INEFFICIENCY OF STOCHASTIC GRADIENT DESCENT WITH LARGER MINI-BATCHES (AND MORE LEARNERS)

Onkar Bhardwaj & Guojing Cong
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{obhardw,gcong}@us.ibm.com

ABSTRACT

Stochastic Gradient Descent (SGD) and its variants are the most important optimization algorithms used in large scale machine learning. Mini-batch version of stochastic gradient is often used in practice for taking advantage of hardware parallelism. In this work, we analyze the effect of mini-batch size over SGD convergence for the case of general non-convex objective functions. Building on the past analyses, we justify mathematically that there can often be a large difference between the convergence guarantees provided by small and large mini-batches (given each instance processes equal number of training samples), while providing experimental evidence for the same. Going further to distributed settings, we show that an analogous effect holds with popular Asynchronous Gradient Descent (ASGD): there can be a large difference between convergence guarantees with increasing number of learners given that the cumulative number of training samples processed remains the same. Thus there is an inherent (and similar) inefficiency introduced in the convergence behavior when we attempt to take advantage of parallelism, either by increasing mini-batch size or by increase the number of learners.

1 INTRODUCTION

Stochastic gradient descent (SGD) and its parallel variants form the backbone of most popular deep learning applications. Consequently, there has been a significant interest in investigating their convergence properties. SGD has been shown to satisfy an asymptotic convergence rate of $O(1/S)$ for convex objective functions [Nemirovski et al. (2009)] and an asymptotic convergence rate of $O(1/\sqrt{S})$ for general non-convex objective functions with mini-batch size 1 in [Ghadimi & Lan (2013)] or with arbitrary mini-batch sizes in [Dekel et al. (2012)].

Although SGD converges *asymptotically* with the same rate irrespective of mini-batch size, it has been reported that for large mini-batch sizes, often it is slower to converge - for example, see Wilson & Martinez (2003) for the detailed graphical illustrations therein for the effect of increasing batch-size or see Bottou (2010) for the comments on the tradeoffs of mini-batching. In this work, we are interested in using theoretical analysis for justifying such practical observations or comments. In particular, we show the following:

- We consider general non-convex objective functions and show that prior to reaching asymptotic regime, SGD convergence could get much slower (inferred from the difference in the convergence rate guarantees from Theorem 2) with using higher mini-batch size, assuming a constant learning rate. Here, to evaluate the convergence rate guarantee we use the measure of average gradient norm since we are considering general non-convex objectives. As a consequence of slower convergence, the number of training samples required to attain a certain convergence guarantee (in terms of average gradient norm) increases as the mini-batch size increases. We build the analysis based on the framework in Ghadimi & Lan (2013).
- Further, we investigate Asynchronous Stochastic Gradient Descent (ASGD) which is one of the most popular asynchronous variants of SGD [Dean et al. (2012); Li et al. (2014a);

Chilimbi et al. (2014)]. Recently, Lian et al. (2015) extended the results of SGD convergence to ASGD and showed that it converges *asymptotically* with a convergence rate of $O(1/\sqrt{S})$. In our analysis we show that prior to the asymptotic regime, with using higher number of learners ASGD convergence could get much slower in terms of average gradient norm attained after cumulatively processing a fixed number of training samples (this slow-down is inferred from the difference in the convergence guarantees from Theorem 4).

- This suggests that there is an inherent limit on harnessing parallelism with SGD either by increasing mini-batch size or increasing the number of learners (even when we do not take into account overheads such as communication cost). The difference in convergence behavior caused by increasing mini-batch size for SGD and by increasing the number of learners with ASGD was found to be similar (See Theorem 2, Theorem 4 and the discussion at the end of Section 4).

For rest of the paper, we use the following notation: Let $F(x, z^i)$ denote a non-convex function of a parameter vector x and a training sample z^i selected from the training set $\{z^1, z^2, \ldots, z^n\}$. Our aim is to find a parameter vector that minimizes the objective function $f(x) = \mathbb{E}_z F(x, z)$. Towards this, we use mini-batch SGD, where in k^{th} iteration, we select a random mini-batch $z_k = \{z_k^1, z_k^2, \ldots, z_k^M\}$ of size M and perform the following update:

$$x_{k+1} = x_k - \gamma \cdot G(x, z_k) = x_k - \gamma \cdot \sum_{i=1}^{M} G(x, z_k^i) \tag{1}$$

In the above equation, γ denotes the learning rate and we have $G(x, z_k) = \sum_{i=1}^{M} G(x, z_k^i)$ where $G(x, z_k^i)$ denotes the gradient of the objective function $f(x)$ with respect to a training sample $z_k^i \in z_k$. We define $D_f := f(x_1) - f(x^*)$ where x_1 is the initial parameter vector and x^* is a local optima towards which SGD proceeds. We also denote by S the total number of training samples to be processed.

