Variance Reduction Methods Help Accelerating Stochastic Gradient Descent

Yuhong Shao [1] **Hanghuan Yang** [1]

Abstract

With the rapid development of deep learning and machine learning, stochastic gradient descent (SGD) becomes increasingly important as an optimization algorithm for predicting model parameters that correspond to the best fit between predicted and actual values. Especially in high-dimensional optimization problems, SGD can effectively reduce high computational burden. However, SGD is not omnipotent, and this algorithm also has some drawbacks such as a slow convergence rate due to the inherent variance. In this paper, we will introduce two new optimization algorithms - Stochastic Dual Coordinate Ascent Method (SDCA) and Stochastic Variance Reduced Gradient(SVRD) (Johnson & Zhang, 2013), which helps with some complex modern hot topics within the area of data science such as some structured prediction problems and neural network learning.

1. Stochastic Gradient Descent

1.1. Introduction to Stochastic Gradient Descent

With the continuous development of computers in the last century, the amount of data that a personal computer can store has grown exponentially, so the computation and optimization methods for processing data have also upgraded and evolved.

In 1951, Herbert Robbins and Sutton Monro introduced an insightful stochastic approximation method, they named it as Robbins–Monro method, and it was considered as the precursor of the modern stochastic gradient descent method (Pedergosa, 2018).

Stochastic gradient descent (also known as SGD) is an optimization algorithm widely-used in deep learning and machine learning to find the model parameters.

*Equal contribution [1]Statistics Department, University of Toronto Scarborough, Scarborough, Canada. Correspondence to: Yuhong Shao <yvonne.shao@mail.utoronto.ca>, Hanghuan Yang <hanghuan.yang@mail.utoronto.ca>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

SGD can be used to solve optimization problems in the form of $f(\cdot) = \mathbb{E}[f_i(\cdot)]$, the most common case is when i can take a finite number of values, the problem becomes:

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

Here, f_i is a loss function or cost function (sometimes we can call it error function). A loss function is used to measure the "cost" of the difference between the predicted and true values of a model. With the help of SGD method we can efficiently minimize the cost of misclassification of each element in the dataset. For example, consider a sequence of n training dataset $(x_1, y_1), \cdots, (x_n, y_n)$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We can choose any cost function (e.g. Quadratic loss function, 0-1 loss function), and to make it clear and simple, we choose the most common one - squared loss function: $f_i(w) = (w^T x_i - y_i)^2$. With all the data we have, the standard gradient descent update rule can be written as (Pedergosa, 2018):

Assume: $t = 1, 2, \cdots$

$$w^{(t)} = w^{(t-1)} - \eta_t \nabla P(w^{(t-1)}) \tag{2}$$

$$= w^{(t-1)} - \frac{\eta_t}{n}\sum_{i=1}^{n} \nabla f_i(w^{(t-1)}) \tag{3}$$

From (3), the computational workload of the gradient descent is massive because it requires evaluation of n derivatives in each iteration; therefore, stochastic gradient descent method can be applied. In each iteration $t = 1, 2, \cdots$, randomly draw i_t from the sequence $1, \cdots, n$, and we can get:

$$w^{(t)} = w^{(t-1)} - \eta_t \nabla f_{i_t}(w^{(t-1)}) \tag{4}$$

We can also get the pseudo-code of stochastic gradient descent, the step size γ is the only free parameter in this algorithm (Pedergosa, 2018).

Algorithm 1 The Stochastic Gradient Method

Input: initial guess x_0, step size sequence γ_t
Require: $\{x^{(i)}y^{(i)}\}$: training dataset
for $t = 0, 1, \cdots$ **to** n **do**
  Randomly choose $i \in \{1, 2, \cdots, n\}$
  $w_{t+1} = w_t - \gamma_t \nabla f_i(w_t)$
end for
Stop until convergence, return w_t

1.2. Advantages and Disadvantages of Stochastic Gradient Descent Method

Although stochastic gradient descent is widely used and has an irreplaceable role in the field of deep learning and machine learning, it still has some drawbacks that cannot be ignored.

