden structure in the observed data. Generally, computing the posterior is intractable and approximation is required. Markov chain Monte Carlo (MCMC) sampling has been the dominant paradigm for posterior computation. It constructs a Markov chain on the hidden variables whose stationary distribution is the desired posterior. Hence, the approximation is based on sampling for a long time to (hopefully) collect samples from the posterior [1].

Recently, variational inference (VI) has emerged as a deterministic alternative approach to Markov chain Monte Carlo (MCMC) sampling. In general, VI tends to be faster than MCMC which makes it more suitable for problems with large data sets. VI turns the inference problem to an optimization problem by positing a simpler family of distributions and finding the member of the family that is closest to the true posterior distribution [2]. Such optimization problem is a non-convex one which requires sophisticated tools to tackle. Stochastic optimisation has been applied to VI in order to cope with massive data [3]. While VI requires

*A. Bouchachia was supported by the European Commission under the Horizon 2020 Grant 687691 related to the project: *PROTEUS: Scalable Online Machine Learning for Predictive Analytics and Real-Time Interactive Visualization*

repeatedly iterating over the whole data set before updating the variational parameters (parameters of the variational objective), stochastic VI (SVI) updates the parameters every time a data example is processed. Therefore, by the end of one pass through the dataset, the parameters will have been updated multiple times. Hence, the parameters converge faster with less computational resources. The idea of SVI is to move the variational parameters at each iteration in the direction of a noisy estimate of the variational objective's natural gradient based on a couple of examples [3]. Following these gradients with certain conditions on the (decreasing) learning rate schedule, SVI provably converges to a local optimum [4].

Although stochastic optimization improves the performance of VI, its serial employment prevents scaling up the inference. Since SVI is basically a stochastic gradient-based optimisation algorithm, horizontal parallelism is straightforward. That is, computing stochastic gradients of a batch of data samples can be done locally in parallel (on multi-core machines) given that the parameters update is synchronised. However, such synchronisation limits the scalability by since slaves need to send their stochastic gradients to the master prior to each parameter update. Hence, synchronous methods suffer from the curse of the last reducer; that is, a single slow slave can dramatically slow down the whole performance. Thus, asynchronous parallel optimization is an interesting alternative provided it maintains comparable convergence rate to its synchronous counterpart. Indeed, asynchronous parallel stochastic gradient-based optimisation algorithms have recently received broad attention [5, 6, 7, 8, 9].

Authors in [6] show that for smooth stochastic convex problems the asynchronisation effects are asymptotically negligible and order-optimal convergence results can be achieved. Since the SVI objective function is non-convex, we are interested in the asynchronous parallel stochastic gradient algorithm (ASYSG) for smooth non-convex optimization [10]. A recent study [11] breaks the usual convexity assumption taken by [6]. Nonetheless, theoretical guarantees (convergence and speed-up) for many recent successes of ASYSG are reported. In this paper, we use the ASYSG algorithm proposed in [6] to come up with an asynchronous SVI (ASYSVI) algorithm for a wide family of Bayesian models. We also adapt the theoretical studies of ASYSG for smooth non-convex optimization from [11] to explain ASYSVIs' convergence and speed-up properties. We also propose a novel contribution to linearly speed up SVI by distributing its stochastic natural gradient computations in an asynchronous way while guaranteeing an ergodic convergence rate $O(1/\sqrt{T})$ under some assumptions. Latent Dirichlet allocation is used as a case study to empirically evaluate ASYSVI.

The rest of the paper is structured as follows. We describe our ASYSVI algorithm and its convergence analysis in Sec. 2. Latent Dirichlet allocation case study is derived in Sec. 3. Empirical evaluation is presented in Sec. 4 and the paper concludes with a discussion in Sec. 5. We also attached an appendix where a background on variational and stochastic variational inference is provided in Sec. A. Related work is discussed in Sec. B.

