Speed Maintained SVRG

Erxue Min
National University of Defense Technology
Email: mex338@qq.com

Yawei Zhao
National University of Defense Technology
Email: zhaoyawei09@163.com

Jun Long
National University of Defense Technology
Email: 14241368@qq.com

***Abstract*—Stochastic gradient descent (SGD) is widely used for large-scale machine learning optimization, but has slow convergence rate due to the highly inherent variance. In recent years, the popular Stochastic Variance Reduced Gradient (SVRG) method mitigates this shortcoming, through computing the full-gradient of the entire dataset occasionally. However, conventional SVRG and its variants usually need a hyper-parameter to identify when to compute such the full gradient, which is essential to the convergence performance. Few previous studies discuss the method to identify such the hyper-parameter, which makes it hard to gain a good convergence performance in practical machine learning tasks. In our paper, we propose a new stochastic gradient descent with variance reduction technique named SMSVRG which computes the full gradient adaptively. Moreover, we propose an improved method denoted by SMSVRG+, which is comparable to and even better than SVRG with best-tuned epoch sizes for smooth and strongly convex functions.**

I. Introduction

Many machine learning tasks such as logistic regression and linear regression can be formulated to be an optimization problem which is described as

$$\min F(\omega), \qquad F(\omega) = \frac{1}{n}\sum_{i=1}^{n} f_i(\omega) + R(\omega). \tag{1}$$

where n is the size of the training data. $F(\omega)$ means the loss function or training loss of a machine model, and ω is its parameter. $R(\omega)$ is the regularizer, which is widely used to avoid overfitting. It is noting that the total number of instances, i.e. n becomes very large with the proliferation of data.

Gradient Descent (GD) is a basic method to solve such the optimization problem. The gradient of $F(\omega)$ is obtained by passing over the entire training data, which is extremely time-consuming when the size of training data, i.e. n becomes large. Besides, GD is an iterative-convergent algorithm, that is, the parameter, i.e. ω, usually needs thousands of iterations to be converged. Since GD needs to compute the gradient of $F(\omega)$ every iteration, when the volume of data is large, the computation cost increases sharply and impairs the convergent performance significantly.

Stochastic Gradient Descent (SGD) mitigates this shortcoming by replacing the calculation of $\nabla F(\omega)$ with a stochastic gradient $\nabla f_i(\omega)$ with $i \in \{1, 2, ..., n\}$. In SGD, i is selected randomly from the entire training data. Thus, SGD outperforms GD on the time efficiency significantly. Take the expectation of i, we obtain $\mathbb{E}[\nabla f_i(\omega)] = \nabla F(\omega)$. The difference between $\nabla f_i(\omega)$ and $\nabla F(\omega)$ represents variance which makes it difficult to achieve the optimum. In order to make the loss function, i.e. $F(\omega)$ converge, a decaying learning rate is usually used to reduce the variance. However, value of the learning rate is decayed to be very small after hundreds of iterations, which impedes the loss function to converge. In a nutshell, SGD with a decaying learning rate incurs a sub-linear convergence rate.

In recent years, variance reduced variants of SGD such as SVRG [4] is proposed to reduce the variance and gain the linear convergence performance with a constant learning rate. In SVRG, a full gradient is computed occasionally during the inexpensive SGD steps to reduce the variance, dividing the optimization procedure into many epochs. On the basis of SVRG, many variants have been proposed to improve its performance. SVRG-BB [12] uses the Barzilai and Borwein (BB) method proposed by Barzilai and Borwein in [2] to compute the step size before every epoch, which generally achieves the comparable convergence performance to SVRG with the best-tuned step size. CHEAPSVRG [10] and SAMPLEVR aim at reducing the expensive cost of full gradient computation through using a surrogate with a subset of the training dataset. mS2GD [6] uses mini-batch method to obtain a full gradient to reduce the variance, which shows a clear advantage for parallel computation. EMGD [14], SVR-GHT [8], Prox-SVRG [13] and SVRG with second-order information [5] modify the update rule of stochastic steps, and show advantages to SVRG in some cases. However, there are few studies discussing about how frequently should a full gradient be computed, i.e. how to set the epoch size m.

Most previous researches present that the epoch size, i.e. m should be constant [4, 12, 10] or increased monotonically [6], regardless of the learning rate. It is recommended that $m = 2n$ for convex problems and $m = 5n$ for non-convex problems in SVRG, without theoretical analysis and further experimental verification.

