An Asynchronous Variance Reduced Framework for Efficient Bayesian Deep Learning

Chi Zhang,* Jiasheng Tang,* Hao Li, Cheng Yang, Shenghuo Zhu, Rong Jin
Institute of Data Science and Technology,
Alibaba Group, Hangzhou, 311000, China
{yutou.zc, jiasheng.tjs, lihao.lh, charis.yangc,
shenghuo.zhu, jinrong.jr}@alibaba-inc.com

Abstract

Although Bayesian deep learning (BDL) combines the power of deep learning and Bayesian inference, its high computational cost makes it difficult to apply to large-scale data. We address this challenge by introducing an asynchronous variance reduced distributed framework, based on Stein Variance Gradient Descent (SVGD), for large-scale BDL training. The key advantage of our framework, termed as AVR-SVGD, is that multiple solutions introduced by SVGD can be significantly different from each other, leading to a greater degree of asynchronousness among solutions and consequentially more efficient computing in optimization. We further apply variance reduction technique to improve the convergence. Empirical studies show the effectiveness of AVR-SVGD.

1 Introduction

Bayesian deep learning was proposed to combine the power of deep learning and Bayesian inference. It has been successfully applied to multiple important problems such as click-through-rate estimation and image classification. We refer to [15] and references therein for a comprehensive survey. Despite the promising results, the high computational cost of BDL makes it difficult to apply for large datasets. Most existing works for BDL are based on either sampling approaches or approximate inference methods [15, 11, 13]. The conventional sampling methods, such as Markov Chain Monte Carlo (MCMC)[1, 2], are usually slow in convergence, and do not scale well to large datasets [9, 16]. Approximate inference methods, variational inference [6, 5] and expectation propagation [10], often have to introduce extra assumptions on latent variables, significantly limiting its applications.

In this work, we focus on developing a distributed framework for BDL. Although multiple distributed computing frameworks have been successfully developed for various machine learning models [17, 7, 3, 4], only few studies were devoted to BDL. [12] introduces a distributed computing framework, based on expectation propagation, for Bayesian learning. Its assumption of exponential family and usage of a centralized server, unfortunately, makes the method not preferred in practice. It is well recognized that, with increasing amount of data, asynchronousness is the key to effectively leverage the power of computer clusters. To this end, we propose and develop an asynchronous variance reduced framework, based on Stein variational gradient descent (SVGD) [8, 14], for efficient BDL. The key advantage of our framework, termed as AVR-SVGD, is that multiple solutions introduced by SVGD can be significantly different from each other, leading to a greater degree of asynchronousness among solutions and consequentially better convergence in optimization. This is in contrast to most distributed optimization frameworks where to ensure appropriate convergence, certain degree of consistency among different copies of solutions has to be maintained, significantly limiting the improvement in computation. In addition, we introduce variance reduction technique to further

*The authors contribute equally to this work.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

improve the computational efficiency. The effectiveness of our method is validated from experiments with real data.

2 An Asynchronous Variance Reduced SVGD

We first introduce the Stein variational gradient descent (SVGD) method for Bayesian deep learning. We then present our asynchronous variance reduced framework for distributed BDL, based on SVGD. We finally describe the technique of variance reduction within our framework.

2.1 Stein Variational Gradient Descent (SVGD) for Bayesian Deep Learning

The SVGD framework was first proposed in [8, 14] for variational inference. It differs from the conventional Bayesian learning approaches in that it explicitly introduces multiple different solutions to approximate the posterior distributions. These solutions are efficiently updated by explicitly minimizing the Kullback-Leibler (KL) divergence in a reproduced kernel Hilbert space. More specifically, let M be the number of solutions, or particles as called from the original work, and let $\kappa(\cdot,\cdot)$ be the kernel function. At each iteration t, we denote by $x_i^t, i \in [M]$ the current particles. To effectively minimize the KL divergence, each particle is updated as followed:

$$x_i^{t+1} \leftarrow x_i^t + \epsilon\phi^*(x_i^t) \quad \text{where} \quad \phi^*(x) = \frac{1}{M}\sum_{m=1}^{M}[\kappa(x_m^t, x)\nabla_{x_m^t}\log p(x_m^t) + \nabla_{x_m^t}\kappa(x_m^t, x)] \quad (1)$$

Our key observation of SVGD is that the multiple solutions $x_i^t, i \in [M]$ maintained by SVGD can be very different from each other. This is in contrast to typical framework of distributed optimization that requires strong degree of consistency among different copies of solutions in order to ensure the convergence of optimization. It is the property that allows us to introduce a large degree of asynchronousness among solutions, which become the basis of our distributed framework for BDL. We should note that early works of SVGD focused on algorithmic development, and did not examine its properties in distributed computing.