Additionally, we make the following standard assumptions, see e.g., [Lian et al. (2015), Ghadimi & Lan (2013)]:

A.1 **Unbiased estimator:** We assume that the expectation of $G(x, z)$ equals to the true value of the gradient, i.e., $\mathbb{E}_z G(x, z) = \nabla f(x)\ \forall x$.

A.2 **Bounded variance:** We assume that there exists a constant σ^2 such that $\mathbb{E}_z(\|G(x, z) - \nabla f(x)\|^2) \leq \sigma^2\ \forall x$.

A.3 **Lipschitzian Gradient:** We assume $f(x)$ to satisfy Lipschitzian smoothness, i.e., there exists a constant L such that $\|\nabla f(x) - \nabla f(y)\| \leq L \cdot \|x - y\|\ \forall x, y$.

The paper is organized as follows: Section 2 discussed some related work. We follow it up by analyzing impact of mini-batch size on SGD in Section 3. Later we extend the analysis to ASGD in Section 4. We provide experimental evidence regarding our analysis in Section 5 and conclude by discussing future directions in Section 6.

2 Related Work

In recent years, there have been several works analyzing convergence properties of SGD and its variants. In Nemirovski et al. (2009), SGD has been shown to have a convergence rate of $O(1/S)$ for convex objective functions, where S is the number of samples seen, and this rate is in terms of distance of objective function from the optimal value. When the objective cost functions are non-convex, as is the case with most deep learning applications, the rate of convergence of SGD in terms of average gradient norm has been shown to be $O(1/\sqrt{S})$ asymptotically by Ghadimi & Lan (2013). The results in Dekel et al. (2012) can be interpreted as showing the applicability of this convergence rate also for the mini-batches of size in M, where now S takes form of MK with K being the number of mini-batches processed.

Among the distributed variants of SGD, ASGD has been the most popular variant Dean et al. (2012); Li et al. (2014a); Chilimbi et al. (2014). Practically it has been observed that ASGD is often slower to converge with increasing number of learners [Seide et al. (2014); Chan & Lane (2014); Dean et al.

(2012)]. Although these works did not ignore communication overhead, in Section 4 we investigate the inherent inefficiency in ASGD without communication overhead costs. In Lian et al. (2015), it was proved that in the asymptotic regime the convergence rate of $O(1/\sqrt{S})$ can be extended to ASGD when the "age" of updates was bounded by the number of learners.

There exist several other sequential and distributed variants of SGD. For example, SGD with variance reduction techniques to mitigate the effects of gradient variance have been discussed in [Johnson & Zhang (2013); Xiao & Zhang (2014)]. SGD with co-ordinate descent/ascent and its distributed variants has been studied in [Hsieh et al. (2008); Richtárik & Takác (2013); Fercoq et al. (2014); Konečnỳ et al. (2014); Qu & Richtárik (2014); Liu et al. (2015); Jaggi et al. (2014); Nesterov (2012)]. The convergence properties of asynchronous stochastic coordinate descent have been analyzed in Liu & Wright (2015). More recently, the authors in Meng et al. (2016) have studied combining variance reduction techniques, randomized block co-ordinate descent [Richtárik & Takáč (2014)] and Nesterov acceleration methods [Nesterov (2013)] and analyzed its theoretical properties.

There have been several recent works which attempt to mitigate the effect of degrading convergence for large mini-batches (e.g., see [Li et al. (2014b)] where in each mini-batch a regularized objective function is optimized to compute updated parameter vector), or there are works which attempt to select mini-batch dynamically for better performance, for example see [Byrd et al. (2012); Tan et al. (2016); De et al. (2016)], or there have been works which attempt to improve SGD performance by intelligent selection of training samples, e.g., [Needell et al. (2014); Bouchard et al. (2015)].

3 The Impact of Minibatch Size on SGD

In this section, we build on the SGD convergence analysis in Ghadimi & Lan (2013). In particular, we consider the convergence guarantees from Theorem 2.1 in Ghadimi & Lan (2013) restricted to constant learning rate. However, we first modify their analysis to allow mini-batches of arbitrary sizes. Building on, we show that SGD with smaller mini-batches can have better convergence guarantees than with using larger mini-batches. As a consequence, we observe that for larger mini-batches, a larger number of samples is needed for the convergence rate to fall below a certain threshold.