The epoch size, i.e. m has a great impact on the convergence performance of SVRG. More specifically, when m is too small, it wastes too much time to compute the full gradient frequently. When m is rather large, the variance between the stochastic gradient and the full gradient increases sharply, making the convergence of training loss extremely difficult. According to the analysis of variance in [YaWei], both the epoch size, i.e. m and learning rate, i.e. η have a significant impact on the convergence performance. However, those previous studies do not provide a practical method to set the value of those hyper-

parameters. Extensive empirical studies illustrates that the choice of good value for those hyper-parameters costs much time in real machine learning tasks. In this paper, we develop a novel algorithm denoted by SMSVRG which can adjust the epoch size adaptively. SMSVRG applies a new stop condition regarding the change of parameters and checks the condition at the same interval, if the condition is satisfied, we stop the current epoch and step into the next one. Besides, we give guidance of how to set the parameters. Since SMSVRG may stop iterations earlier than expectation when η is quite small, we propose an improved algorithm denoted by SMSVRG+. In a nutshell, our contributions are highlighted as follows:

- SMSVRG, this novel algorithm can adjust the epoch size to a suitable value dynamically.
- SMSVRG+, it is an improvement of SMSVRG which is not sensitive to parameters and more practical than SMSVRG.
- Extensive empirical studies shows the effectiveness of our proposed algorithms which outperform their countparts on the convergence performance significantly.

This paper is organized as follows: Section II reviews the related work. Section III describes the SVRG algorithm. Section IV presents the new variant of SVRG, i.e. SMSVRG and its improved method, i.e. SMSVRG+. Section V demonstrates the numerical results of our algorithm. Section VI concludes this paper.

II. Related work

Several variants of SVRG discussing the strategy of adjusting epoch size has been proposed, including SVRG++ [1], S2GD [7], SVRG_Auto_Epoch [1] and so on.

SVRG++ adopts a simple strategy that epoch size m doubles between every consecutive two epochs. This method is absolutely heuristic and sometimes not justified. Our experiments show that when η is large or moderate, the exponential growth of m will incur great variance and impairs convergence.

S2GD designs a probability model of m and shows that a large epoch size is used with a high probability. However it needs to know the lower bound on the strong convexity constant of F, which is hard to estimate in practice. Meanwhile, the maximum of stochastic steps per epoch is also a sensitive parameter.

SVRG_Auto_Epoch is introduced as an additional improvement of SVRG++. It determines the termination of epoch through the quality of the snapshot full gradient. It records $diff_t = \|\nabla f_i(\omega_t^s) - \nabla f_i(\tilde{\omega}^{s-1})\|$ every iteration t and uses it as a tight upper bound on the variance of the gradient estimator. Although this method is reasonable, it has too much parameters to tune. Moreover, it takes much additional computation for every iterations, which impairs performances significantly.

Comparing with the above methods, SMSVRG is apparently reasonable in intuition and does not need to tune extra parameters. Besides, it takes little additional computation cost and outperform the aforementioned three methods.

Algorithm 1 SVRG

Require: learning rate η, epoch size m, initial point $\tilde{\omega}$

for $s = 0, 1, ..., m$ **do**
  $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_s)$
  $\omega_0 = \tilde{\omega}_s$
  for $t = 0, 1, 2, ..$ **do**
    Randomly pick $i_t \in \{1, 2, ..., n\}$
    $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
  end for
  Option I: $\tilde{\omega}_{s+1} = \omega_m$
  Option II: $\tilde{\omega}_{s+1} = \omega_t$ for randomly chosen $t \in \{0, ..., m-1\}$
end for

III. Overview

In this section we review the SVRG algorithm proposed by Johnson and Zhang[4]. As is shown in Algorithm 1, there are two loops in SVRG. Each outer loop is called an $epoch$ while each inner loop is called an $iteration$. In each outer loop, a full gradient $\tilde{\mu}$ is computed at first, and its computation requires one pass over the dataset using $\tilde{\omega}$. In each inner loop, a generalized SGD is conducted: randomly pick $i_t \in \{1, 2, ..., n\}$:

$$\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu}) \tag{2}$$

Note that the expectation of $\nabla f_{i_t}(\tilde{\omega}_s) - \tilde{\mu}$ over i is zero, and the expectation of $\nabla f_{i_t}(\omega_{t-1})$ over i is $\nabla F(\omega_{t-1})$, thus

$$\mathbb{E}[\omega_t | \omega_{t-1}] = \omega_{t-1} - \eta \nabla F(\omega_{t-1})$$

