Effectiveness of Variance Reduction Methods

Ali Zindari
Parham Yazdkhasti
Saarland University

***Abstract*—In this project, we aim to show the effect of variance reduction methods in convex and non-convex optimization problems and we show how they can be better than pure stochastic methods. Moreover, we also show that variance reduction methods are not always the best and they may stick in local minimums when we are in a non-convex regime. We also propose a solution using momentum to overcome this problem. In the end, we present some experiments of the discussed methods when using a neural network for prediction.**

I. Introduction

Stochastic gradient descent (SGD) is a powerful optimization algorithm that offers computational efficiency compared to gradient descent (GD). Due to the random sampling in SGD, we don't need to pass over the whole dataset which could be computationally inefficient. However, this random sampling imposes a noise on the true gradient which can damage the convergence rate. A new line of work has been started for reducing this noise [1], [2], [3]. One of the most popular algorithms for noise reduction is the Stochastic variance reduced gradient (SVRG) proposed in [4]. In the following, we will show the effect of noise reduction by experiment. We also discuss one potential drawback of these methods and show how it can be solved with the help of momentum.

II. Reducing the variance of noise in SGD

We now start by showing the effect of SVRG on a finite-sum convex optimization problem and comparing it with SGD. The minimization problem is in the form of

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \tag{1}$$

We construct a synthetic regression problem in two dimensions in the form of (1) and solve it using SGD and SVRG to compare them. We generate the label of ith data using a linear transformation on this data plus a gaussian noise. So we have

$$y^{(i)} = \mathbf{a}^{(i)}\mathbf{x}^{\star} + \varepsilon, \quad \varepsilon \sim N(0,\sigma^2) \tag{2}$$

$$\mathbf{a}_0^{(i)} \sim U(-0.5, 0.5), \quad \mathbf{a}_1^{(i)} \sim U(-0.125, 0.125) \tag{3}$$

Where $\mathbf{a}_j^{(i)}$ is the jth feature of ith data. Our loss function is $L-$smooth and $\mu-$strongly convex. For $\gamma = \frac{1}{L}$, we have a convergence rate of

$$\mathbb{E}\left\|\mathbf{x}_T - \mathbf{x}^{\star}\right\|^2 \le \exp\left(\frac{-\mu T}{L}\right)\left\|\mathbf{x}_0 - \mathbf{x}^{\star}\right\|^2 + \frac{\sigma^2}{\mu L} \tag{4}$$

The second term in the above equation comes from the noise of SGD. It basically shows that SGD cannot converge to the exact solution when $\gamma = \frac{1}{L}$. Instead, it can only converge to a neighborhood of solution. Figure 1 perfectly captures this term. However, SVRG does not have this issue and it is clear that it reached the global optimum. The reported results in Table I also show that SVRG outperforms SGD in minimizing the loss on our synthetic data in all different noise levels. So SVRG has the computational efficiency of SGD and also the smooth behavior of GD at the same time.

TABLE I
Comparison of the smallest error that each algorithm could reached under different noise levels.

Noise Variance (σ^2)	0.1	0.5	1	1.5
SGD	0.0122	0.3138	1.2990	3.7292
SVRG	0.0097	0.2520	1.0204	2.3384

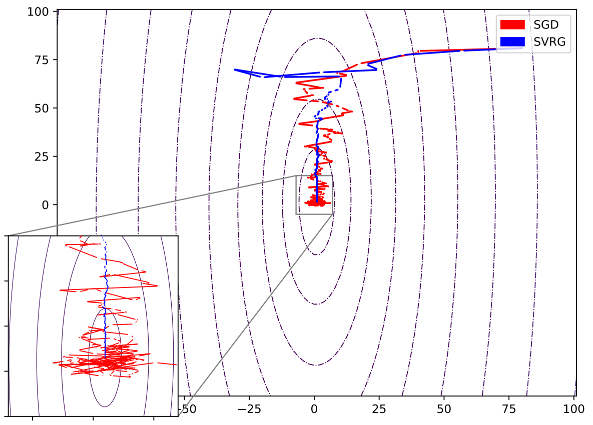

Fig. 1. Comparison of SGD and SVRG in a linear regression problem. Overall, SVRG is more stable and can reach the global optimum. However, SGD can only reach to a neighborhood of the global optima because of the presence of the noise and it fluctuates around it forever.