Lemma 1. *With mini-batches of size M and a constant learning rate γ, after K iterations of SGD, we have*

$$\frac{\sum_{k=1}^{K}\mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \leq \frac{1}{\gamma - \frac{LM\gamma^2}{2}} \cdot \left(\frac{D_f}{S} + \frac{L\sigma^2\gamma^2}{2}\right) \tag{2}$$

Proof outline. Using the property of Lipschitzian gradient, we get

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &= f(x_k) - \gamma_k \cdot \left\langle \nabla f(x_k), \sum_{i=1}^{M} G(x_k, z_k^i)\right\rangle + \frac{\gamma_k^2 L}{2}\left\|\sum_{i=1}^{M} G(x_k, z_k^i)\right\|^2 \end{aligned} \tag{3}$$

Let us define $\delta_k^i = G(x_k, z_k^i) - \nabla f(x_k)$ and $\delta_k = \sum_{i=1}^{M}\delta_k^i$. Using it, the above equation can be rewritten as

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq -\gamma_k \cdot \langle \nabla f(x_k), \delta_k + M \cdot \nabla f(x_k)\rangle + \frac{\gamma_k^2 L}{2}\|\delta_k + M \cdot \nabla f(x_k)\|^2 \\ \Rightarrow f(x_{k+1}) - f(x_k) &\leq -\gamma_k \cdot \langle \nabla f(x_k), \delta_k\rangle - M\gamma_k \cdot \|\nabla f(x_k)\|^2 \\ &\quad + \frac{\gamma_k^2 L}{2}\left(\|\delta_k\|^2 + 2M\langle \delta_k, \nabla f(x_k)\rangle + M^2\|\nabla f(x_k)\|^2\right) \end{aligned}$$

Rest of the proof involves adding such inequalities over first $K-1$ updates and bounding the $\|\delta_k\|^2$ using assumption A.2 and $\langle \delta_k, \nabla f(x_k)\rangle$ using assumption A.1 from Section 1. Finally, we use $f(x_K) - f(x^*) \leq D_f$ and rearrange the terms to get the desired bound. □

In the following theorem, we justify that given a fixed number of samples S to be processed, SGD with smaller mini-batches SGD can have better convergence guarantees than with using larger mini-batches. Note that α used in the theorem statement is a measure of (the square root of) the number of training samples processed.

Theorem 2. *Let $\alpha \coloneqq \sqrt{\frac{S\sigma^2}{LD_f}}$. Let $4M_l \leq \alpha \leq M_h/4$. Then the convergence guarantee for SGD for mini-batch size M_h after processing S training samples can be worse than the convergence guarantee for mini-batch size M_l by a factor of $2M_h/(\alpha\sqrt{2} + M_l)$.*

Proof outline. For a fixed number S of training samples to be processed, we now minimize the right hand side of Equation 2 to find the best convergence guarantee supported by Lemma 1. Let $\gamma = c \cdot \sqrt{D_f/(SL\sigma^2)}$, where c is a scalar multiplier. Substituting it in Equation 2 and after some algebraic manipulations, we get

$$\frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \quad \leq \quad \left(\frac{\frac{1}{c} + \frac{c}{2}}{1 - \frac{cM}{2\alpha}}\right) \cdot \sqrt{\frac{D_f L\sigma^2}{S}} \tag{4}$$

By applying simple calculus, it can be shown that the value c^* of c which minimizes right hand side of the above equation is given by

$$c^* \quad = \quad \frac{M}{\alpha} \cdot \left(-1 + \sqrt{1 + \frac{2\alpha^2}{M^2}}\right) \tag{5}$$

In appendix, we show that with $M = M_l \leq \alpha/4$, we have $c^* \approx \sqrt{2} - M_l/\alpha$ and consequently the coefficient of $\sqrt{D_f L\sigma^2/S}$ from Equation 4 evaluates to approximately $\sqrt{2} + M_l/\alpha$. Whereas for $M = M_h \geq 4\alpha$, we show that in the appendix using that $c^* = \alpha/M_h$ and the coefficient of $\sqrt{D_f L\sigma^2/S}$ from Equation 4 evaluates to approximately $2M_h/\alpha$. Combining these observations, we get that the convergence guarantees for M_l and M_h can differ by a factor of $2M_h/(\alpha\sqrt{2} + M_l)$ after processing S training samples. See the appendix for complete proof. □

Note that while it could be possible to show that in general smaller mini-batches converge faster than larger mini-batches in theory, we used $4M_l \leq \alpha \leq M_h/4$ in Theorem 2 for the sake of simplicity. Also, although we theoretically justified faster convergence smaller mini-batches in Theorem 2, the exact factors by which bigger mini-batches can be worse can vary in practice. Here our purpose is to give theoretical support to the practical observations of smaller mini-batches being faster.

Theorem 3. *The number of samples needs to be processed in order for SGD achieve the same convergence guarantee increases as the mini-batch size increases.*

Proof. For the same values of γ and S, the value of the bound from Equation 2 becomes worse (i.e., increases) as M increases. This is because for a fixed γ the quantity $\gamma - LM\gamma^2/2$ must decrease as M increases. Consequently for given S, the best value of convergence guarantee (i.e., smallest average gradient norm) attained by varying γ must become worse (i.e., higher) as M increases. Thus in order to reach the same convergence guarantee, SGD must process more training samples with increasing mini-batch size. □

Now we will proceed to the next section, where we will show that with ASGD (which is one of the most popular distributed variants of SGD) increasing number of learners can lead to slower convergence given a fixed total number of samples to be processed, and the effect is similar to that of increasing mini-batch size for SGD.