We can be seen that the variance of the update rule (2) is reduced, because when both $\tilde{\omega}$ and ω_t converge to the optimum ω^*, then $\tilde{\mu} \to 0$ and $\nabla f_{i_t}(\omega_{t-1}) \to \nabla f_{i_t}(\tilde{\omega}_s)$, therefore

$$\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu} \to 0$$

Hence, the learning rate for SVRG is allowed to be set as a relatively large constant, which results in high convergence rate. At the end of each epoch, $\tilde{\omega}_{s+1}$ is updated by the output of inner loop. Note that there are two options for the update. Although only the convergence of SVRG with Option I is analyzed in [4], SVRG with Option II has been confirmed numerically to perform better. We adopt Option II in this paper. It is obvious that when m is too large, the SGD with variance reducer will degenerate to native SGD, which results in huge variance. Thus our work focus on how to set a appropriate epoch size.

IV. Speed Maintained SVRG

In this section we describe two novel algorithms: SMSVRG and SMSVRG+, which can set the epoch size adaptively and has superior convergence properties in out experiments. We assume the loss function F and the component functions f_i are convex and L-smooth throughout the paper.

A. smSVRG

1) Idea: It has been proved in [YaWei] that the optimal m is strongly related to η. Our experiments also show that when η is large, the training loss begins to fluctuate after merely a small number of iterations. In the other hands, the algorithm can endure far more than n iterations with a small η. Ideally, if we stop the iterations in one epoch just before the training loss begins to fluctuate, the algorithm will certainly be very efficient and outperforms SVRG with constant epoch size. A direct approach is computing the training loss occasionally. However, the training loss computation requires passing over the entire dataset, which is rather time consuming.

When we apply gradient descent to convex problem, as the ω_t gradually approach the optimal value ω^*, thus the gradient $\nabla F(\omega_t)$ keeps decreasing. We know that

$$\omega_t - \omega_{t-1} = -\eta \nabla F(\omega_{t-1})$$

Then we have $\|\omega_{t+1} - \omega_t\| < \|\omega_t - \omega_{t-1}\|$. When it comes to SVRG, for each iteration in one epoch, we can obtain

$$\begin{aligned} \mathbb{E}[\omega_t - \omega_{t-1}] &= \mathbb{E}[-\eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})] \\ &= -\eta \mathbb{E}[\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu}] \\ &= -\eta \nabla F(\omega_{t-1}) \end{aligned} \tag{3}$$

Intuitively, if ω_t keeps approaching the optimal value ω^*, the $\nabla F(\omega_t)$ decays generally, with slight fluctuation caused by variance. Thus if we consider a window, we can believe that $\|\omega_{t+m_0} - \omega_t\| < \|\omega_t - \omega_{t-m_0}\|$ holds to a certain degree, with the confidence proportional to the window size m_0. On the other hand, when ω_t shakes near the optimal value due to the variance, $\|\omega_{t+m_0} - \omega_t\|$ will keep fluctuating.

Inspired by this, SMSVRG set

$$\|\omega_{t+m_0} - \omega_t\| > \|\omega_t - \omega_{t-m_0}\| \tag{4}$$

as a stop condition in each epoch. When the inequality holds, we deem that the training loss fails to converge and begins to fluctuate, so we have to suspend SGD iterations and compute a full gradient to reduce the variance.

2) Details: As illustrated in Algorithm 2, SMSVRG just requires two parameters: learning rate η and window size m_0. It differs from SVRG only in the inner loop. At line 5, the first condition $t\%m_0 = 0$ means that we check Inequality 4 every m_0 iterations. The second condition $t >= 2m_0$ is obvious as we need at least two windows to check the Inequality. The third one is our stop condition. If the condition holds, we break the inner loop and step into the next epoch. Note that the epoch size is definitely a multiple of m_0, and its minimum is $2m_0$. Hence we should set m_0 to be smaller than $0.5n$ for flexibility. At the same time, m_0 cannot be too smaller as the confidence of condition will be rather low. We recommend to set m_0 between $0.1n$ and $0.2n$.