Advantages:

- The overall algorithm is easy to write in all computer languages and takes less memory space in the computer compared to other optimization methods.
- It is computationally fast as a single selected sample is processed in each iteration, each step only relies on a single derivative ∇f_i [(4)], and thus the computational cost is only $\frac{1}{n}$ that of the gradient descent method [(3)].
- It can converge faster as it causes updates to the parameters more frequently in a relatively larger dataset.

Drawbacks:

- Randomness introduces variance, normally a large variance will slows down the convergence rate.
- Sometimes, it is hard to balance the trade-off between the computation cost per iteration and convergence rate.
- The benefits of vectorized operations are lost as it deals with a single randomly selected sample in each iteration.

We can see that the randomness of data, undoubtedly and greatly affects the result of stochastic gradient, is there a method that data scientist can optimize data more effectively? The answer is obvious, as a member of the gradient descent family, there are a large number of variants of gradient descent methods, each with its own advantages and disadvantages for handling different cases. In this paper, we will introduce two variants - Stochastic Dual Coordinate Ascent (SDCA) and Stochastic Variance Reduced Gradient (SVRG). Meanwhile, we also compare these two new methods with SGD to obtain their differences from SGD.

2. Stochastic Dual Coordinate Ascent Method

2.1. What is Stochastic Dual Coordinate Ascent and why we use this method?

As we mentioned before, a more efficient approach should be able to minimize the impact of the randomness of the data on the variability of each step. Stochastic dual coordinate ascent method (often referred as SDCA) is applied to this situation, it can effectively reduce the variance of the data, which allows us to use a relatively larger learning rate η_t and also leads to a linear convergence rate when $\psi_i(w)$ is smooth and strongly convex.

Now, let us discuss what exactly is SDCA. First, assume n feature vectors $x_1, \cdots, x_n \in \mathbb{R}^d$ be the training examples, $w \in \mathbb{R}^d$ be the weight vector, and let $\phi_i : \mathbb{R} \to \mathbb{R}$ be the convex loss function associated with linear prediction, then our goal is to minimize the following regularized empirical loss (Tran et al., 2015):

$$P(w) = \frac{1}{n}\sum_{i=1}^{n}\phi_i(w^T x_i) + \frac{\lambda}{2}||w||_2^2 \tag{5}$$

, where the non-negative λ is a regularization parameter. To make this simple and clear, we will only focus on l_2 regularization and only consider binary classifications, the linear SVM will be $\phi_i(a) = max\{0, 1 - y_i a\}$, regularized logistic regression will be $\phi_i(a) = log(1 + exp(-y_i a))$ and the loss function ϕ will be the squared error loss. Stochastic Dual Coordinate Ascent (SDCA) works well under current case, the dual problem is to maximize the dual function below (Tran et al., 2015):

$$D(\alpha) = \frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(-\alpha_i) - \frac{\lambda}{2}||\frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i||_2^2 \tag{6}$$

, where our new $\phi^* : \mathbb{R} \to \mathbb{R}$ be the convex conjugate of ϕ_i. Assume $w^* = \arg\min P(w)$ be the primal optimal solutions and $\alpha^* = \arg\min D(\alpha)$ be dual optimal solutions, and define $w(\alpha) = \frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i$, then we can get $w^* = w(\alpha^*)$ and $P(w^*) = D(w^*)$ respectively. We can argue that the solution $w \in \mathbb{R}^d$ is ξ_P-sub-optimal if $P(w) - P(w^*) < \xi_P$.

We can also use pseudo-code to show how we can apply SDCA (Shalev-Shwartz & Zhang, 2013):