2.2 An Asynchronous Variance Reduced SVGD

In the synchronous implementation of SVGD, each particle has to wait for the arrival of ALL other particles before it can be updated, making it computationally less attractive. We address this problem by introducing an asynchronous framework for SVGD. More specifically, for each particle x_i^t, we will randomly select $k < M$ particles, and calculate the gradient of x_i^t as long as the solutions for the selected particles are ready, i.e.

$$\phi_a^*(x_i^t) = \frac{1}{k}\sum_{m=1}^{M} Z(x_m^t)\left[\kappa(x_m^t, x_i^t)\nabla_{x_m^t}\log p(x_m^t) + \nabla_{x_m^t}\kappa(x_m^t, x_i^t)\right] \quad (2)$$

where $Z(x_m^t)$ is an indicator function denoting if x_m^l is selected. It is easy to see that if all $Z(x_m^t)$ are Bernoulli random variable with $\Pr(Z(x_m^t) = 1) = k/M$, we have $\mathrm{E}(\phi_a^*(x_i^t)) = \phi^*(x_i^t)$, and as a result, $\phi_a(x_i^t)$ is an unbiased estimation of $\phi^*(x_i^t)$, ensuring the correctness of our method. By requesting each particle to listen to only a small subset of particles, we could greatly reduce the waiting time of each particle, leading to significant improvement in efficiency. We note that it is the fact that all the particles share different solutions that leads to unbiased estimation of gradients and consequentially more efficient computing. In practice, we simply let each particle to wait for the solutions of the first k arriving particles, which seems to work quite well despite its violation of unbiased estimation of gradients.

Besides the asynchronous framework for SVGD, we also introduce variation reduction technique into the framework to further improve the computational efficiency. The key idea is to let each particle be updated by different batches of data that are randomly chosen. More specifically, at each time t, we select M batches of training data for updating, denoted by $\mathbf{D^t} = (D_1^t, \ldots, D_M^t)$. For each $\log p(x_m^t)$ in (1), it will be computed based on data batch D_m^t. By having each particle computes its gradient over its own data batch, we allow a larger degree of asynchronousness for our framework, leading to even

higher computational efficiency. In addition, this approach introduce additional benefits of variance reduction. Let $\phi^*(x; D)$ denote by the gradient where all the particles use the same data batch D ,and $\phi^*(x; \mathbf{D})$ be the gradient where different particles use different random data batches. It is relatively straightforward to show that under appropriate conditions, $\mathrm{VAR}(\phi^*(x; D)) \geq \mathrm{VAR}(\phi^*(x; \mathbf{D})$.

We summarize the key steps of the proposed AVR-SVGD framework in Algorithm 1. By allowing each particle to be updated as long as the selected messages arrive, we are able to greatly relax the synchronous requirement. By introducing different data batches for computing the gradients of different particles, we reduce the variance in estimating the gradients, and therefore accelerate the optimization process.

Algorithm 1 The Asynchronous Variance Reduced SVGD

In any particle x with index i, $i \in (1, 2, \dots M)$, the learning rate is ϵ
For all t $\in \{1, 2, \dots, T\}$, with training data $\mathbf{D^t} = \{D_1^t, \dots, D_M^t\}$, **do**:
Step 1: Derive $\nabla_{x_i^t} \log p(x_i^t)$
Step 2: Broadcast (Push) x_i^t and $\nabla_{x_i^t} \log p(x_i^t)$ to all other nodes which can be accessed.
Step 3: Derive $\phi_a^*(x_i^t, \mathbf{D}) = \frac{1}{k} \sum_{m=1}^{M} Z(x_m^t)[\kappa(x_j^t, x_i^t) \nabla_{x_j^t} \log p(x_j^t, D_m) + \nabla_{x_j^t} \kappa(x_j^t, x_i^t)]$
Step 4: Update: $x_i^{t+1} = x_i^t + \epsilon \phi_a^*(x_i^t, \mathbf{D})$
End for