3) Advantages:

- SMSVRG is superior to SVRG with prefixed epoch size as it can adapt the epoch size to a appropriate value dynamically. For large η, SMSVRG will adjust the epoch size to be relatively small to constrain variance. On the other hand, when η is small thus each epoch can contain more iterations. SMSVRG will make the epoch size larger than $2n$ to avoid computing the time-consuming full gradient fequently.
- Comparing with SVRG_Auto_Epoch and S2GD, SMSVRG requires far less additional computation cost as it just computes a simple inequality every m_0 iterations, which is very efficient.

B. Optimal Choice of checking interval

In SMSVRG, we use $\|\omega_t - \omega_{t-m_0}\|$ to detect when training loss begins to fluctuate and fails to converge. However, how to choose a suitable value of window size i.e. m_0 becomes an important issue. In this section, we analyze the effects of setting different m_0 and provide guidance on setting a optimal value.

First, as the variance incurred by SGD iterations cannot be ignored, when we set m_0 to be small, the variance of $\|\omega_t - \omega_{t-m_0}\|$ is too large thus the confidence of the inequality is low. As a result, the inequality may hold due to variance when the loss function is still decreasing rapid, which wastes much time on computing full gradients. We use $C(m_0)$ to denote the confidence of inequality holds. It is obvious that $C(m_0)$ is a monotonically increasing function of m_0.

If we set m_0 to be relatively large, the variance becomes small and the confidence of inequality holds is high. However, it will be too late to detect the fluctuation of training loss and waste much time to compute iterations which make no process for optimizing the objective function. We use $D_s(m_0)$ to denote the $AbsoluteDelay$ of detecting fluctuation. Furthermore, the $AbsoluteDelay$ makes different effects depending on the epoch size, so it is better to consider the $RelativeDelay$:

$$D_r(m_0) = \frac{D_s(m_0)}{es + n}$$

Note that es denote the number of iterations in a specific epoch and n denotes the size of the dataset. Function $D_r(m_0)$ is also monotonically increasing with respect to m_0.

It is natural that we want to choice a suitable m_0 which can achieve a large $C(m_0)$ and a small $D_r(m_0)$. In order to obtain a trade-off between $C(m_0)$ and $D_r(m_0)$, we convert the parameter choosing problem to the following maximization problem:

$$\begin{aligned} m_0 &= \max_{m_0} C(m_0) - D_r(m_0) \\ &= \max_{m_0} C(m_0) - \frac{D_s(m_0)}{es + n} \\ &\text{s.t.} 0 < m_0 < n \end{aligned} \tag{5}$$

Note that we do not have the exact expression of C and D_s, instead, we only know their monotonicity. Nonetheless, it is enough to guide us to set the parameter m_0. From equation (5) we can obtain the following conclusions:

1) When epoch size es is small, the D_s tends to be more important than C. Hence we should set m_0 to be relatively small to maximize the objective function. On the contrary, it is recommended to set m_0 to be large.

Algorithm 2 SMSVRG

Require: learning rate η, window size m_0, initial point $\tilde{\omega}$

for $s = 0, 1, ...$ **do**
  $\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_s)$
  $\omega_0 = \tilde{\omega}_s$
  for $t = 0, 1, 2, ..$ **do**
    if $t\%m_0 = 0$ and $t >= 2m_0$ and $\|\omega_t - \omega_{t-m_0}\| > \|\omega_{t-m_0} - \omega_{t-2m_0}\|$ **then**
      break
    end if
    Randomly pick $i_t \in \{1, 2, ..., n\}$
    $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
  end for
  $\tilde{\omega}_{s+1} = \omega_t$
end for
return $\tilde{\omega}_{s+1}$

In spirit of this, we can set m_0 to be proportional to the iteration number of previous epoch.

2) According to our experiment on SVRG, when the learning rate η is large, the loss function begins to fluctuate after merely $n/10$ iterations, so we should set the initial m_0 to the same order of magnitude as $10^{-1} \times n$

C. SMSVRG+

1) Idea: Our experiments on SMSVRG show that it performs well for large and moderate η. However, when η is rather small, the best epoch size for SVRG could be larger than $10n$. And SMSVRG may finish the epoch prematurely while the training loss keeps declining. It is because the inequality (4) may holds due to variance, rather than the fluctuation of training loss. According to the analysis in IV-B, variance of $\|\omega_{t+m_0} - \omega_t\|$ has more impact on the performance of our algorithm when the best epoch size is large. Intuitively, we should adjust the window size m_0 of next epoch according to current epoch size, instead of a constant value.