Algorithm 1 ASYSVI-Master

```
1: **Input:** number of iteration $T$ and step-size $\{\rho_t\}_{t=0,...,T-1}$
2: **initialize:** $\boldsymbol{\lambda}^0$ randomly and $t$ to 0
3: **while** ($t < T$) **do**
4:     Aggregate $M$ stochastic natural gradients from the slaves: $\hat{\nabla}\mathcal{L}_1(\boldsymbol{\lambda}^{t-\tau_{t,1}}), ..., \hat{\nabla}\mathcal{L}_M(\boldsymbol{\lambda}^{t-\tau_{t,M}})$
5:     Average the $M$ stochastic natural gradients. $G_M^t = \sum_m \hat{\nabla}\mathcal{L}_m(\boldsymbol{\lambda}^{t-\tau_{t,m}})$
6:     Update the current estimate of the global variational parameter. $\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho_t G_M^t$
7:     $t = t + 1$
8: **end while**
```

Algorithm 2 ASYSVI-Slave

```
1: **Input:** data size $D$
2: **while** (True) **do**
3:     Sample a data point $\boldsymbol{x}_i$ uniformly from the data set
4:     Pull a global variational parameter $\boldsymbol{\lambda}^*$ from the master
5:     Compute the local variational parameters $\boldsymbol{\phi}_i^*(\boldsymbol{\lambda}^*)$ corresponding to the data point $\boldsymbol{x}_i$ and the global variational parameter $\boldsymbol{\lambda}^*$, $\boldsymbol{\phi}_i^*(\boldsymbol{\lambda}^*) = \arg\max_{\boldsymbol{\phi}_i} \mathcal{L}_i(\boldsymbol{\lambda}^*, \boldsymbol{\phi}_i)$
6:     Compute the stochastic natural gradient with respect to the global parameter $\boldsymbol{\lambda}$, $\boldsymbol{g}_i(\boldsymbol{\lambda}) = \boldsymbol{\alpha} + DE_{\boldsymbol{\phi}_i(\boldsymbol{\lambda})}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] - \boldsymbol{\lambda}$
7:     Push $\boldsymbol{g}_i(\boldsymbol{\lambda}^*)$ to the master
8: **end while**
```

2 Asynchronous Stochastic Variational Inference

ASYSVI is presented analogously to ASYSG in [11] but in the context of VI. The architecture of the computer network on which ASYSVI is run is known as the *star-shaped* network. In this network, a master machine maintains the global variational parameter $\boldsymbol{\lambda}$, whereas other machines serve as slaves which independently and simultaneously compute the local variational parameters $\boldsymbol{\phi}$ and stochastic gradients of ELBO $\mathcal{L}(\boldsymbol{\lambda})$. The slaves only communicate with the master to exchange information in which they access the state of the global variational parameter and provide the master with the stochastic gradients. These gradients are computed with respect to $\boldsymbol{\lambda}$ based on few data points acquired from distributed sources. The master aggregates predefined amounts of stochastic gradients from slaves nonchalantly about the sources of the collected stochastic gradients. Then, it updates its current global variational parameter. The update step is performed as an atomic operation where slaves cannot read the value of the global variational parameter during this step. However, vertical parallelism can be achieved by adopting the ASYSG algorithm proposed in [5]. Furthermore, a hybrid horizontal-vertical parallelism could be achieved by combining the mechanism used in [12] with ASYSVI (more details in Sec. 5).

The key difference between ASYSVI and the synchronous parallel SVI is that ASYSVI does not lock the slaves until the master's update step is done. That is, the slaves might compute some stochastic gradients based on early value of the global variational parameter. By allowing delayed and asynchronous updates, one might expect slower convergence if any. In the next section, we apply the study of [11] on SVI to show that the effect of stochastic gradients delay will vanish asymptotically. The algorithms of ASYSVI-mater and ASYSVI-salve are shown in Alg. 1 and Alg. 2. We denote by $\tau_{t,m}$ the delays between the current iteration t and the one when the slave pulled the global variational parameter at which it computed the stochastic gradient.

2.1 Convergence Analysis

Following [11, 6], we take the same assumptions, but replace the gradient with the natural gradient:

- Unbiased gradient: the expectation of the stochastic natural gradient of Eq. (16) is equivalent to the natural gradient of Eq. (13):

$$\hat{\nabla}\mathcal{L}(\boldsymbol{\lambda}) = E[\hat{\nabla}\mathcal{L}_i(\boldsymbol{\lambda})] \tag{1}$$

  where $\hat{\nabla}$ denotes natural gradient. This assumption already holds in SVI problems for the family of models shown in Sec. A.