Algorithm 2 Stochastic Dual Coordinate Ascent Method

Input: initial step $w^{(0)} = w(\alpha^{(0)})$, T: number of iterations
Require: $\{x^{(i)}y^{(i)}\}$: training dataset
for $t = 0, 1, \cdots T$ **to** n **do**
  Randomly choose $i \in \{1, 2, \cdots, n\}$
  Find $\nabla\alpha_i$ to maximize $-\phi_i^*(-(\alpha_i^{(t-1)} + \nabla\alpha_i)) - \frac{\lambda n}{2}||w^{(t-1)_i} + \frac{\nabla\alpha_i x_i}{\lambda n}||^2$
  Set: $\alpha^{(t)} \leftarrow \alpha^{(t-1)} + \nabla\alpha_i x_i$
  Set: $w^{(t)} \leftarrow w^{(t-1)} + \frac{\nabla\alpha_i x_i}{\lambda n}$
end for
Output Option I : Averaging Option
  Let $\bar{\alpha} = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T}\alpha^{(t-1)}$
  Let $\bar{w} = w(\bar{\alpha}) = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T}w^{(t-1)}$
Return: $\bar{w}$
Output Option II : Random Option
  Randomly select $t \in \{T_0 + 1, \cdots, T\}$, let $\bar{w} = w^{(t)}$ and $\bar{\alpha} = \alpha^{(t)}$
Return: $\bar{w}$

2.2. Disadvantages of Stochastic Dual Coordinate Ascent Method

Unlike SGD only a single gradient is stored in each iteration. From the pseudo-code Algorithm 2, we can see that all the gradients will be stored throughout the iterations, which means more computer memory is used when we use SDCA. While this may not be problematic for training simple regularized linear prediction model such as least squares regression, it is not suitable for more complex and high-dimensional applications where storing all the gradients is impractical. As we needed to make a trade-off between reducing the variance of the data and using less memory space, a more efficient variant of the gradient descent optimization method was devised, named Stochastic Variance Reduced Gradient (usually referred as SVRD).

3. Stochastic Variance Reduced Gradient

3.1. Introduction to Stochastic Variance Reduced Gradient

As we mentioned before, we now need a variant method that minimizes randomness and reduce memory usage, and Stochastic Variance Reduced Gradient is a perfect choice. we may keep a snapshot of $\tilde{w}$ for every m SGD iterations, the average gradient descent will be:

$$\tilde{\mu} = \nabla P(\tilde{w}) = \frac{1}{n}\sum_{i=1}^{n}\nabla\psi_i(\tilde{w}), \tag{7}$$

only a single $\tilde{w}$ has been passed throughout the computation. The new updating rule is randomly select i from the sequence $\{1, \cdots, n\}$:

$$w^{(t)} = w^{(t-1)} - \eta_t(\nabla\psi_i(w^{(t-1)}) - \nabla\psi_{i_t}(\tilde{w}) + \tilde{\mu}) \tag{8}$$
$$\text{with} \quad \mathbb{E}[w^{(t)}|w^{(t-1)}] = w^{(t-1)} - \eta_t\nabla P(w^{(t-1)}) \tag{9}$$

Since we have $\sum_{i=1}^{n}(\nabla\psi_i(\tilde{w}) - \tilde{\mu}) = 0$,

$$P(w) = \frac{1}{n}\sum_{i=1}^{n}\psi_i(w) = \frac{1}{n}\sum_{i=1}^{n}\tilde{\psi}_i(w) \tag{10}$$

We can see that the entire idea is still based on gradient descent, the only difference is the update rule (8), the variance of the data is reduced, that is the reason why this variant is called stochastic variance reduced gradient (SVRG) because it explicitly reduces the variance of SGD. We can also use pseudo-code to show how we can apply SVRG (Johnson & Zhang, 2013):

Algorithm 3 Stochastic Variance Reduced Gradient Method

Parameters: update frequency m and learning rate η
Initialize: $\tilde{w}_0$
for $s = 1, 2, \cdots$ **do**
  $\tilde{w} = \tilde{w}_{s-1}$
  $\mu = \frac{1}{n}\sum_{i=1}^{n}\bigtriangledown\phi_i(\tilde{(w)})$
  $w_0 = \tilde{w}$
  for $t = 1, 2, \cdots m$ **do**
    Randomly pick $i_t \in \{1, ..., n\}$ and update weight $w_t = w_{t-1} - \eta(\bigtriangledown\phi_{i_t}(w_t - 1) - \bigtriangledown\phi_{i_t}(\tilde{w}) + \tilde{\mu})$
  end for
  option I: set $\tilde{w}_s = w_m$
  option II: set $\tilde{w}_s = w_t$ for randomly chosen $t \in 0, \cdots, m-1$
end for