2) Details: As illustrated in Algorithm 3, we recompute m_0 after the inner loop in each epoch. m_0 is set to be $(int(es/n)+1) * (0.1n)$, where es denotes the iteration number of current epoch size. It means that m_0 is incremented by $0.1n$ when es exceeds $n, 2n, 3n$ and so on.

3) Advantages:

- SMSVRG+ outperforms SMSVRG as it will enlarge the window size to reduce variance when necessary, avoiding early stopping of epochs. It shows good performance regardless the learning rate and datasets in our experiments.
- SMSVRG+ is not sensitive to the initialized m_0, as it adapts m_0 to an appropriate size rapidly. On the contrary, the performance of SMSVRG fairly depends on the choice of m_0.

V. Numerical Experiments

In this section, we conduct some experiments to demonstrate the efficiency of our proposed algorithm. We evaluate our algorithm on eight training datasets, which are public on the LIBSVM website[1]. In our experiments, algorithms are applied for two standard machine learning tasks: $l2$-regularized logistic regression and $l2$-regularized ridge regression.

Algorithm 3 SMSVRG+

Require: learning rate η, window size m_0, initial point $\tilde{\omega}$

for $s = 0, 1, ...$ **do**
  $\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_s)$
  $\omega_0 = \tilde{\omega}_s$
  for $t = 0, 1, 2, ..$ **do**
    if $t > m_0$ and $t\%m_0 = 0$ and $\|\omega_t - \omega_{t-m_0}\| > \|\omega_{t-m_0} - \omega_{t-2m_0}\|$ **then**
      break
    end if
    Randomly pick $i_t \in \{1, 2, ..., n\}$
    $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
  end for
  $es = t$
  $m_0 = (int(es/n) + 1) * (0.1n)$
  $\tilde{\omega}_{s+1} = \omega_t$
end for
return $\tilde{\omega}_{s+1}$

The $l2$-regularized logistic regression task is conducted on the two datasets: ijcnn1, a9a, w8a and mushrooms. Since the label of each instance in these datasets is set to be 1 or -1, the loss function of $l2$-regularized logistic regression task is:

$$\min_{\omega} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i \omega^{\mathrm{T}} x_i}) + \lambda \parallel \omega \parallel^2 . \tag{6}$$

The $l2$-regularized ridge regression task is conducted on the four datasets: YearPredictMSD, cadata, mg, abalone. The loss function of $l2$-regularized ridge regression task is:

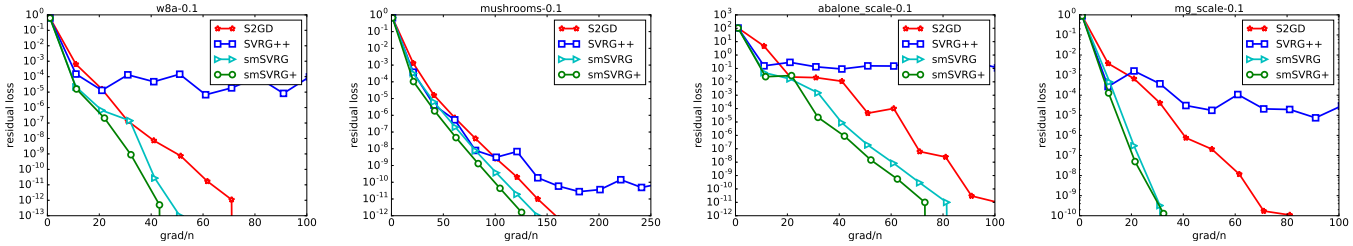
$$\min_{\omega} \frac{1}{n}\sum_{i=1}^{n} \left(\omega^{\mathrm{T}} x_i - y_i\right)^2 + \lambda \parallel \omega \parallel^2 . \tag{7}$$

We scale the value of all features to $[-1, 1]$ and set the weighting parameter λ to 10^{-4} for all evaluations. In all figures, the x-axis denotes the computational cost, which is measured by the number of gradient computation divided by the size of training data, i.e. n. The y-axis denotes training loss residual, i.e. $F(\tilde{\omega}_s) - F(\omega^*)$. Note that the optimum ω^* is estimated by running the gradient descent for a long time. Our numerical experiments include three parts: comparing with existing related methods, comparing with SVRG by varying learning rate and sensitivity test on m_0. The experiments show the superior performance of our methods.