- Bounded variance: the variance of the stochastic natural gradient is bounded for all $\lambda \in \mathcal{G}$, $E[||\hat{\nabla}\mathcal{L}_i(\boldsymbol{\lambda}) - \hat{\nabla}\mathcal{L}(\boldsymbol{\lambda})||^2] \leq \sigma^2$. By applying SVI natural gradient, we obtain:

$$E[||nE_{\boldsymbol{\phi}_i(\boldsymbol{\lambda})}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] - \sum_{i=1}^{n} E_{\boldsymbol{\phi}_i(\boldsymbol{\lambda})}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)]||^2] \leq \sigma^2 \tag{2}$$

- Lipschitz-continuous gradient: the natural gradient is L-Lipschitz-continuous for all $\lambda \in \mathcal{G}$ an $\lambda' \in \mathcal{G}$, $||\hat{\nabla}\mathcal{L}(\boldsymbol{\lambda}) - \hat{\nabla}\mathcal{L}(\boldsymbol{\lambda}')|| \leq L||\lambda - \lambda'||$. By applying SVI natural gradient, we end up with the following formulation:

$$||\sum_{i=1}^{n} E_{\boldsymbol{\phi}_i(\boldsymbol{\lambda})}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] - \lambda - \sum_{i=1}^{n} E_{\boldsymbol{\phi}_i(\boldsymbol{\lambda}')}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] + \lambda'|| \leq L||\lambda - \lambda'|| \tag{3}$$

- Bounded delay: All delay variables $\tau_{t,m}$ are bounded:

$$\max_{t,m} \tau_{t,m} \leq B \tag{4}$$

In addition to these assumptions, authors [11, 6] assume that each slave receives a stream of independent data points. Although this assumption might not be satisfied strictly in practice, we follow the same assumption for analysis purpose. Thus, the same theoretical results obtained by [11] can be applied for ASYSVI, namely, an ergodic convergence rate $O(1/\sqrt{MT})$ provided that T is greater than $O(B^2)$. The results also show that, since the number of slaves is proportional to B, the ergodic convergence rate is achieved as long as the number of salves is bounded by $O(\sqrt{T/M})$. Note that $O(1/\sqrt{MT})$ is consistent with the serial stochastic gradient (SG) and the stochastic variational inference (SVI). Thus, ASYSG and ASYSVI allow for a linear speed-up if $B \leq O(\sqrt{T/M})$.

3 Case Study: Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is an instance of the family of models described in Sec A where the global, local, observed variables and their distributions are set as follows:

- the global variables $\{\boldsymbol{\beta}\}_{k=1}^{K}$ are the topics in LDA. A topic is a distribution over the vocabulary, where the probability of a word w in topic k is denoted by $\beta_{k,w}$. Hence, the prior distribution of $\boldsymbol{\beta}$ is a Dirichlet distribution $p(\boldsymbol{\beta}) = \prod_k Dir(\boldsymbol{\beta}_k; \boldsymbol{\eta})$
- the local variables are the topic proportions $\{\boldsymbol{\theta}_d\}_{d=1}^{D}$ and the topic assignments $\{\{z_{d,w}\}_{d=1}^{D}\}_{w=1}^{W}$ which index the topic that generates the observations. Each document is associated with a topic proportion which is a distribution over topics, $p(\boldsymbol{\theta}) = \prod_d Dir(\boldsymbol{\theta}_d; \boldsymbol{\alpha})$. The assignments $\{\{z_{d,w}\}_{d=1}^{D}\}_{w=1}^{W}$ are indices, generated by $\boldsymbol{\theta}_d$, that couple topics with words, $p(\boldsymbol{z}_d|\boldsymbol{\theta}) = \prod_w \theta_{d,z_{d,w}}$
- the observations $\boldsymbol{x}_d$ are the words of the documents which are assumed to be drawn from topics $\boldsymbol{\beta}$ selected by indices $\boldsymbol{z}_d$, $p(\boldsymbol{x}_d|\boldsymbol{z}_d, \boldsymbol{\beta}) = \prod_w \beta_{z_{d,w},x_{d,w}}$

In LDA, documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [13]. LDA assumes the following generative process:

1 Draw topics $\boldsymbol{\beta}_k \sim Dir(\eta, ..., \eta)$ for $k \in \{1, ..., K\}$

2 Draw topic proportions $\boldsymbol{\theta}_d \sim Dir(\alpha, ..., \alpha)$ for $d \in \{1, ..., D\}$

  2.1 Draw topic assignments $z_{d,w} \sim Mult(\boldsymbol{\theta}_d)$ for $w \in \{1, ..., W\}$

    2.1.1 Draw word $x_{d,w} \sim Mult(\boldsymbol{\beta}_{z_{d,w}})$

According to Sec. A, each variational distribution is assumed to come from the same family of the true one. Hence, $q(\boldsymbol{\beta}_k|\boldsymbol{\lambda}_k) = Dir(\boldsymbol{\lambda}_k)$, $q(\boldsymbol{\theta}_d|\boldsymbol{\gamma}_d) = Dir(\boldsymbol{\gamma}_d)$ and $q(z_{d,w}|\boldsymbol{\phi}_{d,w}) = Mult(\boldsymbol{\phi}_{d,w})$. To compute the stochastic natural gradient g_i in Alg. 2 for LDA, we need to find the sufficient statistic $t(.)$ presented in Eq. (7). By writing the likelihood of LDA in the form of Eq. (7), we obtain $t(\boldsymbol{x}_d, z_d) = \sum_{w=1}^{W} \mathbf{I}_{z_{d,w},x_{d,w}}$, where $\mathbf{I}_{i,j}$ is equal to 1 for entry (i, j) and 0 for all the rest. Hence, the stochastic natural gradient $g_i(\boldsymbol{\lambda}_k)$ can be written as follows:

$$g_i(\boldsymbol{\lambda}_k) = \eta + D \sum_{w=1}^{W} \phi_{i,w}^{k} \mathbf{I}_{k,x_{i,w}} - \boldsymbol{\lambda}_k \qquad (5)$$

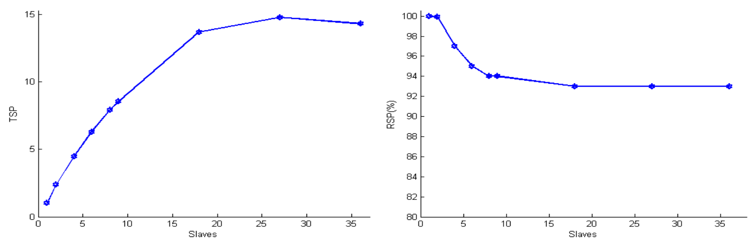
Table 1: Parameters settings

Data sets	Enron emails				NYTimes news articles				Wikipedia articles			
batch	16	64	256	1024	16	64	256	1024	16	64	256	1024
κ	0.7	0.7	0.5	0.5	0.7	0.7	0.5	0.5	0.7	0.7	0.5	0.5
τ_0	1024	24	24	1	1024	24	24	1	1024	1024	1024	1024
perplexity	5919	5348	5264	4771	11989	10156	9015	5501	1446	1390	1355	1332

Details on how to compute the local variational parameters $\boldsymbol{\phi}_i^*(\boldsymbol{\lambda}^*)$ in Alg. 2 can be found in [3].

Having computed the elements needed to run ASYSVI's algorithms 1 and 2, we move to the convergence analysis. Since the data is assumed to be subsampled uniformly, the unbiased gradient assumption holds for LDA. We can always find a constant variable to bound the variance. At the worst case, the variance of the stochastic natural gradient of LDA can be bounded by $DW\big(max_{i,w}(\phi_{i,w}^k)^2 - min_{i',w'}(\phi_{i',w'}^k)^2\big)$, $\forall k$. Therefore, it can be bounded by $O((DW)^2)$. It is clear that the Lipschitz-continuous gradient can be satisfied for any class of the family models proposed in Sec. A and hence, for LDA. Finally, the bounded delay can be guaranteed through the implementation. Therefore, ASYSVI of LDA can converge since the aforementioned assumptions can be satisfied.

4 Experimental Results

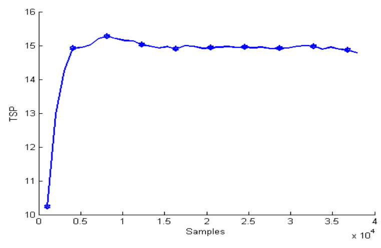


(a) TSP (b) RSP

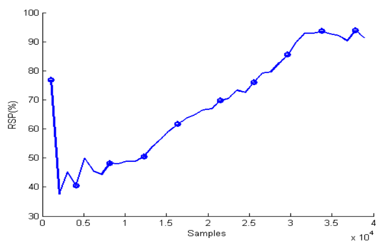
Fig. 1: Comparing ASYSVI LDA to online LDA on *Enron* dataset

In the following, we demonstrate the usefulness of distributing the computation of SVI, mainly the speed-up advantages of ASYSVI. For this purpose, we compare the speed-up of ASYSVI LDA against serial SVI LDA (online LDA [14]). The two versions are evaluated on three datasets consisting of very large document collections. We also evaluate ASYSVI LDA in the streaming setting where new documents arrive in the form of stream. The implementation is available in PROTEUS SOLMA Library [1]. The evaluation is done using held-out perplexity as a measure of model fit. Perplexity is defined as the geometric mean of the inverse marginal probability of each word in the held-out set of documents [13].