III. Noise Reduction is not always the best strategy

In the previous section, we discussed the benefits of noise reduction in the optimization. While it may initially appear that using noise reduction is always advantageous, this section aims to highlight cases where SGD can outperform other optimization methods. The underlying concept is that there might be shallow local minima along the optimization path that the optimizer traverses before reaching the global minimum. Surprisingly, the noise present in SGD can help the optimizer to escape these local minima. On the contrary, optimizers utilizing variance reduction techniques may struggle to break free from such local minima. To gain a better understanding of this phenomenon, let's consider an extreme scenario. An ideal variance reduction method would resemble Gradient Descent (GD). Since the primary objective of variance reduction is to eliminate the noise generated by SGD sampling, this method should completely eradicate noise and behave like GD. However, GD is known to become trapped in local minima once encountered. Hence, reducing the noise in SGD diminishes the optimizer's capability to escape local minima. In this section, we present an optimization experiment that demonstrates the following scenario: while SGD successfully reaches the global optimum, SVRG gets stuck in a local minimum. We define $f(x)$ in the following manner:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} a_i x^2 + \underbrace{c_1 e^{-(x-c_2)^2}}_{h(x)} \tag{5}$$

Now, let's examine the loss landscape in our simple one-dimensional experiment illustrated in Figure 2. To create a local minimum around $x = -6.4$, we set $c_1 = 4$ and $c_2 = -5.5$. Additionally, the initialization of all a_i values ensures that $\frac{1}{n}\sum_{i=1}^{n} a_i = \frac{1}{4}$. In this experiment, we demonstrate that SGD, with a sufficient level of noise, is capable of escaping local minima and converging to $x = 0$, which serves as the global minimizer for this problem. However, before proceeding, we need to find a method to control the noise experienced by SGD in this particular scenario. By assuming that the index i is uniformly sampled at each iteration, we can compute the variance of noise imposed on true gradients by SGD for this specific setting according to Lemma 1 as

$$noise(SGD) = 4x^2.Var(a_i) \tag{6}$$

We can observe that by adjusting the variance of the coefficients, a_i, we have the ability to regulate the gradient noise in the experiment. Now let's take a closer look at what happens in the local minima; we know that in the neighborhood of the center of this local minimum (roughly $x = -6.4$) norm of the gradient starts to vanish. Consequently, our only chance of escaping this local minimum lies in the presence of noise within the gradient. Without the noise, we would be trapped in this local minimum indefinitely. From the Lemma 2 we have

$$\mathbb{E}\,\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \approx \gamma^2\,\mathbb{E}\,\|g_t - \nabla f(\mathbf{x}_t)\|^2 \tag{7}$$

So here we can see that noisy steps are not always a bad thing. The only duty of any variance reduction method is to reduce $\mathbb{E}\,\|g_t - \nabla f(\mathbf{x}_t)\|^2$ as much as possible. As a consequence, the norm of its steps $\mathbb{E}\,\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2$, tends to approach zero when it enters a neighborhood of a local minimum. This phenomenon is illustrated in Figure 2, where SGD and SVRG begin from the same initial point. While SGD successfully escapes the local minimum, SVRG fails to do so. This issue may not be of great concern for simpler optimization problems, such as convex problems that lack local minima entirely. However, for optimizing non-convex problems, it becomes a significant challenge. Training deep networks serves as an extreme example, as the loss landscape of deep networks is highly non-convex, featuring numerous local minima along the optimizer's path before reaching the global optimum.

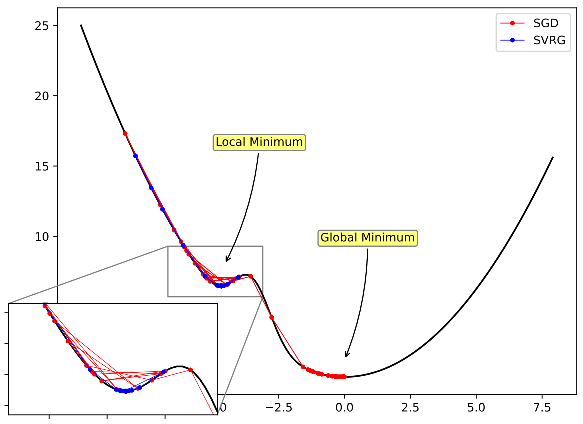

Fig. 2. An example of a non-convex function in which SVRG fails to optimize and is stuck in a local minimum. However, SGD with the same step size can jump over this local minima and reach the global minima.

IV. Momentum can save us!

Momentum-based methods have been widely used in the context of optimization in order to speed up the convergence of SGD. Momentum addresses this issue by taking the history of gradients into account. We first construct a synthetic non-convex function in two dimensions and minimize it using four different optimizers. We define the function $f(x, y)$ as follows

$$f(x,y) = \frac{1}{n}\sum_{i=1}^{n}\Big(a_{i1}x^2 + a_{i2}y^2\Big) - ce^{-(\frac{x-c_1}{s_1})^2 - (\frac{y-c_2}{s_2})^2} \tag{8}$$

Where $c = 24.5$, $c_1 = c_2 = 6.5$ and $s_1 = s_2 = 3.5$. We also ensure that $\frac{1}{n}\sum_{i=1}^{n} a_{i1} = \frac{1}{4}$ and we set $a_{i2} = 2a_{i1}$. In this section, we introduce a method called **MOSVRG** which combines momentum and noise reduction at the same time (1). From the experiment in Figure 3, we see that by combining these two methods, we can escape from the local minima and also we can get very close to the global optimum. So we somehow enjoy the benefits of SGD and SVRG at the same time. We can take a closer look in Figure 4. We see that momentum can help to escape from the local optimum but without using variance reduction, we see the noisy behavior around the global optimum. So in this case, SVRG+momentum would be the best choice.