A. Comparing with existing related methods

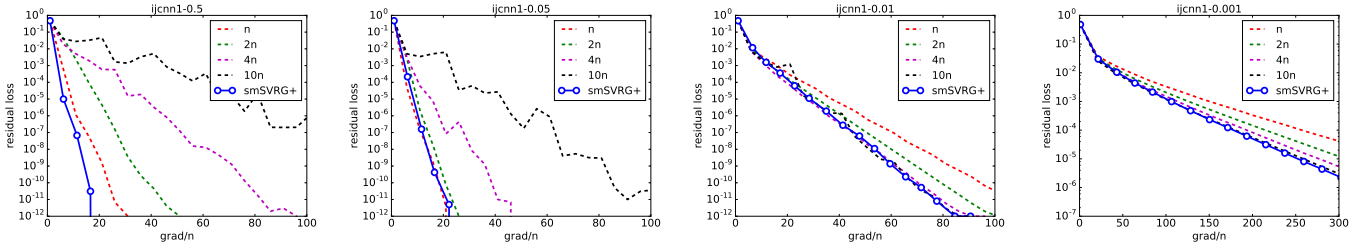
In this section, we compare our SMSVRG and SMSVRG+ with two aforementioned existing methods: SVRG++ and S2GD. We do not compare with SVRG_Auto_Epoch in that

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/



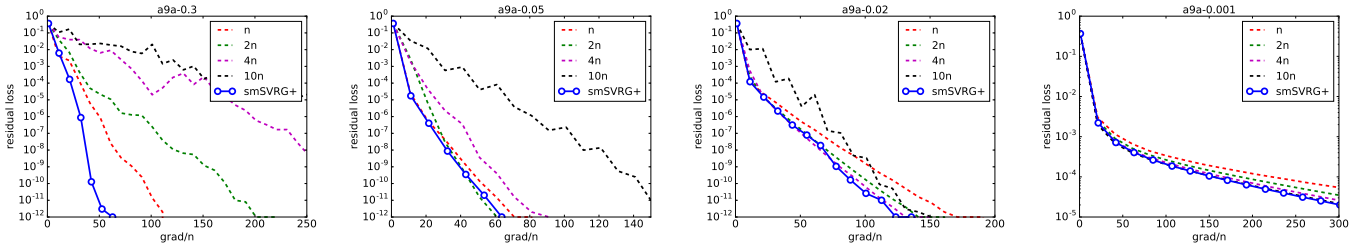
(a) w8a $\eta = 0.1$ (b) mushrooms $\eta = 0.1$ (c) abalone $\eta = 0.1$ (d) mg $\eta = 0.1$

Fig. 1. Comparison of SMSVRG, SMSVRG+, SVRG++, S2GD on four datasets.



(a) ijcnn1 $\eta = 0.5$ (b) ijcnn1 $\eta = 0.05$ (c) ijcnn1 $\eta = 0.01$ (d) ijcnn1 $\eta = 0.001$

Fig. 2. Generally, SMSVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized logistic regression



(a) a9a $\eta = 0.3$ (b) a9a $\eta = 0.05$ (c) a9a $\eta = 0.02$ (d) a9a $\eta = 0.001$

Fig. 3. Generally, SMSVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized logistic regression

TABLE I
DETAIL INFORMATION OF DATASETS AND MODELS

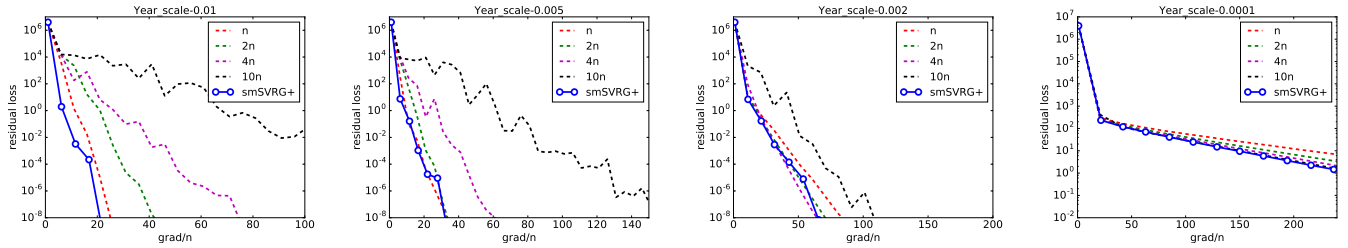
Dataset	size	dimension	model	λ
ijcnn1	49990	22	logistic	10^{-4}
a9a	32561	123	logistic	10^{-4}
YearPredictionMSD	463715	90	linear	10^{-4}
cadata	20640	8	linear	10^{-4}