3 Experiments

We demonstrate our method AVR-SVGD on the ImageNet 2012 Dataset. For fast experiment, we first extract features from the outputs of the convolution layer of a ResNet-18. We compare our AVR-SVGD with SVGD. In AVR-SVGD, we select three scenarios $k/M = \{0.4, 0.7, 1\}$ for illustration. The number of particles is set to 20 in all methods. The size of data batch for each particle is set to be 16, thus the total batch size for each iteration is 320 in AVR-SVGD. The validation losses of all methods are plotted in Figure 1. We first see that when AVR-SVGD significantly outperforms SVGD when $k/M = 1$, indicating that our variance reduction technique works well. We then observe a further improvement in optimization when $k/M = 0.7$, implying that our asynchronous scheme gives additional boost in optimization performance. We also compare the accuracy of AVR-SVGD with that of SVGD, and the results are shown in Table 1. We find that AVR-SVGD performs slightly better than SVGD. Therefore, AVR- SVGD is able to not only accelerate the training process, but also deliver a strong prediction power. Moreover, the accuracy of the DNN model (non-bayesian SGD trained model) is also listed in Table 1. We can clearly see that the AVR-SVGD outperforms the DNN model under all k/M scenarios, indicating that AVR-SVGD is more effective in learning deep models than DNN.

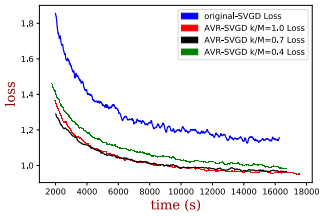

Figure 1: Validation Loss

Method	k/M	Top 1 Accu	Top 5 Accu
DNN	None	66.6%	87.4%
SVGD	None	67.1%	87.8%
AVR-SVGD	1	**67.5%**	**88.1%**
AVR-SVGD	0.7	67.5%	88.0%
AVR-SVGD	0.4	67.3%	87.8%

Table 1: Prediction Accuracy

4 Conclusion

In this paper, we propose an efficient AVR-SVGD framework for BDL. This framework fully leverage the advantage of SVGD and introduces appropriate asynchronous mechanism and variance reduction mechanism into the updating equations of SVGD, making it significantly more efficient compared to the synchronous implementation. The efficiency of our training framework is proved from real data experiments. Currently, we are investigating the efficiency of our framework for E-commence

data from Alibaba that is comprised of millions of unbalanced records. We are also considering to deliver the end-to-end deep network training for AVR-SVGD, instead of utilizing the dense feature from a ResNet-18 in the ImageNet experiment. The last but not the least, we will further compare AVR-SVGD and other ensemble models to show the benefit by applying the Bayesian mechanism.

References

[1] Christophe Andrieu, Nando De Freitas, and Arnaud Doucet. Sequential mcmc for bayesian model selection. In *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*, pages 130–134. IEEE, 1999.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3(1):993–1022, 2003.

[3] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[5] 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.

[6] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[7] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 1, page 3, 2014.

[8] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

[9] Dougal Maclaurin and Ryan P Adams. Firefly monte carlo: Exact mcmc with subsets of data. In *UAI*, pages 543–552, 2014.

[10] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *UAI*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[11] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 569–577. ACM, 2008.

[12] Yee Whye Teh, Leonard Hasenclever, Thibaut Lienart, Sebastian Vollmer, Stefan Webb, Balaji Lakshminarayanan, and Charles Blundell. Distributed bayesian learning with stochastic natural-gradient expectation propagation and the posterior server. *arXiv preprint arXiv:1512.09327*, 2015.

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

[14] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

[15] Hao Wang and Dit-Yan Yeung. Towards bayesian deep learning: A survey. *arXiv preprint arXiv:1604.01662*, 2016.

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

[17] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.