Stochastic Variance Reduction with Adaptive Optimization Methods

Luiz Felipe Santana dos Santos
Data Science Team
Itaú Unibanco
São Paulo, Brazil
luizf@outlook.com

Luis Felipe Bueno
Institute of Science and Technology
Federal University of São Paulo
São José dos Campos, Brazil
lfelipebueno@gmail.com

***Abstract*—In this paper, we study the optimization technique known as *Stochastic Variance Reduced Gradient* (SVRG), a variant of *Stochastic Gradient Descent* (SGD), and analyze the effects of swapping its internal SGD evaluation with other algorithms, such as Adam and RMSProp on Machine Learning problems. We show that the direct application of this technique to neural networks does not result in improvements over SGD or Adaptive SGD methods.**

***Index Terms*—gradient, SVRG, SGD, Adam, RMSProp, variance reduction**

I. Introduction

A common objective in Machine Learning is to find a minima for a given loss function by performing some optimization procedure on it. Usually this arises as a least squares problems where one wants to adjust parameters so that a model reflects actual observations in a training set. Since many of these problems are, in part, defined by the data fed to them, the gradient evaluation can be expensive and memory-wise demanding on large datasets.

Efforts in tackling this issue include the *Stochastic Gradient Descent* (*SGD*) [1] method, which, instead of calculating gradients with respect to an entire dataset, uses gradient evaluations in smaller subsets of data iteratively until a local minima is reached. However, due to the stochastic nature and inherent variance of this technique, the path to a local minima can be noisy, leading to slow convergence. Moreover, convergence is drawn to a noise ball whose radius depends on the variance of the analyzed process. This issue can be reduced by increasing the size of the data subsets to include more examples, coming closer to the full representation of the dataset, but this usually leads to poor generalization of the model over previously unseen data [2].

More effective procedures include adaptive moment methods such as *Adam* [3], *RMSProp* [4] or *Nesterov Momentum* [5], where the search direction is combined with the previous ones, reducing noise. Another way to achieve the same objectives is to use explicit variance reduction techniques, such as the *Stochastic Variance Reduced Gradient* (*SVRG*) [6]. In this method the gradient calculation is estimated by combining the current evaluation in certain samples with evaluations in previous ones.

This paper analyzes a modified *SVRG* algorithm, by replacing its internal *SGD* evaluation with other techniques (henceforth referred to as *optimization kernels*), which are known to provide better convergence rates in some scenarios, particularly non-convex problems such as neural networks.

Any opinions, findings, and conclusions expressed in this manuscript are those of the authors and do not necessarily reflect the views, official policy or position of Itaú Unibanco.

II. Related Work

Min, Long and Cui [7] further studied the convergence of SVRG and created a framework to generalize its results, creating an SVRG-based model where sampling is used to approximate the full gradient, improving performance.

Raj and Stich [8] also developed an SVRG variant where only a subset of the weights, comparable to the available memory, is used to compute an approximation of the full gradient.

Shang, Jiao *et al* [9] add momentum acceleration to SVRG, making it comparable to other momentum methods but, unlike other acceleration tricks, it only adds one additional variable and one momentum parameter to the optimization problem.

Defazio and Bottou [10] recently published a thorough analysis of SVRG methods applied to Deep Learning, and concluded that for larger neural networks, SVRG does not provide significant variance reduction. In their conclusion, the authors also suggest that further analysis combining adaptive methods such as Adam [3] should be conducted, which is contained in the scope of this paper (section IV-B).

III. Algorithms

Supervised Machine Learning problems usually take the form of minimizing an objective function such as

$$\min P(w), \quad P(w) := \frac{1}{n}\sum_{i=1}^{n}\phi_i(w) \tag{1}$$

where $\phi_1, \cdots, \phi_n$ is a sequence of vector functions from $\mathbb{R}^d$ to $\mathbb{R}$ and are loss functions of the parameter w with respect to the training examples $(x_1, y_1), \cdots, (x_n, y_n)$. commonly ϕ is a measure of classification loss (cross-entropy loss will be used throughout this article) and some regularization loss on the weights.

The estimation of w can be done by performing *gradient descent*

$$w := w - \eta \nabla P(w) \tag{2}$$

where η is a parameter called the *learning rate*. However, a large number of components $\phi_i(w)$ turns this into a difficult problem, since n gradients have to be calculated for a single update of w. This problem can be mitigated in many different ways, but some of the current best-performing methods for neural networks and related models are based on SGD, explained below.