V. Deep Learning experiments

In this section, we compare four different optimizers introduced in the previous section. We use two widely used datasets, CIFAR10 and FashionMNIST for this task. For each dataset, 10% of the total training and test set is used due to the limited resources. The data is first passed through a ResNet50 pre-trained network and features are extracted, then a 4-layer feed-forward network is trained on top of that for the classification. The performance of each optimizer during the training is plotted in Figure VI. With a large learning rate of 0.01, we can see that SGD+Momentum couldn't converge. It shows that this learning rate is large while using momentum. However, other methods performed relatively the same and converged fast. On the other hand, with a small learning rate of 0.001 we can see that SGD+Momentum is the best and that's because momentum itself accelerates the convergence. For the other methods, it took a longer time to converge. Finally, for a fair comparison, we used the best learning rate for each algorithm (small for SGD+Momentum and large for the others). It seems that all of them performed relatively the same. So we consider another experiment to evaluate their generalization performance. From the results in Table II we can see that SVRG has the best generalization performance among the other algorithms. However, due to the huge amount of memory that is required to store the gradients, this method is not efficient in Deep Learning (DL). So SGD or SGD+Momentum is still preferred when we have neural networks. In addition, if we use some techniques like batch normalization, data augmentation, and dropout in our problem, the pure SRVG method is a bad choice according to [5].

VI. Conclusion

In this project, we've seen the effectiveness of variance reduction methods in simple convex and non-convex problems. We also saw why in some cases these methods can be problematic and be stuck in local minima and how momentum can help us. Also in the context of DL, although the results show that SVRG is better on the test set, we conjecture that it might be because of the simplicity of the model. In much deeper models, the noise of SGD could be of advantage for escaping from some local minima which is not in the scope of this project.

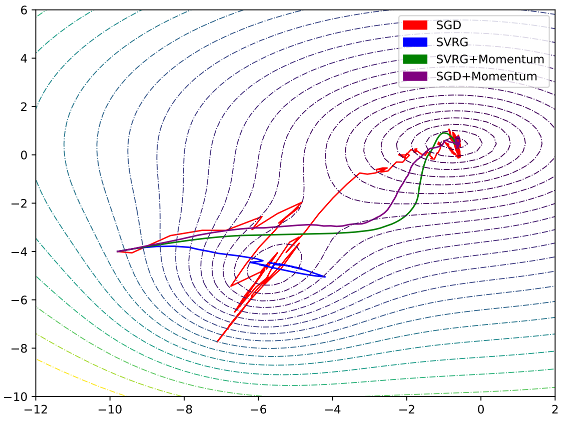

Fig. 3. Comparison of 4 different optimizers on a 2-dimensional non-convex problem.

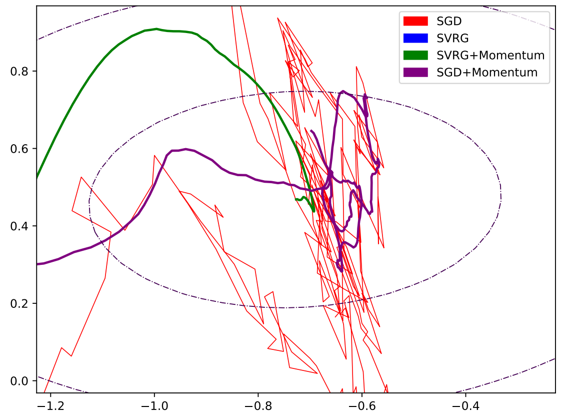

Fig. 4. Comparison of the behavior of different optimizers when they get very close to global optima. As you can see, SGD has the noisiest behavior because it doesn't have any mechanism for noise reduction. SGD+Momentum is the second noisiest. It's because of the presence of momentum which acts as a noise reduction method. SVRG+Momentum has the smoothest behavior and can approximately reach the global optimum because both SVRG and momentum reduce the noise.