[1]https://github.com/proteus-h2020/proteus-solma/tree/master/src/main/scala/eu/proteus/solma/asvi

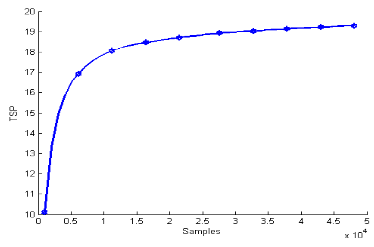


(a) TSP on *Enron* dataset

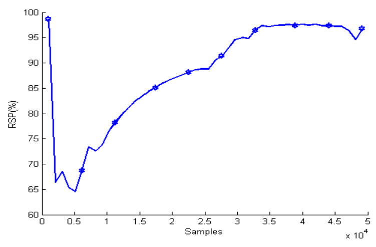


(b) RSP on *Enron* dataset

Fig. 2: TSP and RSP with respect to streamming samples on *Enron* dataset

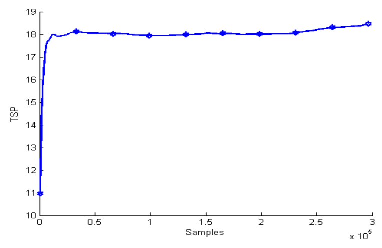


(a) TSP on *NYTimes* dataset

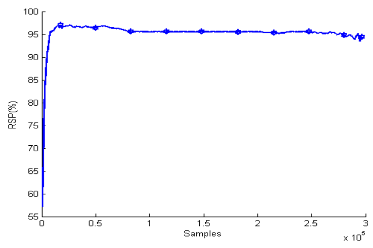


(b) RSP on *NYTimes* dataset

Fig. 3: TSP and RSP with respect to streamming samples on *NYTimes* dataset



(a) TSP on *Wikipedia* dataset



(b) RSP on *Wikipedia* dataset

Fig. 4: TSP and RSP with respect to streaming samples on *Wikipedia* dataset

To validate the speed-up properties, following [11], we compute the running time speed-up (TSP):

$$TSP = \frac{\text{running time of SVI-LDA}}{\text{running time of DPSVI-LDA}}$$

The running time of both models is taken when they achieve the same final held-out perplexity.

Datasets: we perform all comparisons and evaluations on three corpora of documents. The first two corpora are available on [15]. The third corpus was used in [14].

- *Enron* emails: contains $39,861$ email messages from about 150 users. Data is pre-processed by removing all words not in a vocabulary dictionary of $28,102$ words.
- *NYTimes* news dataset: contains $300,000$ news articles from *New York Times*. Data is pre-processed by removing stopwords (not in a dictionary of $102,660$ words).
- *Wikipedia* articles: contains $1M$ documents downloaded from *Wikipedia*. Data is proceeded before usage by removing all words not in a vocabulary dictionary of $7,700$ words.

Settings the parameters: In all experiments, α and η are fixed at 0.01 and the number of topics $K = 50$. We evaluated a range of settings of the learning parameters, κ, τ_0, and *batch* on all corpora. The parameters κ and τ_0, defined in [14], control the learning step-size ρ_t. We use $29,861$ emails from *Enron* data, $50,000$ news articles from *NYTimes* data and $300,000$ documents from *Wikipedia* data as training sets. We also reserve $5,000$ documents as a validation set and another $5,000$ documents as a testing set. The online LDA is run (one time per corpus) on the training sets for $\kappa \in \{0.5, 0.7, 0.9\}$, $\tau_0 \in \{1, 24, 256, 1024\}$, and *batch* $\in \{16, 64, 256, 1024\}$. Table 1 summarises the best settings of each *batch* along with the resulting perplexity on the test set for each corpus.

Comparing Serial online LDA and asynchronous LDA: for each dataset, we set the parameters setting that give the best performance (least perplexity). ASYSVI LDA is then compared against serial SVI LDA using the same parameters setting.