3.2. Intuition of Stochastic Variance Reduced Gradient

In the case of stochastic gradient descent, in each iteration we choose a random data point whose index is $i_k \in \{1, ...n\}$ and calculate an estimate of gradient $\bigtriangledown\tilde{f_{i_k}}(w_k)$ accordingly. The expected value of $\bigtriangledown\tilde{f_{i_k}}(w_k)$ is

$$E[\bigtriangledown f_{i_k}\tilde{(w_k)}|w_k] = \frac{1}{n}\sum_{i=1}^{n}\bigtriangledown f_i(w_k) = \bigtriangledown f(w_k) \tag{11}$$

Therefore this is an unbiased estimator. However, the variance of $\bigtriangledown\tilde{f_{i_k}}(w_k)$ is bounded by

$$Var(\bigtriangledown \tilde{f}(w)) \leq C||E[\bigtriangledown \tilde{f}(w)]||_2^2 = C||\bigtriangledown f(w)||_2^2 \tag{12}$$

$$i.e.\ Var(\bigtriangledown \tilde{f}(w)) \leq C||w - w^*||_2^2 \tag{13}$$

Assume we know the optimal value w^* before hand, then we can improve the the gradient estimator by defining it as $\bigtriangledown \tilde{g}(x) = \bigtriangledown \tilde{f}(w) - \bigtriangledown \tilde{f}(w^*)$. Then, variance of gradient estimator is bounded for loss function that is $\sqrt{C}$-Lipschitz continuous by

$$||\bigtriangledown \tilde{g}(x)||_2^2 = ||\bigtriangledown f(w) - \bigtriangledown f(w^*)||_2^2 \leq C||w - w^*||_2^2 \tag{14}$$

This means that linear convergence is only guaranteed under two conditions. Firstly, loss function f is $\sqrt{C}$-Lipschitz continuous. Secondly, we know the optimal solution for the said optimization problem, which is not practical.

When we don't know the optimal solution to the problem, we can approximate this solution by using an approximation $\hat{w}$. So our gradient estimator evolves to $\bigtriangledown \tilde{g}(w) = \bigtriangledown \tilde{f}(w) - \bigtriangledown \tilde{f}(\hat{w})$. Although variance could now be bounded, a new problem emerges: this estimator is biased.

To balance out the bias introduced by term $-\bigtriangledown \tilde{f}(\hat{x})$, we simply add the expected value $E[\bigtriangledown \tilde{f}(\hat{x})]$. Therefore, our final version of gradient estimator is $\tilde{g}(w) = \bigtriangledown \tilde{f}(w) - \bigtriangledown \tilde{f}(\hat{w}) + E[\bigtriangledown \tilde{f}(\hat{w})]$. In the case of SVRG, since we update the estimated optimal value $\hat{w}$ after every certain number of iterations, $\hat{x}$ converges to the true optimal value at a linear rate, meaning that the variance of gradient estimator also converges to zero at a linear rate. This technique ensures that the estimate of weights doesn't converge to a stochastic ball as in the case of SGD.

4. Examples and Counterexamples

4.1. Example: Strongly Convex Optimization

4.1.1. Convergence of SGD

We discuss the strongly convex optimization problem of a linear regression analysis with n pairs of observations $\{x_n, y_n\}$. The loss function is square loss $f_{x_i,y_i}(w) = (y_i - w^T x_i)^2$, so $||\bigtriangledown^2 f_{x_i,y_i}(w)||_2 = \lambda = 2$ and loss is strongly convex. Assume w^* represents the optimal weight of this optimization problem.

