Accelerating Distributed Stochastic L-BFGS by sampled 2^{nd}-Order Information

Jie Liu[1,2] **Yu Rong**[1] **Martin Takáč**[2] **Junzhou Huang**[1]
[1]Tencent AI Lab [2]Lehigh University
jie.liu@alum.lehigh.edu, royrong@tencent.com, Takac.MT@gmail.com, joehhuang@tencent.com

Abstract

This paper proposes a framework of L-BFGS based on the (approximate) second-order information with stochastic batches, as a novel approach to the finite-sum minimization problems. Different from the classical L-BFGS where stochastic batches lead to instability, we use a smooth estimate for the evaluations of the gradient differences while achieving acceleration by well-scaling the initial Hessians. We provide theoretical analyses for both convex and nonconvex cases. In addition, we demonstrate that within the popular applications of least-square and cross-entropy losses, the algorithm admits a simple implementation in the distributed environment. Numerical experiments support the efficiency of our algorithms.

1 Introduction

We consider the finite-sum minimization problem of the form

$$\min_{w\in\mathbb{R}^d}\left\{F(w) := \frac{1}{n}\sum\nolimits_{i\in[n]} f(w;x_i,z_i) \stackrel{\text{def}}{=} \frac{1}{n}\sum\nolimits_{i\in[n]} f_i(w)\right\}, \tag{1}$$

where $i\in[n]\stackrel{\text{def}}{=}\{1,\ldots,n\}$, and $\{(x_i,z_i)\}_{i=1}^{i=n}$ are the data pairs. Throughout the paper, we assume there exists a global optimal solution w^* of (1); in other words, we have a lower bound $F(w^*)$ of (1).

In general, the problem of form (1) covers a wide range of convex and nonconvex problems including logistic regression [9], multi-kernel learning [3, 40], conditional random fields [20], neural networks [14], etc. Classical first-order methods to solve (1) are gradient descent (GD) [33] and stochastic gradient descent (SGD) [36, 38]. A large class of optimization methods can be used to solve (1), where the iterative updates can be generalized as follows,

$$w_{k+1} = w_k + \alpha_k p_k, \text{with } p_k = -H_k g_k, \tag{2}$$

where p_k is some descent direction, H_k is an inverse Hessian approximation of F at w_k, and g_k is an estimate of $\nabla F(w_k)$.

When H_k is an identity matrix, the update is considered a first-order method. Numerous work has focused on the choice of g_k such as SAG/SAGA [37, 11], MISO/FINITO [23, 12], SDCA [39], SVRG/S2GD [17, 19], SARAH [30, 31]. Nevertheless, with the importance of second-order optimization providing potential curvature around local optima and thus promoting fast convergence, the choice of non-identity H_k is crucial to the development of modern optimization algorithms.

Within the framework of second-order optimization, a popular choice for H_k is the inverse Hessian; however, we lack an efficient way to invert matrices, leading to increases in computation and communication costs to a problem for the distributed setting. Motivated by this, quasi-Newton methods, among which BFGS is one of the most popular, were developed, including a practical variant named limited-memory BFGS (L-BFGS) [33]. It has been widely known that batch methods

Preprint. Under review.

have been successfully applied in first-order algorithms and provide effective improvements, but it remains a problem for L-BFGS due to the instability caused by randomness between different gradient evaluations.

Related Works In particular, Agarwal et al. [1] introduce a method to approximate the inverse Hessian with Taylor expansion. Bach et al. [4] try to combine SGD and least-mean-square algorithms with Hessian information and formulates a stochastic algorithm which inherits the $\mathcal{O}(1/t)$ convergence rate. Byrd et al. [7] provides a framework for the batch L-BFGS, and subsequently, Moritz et al. [29] proposes a stochastic L-BFGS algorithm as to combine SVRG stochastic gradient with the L-BFGS to support linear convergence for strongly convex objectives. Berahas et al.[6] proposes a multi-batch L-BFGS algorithm, evaluating the y_k based on overlaps of two consecutive batches in L-BFGS and they provide a sub-linear convergence for non-convex objectives under reasonable assumptions. This work is extended e.g. in [16, 5, 15]

Our contributions In this paper, we analyze L-BFGS with stochastic batches for both convex and nonconvex optimization (Section B), as well as its distributed implementation. We propose a few variants of LBFGS algorithm (see Section A for more details). Instead of using the differences of gradients, LBFGS-H uses Hessian information directly [7, 29]. LBFGS-F combines L-BFGS with Fisher information matrix from the natural gradient algorithm [2, 25, 34]. We show that they are efficient for minimizing finite-sum problems both in theory and in practice. The key contributions of our paper are summarized as follows.