4 Asynchronous Stochastic Gradient Descent

ASGD typically has a parameter server maintaining the parameters (i.e., the weights in the neural network) and multiple learners. Each learner asynchronously repeats the following:

- **Pull:** Get the parameters from the server.

- **Compute:** Compute the gradient with respect to randomly selected mini-batch (i.e., a certain number of samples from the dataset).
- **Push and update:** Communicate the gradient to the server. Server then updates the parameters by subtracting this newly communicated gradient multiplied by the learning rate.

We assume that the update performed by the server is atomic, i.e., the server does not send or receive parameters while it updates the parameters. Now we express k^{th} update step of the ASGD algorithm in terms of our notation. Note that for k^{th} update, the partial gradient computed by a learner can be with respect to an older parameter vector. This is because while computing the partial gradient, the parameter vector could have been updated because of the partial gradients sent in by other learners. Let $x_{k-\tau}$ be the parameter vector used by a learner to compute the partial gradient to be used in k^{th} update. Then the equation for k^{th} update of ASGD becomes:

$$x_{k+1} = x_k - \gamma \cdot G(x_{k-\tau}, z) \tag{6}$$

Lian et al. (2015) showed that when the age of the updates τ is bounded by the number of learners N, then ASGD asymptotically converges with a rate of $O(1/\sqrt{S})$ where S is the cumulative number of training samples processed. From Theorem 1 in Lian et al. (2015), the convergence rate guarantee (expressed in the terms of average gradient norm) for ASGD with N learners after processing K updates becomes

$$\frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \leq \frac{2D_f}{MK\gamma} + \sigma^2 L\gamma + 2\sigma^2 L^2 MN\gamma^2 \tag{7}$$

$$\text{s.t.} \quad LM\gamma + 2L^2M^2N^2\gamma^2 \leq 1 \tag{8}$$

The terms independent of the number of updates K in Equation 7 indicate that with a constant learning rate, there is a limit on how close the algorithm can reach to the optimum without lowering the learning rate. Although asymptotically, it can be shown that Equation 7-8 lead to $O(1/\sqrt{S})$ convergence (see Lian et al. (2015)), we now investigate the convergence behavior prior to such a regime. We have the following theorem about the effect of increasing the number of learners on ASGD convergence guarantee:

Theorem 4. *Let $N > 1$ be the number of learners and let $\alpha = \sqrt{\frac{K\sigma^2}{MLD_f}} \leq N$, then the optimal ASGD convergence rate guarantee for 1 learner and N learners can differ by a factor of approximately $\frac{N}{\alpha}$.*

The proof of above theorem is in the same spirit as that of Theorem 2 and can be found in the appendix. Note that without asynchronous nature of ASGD, the analysis for *synchronous* distributed SGD would be the same as the analysis for SGD from Section 3. This is because synchronous SGD, where each of the N learners compute the gradient for a random mini-batch of size M, equivalently represents SGD with mini-batch size MN. The asynchronous nature of ASGD introduces extra factors to be taken into account such as the "age" of the updates (i.e., the situation where the gradient returned by a learner may have been computed by an older parameter vector).

Theorem 5. *For a constant mini-batch size M, the number of samples needs to be processed in order to achieve the same convergence guarantee increases as the number of learners increases.*

Proof outline. The range of γ permissible by Equation 8 becomes smaller as N increases. Rest of the proof combines the observations that the minimum attained by the convergence guarantee by Equation 7 must become worse if the range of γ decreases and N increases. For complete proof, please see the appendix. □

Discussion: From Theorem 2, for sequential SGD, there could be a difference of $2M_h/(\alpha\sqrt{2}+M_l)$ between the convergence guarantee of mini-batch sizes M_l, M_h with $4M_l \leq \alpha \leq M_h/4$. Assuming M_l to be far smaller than α, this factor becomes approximately $\sqrt{2}M_h/\alpha$. This is comparable with the difference N/α between ASGD convergence guarantee of 1 learner and N learners. Although exact numerical multipliers may differ depending on the tightness of the original convergence bound, it points to the similarities between slow-down caused by bigger mini-batch sizes with SGD and larger number of learners with ASGD. At a high level, ASGD with higher number of learners (with

Input: mini-batch of M RGB images
$\downarrow$
Convolution: (nfeat, nkern, height, width) $= (3, 64, 5, 5)$
Rectified Linear Unit (ReLU)
Max-Pooling: (height, width) $= (2, 2)$
Dropout: prob. $= 0.5$
$\downarrow$
Convolution: (nfeat, nkern, height, width) $= (64, 128, 3, 3)$
Rectified Linear Unit (ReLU)
Max-Pooling: (height, width) $= (2, 2)$
Dropout: prob. $= 0.5$
$\downarrow$
Convolution: (nfeat, nkern, height, width) $= (128, 256, 3, 3)$
Rectified Linear Unit (ReLU)
Max-Pooling: (height, width) $= (2, 2)$
Dropout: prob. $= 0.5$
$\downarrow$
Convolution: (nfeat, nkern, height, width) $= (256, 128, 2, 2)$
Rectified Linear Unit (ReLU)
Max-Pooling: (height, width) $= (2, 2)$
Dropout: prob. $= 0.5$
$\downarrow$
Fully connected layer: 128×10
$\downarrow$
Cross-entropy error

Table 1: Convolutional Neural Network for CIFAR10. For convolutional layers nfeat denotes the number of input feature maps and nkern denotes the number of kernels.

bounded age/staleness of updates) can be thought of SGD with some *effective* mini-batch size. This effective mini-batch size could be dependent on the number of learners as well as the age/staleness of updates.

