SVRG with Adaptive Epoch Size

Erxue Min
National University of
Defense Technology

Yawei Zhao
National University of
Defense Technology

XXXXX
National University of
Defense Technology

***Abstract*—Stochastic gradient descent (SGD) is widely used for large-scale machine learning tasks, but converges slowly 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 this paper, we propose a new stochastic gradient descent with the variance reduction technique named AESVRG which computes the full gradient adaptively. Moreover, we propose an improved method denoted by AESVRG+, which is comparable to and even better than SVRG with best-tuned epoch sizes for smooth and strongly convex functions. Extensive empirical studies illustrate the significant advantage of our proposed algorithms.**

I. Introduction

Many machine learning tasks such as neural networks [5], logistic regression [12] and matrix factorization [13] 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 learning 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 in the training dataset, i.e. n becomes very large with the proliferation of data.

Gradient Descent (GD) [4] is a basic method to solve such the optimization problem. The gradient of $F(\omega)$ denoted by $\nabla F(\omega)$ 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) [11] mitigates this shortcoming by replacing the calculation of $\nabla F(\omega)$ with a stochastic gradient denoted by $\nabla f_i(\omega)$ 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 [6] are proposed to reduce the variance. Those methods gain the linear convergence performance with a constant learning rate. In SVRG, a full gradient is computed occasionally during the inexpensive SGD steps, and thus divides the optimization procedure into many epochs. SVRG uses such the full gradient to reduce the variance during the train of parameters. On the basis of SVRG, many variants have been proposed to improve its performance. SVRG-BB [15] uses the Barzilai and Borwein (BB) method to compute the step size before every epoch [3]. CHEAPSVRG [14] aims at reducing the expensive computational cost of full gradient through using a surrogate with a subset of the training dataset. mS2GD [8] uses mini-batch method to obtain a full gradient to reduce the variance, which shows a clear advantage for parallel computation. EMGD [17], SVR-GHT [10], Prox-SVRG [16] and SVRG with second-order information [7] modify the update rule of stochastic steps, and show advantages to SVRG in some cases. However, most previous researches present that the epoch size, i.e. m should be constant [6, 15, 14] or increased monotonically [8], 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. Extensive empirical studies illustrate that the choice of good value for those hyper-parameters costs much time in real machine learning tasks.

Generally, 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 [1], 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.

Since SVRG applies SGD with variance reducer to optimize the object function, the training loss decreases rapidly at the beginning of each epoch but tends to fluctuate after excessive iterations. If we can detect the fluctuation and break the current epoch as soon as it happens, we are sure to achieve a better performance than the original SVRG. Thus the key point is finding a effective strategy to detect the fluctuation in real time.

In this paper, we develop a novel algorithm denoted by AESVRG which can adjust the epoch size adaptively. AESVRG applies a new stop condition regarding the variance and the change rate of parameters. Besides, such the stop condition terminates the iterations in an epoch by considering the value of learning rate. Additionally, we give guidance of how to set the parameters in practical machine learning tasks. Since AESVRG may stop iterations earlier than expectation when η is quite small, we propose an improved algorithm denoted by AESVRG+. In a nutshell, our contributions are highlighted as follows:

- AESVRG, this novel algorithm can adjust the epoch size to a suitable value dynamically.
- AESVRG+, it is an improvement of AESVRG which is not sensitive to parameters and more practical than AESVRG.
- 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. AESVRG and its improved method, i.e. AESVRG+. Section V demonstrates the numerical results of our algorithm. Section VI discusses the strengths and weaknesses. Section VII concludes this paper.

II. Related work

Several variants of SVRG discussing the strategy of adjusting epoch size has been proposed, including SVRG++ [2], S2GD [9], SVRG_Auto_Epoch [2] and so on.

SVRG++ adopts a simple strategy that epoch size, i.e. 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 of the value of strong convexity of $F(w)$ in Equation 1, 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, AESVRG can adaptively adjusting the epoch size without tuning extra hyper-parameters. Moreover, it takes little additional computation cost and outperforms those previous methods.