A. Mini-batch SGD

The *Mini-batch Stochastic Gradient Descent* method performs the optimization of the objective function defined in (1) using an iterative stochastic approximation of gradient descent direction. This is done by, instead of calculating the gradient with respect to the entire dataset, using only a few randomly selected samples at each iteration. The selected samples are called a mini-batch.

This results in a more economical computation scheme, which allows one to solve some models that would be impracticable if the entire dataset is used, with the caveat that the solution won't converge to a single minima, but instead to a region (a *noise ball*), with some variance near the minima.

B. RMSProp

The *Root Mean Square Propagation* method modifies *SGD* by using an adaptive learning rate for the gradient descent algorithm [4]. This means that, on equation (2), the learning rate is divided by a weight, which modifies the iteration to

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla \phi_i(w) \tag{3}$$

where $v(w,t)$ is a running exponentially decaying average of the magnitudes of the squared gradients, given by:

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla \phi_i(w))^2 \tag{4}$$

where $v(w,0) = 0$ and γ is the forgetting factor, usually chosen as 0.9.

This enables the algorithm to have a large effective learning rate at the beginning of the training procedure and, on the other hand, that rate becomes smaller as the model gets closer to a local minima.

C. Adam

The *Adaptive Moment Estimation* method is similar to *RMSProp* in which it also modifies the learning rate to produce better convergence. However, it does this by calculating the first and second moments of the gradient, as follows:

$$m := \beta_1 m + (1-\beta_1) \nabla \phi_i \tag{5}$$

$$v := \beta_2 v + (1-\beta_2)(\nabla \phi_i)^2 \tag{6}$$

$$\hat{m} = \frac{m}{1-\beta_1^{t+1}} \tag{7}$$

$$\hat{v} = \frac{v}{1-\beta_2^{t+1}} \tag{8}$$

$$w := w - \eta \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \tag{9}$$

where β_1 and β_2 are the forgetting factors for the first and second moments and are usually equal to 0.9 and 0.999 [3], respectively. t is the algorithm's current iteration and ϵ is a scalar to prevent division by zero (equal to 10^{-8}).

D. SVRG

The *Stochastic Variance Reduced Gradient* method deals with the inherent variance of SGD by doing explicit variance reduction, expressed by the equation:

$$\gamma = \nabla \phi_i(w) - \nabla \phi_i(\tilde{w}) + \tilde{\mu} \tag{10}$$

where the first term is calculated by the regular SGD algorithm and the two extra terms do the variance reduction. The second term keeps a version of estimated weights $\tilde{w}$ and is called the *variance reducer* term while the third one is denoted by *progressive direction* which keeps γ not too far away from the full gradient [7].

The complete procedure for SVRG calculation is detailed on Algorithm 1.

Algorithm 1: SVRG Method

```
Result: Vector with w_i weights
Choose a starting value for w;
Define max iterations K;
for s = 1, 2, ... do
    w̃ = w̃_{s-1};
    μ̃ = Σ_{i=1}^{n} ∇φ_i(w̃);
    w_0 = w̃;
    for t = 1, 2, ..., m do
        Select a random sample of the data;
        γ = ∇φ_i(w) − ∇φ_i(w̃) + μ̃;
        w := w − η(γ);
    end
end
```

It is easy to notice that lines 8 through 10 are a slightly altered version of SGD, which leads to the possibility of modifying this section of the algorithm to use other optimization kernels, such as the ones previously mentioned on this section.

IV. Experiments

We investigate the suitability of SVRG with modified optimization kernels by applying the technique to logistic regression and multi-layer fully connected neural networks.

We use the *Apache MXNet* Deep Learning Framework [11] to build the architectures and run them over GPUs, when applicable.

All optimization algorithms are initialized with the same weights and parameters and the learning rates are chosen by a grid-search process where only the best setting is reported.

Since the implementations of SVRG and the other optimization algorithms are not equivalent (for instance, the implementation doesn't fully support parallel computing) the comparisons are made, instead of directly measuring computation times, with respect to the number of gradient evaluations done throughout the training process. This means that, for every m gradient evaluations done by the non-SVRG algorithms, SVRG ones execute $2m + n$ evaluations. We choose $m = 2n$ when performing logistic regression and $m = 5n$ on neural networks, as SVRG authors have done in their original paper [6], resulting in $5/2$ and $11/5$ gradient computations for SVRG versus every single evaluation on SGD, Adam or RMSProp.