To derive the convergence rate, we first derive a bound on the step size in one single iteration by taking the l2-norm of $w_{t+1} - w^*$:

$$E[||w_{t+1} - w^*||_2^2] \tag{15}$$

$$= ||E[w_{t+1} - w^*]||_2^2 + \frac{\lambda}{2} Var(w_{t+1} - w^*) \tag{16}$$

$$= ||E[w_{t+1} - w^*]||_2^2 + Var(w_{t+1} - w^*) \tag{17}$$

For the first term $||E[w_{t+1} - w^*]||_2^2$, we apply the gradient descent convergency theorem and get a bound:

$$||E[w_{t+1} - w^*]||_2^2 \leq (1 - 2\alpha)^2 E[||x_t - w^*||_2^2] \tag{18}$$

For the second term we derive a bound:

$$Var(w_{t+1} - w^*) \tag{19}$$

$$= Var(w_t - w^* - \alpha(\bigtriangledown f_{x_{i_t},y_{i_t}}(w_t) - \frac{1}{n}\sum_{i=1}^{n} \bigtriangledown f_{x_i,y_i}(w^*)) \tag{20}$$

$$= Var((I - \alpha \bigtriangledown^2 \frac{1}{n}\sum_{i=1}^{n} \bigtriangledown^2 f_{x_i,y_i}(z_t))(w_t - w^*) \tag{21}$$

$$- \alpha(\bigtriangledown f_{x_{i_t},y_{i_t}}(w_t) - \frac{1}{n}\sum_{i=1}^{n} \bigtriangledown f_{x_i,y_i}(w_t))) \tag{22}$$

$$= \alpha^2 Var(\bigtriangledown f_{x_{i_t},y_{i_t}}(w_t) - \frac{1}{n}\sum_{i=1}^{n} \bigtriangledown f_{x_i,y_i}(w_t))) \tag{23}$$

$$\leq \alpha^2 M \tag{24}$$

Putting these two terms together, we have:

$$E[||w_{t+1} - w^*||_2^2] \tag{25}$$

$$\leq (1 - 2\alpha)^2 E[||w_t - w^*||_2^2] + \alpha^2 M \tag{26}$$

Adding the telescoping terms $w_0, ...w_{t+1}$ together, we then have:

$$E[||w_{t+1} - w^*||_2^2] \leq \frac{\alpha M}{4 - 4\alpha} \tag{27}$$

Therefore we see that instead of converging to a specific point, the weight approximation of the SGD algorithm converges to a fix-size L2-ball because of the variance term $Var(w_{t+1} - w^*)$.

4.1.2. Convergence of SVRG

Our objective is to construct a gradient estimator $\bigtriangledown \tilde{g}(w)$ such that the variance of the gradient estimator goes to

zero as the number of iterations increase. Assume the loss function f is L-Lipschitz.

First we try to determine the variance of gradient update by L-lipschitz properties and triangular inequality:

$$Var[\bigtriangledown f_{i_t}(w_t) - \bigtriangledown f_{i_t}(\hat{w}) + \sum_{i=1}^{n} \frac{\bigtriangledown f_i(\hat{w}))}{n}] \tag{28}$$
$$= E[|| \bigtriangledown f_{i_t}(w_t) - \bigtriangledown f_{i_t}(\hat{w})||_2^2] \tag{29}$$
$$\leq L^2||w_t - \hat{w}||_2^2 \tag{30}$$
$$\leq 2L^2||w_t - w^*|| + 2L^2||\hat{w} - w^*||_2^2 \tag{31}$$

Since we know that w_t and $\hat{w}$ converges to w^*, we assert that variance gradient estimator is indeed diminishing.

Then we try to derive a bound for the step size of each iteration of the inner loop of SVRG using L-lipschitz properties and result (31) we derived above:

$$E[||w_{t+1} - w^*||_2^2] \tag{32}$$
$$= E[||w_t - w^* - \alpha(\bigtriangledown f_{x_{i_t}, y_{i_t}}(w_t) \tag{33}$$
$$- \bigtriangledown f_{x_{i_t}, y_{i_t}}(\hat{w}) + \frac{1}{n}\sum_{i=1}^{n} \bigtriangledown f_{x_i, y_i}(\hat{w}))||_2^2] \tag{34}$$
$$= ||w_t + w^*||_2^2 \tag{35}$$
$$+ \alpha^2 Var[\bigtriangledown f_t(w_t) - \bigtriangledown f_t(\hat{w}) + \sum_{i=1}^{n} \frac{\bigtriangledown f_i(\hat{w}))}{n}] \tag{36}$$
$$+ \alpha^2 ||\frac{1}{n}\sum_{i=1}^{n} \bigtriangledown f_{x_i, y_i}(w_t))||_2^2 \tag{37}$$
$$- 2\alpha(w_t - w^*)\frac{1}{n}\sum_{i=1}^{n} \bigtriangledown f_{x_i, y_i}(w_t)) \tag{38}$$
$$\leq ||w_t + w^*||_2^2 \tag{39}$$
$$+ 2\alpha^2 L^2||w_t - w^*|| + 2\alpha^2 L^2||\hat{w} - w^*||_2^2 \tag{40}$$
$$\tag{41}$$
$$+ \alpha^2 L^2||w_t - w^*||_2^2 \tag{42}$$
$$- 4\alpha^2||w_t - w^*||_2^2 \tag{43}$$
$$= (1 - 4\alpha + 3\alpha^2 L^2)||w_t - w^*||_2^2 \tag{44}$$
$$+ 2\alpha^2 L^2||\hat{w} - w^*||_2^2 \tag{45}$$