Algorithm 1 SVRG

Require: the learning rate η, the epoch size m, and an initial point $\tilde{\omega}_0$.

for $s = 0, 1, 2, ...$ **do**
  $\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_s)$
  $\omega_0 = \tilde{\omega}_s$
  for $t = 1, 2, ..., m$ **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. Preliminaries

In this section we review the SVRG algorithm proposed by Johnson and Zhang[6]. 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 calculation requires to scan the entire dataset. In each inner loop, every iteration needs to pick $i_t \in \{1, 2, ..., n\}$ randomly. The update rule of the parameters is illustrated in Equation 2:

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

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

$$\mathbb{E}[\omega_t | \omega_{t-1}] = \omega_{t-1} - \eta \nabla F(\omega_{t-1}) \tag{3}$$

It can be seen that the variance of the update rule, i.e. Equation 2 is reduced. 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 against SGD, which results in a 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 [6], 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, SGD with variance reducer will degenerate to the basic SGD, which results in huge variance. Thus our work focuses on how to set an appropriate epoch size.

IV. SVRG WITH ADAPTIVE EPOCH SIZE

In this section we describe two novel algorithms: AESVRG and AESVRG+, which can set the epoch size adaptively and have significant convergence performance superior to the previous studies. We assume the loss function $F(\omega)$ and the component functions $f_i(\omega)$ in Equation 1 are convex and smooth throughout the paper.

A. AESVRG

1) Idea: Generally, the optimal m is strongly related to η. Our experimental results also report that when η is large, the training loss begins to fluctuate after merely a small number of iterations. On the other hands, SVRG can endure far more than n iterations with a small η. In specific, if we stop the iterations in one epoch just before the training loss begins to fluctuate, the new algorithm will certainly be very efficient and outperform SVRG with constant epoch size. A direct approach is to evaluate the training loss occasionally. However, the training loss computation requires to pass over the entire training dataset, which is rather time consuming.

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

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

Then we have $\|\omega_{t+1} - \omega_t\| < \|\omega_t - \omega_{t-1}\|$. Consider the update rule of SVRG, and take the expectation of i_t, we can obtain

$$\begin{aligned} \mathbb{E}[\omega_t - \omega_{t-1}] &= \mathbb{E}[-\eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})] \\ &= -\eta \mathbb{E}[\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu}] \\ &= -\eta \nabla F(\omega_{t-1}) \end{aligned} \tag{4}$$

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

Inspired by this observation, AESVRG sets

$$\|\omega_{t+m_0} - \omega_t\| > \|\omega_t - \omega_{t-m_0}\| \tag{5}$$

as a stoping condition in each epoch. If the Inequality 5 holds, we deem that the training loss fails to converge and begins to fluctuate, we have to suspend the iterations in such epoch and compute a full gradient in order to reduce the variance.

2) Details: As illustrated in Algorithm 2, AESVRG 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, i.e. $t\%m_0 = 0$ means that we check Inequality 5 every m_0 iterations. The second condition, i.e. $t >= 2m_0$ is trivial as we need at least two windows to check the stoping conditions. If the condition holds, we break the inner loop, and step out of the current epoch, and into the next epoch. Note that the epoch size is definitely a multiple of m_0. Hence we should set m_0 to be smaller than $0.5n$ for flexibility. At the same time, m_0 cannot be too small because the stop condition may usually hold, which is caused by the random pick of the instances. We recommend to set m_0 between $0.1n$ and $0.2n$ according to extensive empirical sutdies.

Algorithm 2 AESVRG

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

1: **for** $s = 0, 1, ...$ **do**
2: $\quad \tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_s)$
3: $\quad \omega_0 = \tilde{\omega}_s$
4: $\quad$ **for** $t = 1, 2, ..$ **do**
5: $\quad\quad$ **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**
6: $\quad\quad\quad$ break
7: $\quad\quad$ **end if**
8: $\quad\quad$ Randomly pick $i_t \in \{1, 2, ..., n\}$
9: $\quad\quad$ $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t}(\omega_{t-1}) - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
10: $\quad$ **end for**
11: $\quad$ $\tilde{\omega}_{s+1} = \omega_t$
12: **end for**
13: **return** $\tilde{\omega}_{s+1}$

3) Advantages:

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

B. Optimal Choice of the window size

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

Since 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, and the probability of the stop condition of AESVRG which is denoted by $C(m_0)$, is relatively high. As a result, the Inequality 5 may often holds due to variance, even though the loss function is still decreased during iterations. Therefore, the full gradient will be computed frequently, which thus wastes much time. It is obvious that $C(m_0)$ is a monotonically increasing function of m_0.