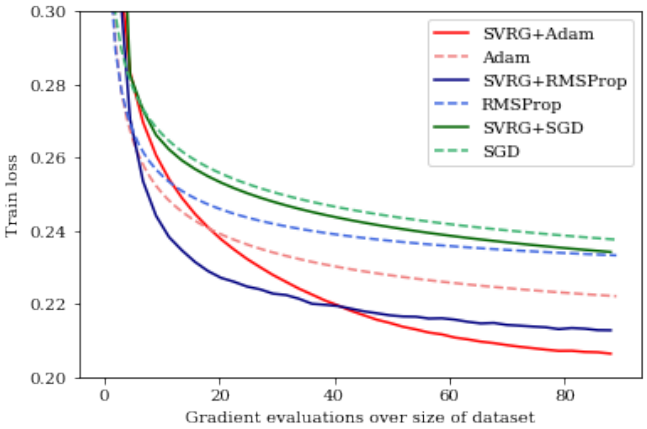
All experiments are performed on the MNIST [12] and Fashion MNIST [13] datasets. Both are comprised of 60 thousand labeled images for training and 10 thousand for validation. Each image is a 28x28 grayscale picture, associated with a label from 10 classes, which are numbers from 0 to 9 (on MNIST) or different clothing items (on Fashion MNIST).

A. *Experiment: Logistic Regression*

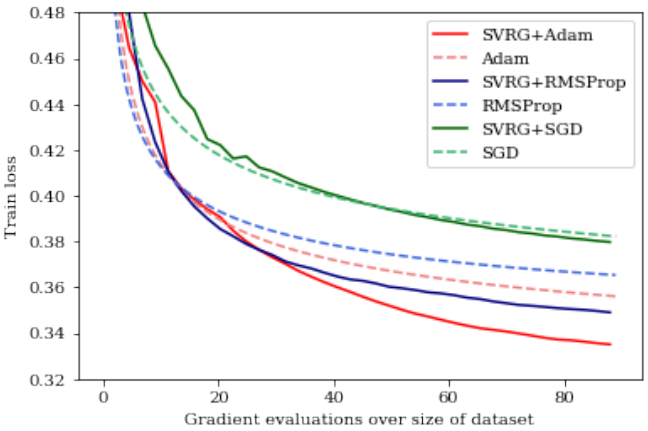
We compare SVRG with different optimization kernels to the non variance reduced algorithms by means of a multi-class logistic regression model with mini-batch size of 256.

Figure 1 shows some performance improvement between SVRG-based models versus the original ones, with a significant improvement on both Adam and RMSProp variants, which suggests that SVRG with adaptive methods probably has some advantages with respect to the original SVRG method using similar computing power.

Regarding validation loss (on Figure 2), it reaches a similar minima for all momentum optimizers, with or without SVRG,



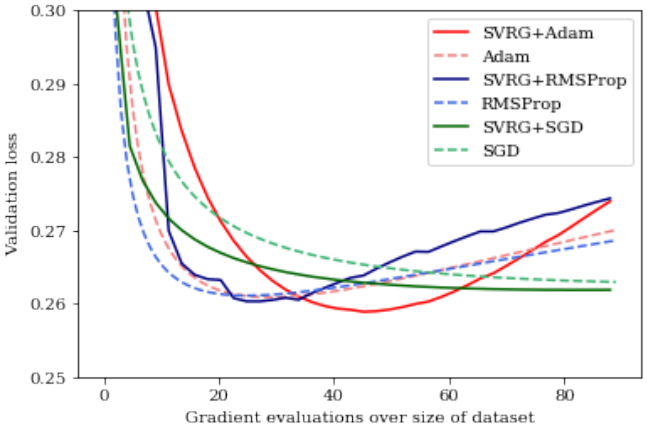
(a) Logistic Regression on MNIST dataset.