The empirical results shown in this paper are obtained from a python implementation on high-performance computing (HPC) environment using message passing interface (MPI) for python (MPI4py). The cluster consists of 10 nodes, excluding the head node, with each node is a four-core processor. We run AYSVI LDA on *Enron* dataset for number of workers $nW \in \{2, 4, 6, 8, 9, 18, 27, 36\}$, B is set to 5. The number of employed nodes is equal to nW as long as nW is less than 9. As nW becomes higher than the available nodes, the processors' cores of nodes are employed as slaves until all the cores of all the nodes are used i.e., $9 \times 4 = 36$. Since the batch size is fixed to 1024, each slave processes a batch of data of size $S = 1024/M$ per iteration, where M is fixed to 36. Thus, the gradient computed by each slave will be multiplied by D/S. Hence, line number 6 of Alg. 2 becomes, $\boldsymbol{g}_i(\boldsymbol{\lambda}) = \boldsymbol{\alpha} + (D/S)E_{\boldsymbol{\phi}_i(\boldsymbol{\lambda})}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] - \boldsymbol{\lambda}$.

Figure 1 summarises the total speed-up (i.e., TSP measured at the end of the algorithm) as well as the ratio of serial LDA perplexity to parallel LDA (RSP) on the test set for *Enron* dataset. It shows TSP and RSP results with respect to the number of slaves. It is clear that as long as each node is assigned one slave, the speed-up is linear which demonstrates the convergence analysis done in Sec. 2. Linear speed-up slowly converts to sub-linear as solo machines host more than one slave. The main reason of such behaviour is the communication delay caused by the increase of the network traffic. Hence, TSP is affected by the hardware. The communication cost starts affecting the speed-up when it becomes comparable to the local computation. Hence, increasing the local computation by increasing the batch size can be adopted to soften the communication effect. However, this decreases the convergence rate and increase local memory load. Hence, a balanced trade-off should be considered. RSP in Figure 1 shows that although the speed of online LDA has been increased up to 15 times, performance is not seriously affected. We also evaluate TSP and RSP on *NYTimes* and *Wikipedia* for $nW = 27$. The processing speed of Online LDA on *NYTimes* has been increased 19.29 times, $TSP = 19.29$, with slight loss of performance, $RSP = 0.97$. For *Wikipedia*, $TSP = 18.58$ and $RSP = 0.94$.

Figures 2, 3 and 4 present TSP and RSP with respect to streaming samples from *Enron*, *NYTimes* and *Wikipedia* datasets. These figures show the performance of ASYSVI in a true online setting where the algorithm continually collects samples from the hard driver for the case of *Enron* and *NYTimes* or by downloading online in the case of *Wikipedia*. The perplexity is obtained online on the coming batches before being used to update the model parameters. The plots in the figures are slightly softened using a low-pass filter in order to make them easy to read. These plots show that the speed-up becomes invariant as more samples are processed. The poor speed-up in the beginning is normally caused by initialization and loading process. It can be noticed that the performance of ASYSVI LDA suffers at the beginning then it becomes comparable to online LDA after certain number of iterations. This behaviour can be explained by the convergence condition shown in Sec. 2 (T is greater than $O(B^2)$). Thus, as the number of iterations increases, the convergence of ASYSVI LDA is guaranteed and its performance becomes comparable to that of online LDA. Hence, RSP approaches 1.

5 Conclusion and Discussion

We have introduced ASYSVI, an asynchronous parallel implementations for SVI on computer cluster. ASYSVI leads to linear speed-up, while guaranteeing an asymptotic convergence rate given some assumptions involving the number of the slaves and iterations. Empirical results using latent Dirichlet allocation topic model as a case study have demonstrated the advantages of ASYSVI over SVI, particularly with respect to the key issue of speeding-up the computation while maintaining comparable performance to SVI.

In future work, vertical parallelism can be adopted along with the proposed horizontal one leading to a hybrid horizontal-vertical parallelism. In such case, multi-core processors will be used for the vertical parallelism, while horizontal parallelism is achieved on a multi-node machine. Another avenue of interest is to derive an algorithm for streaming, distributed, asynchronous inference where the number of instances is not known. Moreover, it is interesting to apply ASYSVI on very large scale problems and particularly on other models of the family discussed in Sec. A and studying the effect of the statistical properties of those models.

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[2] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[3] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[4] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[5] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[6] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[7] Ruiliang Zhang and James T Kwok. Asynchronous distributed admm for consensus optimization. In *ICML*, pages 1701–1709, 2014.

[8] Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754, 2016.

[9] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

[10] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[11] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745, 2015.

[12] Parameswaran Raman, Jiong Zhang, Hsiang-Fu Yu, Shihao Ji, and SVN Vishwanathan. Extreme stochastic variational inference: Distributed and asynchronous. *arXiv preprint arXiv:1605.09499*, 2016.