5 EXPERIMENTS

Experiment setup: We carry our experiments with *CIFAR-10* dataset [cif (accessed January 11, 2016a)] which contains $50,000$ training samples and $10,000$ test samples, each associated with 1 out of 10 possible labels For the *CIFAR-10* dataset, our aim is to predict the correct label the input images. For our experiments, we train convolutional neural network shown in Table 1, which is taken from [cif (accessed January 11, 2016b)]. It is a fairly standard convolutional network design consisting of a series of convolutional layers interspersed by max-pooling layers. The convolutional layers outputs are filtered with rectified linear unit before applying max-pooling. Additionally, it also uses Dropout layers which act as regularization [Srivastava et al. (2014)]. At the end, it has a fully connected layer with 10 outputs (equal to the number of labels). The number of parameters to be learned for *CIFAR-10* network is ≈ 0.5 million.

We use cross-entropy between the input labels and the predicted labels in the final output layer, i.e., $F(x, z)$ (see Section 1 for the notation) is the cross-entropy error and $f(x)$ is the average cross-entropy error over all training samples. The implementation of neural network was done using Torch. The target platform for our experiments is a Magma system with 16 GPUs connected to an IBM Power8 host. In our ASGD *Downpour* implementation, the learners are run on the GPUs, while the parameter server is run on the CPU.

We carried our experiments for 100 epochs, here by an epoch we mean a complete pass of the training data. We chose the learning rate to be 0.01 as it was seen to be performing well at the end

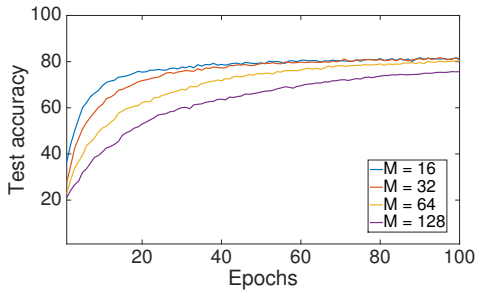

Figure 1: SGD experiments with *CIFAR-10*: convergence becomes slower as mini-batch M size increases.

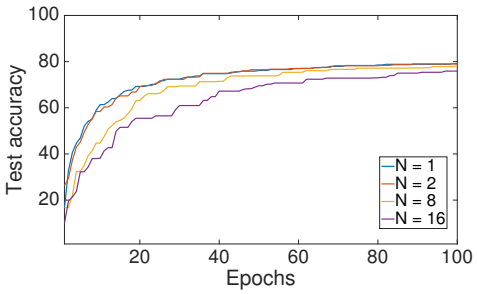

Figure 2: ASGD convergence for *CIFAR-10*: convergence becomes slower as the number of learners N increases.

of our experiments with respect to test accuracy (i.e., classification accuracy on the test data). For ASGD part of experiments, we randomly partitioned the training data between all N learners in the beginning of each epoch. At the end of each epoch we measured the test accuracy. See Figure 1 for the results of our SGD experiments. We can observe that as the mini-batch size increases, the test error converges slower with respect to the number of epochs. See Figure 2 for the result of our experiments with ASGD experiments. Again, we can observe that as the number of learners increases, the convergence of test error becomes slower.

These observations agree with our justifications from Section 3 and 4. Moreover, they show that there are similarities between the slow-down caused by increasing mini-batch size with SGD and increasing the number of learners with ASGD. Thus, exploiting parallelism, either by increasing mini-batch size or by increasing the number of learners introduces an inherent inefficiency in the convergence behavior, even after disregarding other overheads such as communication time when we increase the number of learners.

6 Conclusion and Future Directions

In this paper, we theoretically justified faster convergence (in terms of average gradient norm attained after processing a fixed number of samples) of SGD with small mini-batches or that of ASGD with smaller number of learners. This indicates that there is an inherent inefficiency in the speed-up obtained with parallelizing gradient descent methods by taking advantage of hardware. It would be interesting to see if such a conclusion holds for more advanced update methods than vanilla SGD, for example methods using momentum and its variants.

REFERENCES

Cifar10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html, accessed January 11, 2016a.