- We apply a technique for approximating the differences of gradients in the stochastic L-BFGS algorithm [7, 29] that ensures stability for the general finite-sum problems. Further, we introduce a variant of stochastic L-BFGS called LBFGS-F where the Hessian matrix is replaced by the Fisher information matrix [25] and demonstrate it applicable to a distributed environment with popular loss functions.
- We show that under standard assumptions [6], with any unbiased stochastic gradient, the framework converges to a neighborhood of the optimal solution with a linear convergence for strongly convex functions and a sublinear convergence for nonconvex functions. Additionally, under the same assumptions, it comes with a sublinear convergence for strongly convex objectives.
- With a potential acceleration in practice using ADAM techniques [18], we verify the competitive performances of both LBFGS-H and LBFGS-F against mainstream first- and second-order optimization methods in both convex and nonconvex applications.

2 Algorithms

In this paper we extend the classical L-BFGS algorithm (Algorithm 1) in various ways (see Section A for deeper motivation and derivations). Hence in this paper we will consider a few variants of L-BFGS algorithm, namely

- **L-BFGS** is the classical batch version of the L-BFGS algorithm from [33]. The full gradient is computed (i.e. using all n samples).
- **L-BFGS-H** is a stochastic version of L-BFGS but with a small change how y_k is computed. In this case we define y_k to be $y_k = B_k^{S_k}(w_{k+1} - w_k)$, where S_k is the stochastic batch picked at iteration k and $B_k^{S_k} \overset{\text{def}}{=} \frac{1}{|S_k|}\sum_{i\in S_k} \nabla^2 f_i(w_k)$. Note that one is not forming the $B_k^{S_k}$ matrix, but y_k us computing using a Hessian-vector multiplication.
- **L-BFGS-F** is a modification of **L-BFGS-H**, where instead of using $B_k^{S_k}$ we are using a Fisher information matrix (FIM) [25]. Note that when the predictor is linear, i.e. $h(w; x_i) = x_i^T w$, with the loss function L as either the cross-entropy or the least-squares, LBFGS-F is identical to LBFGS-H (GGN = FIM = Hessian). Similarly, this also applies to the batch version of the FIM.

Algorithm 1 L-BFGS

Initialize: x_0, integer $m > 0$
for $k = 1, 2, \ldots$ **do**
  Choose H_k^0
  Compute a direction $p_k = -H_k \nabla f(w_k)$ using two-loop recursion using $\{(s_i, y_i)\}_{i=k-m}^{k-1}$
  Choose a learning rate $\alpha_k > 0$
  Update the iterate: $w_{k+1} = w_k + \alpha_k p_k$
  Update the curvature pairs:
    $s_k = w_{k+1} - w_k, y_k = \nabla F(w_{k+1}) - \nabla F(w_k)$
  if $k \geq m$ **then**
    Replace the oldest pair (s_{k-m}, y_{k-m}) by (s_k, y_k)
  else
    Store the vector pair (s_k, y_k)
  end if
end for

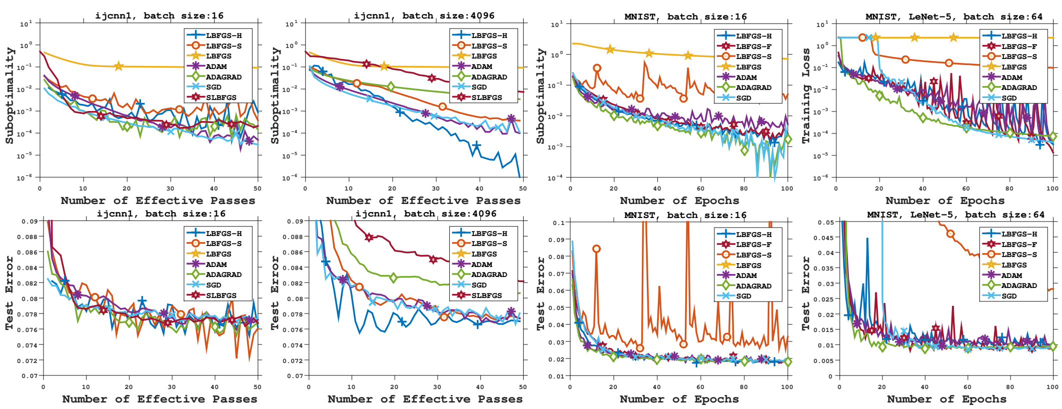

Figure 1: Comparisons of sub-optimality (top) and test errors (bottom) for different algorithms with batch sizes 16, 4096 on *ijcnn1* (logistic regression) and 16, 64 on *MNIST* with 1 hidden layer neural network and LeNet-5.