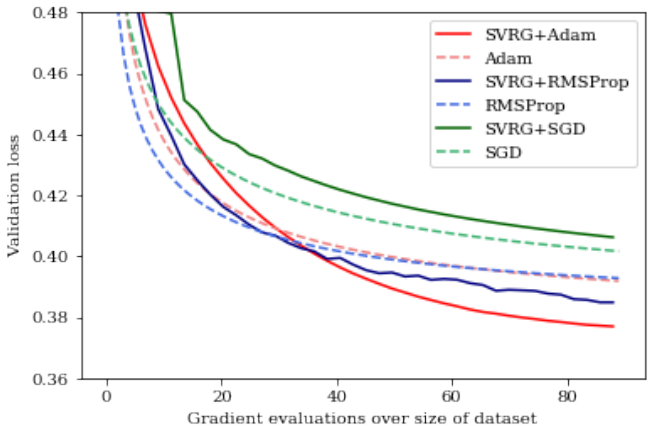
(b) Logistic Regression on Fashion MNIST dataset.

Fig. 1: Training loss on Logistic Regression.

but the SVRG-based ones display some higher tendency to overfit on the MNIST dataset, probably due to its simplicity.



(a) Logistic Regression on MNIST dataset.

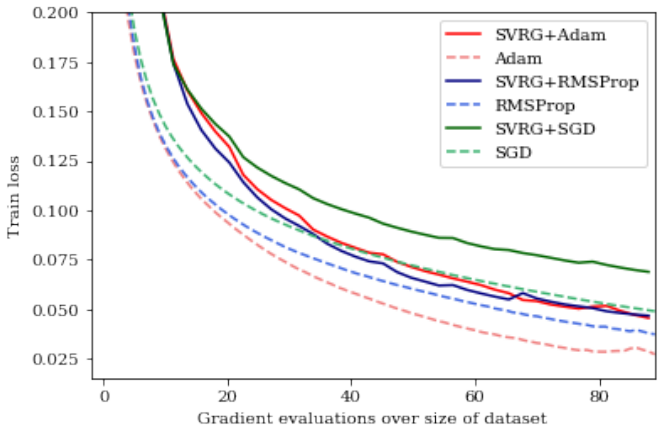


(b) Logistic Regression on Fashion MNIST dataset.

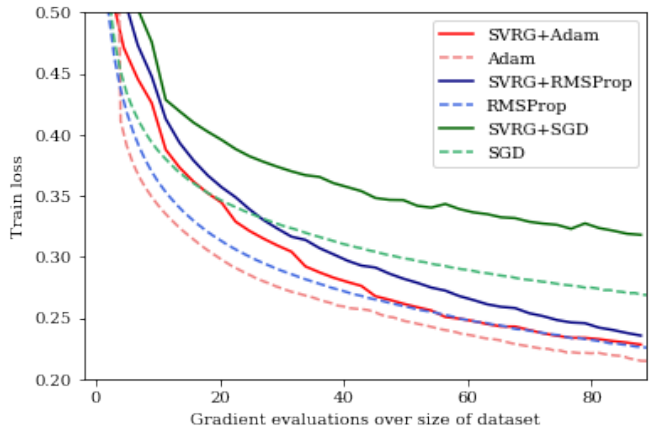
Fig. 2: Validation loss on Logistic Regression.

B. Experiment: Multi-layer Neural Networks

Multi-layer neural networks are models with non-convex objective functions. To test whether the procedure described on this paper is applicable to non-convex problems, we developed a fully-conected, two hidden-layer network with 1000 neurons on each layer, ReLU activation [14], 0.25 dropout [15], softmax output and a mini-batch size of 256.



(a) Neural network on MNIST dataset.



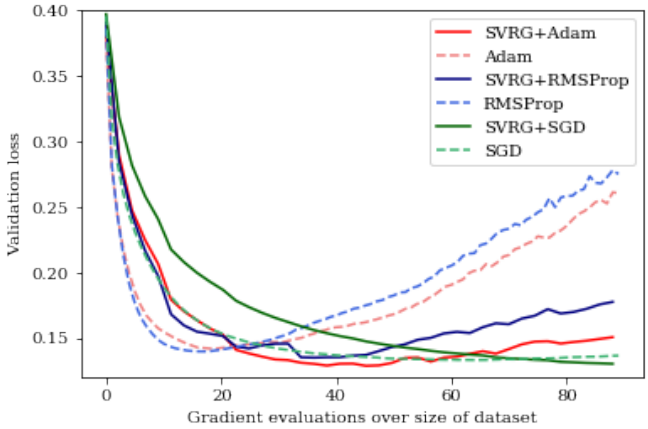
(b) Neural network on Fashion MNIST dataset.