Cifar10 model. https://github.com/eladhoffer/ConvNet-torch/blob/master/Models/Model.lua, accessed January 11, 2016b.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Accelerating stochastic gradient descent via online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

William Chan and Ian Lane. Distributed asynchronous optimization of convolutional neural networks. In *INTERSPEECH*, pp. 1073–1077, 2014.

Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 571–582, 2014.

Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Big batch sgd: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

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*, pp. 1223–1231, 2012.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for non-strongly convex losses. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2014.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pp. 408–415. ACM, 2008.

Martin Jaggi, Virginia Smith, Martin Takac, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pp. 3068–3076, 2014.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Jakub Konečnỳ, Zheng Qu, and Peter Richtárik. Semi-stochastic coordinate descent. *arXiv preprint arXiv:1412.6293*, 2014.

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 *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 583–598, 2014a.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014b.

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

Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

Ji Liu, Stephen J Wright, Christopher Re, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *Journal of Machine Learning Research*, 16(285-322):1–5, 2015.

Qi Meng, Wei Chen, Jingcheng Yu, Taifeng Wang, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous accelerated stochastic gradient descent. In *Proceedings of the 25th international joint conference on Artificial Intelligence*, 2016.

Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pp. 1017–1025, 2014.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *arXiv preprint arXiv:1412.8060*, 2014.

Peter Richtárik and Martin Takác. Distributed coordinate descent method for learning with big data. 2013.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. On parallelizability of stochastic gradient descent for speech dnns. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 235–239. IEEE, 2014.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent. *arXiv preprint arXiv:1605.04131*, 2016.

D Randall Wilson and Tony R Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

A APPENDIX

Lemma 1. *With mini-batches of size M and a constant learning rate γ, after K iterations of SGD, we have*

$$\frac{\sum_{k=1}^{K}\mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \leq \frac{1}{\gamma - \frac{LM\gamma^2}{2}} \cdot \left(\frac{D_f}{S} + \frac{L\sigma^2\gamma^2}{2}\right) \tag{2}$$

Proof. Using the property of Lipschitzian gradient, we get

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &= f(x_k) - \gamma_k \cdot \left\langle \nabla f(x_k), \sum_{i=1}^{M} G(x_k, z_k^i)\right\rangle + \frac{\gamma_k^2 L}{2}\left\|\sum_{i=1}^{M} G(x_k, z_k^i)\right\|^2 \end{aligned} \tag{9}$$

Let us define $\delta_k^i = G(x_k, z_k^i) - \nabla f(x_k)$ and $\delta_k = \sum_{i=1}^{M}\delta_k^i$. Using it, the above equation can be rewritten as

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq -\gamma_k \cdot \langle \nabla f(x_k), \delta_k + M \cdot \nabla f(x_k)\rangle + \frac{\gamma_k^2 L}{2}\|\delta_k + M \cdot \nabla f(x_k)\|^2 \\ \Rightarrow f(x_{k+1}) - f(x_k) &\leq -\gamma_k \cdot \langle \nabla f(x_k), \delta_k\rangle - M\gamma_k \cdot \|\nabla f(x_k)\|^2 \\ &\quad + \frac{\gamma_k^2 L}{2}\left(\|\delta_k\|^2 + 2M\langle \delta_k, \nabla f(x_k)\rangle + M^2\|\nabla f(x_k)\|^2\right) \end{aligned}$$

Generating such inequalities over K mini-batches and adding them, we get

$$\begin{aligned} -D_f &\leq f(x_{K+1}) - f(x_1) \\ &\leq \sum_{k=1}^{K}\Big(-\gamma_k \cdot \langle \nabla f(x_k), \delta_k\rangle - M\gamma_k\|\nabla f(x_k)\|^2 \\ &\quad + \frac{\gamma_k^2 L}{2}\left(\|\delta_k\|^2 + 2M\langle \delta_k, \nabla f(x_k)\rangle + M^2\|\nabla f(x_k)\|^2\right)\Big) \end{aligned}$$

Simple rearrangement of terms gives us

$$\begin{aligned} &\sum_{k=1}^{K}\left(M\gamma_k - \frac{\gamma_k^2 LM^2}{2}\right)\cdot\|\nabla f(x_k)\|^2 \\ &\leq D_f + \sum_{k=1}^{K}\left(-\gamma_k \cdot \langle \nabla f(x_k), \delta_k\rangle + \frac{\gamma_k^2 L}{2}\left(\|\delta_k\|^2 + 2M \cdot \langle \delta_k, \nabla f(x_k)\rangle\right)\right) \end{aligned} \tag{10}$$