3 Numerical Experiments

In this section, we present numerical results to illustrate the properties and performance of our proposed algorithms (LBFGS-H and LBFGS-F) on both convex and nonconvex applications. For comparison, we show performance of popular stochastic gradient algorithms, namely, ADAM [18], ADAGRAD [13] and SGD (momentum SGD). Meanwhile, in Figure 1, we cover the performance of SLBFGS [29] in the convex examples since the convergence analysis of SLBFGS is only provided for the strongly convex setting. Besides, we include the performance for classical L-BFGS where $H_k^0 = \frac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}} I$, and a stochastic L-BFGS as LBFGS-S where we set $y_k = \nabla F^{S_k}(w_k) - \nabla F^{S_{k-1}}(w_{k-1})$.

In the convex setting, we test logistic regression problem on *ijcnn1* [1], where LBFGS-H is identical to LBFGS-F because of the linear predictor, so we omit the results for LBFGS-F (details available in Section A.3). On the other hand, we show performance of 1-hidden layer neural network (with 300 neurons) and LeNet-5 (a classical convolutional neural network) [21] on *MNIST* [2]. Across all the figures, each epoch refers to a full pass of the dataset, i.e., n component gradient evaluations. Weight decay regularizers have been added to all the experiments to enforce regularization.

Figure 1 shows sub-optimality $F(w_k) - F(w_*)$ (training loss $F(w_k)$ for the last column) and test errors of various methods with batch sizes 16 and 4096 on the logistic regression problem with *ijcnn1* for the first two columns. For a mini-batch size $b = 16$, LBFGS-H exhibits competitive performance with ADAM, SGD, ADAGRAD and SLBFGS while LBFGS-S seems highly unstable; however, with

[1] Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`; we use the test/train sets for training/testing, respectively, since the test set has more data samples.

[2] Available at `http://yann.lecun.com/exdb/mnist/`.

a larger bath size $b = 4096$, LBFGS-H maintains the competitive performance as for the small batch size while the performances of others suffer from the large batch size. On the nonconvex examples for the last two columns in the figure, similar results are presented with LBFGS-S to be extremely unstable and slow for small batch sizes [3].

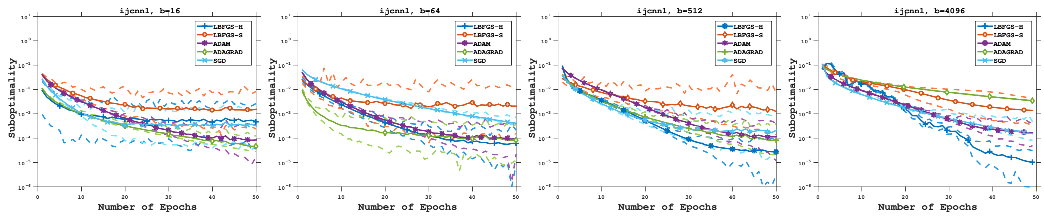

Figure 2: Comparisons of sub-optimality (top) and test errors (bottom) for different stochastic methods with batch sizes 16, 64, 512, 4096 on *ijcnn1*, convex, logistic regression.

To further show the robustness of LBFGS-H (LBFGS-F), we run each method with different batch sizes and 100 different random seeds on the logistic regression problem with dataset *ijcnn1* in Figure 2, and report the results. The dotted lines represent the best and worst performance of the corresponding algorithm and the solid line shows the average performance. Obviously, with large batch sizes, the performance of ADAM, ADAGRAD and SGD worsen while LBFGS-H behaves steadily fast and outperforms the others in sub-optimality. This also conveys that to achieve the same accuracy, fewer epochs are needed, leading to fewer communications for our framework when the batch size is large.

The ability to use a large batch size is of particular interest in a distributed environment since it allows us to scale to multiple GPUs without reducing the per-GPU workload and without sacrificing model accuracy. In order to illustrate the benefit of large batch sizes, we evaluate the stochastic gradient $\nabla F^S(w)$ on a neural network with different batch sizes ($b = 2^0, 2^1, \ldots, 2^{14}$) on a single GPU (Tesla K80), and compare the computational time against that of the pessimistic and utopian cases in Figure 3. Up to $b = 2^6$, the computational time stays almost constant; nevertheless, with a sufficiently large batch size ($b > 2^8$), the problem becomes computationally bounded and suffers from the computing resource limited by the single GPU, hence doubling batch size leads to doubling computational time. Therefore, the efficiency of our proposed algorithm shown in Algorithm 2 can benefit tremendously from a distributed environment.

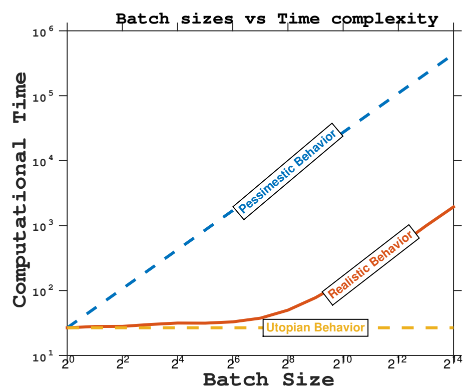

Figure 3: Batch size vs time complexity.