We see two terms in the bound, one is dependent on w_t and the other is dependent on $\hat{w}$. We are interested in the term containing $\hat{w}$, specifically, the converge rate of the term $2\alpha^2 L^2||\hat{w} - w^*||_2^2$. In order for the series to converge, the number of iteration we need so that the optimization is guaranteed to have less error than ϵ is (Sa, 2017):

$$\frac{5L^2 e}{4} log(\frac{||\hat{w}_0 - w^*||^2}{\epsilon}) \in O(log(\frac{1}{\epsilon})) \tag{46}$$

4.2. Counterexample

Since the derivation of bound for SVRG requires the loss function f begin L-lipschitz, once we remove this conditional it would be impossible to derive a bound for gradient update steps or iteration guarantee in the form of $\frac{5L^2 e}{\mu^2} log(\frac{||\hat{w}_0 - w^*||^2}{\epsilon})$.

5. Empirical Studies

5.1. Convex Analysis

To discover the performance of SVRG in convex optimization, compared to that of fixed-learning-rate SGD's, L2-regrularized logistic regression with cross-entropy loss is performed. The regularization parameter is $\lambda = 0.0001$, and the total number of gradient operations for both algorithms is fixed to 10000. We used the same training dataset as Johnson and Zhang's (Johnson & Zhang, 2013). Data is retrieved from Python package tensorflow.examples.tutorials.mnist in version 1.14.0 and directly used in analysis with no preprocessing procedures.

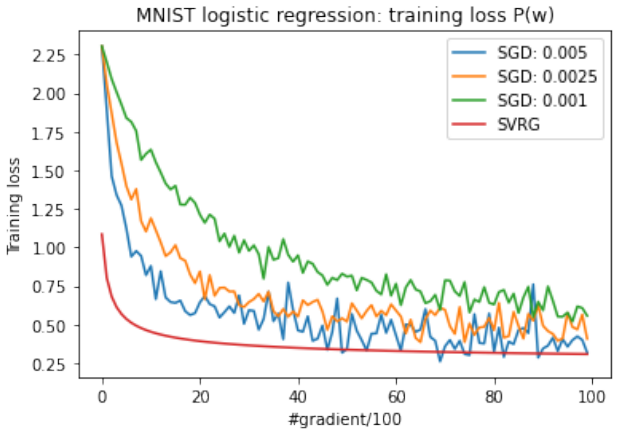

Figure 1.

For Figure 1., we plot the cross-entropy training loss against gradient numbers. In order to observe the trend of training loss better, we only plot the training loss once every 100 gradient operations to prevent the plot from becoming too dense. For SGD algorithms with different learning rates, the convergence speed is negatively correlated to the learning rate, so the decreasing speed of training loss is slower for SGD with smaller learning rate. However, after approximately 6000 iterations, we no longer see significant difference between

the training loss for SGD algorithms with different learning rates since all training loss oscillate between the interval $[0.25, 0.75]$. This confirms the disadvantage of fix-learning-rate SGD algorithms where convergency ends up towards a fix-size stochastic ball (Gower et al., 2020). On the other hand, convergence rate of SVRG algorithm exceeds that of all SGD's, even when the learning rate of SVRG is greater than the leaning rates of SGDs. In addition, the training loss curve for SVRG is smooth, indicating that the training loss continues decreasing with almost no oscillation as iterations increases, which is useful when we need approximation with particularly low training loss. When compared with the original results from Johnson and Zhang (Johnson & Zhang, 2013), the behavior of training loss is similar; however, the training loss in original results is lower, which we suspect may be due to different initialization strategies and different pre-processing procedures.