Now we observe that $\mathbb{E}(\|\delta_k\|^2) = \mathbb{E}(\|\sum_{i=1}^{M}\delta_k^i\|^2) \leq M \cdot \sum_{i=1}^{M}\mathbb{E}(\|\delta_k^i\|^2) \leq M\sigma^2$, using Assumption A.2. From the assumption of the stochastic gradient being unbiased estimator of the true gradient (Assumption A.1), we also know that $\mathbb{E}(\langle \delta_k, \nabla f(x_k)\rangle) = \sum_{i=1}^{M}\mathbb{E}(\langle \delta_k^i, \nabla f(x_k)\rangle) = \left\langle \sum_{i=1}^{M}\mathbb{E}(\delta_k^i), \nabla f(x_k)\right\rangle = 0$. Taking the expectation of both sides in Equation 10 with respect to randomly selected samples and using these observations we get the following equation:

$$\sum_{k=1}^{K}\left(M\gamma_k - \frac{LM^2\gamma_k^2}{2}\right)\cdot\mathbb{E}(\|\nabla f(x_k)\|^2) \leq D_f + \frac{LM\sigma^2}{2}\sum_{k=1}^{K}\gamma_k^2$$

The above equation is equivalent to Equation 2.11 from Ghadimi & Lan (2013) modified to allow arbitrary mini-batch sizes. Restricting ourselves to allow constant learning rate and after simplifying the above equation, we get

$$\frac{\sum_{k=1}^{K}\mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \leq \frac{1}{\gamma - \frac{LM\gamma^2}{2}} \cdot \left(\frac{D_f}{S} + \frac{L\sigma^2\gamma^2}{2}\right) \tag{11}$$

□

Theorem 2. *Let $\alpha \coloneqq \sqrt{\frac{S\sigma^2}{LD_f}}$. Let $4M_l \leq \alpha \leq M_h/4$. Then the convergence guarantee for SGD for mini-batch size M_h after processing S training samples can be worse than the convergence guarantee for mini-batch size M_l by a factor of $2M_h/(\alpha\sqrt{2} + M_l)$.*

Proof outline. For a fixed number S of training samples to be processed, we now minimize the right hand side of Equation 2 to find the best convergence guarantee supported by Lemma 1. Let $\gamma = c \cdot \sqrt{D_f/(SL\sigma^2)}$, where c is a scalar multiplier. Substituting it in Equation 2 and after some algebraic manipulations, we get

$$\frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \quad \leq \quad \left(\frac{\frac{1}{c} + \frac{c}{2}}{1 - \frac{cM}{2\alpha}}\right) \cdot \sqrt{\frac{D_f L\sigma^2}{S}} \tag{12}$$

By applying simple calculus, it can be shown that the value c^* of c which minimizes right hand side of the above equation is given by

$$c^* \quad = \quad \frac{M}{\alpha} \cdot \left(-1 + \sqrt{1 + \frac{2\alpha^2}{M^2}}\right) \tag{13}$$

- Consider the case of $M = M_l \leq \alpha/4$. Denote c^* by c_l^* for this case. With $M = M_l \leq \alpha/4$, we have $\sqrt{1 + 2\alpha^2/M_l^2} \approx \sqrt{2}\alpha/M_l$. Thus we get $c_l^* \approx \sqrt{2} - M_l/\alpha$. Further, we can write the following using M_l is little compared to α and other simple known approximations:

$$\frac{1}{1 - \frac{c_l^* M_l}{2\alpha}} \quad = \quad \frac{1}{1 - (\sqrt{2} - \frac{M_l}{\alpha})\frac{M_l}{2\alpha}} \approx \frac{1}{1 - \frac{M_l}{\sqrt{2}\alpha}} \approx 1 + \frac{M_l}{\sqrt{2}\alpha} \tag{14}$$

$$\frac{1}{c_l^*} \quad = \quad \frac{1}{\sqrt{2} - \frac{M_l}{\alpha}} = \frac{1}{\sqrt{2}} \cdot \frac{1}{1 - \frac{M_l}{\sqrt{2}\alpha}} \approx \frac{1}{\sqrt{2}} \cdot \left(1 + \frac{M_l}{\sqrt{2}\alpha}\right) \approx \frac{1}{\sqrt{2}} + \frac{M_l}{2\alpha} \tag{15}$$

$$\frac{c_l^*}{2} \quad = \quad \frac{1}{\sqrt{2}} - \frac{M_l}{2\alpha} \tag{16}$$

  Using Equation 14, 15, 16, we get that the coefficient of $\sqrt{D_f L\sigma^2/S}$ from Equation 12 evaluates to approximately $\sqrt{2} + M_l/\alpha$.

- Consider $M_h \geq 4\alpha$. Denote c^* by c_h^* for this case. With $M = M_l \leq \alpha/4$, we have $\sqrt{1 + 2\alpha^2/M_h^2} \approx 1 + \alpha^2/M_h^2$, using the approximation $\sqrt{1+\epsilon} \approx 1 + \epsilon/2$ for small ϵ. Thus $c_h^* \approx \alpha/M_h$. Since α is much smaller than M_h, we can approximate $1/c_h^* + c_h^*/2 \approx 1/c_h^* \approx M_h/\alpha$ and we also have $1 - c_h^* M/(2\alpha) = 1/2$. Thus the coefficient of $\sqrt{D_f L\sigma^2/S}$ from Equation 12 evaluates to approximately $2M_h/\alpha$.