Fig. 3: Training loss on neural networks.

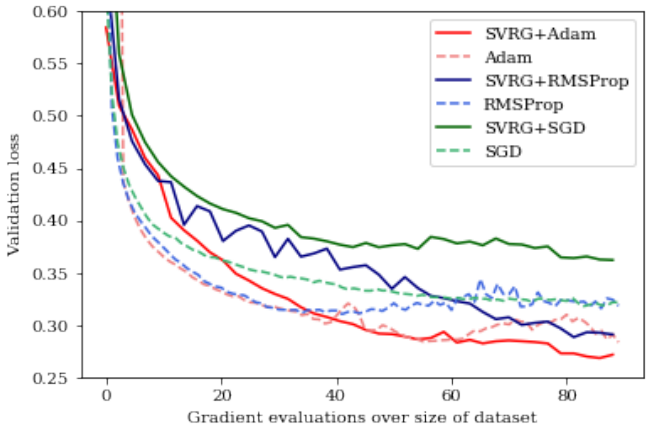
The neural network experiment shows a different result for the SVRG variants when compared to the Logistic Regression one. Figure 3(a) shows that all SVRG variants have performed worse with respect to training loss than their non-SVRG counterparts. Regarding validation loss (Figure 4), the results between the variants are similar, reaching similar lower bounds of error.

Reference [10] studies the behavior of SVRG-SGD on Deep Learning (non-convex) problems. Specifically, it is known [16] that some step correlation is desirable on the search for an optimal solution, and that some of the effectiveness of adaptive methods is derived from this property. SVRG works in a similar fashion, aggregating successive step directions on the last two terms of Equation 10. However, [10] found that these correction factors quickly become uncorrelated with the stochastic gradient and end up increasing the method's variance. In a way, SVRG needs that both the snapshot points (outer loop of Algorithm 1) and the current iterate be similar, but there's evidence that the current iterate w_k moves too quickly for the snapshot weights $\tilde{w}$ to be useful.

Our experiment suggests that this property also holds true for adaptive SVRG methods, albeit applied on shallow neural networks.



(a) Neural network on MNIST dataset.



(b) Neural network on Fashion MNIST dataset.

Fig. 4: Validation loss on neural networks.

V. Conclusion

We have shown that the usage of SVRG with adaptive optimization kernels does not provide a significant improvement over the non-SVRG methods on neural networks, suggesting that the source of variance on this domain cannot be mitigated by these techniques, and that further analysis is needed on applying these methods to neural network models. However, the large improvement seen on logistic regression tasks suggests some future avenues for research on variance reduction tasks.

Future work on this topic should include other optimization kernels applied to SVRG, improve the hyperparameter tuning

for adaptive methods and the update frequency for variance reduction.

VI. Acknowledgement

L. F. S. dos Santos thanks both the Technological Institute of Aeronautics (ITA) for the Data Science Specialization Program and Itaú Unibanco for sponsoring and organizing the initiative, as well as all colleagues within the company for the thoughtful discussions and encouragements.

References

[1] H. Robbins and S. Monro, "A stochastic approximation method," in *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[2] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[4] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[5] Y. Nesterov *et al.*, "Gradient methods for minimizing composite objective function," 2007.

[6] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[7] E. Min, J. Long, and J. Cui, "Analysis of the variance reduction in svrg and a new acceleration method," *IEEE Access*, vol. 6, pp. 16 165–16 175, 2018.

[8] A. Raj and S. Stich, "k–svrg: Variance reduction for large scale optimization," 2018.

[9] F. Shang, L. Jiao, K. Zhou, J. Cheng, Y. Ren, and Y. Jin, "Asvrg: Accelerated proximal svrg," *arXiv preprint arXiv:1810.03105*, 2018.

[10] A. Defazio and L. Bottou, "On the Ineffectiveness of Variance Reduced Optimization for Deep Learning," *arXiv e-prints*, p. arXiv:1812.04529, Dec. 2018.

[11] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," *arXiv e-prints*, p. arXiv:1512.01274, Dec. 2015.

[12] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[13] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[16] R. Kidambi, P. Netrapalli, P. Jain, and S. Kakade, "On the insufficiency of existing momentum schemes for stochastic optimization," in *International Conference on Learning Representations*, 2018.