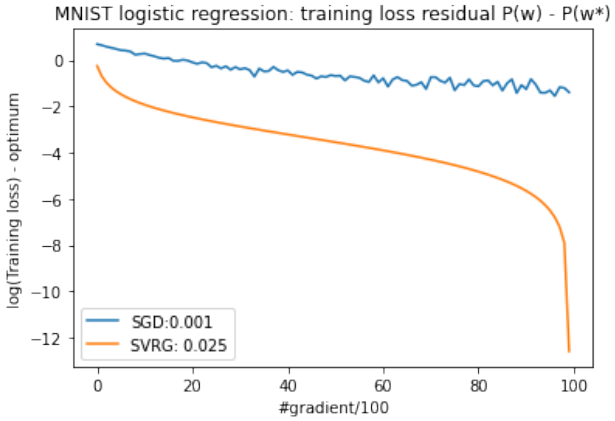

Figure 2.

For Figure. 2, we plot the logarithm of training loss residual against gradient numbers for both SGD with learning rate of 0.001 and SVRG with learning rate of 0.025. Note that we compute the optimal gradient by running 1000 iterations in SVRG algorithm, in other words, the number of gradient update operations is 100000. Due to the limitation in computation power of our devices, the estimated optimal weight w^* is fairly close to the weight estimates, especially for the weight estimates after approximately 6000 iterations. Such proximity is then amplified by the logarithm operation, causing large fluctuations in the right-half of the figure, thus causing the training loss residuals for large gradient numbers to become unreliable. If we focus on the training loss residuals for small gradient numbers, we will observe that the logarithm of training loss residual roughly forms a straight line, indicating that the training loss residual for both SGD and SVRG decreases exponentially. In addition, since the slope of the line for SVRG is slightly steeper than that if SGD, we assert that SVRG converges faster than SGD. The results in this graph is similar to that of Johnson, R. and Zhang, T's (Johnson & Zhang, 2013).

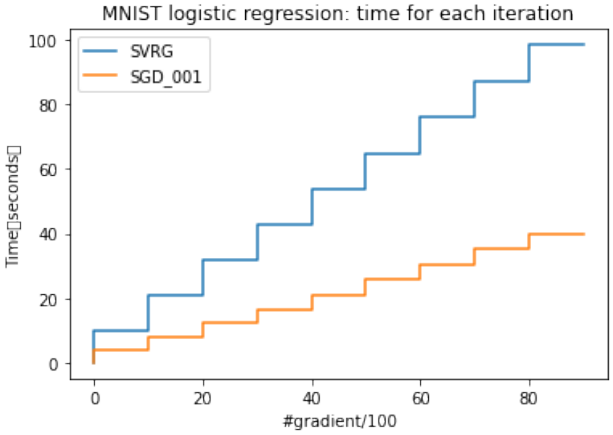

Figure 3.

For Figure 3., we use a step diagram to demonstrate the time consumption of gradient update operations for different algorithms. For demonstration purpose, each step increase represents the time consumption of 100 gradient update operations. We observe that each gradients update operation for SVRG algorithm roughly takes twice as long as SGD's, which is reasonable since we need to calculate two different gradient values when calculating the gradient approximation in one SVRG iteration. Combine the result in this plot with the result in Figure.1, we can confirm that there is a trade-off between convergence speed and computation time per iteration (Johnson & Zhang, 2013).

5.2. Non-convex Analysis

To evaluate the performance of SVRG in nonconvex optimization, compared to that of fixed-learning-rate SGD's, we used the PyTorch classes from GitHub repository (Fatras, 2018) to conduct neural network analysis with two convolution layers. The regularization parameter is $\lambda = 0.01$, and the total number of gradient calculation operations for both algorithms is fixed to 25000. We used the same training dataset CIFAR as Johnson and Zhang's (Johnson & Zhang, 2013). Data is retrieved from Python package torchvision.datasets and used in analysis after normalization.