Combining the above observations, we get that the convergence guarantees for M_l and M_h can differ by a factor of $2M_h/(\alpha\sqrt{2} + M_l)$ after processing S training samples. □

Theorem 4. *Let $N > 1$ be the number of learners and let $\alpha = \sqrt{\frac{K\sigma^2}{MLD_f}} \leq N$, then the optimal ASGD convergence rate guarantee for 1 learner and N learners can differ by a factor of approximately $\frac{N}{\alpha}$.*

Proof. We have $\gamma = c \cdot \sqrt{D_f/(MKL\sigma^2)} = \frac{c}{\alpha ML}$ from the definition of α. Substituting this in Equation 7, we get

$$\begin{aligned}\frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \quad &\leq \quad \frac{2D_f}{MKc \cdot \sqrt{\frac{D_f}{MKL\sigma^2}}} + \sigma^2 L \cdot c \cdot \sqrt{\frac{D_f}{MKL\sigma^2}} \\ &\quad + 2\sigma^2 L^2 MN \cdot \frac{c}{\alpha ML} \cdot c \cdot \sqrt{\frac{D_f}{MKL\sigma^2}} \\ &= \quad \left(\frac{2}{c} + c + \frac{2Nc^2}{\alpha}\right) \cdot \sqrt{\frac{D_f L\sigma^2}{MK}}\end{aligned} \tag{17}$$

From the definition of α, we have $K = \alpha^2 MLD_f/\sigma^2$. Using it in the above equation, we get

$$\frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \quad \leq \quad \left\{ \left(\frac{2}{c} + c + \frac{2Nc^2}{\alpha} \right) \cdot \frac{1}{\alpha} \right\} \cdot \frac{\sigma^2}{M} \tag{18}$$

Similarly, given $\gamma = c \cdot \sqrt{\frac{D_f}{MKL\sigma^2}} = \frac{c}{\alpha ML}$, the condition in Equation 8 can also be expressed as:

$$\frac{c}{\alpha} + \frac{2N^2c^2}{\alpha^2} \leq 1 \;\; \Rightarrow \;\; 2N^2c^2 + \alpha c - \alpha^2 \leq 0$$

Since learning rate (and hence c) is always positive, the above equation gives us

$$0 \leq c \leq \frac{\alpha}{4N^2} \cdot (-1 + \sqrt{1 + 8N^2})$$

Thus finding the optimal learning rate (within the regime of Equation 7 and 8) is equivalent to solving the following:

$$\text{minimize} \qquad \left(\frac{2}{c} + c + \frac{2Nc^2}{\alpha} \right) \cdot \frac{1}{\alpha} \cdot \frac{\sigma^2}{M} \tag{19}$$

$$\text{s.t.} \qquad 0 \;\leq\; c \;\leq\; \frac{\alpha}{4N^2} \cdot (-1 + \sqrt{1 + 8N^2}) \tag{20}$$

Now, by means of solving the above optimization, we will investigate how much the convergence guarantee can differ as the number of learners increase. In particular, we will look at the difference in the guarantee for 1 learner and N' learners where $N' \geq 16$. Taking the derivative of Equation 19 with respect to c and setting it to 0, we get the following:

$$4Nc^3 + \alpha c^2 - 2\alpha = 0 \tag{21}$$

Let c_1^* and $c_{N'}^*$ denote the solutions to the above equation for 1 and N' learners respectively. Notice that for $N = 1$ and for $\alpha \geq 16$, the square term dominates in Equation 21 and $c_1^* \approx \sqrt{2}$ (which also satisfies the constraint in Equation 20). Thus we get

$$\text{For } N = 1\text{:} \qquad \frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \lesssim 2\sqrt{2} \cdot \frac{\sigma^2}{\alpha M} \tag{22}$$

However, for $N = N'$ and $16 \leq \alpha \leq N'$, the cubic term dominates in Equation 21. Thus the value of c which satisfies Equation 21 is approximately $\sqrt[3]{\alpha/(2N')}$. However, the upper bound in Equation 20 for large N' becomes $c \lesssim \alpha/(\sqrt{2}N')$. For the range of α under consideration, this is smaller than $\sqrt[3]{\alpha/(2N')}$, thus we get $c_{N'}^* = \alpha/(\sqrt{2}N')$. Thus for $16 \leq \alpha \leq N'$, Equation 18 becomes

$$\text{For } N = N'\text{:} \qquad \frac{\sum_{k=1}^{K} \mathbb{E}(\|\nabla f(x_k)\|^2)}{K} \lesssim \frac{2\sqrt{2}\,N'}{\alpha} \cdot \frac{\sigma^2}{\alpha M} \tag{23}$$

Thus comparing Equation 22 and 23, we see that the ASGD convergence guarantee for $N = 1$ and $N = N'$ learners can differ by a factor of $\frac{N'}{\alpha}$ for $16 \leq \alpha \leq N'$. □