[13] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[14] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] Antti Honkela and Harri Valpola. On-line variational bayesian learning. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pages 803–808, 2003.

[17] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. In *Advances in Neural Information Processing Systems*, pages 1727–1735, 2013.

[18] Willie Neiswanger, Chong Wang, and Eric Xing. Embarrassingly parallel variational inference in nonconjugate models. *arXiv preprint arXiv:1510.04163*, 2015.

A Background

We derive the model family studied in this paper and review SVI following the same pattern as in [3].

Model family. Our family of models consists of three random variables: observations $\boldsymbol{x} = \boldsymbol{x}_{1:n}$, local hidden variables $\boldsymbol{z} = \boldsymbol{z}_{1:n}$, global hidden variables $\boldsymbol{\beta}$ and fixed parameters $\boldsymbol{\alpha}$. The model assumes that the distribution of the n pairs of $(\boldsymbol{x}_i, \boldsymbol{z}_i)$ is conditionally independent given $\boldsymbol{\beta}$. Further, their distribution and the prior distribution of $\boldsymbol{\beta}$ are in an exponential family:

$$p(\boldsymbol{\beta}, \boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\alpha}) = p(\boldsymbol{\beta}|\boldsymbol{\alpha}) \prod_{i=1}^{n} p(\boldsymbol{z}_i, \boldsymbol{x}_i|\boldsymbol{\beta}), \tag{6}$$

$$p(\boldsymbol{z}_i, \boldsymbol{x}_i|\boldsymbol{\beta}) = h(\boldsymbol{x}_i, \boldsymbol{z}_i) \exp\left(\boldsymbol{\beta}^T t(\boldsymbol{x}_i, \boldsymbol{z}_i) - a(\boldsymbol{\beta})\right), \tag{7}$$

$$p(\boldsymbol{\beta}|\boldsymbol{\alpha}) = h(\boldsymbol{\beta}) \exp\left(\boldsymbol{\alpha}^T t(\boldsymbol{\beta}) - a(\boldsymbol{\alpha})\right) \tag{8}$$

Here, we overload the notation for the base measures $h(.)$, sufficient statistics $t(.)$ and log normalizer $a(.)$. While the proposed approach is generic, we assume a conjugacy relationship between $(\boldsymbol{x}_i, \boldsymbol{z}_i)$ and $\boldsymbol{\beta}$. That is, the distribution $p(\boldsymbol{\beta}|\boldsymbol{x}, \boldsymbol{z})$ is in the same family as the prior $p(\boldsymbol{\beta}|\boldsymbol{\alpha})$.

Note that this innocent looking family of models includes (but is not limited to) latent Dirichlet allocation [13], Bayesian Gaussian mixture, probabilistic matrix factorization, hidden Markov models, hierarchical linear and probit regression, and many Bayesian non-parametric models.

Mean-field variational inference. Variational inference (VI) approximates intractable posterior $p(\boldsymbol{\beta}, \boldsymbol{z}|\boldsymbol{x})$ by positing a family of simple distributions $q(\boldsymbol{\beta}, \boldsymbol{z})$ and find the member of the family that is closest to the posterior (closeness is measured with KL divergence). The resulting optimization problem is equivalent maximizing the evidence lower bound (ELBO):

$$\mathcal{L}(q) = E_q[\log p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\beta})] - E_q[\log p(\boldsymbol{z}\boldsymbol{\beta})] \leq \log p(\boldsymbol{x}) \tag{9}$$

Mean-field is the simplest family as it allows the distribution over hidden variables to factorize:

$$q(\boldsymbol{\beta}, \boldsymbol{z}) = q(\boldsymbol{\beta}|\boldsymbol{\lambda}) \prod_{i=1}^{n} p(\boldsymbol{z}_i|\boldsymbol{\phi}_i) \tag{10}$$

Each variational distribution is assumed to come from the same family of the true one. Mean-field VI optimizes the new ELBO with respect to the local and global variational parameters $\boldsymbol{\phi}$ and $\boldsymbol{\lambda}$:

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi}) = E_q\left[\log \frac{p(\boldsymbol{\beta})}{q(\boldsymbol{\beta})}\right] + \sum_{i=1}^{n} E_q\left[\log \frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{\beta})}{q(\boldsymbol{z}_i)}\right] \tag{11}$$