we find its termination condition of epoch is never satisfied and keeps doing SGD iteration, resulting in nonconvegence. For SVRG++, we initialize $m = n$. For S2GD, we set the maximum of m to be $4n$. For both SMSVRG and SMSVRG+, we set the window size, i.e. m_0 to be $0.1n$. We evaluate these methods by running logistic regression on dataset ijcnn1. As illustrated in Figure 1, we can see that SVRG++ always fluctuates violently and fails to converge due the large variance caused by too large epoch size. It is shown that SMSVRG and SMSVRG+ always outperform SVRG++ and S2GD and converge rapidly. Besides, the performance of SMSVRG+ is superior to that of SMSVRG. The main reason is that SMSVRG+ can adjust the windows size to a suitable value adaptively.

B. Comparing with SVRG by varying learning rate

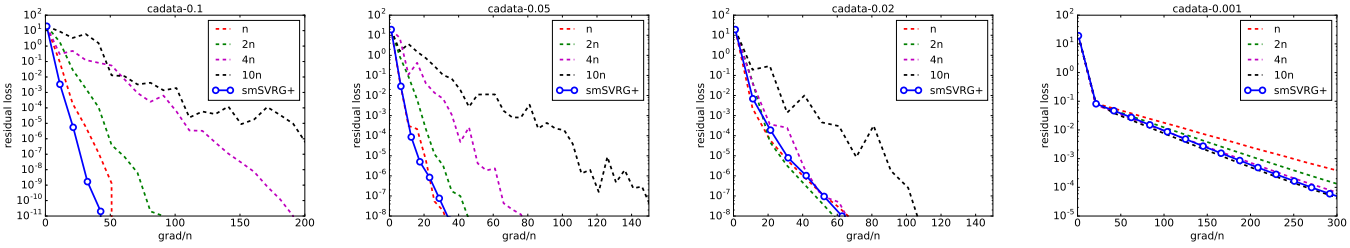
Since SMSVRG+ performs better than SMSVRG, only SMSVRG+ is used to conduct the comparison with SVRG. For SVRG, we increase the epoch size, i.e. m as four different values: n, $2n$, $4n$, $10n$. The first three values are chosen because it is recommended to choose m to be the same order of n but slightly larger[4]. Besides, our experiments show that epoch size larger than $10n$ have similar performance as the extremely large epoch size decrease the impact of full gradient computation. In all figures, the dashed lines represents SVRG with fixed epoch size; while the blue solid lines with markers stand for SMSVRG+.

As illustrated in Figures 2, 3, 4, 5, SMSVRG+ can always have the similar performance as SVRG with most best-tuned epoch size. We observe that when η is large, and m is set to be a small value, e.g. n, can achieve best performance. The main reason is that when η is large, the variance becomes



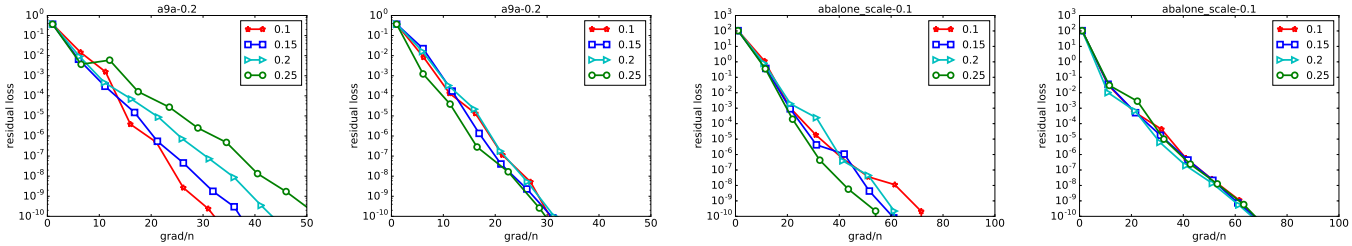
(a) YearPredictionMSD $\eta = 0.01$ (b) YearPredictionMSD $\eta = 0.005$ (c) YearPredictionMSD $\eta = 0.002$ (d) YearPredictionMSD $\eta = 0.0001$

Fig. 4. Generally, SMSVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized linear regression



(a) cadata $\eta = 0.1$ (b) cadata $\eta = 0.05$ (c) cadata $\eta = 0.02$ (d) cadata $\eta = 0.001$

Fig. 5. Generally, SMSVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized linear regression