For Figure 4., we plot the cross-entropy loss training loss against every 1000 iterations. We observe that after 25000 iterations, the weight approximations of SGD and SVRG is similar, however the training loss curve of SGD algorithm experiences large oscillations, while the training loss curve of SGD algorithm remains smooth. This confirms the fact that the gradient approximation of SVRG algorithm has

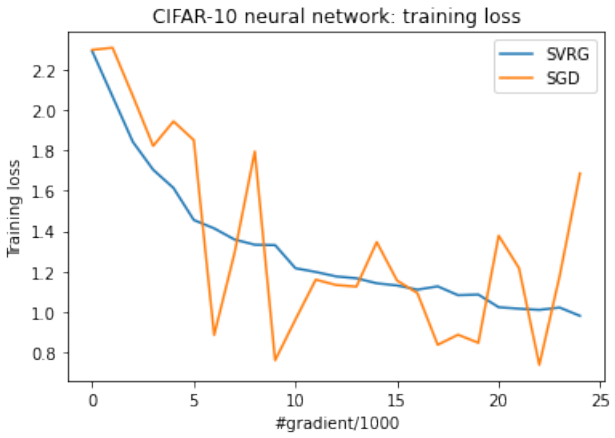

Figure 4.

smaller variance when being compared to SGD algorithm.

6. Limitations

One of the limitations of SVRG algorithm is that since the algorithm requires we update the estimated optimal weight $\hat{w}$ every few iterations, it would be hard to develop an online version of SVRG that could constantly optimize weights when given a series of $\{x_t, y_t | t \in N\}$. I think it is still possible to come up with an online version of SVRG, however it would be hard to tune hyperparameters such as inner loop size. In contrast, the structure of SDCA algorithm is more suitable for stream learning.

Although empirical results show that SVRG and SDCA algorithm demonstrate fairly good performance in non-convex optimization problems such as finding weights in neural network analysis (Johnson & Zhang, 2013), there is no explicit form of convergence rate. It would be reasonable to assume that these variance reduction algorithms may not perform as well in other forms of non-convex optimization problems such as leaning a convex basis model ensembles (Nguyen, 2020).

7. Conclusion and Future Work

We have reviewed the SGD method and briefly analyzed and evaluated two new variant techniques for optimizing the data set - SDCA and SVRG. We have used the graphs to show the efficiency of each methods and have mentioned the suitable cases for each method. For smooth and strongly convex functions, SVRG method has the same fast convergence rate as the SGD and SDCA, and it requires less memory space compared with SDCA method; therefore, it is more suitable for some complex high-dimensional problems such as neural network learning; whereas, SDCA is usually used in relatively smaller scale supervised learning problems but the function can be less convex.

In the last decade of machine learning, many algorithms and theories are evolving, and the gradient descent family has produced various variants and branches, such as SAG (stochastic average gradient), all of which have good optimization results in specific cases. However, no method is perfect, they all have more or less their own drawbacks. Therefore, we believe that while we keep studying gradient descent, a good trade-off theory should also be established, through which we can effectively balance the variance of data, memory usage and the running time. Only in this way, we can more efficiently and directly choose a suitable optimization method instead of using multiple gradient descent methods and comparing their results at the end.

References

Fatras, K. Stochastic variance reduced algorithms : Implementation of svrg and saga optimization algorithms for deep learning. https://github.com/kilianFatras/variance_reduced_neural_networks, 2018.

Gower, R. M., Schmidt, M., Bach, F., and Richtárik, P. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

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

Nguyen, T. T. *Selected non-convex optimization problems in machine learning*. PhD thesis, Queensland University of Technology, 2020.

Pedergosa, F. The stochastic gradient method. *Google AI*, 2018.

Sa, C. Online vs. offline learning, variance reduction, and svrg. https://www.cs.cornell.edu/courses/cs6787/2017fa/Lecture5.pdf, 2017.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(2), 2013.

Tran, K., Hosseini, S., Xiao, L., Finley, T., and Bilenko, M. Scaling up stochastic dual coordinate ascent. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1185–1194, 2015.