It iteratively updates each variational parameter holding the others fixed. With the assumptions taken so far, each update has a closed form solution. The local parameters are a function of the global ones:

$$\boldsymbol{\phi}(\boldsymbol{\lambda}_t) = \arg\max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\lambda}_t, \boldsymbol{\phi}) \tag{12}$$

The global parameters summarise the dataset (clusters in Bayesian Gaussian mixture, topics in LDA):

$$\mathcal{L}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi}) \tag{13}$$

To find optimal $\boldsymbol{\lambda}$ given fixed $\boldsymbol{\phi}$, we compute the natural gradient of $\mathcal{L}(\boldsymbol{\lambda})$ and set it to zero by setting:

$$\boldsymbol{\lambda}^* = \boldsymbol{\alpha} + \sum_{i=1}^{n} E_{\boldsymbol{\phi}_i(\boldsymbol{\lambda}_t)}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] \tag{14}$$

Thus, the new optimal global parameters are $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}^*$. The algorithm works by iterating between computing the optimal local parameters given the global ones $\big($Eq. (12)$\big)$ and vice versa $\big($Eq. (14)$\big)$.

Stochastic variational inference. Rather than analysing all the data to compute $\boldsymbol{\lambda}^*$ at each iteration, stochastic optimization can be used. Assuming that the data is uniformity at random selected from the dataset, an unbiased noisy estimator of $\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi})$ can be developed based on a single data point:

$$\mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\phi}_i) = E_q\left[\log \frac{p(\boldsymbol{\beta})}{q(\boldsymbol{\beta})}\right] + nE_q\left[\log \frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{\beta})}{q(\boldsymbol{z}_i)}\right] \tag{15}$$

The unbiased stochastic approximation of the ELBO as a function of $\boldsymbol{\lambda}$ can be written as follows:

$$\mathcal{L}_i(\boldsymbol{\lambda}) = \max_{\boldsymbol{\phi}_i} \mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\phi}_i) \tag{16}$$

Following the same step in the previous section, we obtain a noisy unbiased estimate of Eq. (14):

$$\hat{\boldsymbol{\lambda}} = \boldsymbol{\alpha} + nE_{\boldsymbol{\phi}_i(\boldsymbol{\lambda}_t)}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] \tag{17}$$

Iteratively, we move the global parameters a step-size ρ_t in the direction of the noisy natural gradient:

$$\boldsymbol{\lambda}_{t+1} = (1 - \rho_t)\boldsymbol{\lambda}_t + \rho_t \hat{\boldsymbol{\lambda}} \tag{18}$$

With certain conditions on ρ_t, the algorithm converges ($\sum_{t=1}^{\infty} \rho_t = \infty$, $\sum_{t=1}^{\infty} \rho_t^2 < \infty$)[4].

B Related Work

Few work has been proposed to scale VI to large datasets. We can distinguish two major classes. The first class is based on the Bayesian filtering approach [16, 17]. That is, the sequential nature of Bayes theorem is exploited to recursively update an approximation of the posterior. Particularly, VI is used between the updates to approximate the posterior which becomes the prior of the next step. Author in [16] uses forgetting factors to decay the contribution of old data in favour of a new better one. The algorithm proposed in [17] considers a sequence of data batches and iterates over the data in each batch until convergence. Relying on a master-slave architecture, the computation of the batches posterior is done in a distributed and asynchronous manner. That is, the algorithm applies VI by performing asynchronous Bayesian updates to the posterior as data batches arrive continuously.

The second class of work is based on optimization [3, 18, 12]. As we already discussed, SVI proposed by [3] employs stochastic optimization to scale up Bayesian computation to massive data. SVI is inherently serial and requires the model parameters to fit in the memory of a single processor. Authors in [18] present a VI based inference algorithm that runs in parallel on data divided across several slaves. However at each iteration, the slaves are synchronized to combine their obtained parameters. Such synchronisation limits the scalability and decreases the speed of the update to that of the slowest slave. To avoid bulk synchronization, authors in [12] propose an asynchronous and lock-free update. In this update, vertical parallelism is adopted, where each processor asynchronously updates a subset of the parameters based on a subset of attributes. In contrast, we adopt horizontal parallelism update based on few (mini-batched or single) data points acquired from distributed sources. The update steps are aggregated to form the global update. Our proposed approach can make use of the mechanism proposed by [12] to achieve a hybrid horizontal-vertical parallelism. On the contrary to [12], our approach is not customised for LDA and can be applied to any of the model family presented in Sec. A