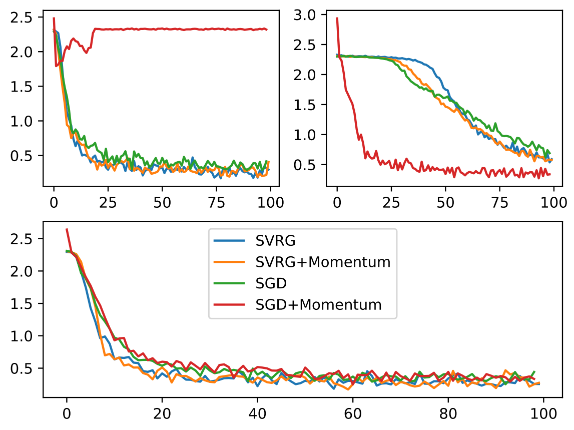

Fig. 5. These figures illustrate the loss value of the four discussed methods. On the top left of the first row, all algorithms used $\gamma = 0.01$ as the step size. on the top right, step size $\gamma = 0.001$ was used; and finally, on the bottom we compared the performance of these algorithms, using the best step size for each of them.

TABLE II
Comparison of accuracy achieved by four algorithms on two datasets.

	SVRG	SVRG+Momentum	SGD	SGD+Momentum
CIFAR10	85.4	82.4	85.2	84.4
FasionMNIST	83.2	83.0	82.5	83.0

REFERENCES

[1] M. Schmidt, N. Le Roux, and F. Bach, “Minimizing finite sums with the stochastic average gradient,” *Mathematical Programming*, vol. 162, pp. 83–112, 2017.

[2] A. Defazio, F. Bach, and S. Lacoste-Julien, “Saga: A fast incremental gradient method with support for non-strongly convex composite objectives,” *Advances in neural information processing systems*, vol. 27, 2014.

[3] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, “Sarah: A novel method for machine learning problems using stochastic recursive gradient,” in *International conference on machine learning*. PMLR, 2017, pp. 2613–2621.

[4] R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” *Advances in neural information processing systems*, vol. 26, 2013.

[5] A. Defazio and L. Bottou, “On the ineffectiveness of variance reduced optimization for deep learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

Appendix

Lemma 1. *Let $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ be an arbitrary function we are trying to optimize using SGD. The noise imposed by SGD while computing the gradients can be formulated as*

$$noise(SGD) = 4x^2.Var(a_i) \tag{9}$$

Proof. We start by the definition of variance

$$\begin{aligned}
&\mathbb{E}\left[\|\nabla f_i(x) - \nabla f(x)\|^2 \,\middle|\, x\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(2a_i x + \nabla h(x) - \frac{1}{n}\sum_{i=1}^{n}\left(a_i x^2 + \nabla h(x)\right)\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(2a_i x - \left(\frac{2}{n}\sum_{i=1}^{n} a_i\right) x\right)^2 \\
&= \frac{4}{n}\sum_{i=1}^{n}\left(a_i - \left(\frac{1}{n}\sum_{i=1}^{n} a_i\right)\right)^2 x^2 = 4x^2.Var(a_i)
\end{aligned}$$

□

Lemma 2. *Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t, \mathbf{x}_{t+1}$ be the sequence of parameters generated by SGD. If we are sufficiently close to the optima, we can write the distance between two steps in the form of*

$$\mathbb{E}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \approx \gamma^2\,\mathbb{E}\|g_t - \nabla f(\mathbf{x}_t)\|^2 \tag{10}$$

Proof.

$$\begin{aligned}
&\mathbb{E}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\
&= \mathbb{E}\|-\gamma g_t\|^2 \\
&= \gamma^2\,\mathbb{E}\|g_t - \nabla f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)\|^2 \\
&= \gamma^2\,\mathbb{E}\|g_t - \nabla f(\mathbf{x}_t)\|^2 + \gamma^2\,\mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2 + \underbrace{\mathbb{E}\left[2\gamma\nabla f(\mathbf{x}_t)^T\left(g_t - \nabla f(\mathbf{x}_t)\right)\right]}_{=0} \\
&= \gamma^2\,\mathbb{E}\|g_t - \nabla f(\mathbf{x}_t)\|^2 + \underbrace{\gamma^2\,\mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2}_{\approx 0 \text{ around local minimum}} \\
&\approx \gamma^2\,\mathbb{E}\|g_t - \nabla f(\mathbf{x}_t)\|^2
\end{aligned}$$

□

Algorithm 1: MOSVRG

```
Input: $\mathbf{x}_0, k, \alpha, T, \gamma$
Initialize: $\widetilde{\mathbf{x}} = \mathbf{x}_0$
for $i = 1, 2, ..., T$ do
    Compute $\nabla f_i(\widetilde{\mathbf{x}})$ for all $i$
    for $t = 1, 2, ..., k$ do
        $v_t = \nabla f_i(\mathbf{x}_t) - \nabla f_i(\widetilde{\mathbf{x}}) + \nabla f(\widetilde{\mathbf{x}})$
        $m_t = \alpha m_{t-1} + (1-\alpha) v_t$
        $\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma m_t$
    end
    $\widetilde{\mathbf{x}} = \mathbf{x}_k$;
end
```