4 Conclusion

We developed a novel framework for the L-BFGS method with stochastic batches that is stable and efficient. Based on the framework, we proposed two variants – LBFGS-H and LBFGS-F, where the latter tries to employ Fisher information matrix replacing the Hessian to approximate the difference of gradients. LBFGS-F also admits a distributed implementation. We show that our framework converges linearly to a neighborhood of the optimal solution for the strongly convex setting while sublinearly for the nonconvex setting under standard assumptions. In addtion, sublinear convergence to the optimal solution has been validated for strongly convex objectives. We provide numerical experiments on both convex applications and nonconvex neural networks and compare our framework with the prevalent optimization algorithms.

[3]Performances of larger batch sizes for nonconvex objectives exhibit similar trends and are available in Appendix E.

Acknowledgements

Jie Liu was partially supported by the IBM PhD Fellowship. Martin Takáč was partially supported by the U.S. National Science Foundation, under award number NSF:CCF:1618717, NSF:CMMI:1663256 and NSF:CCF:1740796. We would like to thank Courtney Paquette for her valuable advice on the paper.

References

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1):4148–4187, 2017.

[2] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[3] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality and the smo algorithm. In *ICML*, 2004.

[4] Francis R. Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o(1/n). In *NIPS*, pages 773–781, 2013.

[5] Albert S Berahas, Majid Jahani, and Martin Takáč. Quasi-newton methods for deep learning: Forget the past, just sample. *arXiv preprint arXiv:1901.09997*, 2019.

[6] Albert S. Berahas, Jorge Nocedal, and Martin Takáč. A multi-batch L-BFGS method for machine learning. In *NIPS*, pages 1055–1063, 2016.

[7] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[8] Weizhu Chen, Zhenghao Wang, and Jingren Zhou. Large-scale L-BFGS using mapreduce. In *NIPS*, pages 1332–1340, 2014.

[9] David Roxbee Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society*, 20(2):215–242, 1958.

[10] Yu-Hong Dai. Convergence properties of the BFGS algoritm. *SIAM Journal on Optimization*, 13(3):693–701, 2002.

[11] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, pages 1646–1654, 2014.

[12] Aaron Defazio, Justin Domke, and Tibério Caetano. A faster, permutable incremental gradient method for big data problems. In *ICML*, pages 1125–1133, 2014.

[13] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2nd edition, 2009.

[15] Majid Jahani, Xi He, Chenxin Ma, Aryan Mokhtari, Dheevatsa Mudigere, Alejandro Ribeiro, and Martin Takáč. Efficient distributed hessian free algorithm for large-scale empirical risk minimization via accumulating sample strategy. *arXiv preprint arXiv:1810.11507*, 2018.

[16] Majid Jahani, Mohammadreza Nazari, Sergey Rusakov, Albert S Berahas, and Martin Takáč. Scaling up quasi-newton algorithms: Communication efficient distributed sr1. *arXiv preprint arXiv:1905.13096*, 2019.

[17] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pages 315–323, 2013.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Jakub Konečný, Jie Liu, Peter Richtárik, and Martin Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10:242–255, 2016.

[20] John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[21] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[22] Dong-Hui Li and Masao Fukushima. A modified BFGS method and its global convergence in nonconvex minimization. *Journal of Computational and Applied Mathematics*, 129(1-2):15–35, 2001.

[23] Julien Mairal. Optimization with first-order surrogate functions. In *ICML*, pages 783–791, 2013.

[24] James Martens. Deep learning via hessian-free optimization. In *ICML*, pages 735–742, 2010.

[25] James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

[26] James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, pages 2408–2417, 2015.

[27] Walter F Mascarenhas. The BFGS method with exact line searches fails for non-convex objective functions. *Mathematical Programming*, 99(1):49–61, 2004.

[28] A. Meenakshi and C. Rajian. On a product of positive semidefinite matrices. *Linear Algebra and its Applications*, 295(1):3–6, 1999.

[29] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic l-bfgs algorithm. In *Artificial Intelligence and Statistics*, pages 249–258, 2016.

[30] Lam Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *ICML*, pages 2613–2621, 2017.

[31] Lam Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv:1705.07261*, 2017.

[32] Lam Nguyen, Phuong Ha Nguyen, Marten van Dijk, Peter Richtárik, Katya Scheinberg, and Martin Takáč. SGD and Hogwild! convergence without the bounded gradients assumption. In *ICML*, 2018.

[33] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[34] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *ICLR*, 2014.

[35] Michael JD Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. *Nonlinear programming*, 9(1):53–72, 1976.

[36] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[37] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, pages 1–30, 2016.

[38] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

[39] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, 2013.

[40] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.