(a) SMSVRG on a9a (b) SMSVRG+ on a9a (c) SMSVRG on abalone (d) SMSVRG+ on abalone

Fig. 6. Sensitivity test on m_0 for SMSVRG and SMSVRG+. The four numbers on the legends means four different choices of m_0.

significant simultaneously, so m must be set to be small in order to bound the variance. As η decays, the optimal value of m increases, which means that the algorithm can tolerate more variance induced by extra iterations. As illustrated in Figures 2(a), 3(a), 4(a), 5(a), our method is comparable to and sometimes even better than SVRG with best-tuned epoch sizes when learning rate is large or medium. However, As illustrated in Figures 2(d), 3(d), 4(d), 5(d), if η is set to be too small, SMSVRG+ performs slightly inferior to SVRG with large epoch sizes, but outperforms SVRG with recommended epoch sizes, i.e. n and $2n$. It is noting that setting η to be too small is not a practical approach when using SVRG or its variants, because the convergence rate will be extremely low. Therefore, the sub-optimal performance of SMSVRG+ with rather small η is acceptable.

C. Sensitivity test on m_0

In this section, we conduct the sensitivity test on both SMSVRG and SMSVRG+ regarding window size m_0. As analyzed in IV-B, we should choose the initial m_0 to be the same order of $0.1n$. Thus we choose the following four values: $0.1, 0.15, 0.2, 0.25$ to test the sensitivity of SMSVRG and SMSVRG+ on m_0. We conduct the experiments on two datasets: a9a and abalone. As illustrated in 6(a) and 6(c), the performance of SMSVRG varies with the m_0 significantly. And we can see in 6(b) and 6(d) that the performance of SMSVRG+ is not sensitive to the choice of m_0. The main reason is that SMSVRG+ will adjust the m_0 adaptively regardless of the initialization. Hence SMSVRG+ is more practical than SMSVRG in reality.

VI. Conclusion

In this paper we propose a novel stop condition for each epoch in SVRG, leading to a new variant of SVRG: SMSVRG, which can adjust the epoch size adaptively. We analyze how to choose the optimal value of parameters and developed an improved method called SMSVRG+. We conduct numerical experiments on real datasets to demonstrate the performance of SMSVRG and SMSVRG+. The experiments show that both SMSVRG and SMSVRG+ are superior to existing methods.

Moreover, the SMSVRG+ is comparable to and sometimes even better than the original SVRG with best-tuned epoch sizes.

REFERENCES

[1] Zeyuan Allen-Zhu and Yang Yuan. Improved SVRG for non-strongly-convex or sum-of-non-convex objectives. In International Conference on Machine Learning, New York, USA, June 2016.

[2] Jonathan Barzilai and Jonathan M. Borwein. Two-point step size gradient methods. IMA Journal of Numerical Analysis, 8(1):141–148, 1988.

[3] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Advances in Neural Information Processing Systems, pages 1646–1654, Montréal, Canda, December 2014.

[4] R Johnson and T Zhang. Accelerating stochastic gradient descent using predictive variance reduction. Advances in Neural Information Processing Systems, pages 315–323, December 2013.

[5] Ritesh Kolte and Murat Erdogdu. Accelerating svrg via second-order information. 2016.

[6] Jakub Konečnỳ, Jie Liu, Peter Richtárik, and Martin Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. IEEE Journal of Selected Topics in Signal Processing, 10(2):242–255, 2016.

[7] Jakub Konečnỳ and Peter Richtárik. Semi-stochastic gradient descent methods. arXiv preprint arXiv:1312.1666, 2013.

[8] Xingguo Li, Tuo Zhao, Raman Arora, Han Liu, and Jarvis Haupt. Stochastic variance reduced optimization for nonconvex sparse learning. In International Conference on Machine Learning, New York, USA, June 2016.

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

[10] Vatsal Shah, Megasthenis Asteris, Anastasios Kyrillidis, and Sujay Sanghavi. Trading-off variance and complexity in stochastic gradient descent. arXiv preprint arXiv:1603.06861, 2016.

[11] Shai Shalev-Shwartz. SDCA without duality, regularization, and individual convexity. In International Conference on Machine Learning, New York, USA, June 2016.

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

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

[14] Lijun Zhang, Mehrdad Mahdavi, and Rong Jin. Linear convergence with condition number independent access of full gradients. In Advances in Neural Information Processing Systems, pages 980–988, Lake Tahoe, USA, December 2013.