If we set m_0 to be relatively large, it will be too late to detect the oscillation of training loss. Therefore, it wastes much time, and makes no process to optimize the objective function. We use $D_s(m_0)$ to denote the Absolute Delay of detecting fluctuation. Furthermore, the Absolute Delay makes different effects on the convergence performance, which depends on the epoch size. Thus, it is better to consider the Relative Delay:

$$D_r(m_0) = \frac{D_s(m_0)}{\upsilon + n}. \tag{6}$$

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

It is necessary to choice a suitable m_0 which should ensure that $C(m_0)$ is large and $D_r(m_0)$ is small. 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:

$$\begin{aligned} m_0 &= \max_{m_0}\{C(m_0) - D_r(m_0)\} \\ &= \max_{m_0}\{C(m_0) - \frac{D_s(m_0)}{\upsilon + n}\} \end{aligned} \tag{7}$$

s.t.

$$0 < m_0 < n \tag{8}$$

From equation (7) we can obtain the following conclusions:

1) When epoch size υ 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. 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 $0.1n$.

C. AESVRG+

1) Idea: Our experiments on AESVRG show that it performs well for large and moderate η. However, when η is rather small, the best epoch size for SVRG will be larger than $10n$. And AESVRG may finish the epoch prematurely while the training loss keeps declining. It is because the inequality (5) 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 the next epoch according to the 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 $(\lfloor \upsilon/n \rfloor + 1) \times (0.1n)$, where υ denotes the iteration number of current epoch size. It means that m_0 is incremented by $0.1n$ when υ exceeds $n, 2n, 3n$ and so on. In other words, for each epoch, we have an expected epoch size equal to $(10m_0 - n)$, if the real epoch size is smaller than the expected one, AESVRG+ will decrease the value of m_0 according to the size of current epoch. Otherwise it will set m_0 to be larger.

Algorithm 3 AESVRG+

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 $t = 1, 2, ..$ **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
  $\upsilon = t$
  $m_0 = (\lfloor \upsilon/n \rfloor + 1) \times (0.1n)$
  $\tilde{\omega}_{s+1} = \omega_t$
end for
return $\tilde{\omega}_{s+1}$

3) Advantages:

- AESVRG+ outperforms AESVRG as it will enlarge the window size to reduce variance when necessary, avoiding to stop an epoch early. It shows good performance regardless the learning rate and datasets in our experiments.
- AESVRG+ is not sensitive to the initialized m_0, as it tunes m_0 to an appropriate size adaptively. By contrast, the performance of AESVRG fairly depends on the choice of m_0.

V. Numerical Experiments

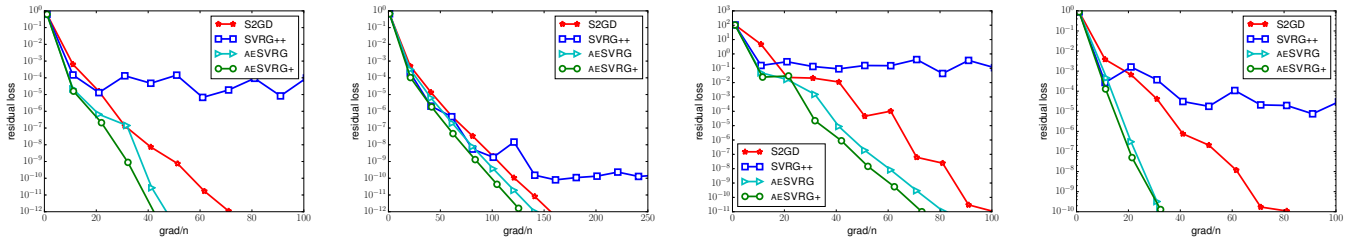
A. Experimental settings

In this section, we conduct extensive experiments to demonstrate the advantages of our proposed algorithms. We evaluate our algorithms on eight training datasets, which are public on the LIBSVM website[1]. More specifically, $l2$-regularized logistic regression on datasets: ijcnn1, a9a, w8a and mushrooms, and $l2$-regularized ridge regression on datasets: YearPredictMSD, cadata, mg, and abalone are conducted to evaluate our proposed algorithms and the previous work. The details of those datasets are illustrated in Table I. Note that the five columns denote the name of the dataset, the size of the dataset, the dimension of features, the model applied, the coefficient of regularizer respectively. Besides, logistic denotes logistic regression and ridge denotes ridge regression.

The $l2$-regularized logistic regression task is conducted on datasets: ijcnn1, a9a, w8a and mushrooms where the label of instances is set to be 1 or -1. Thus, the loss function of $l2$-regularized logistic regression task is formulated:

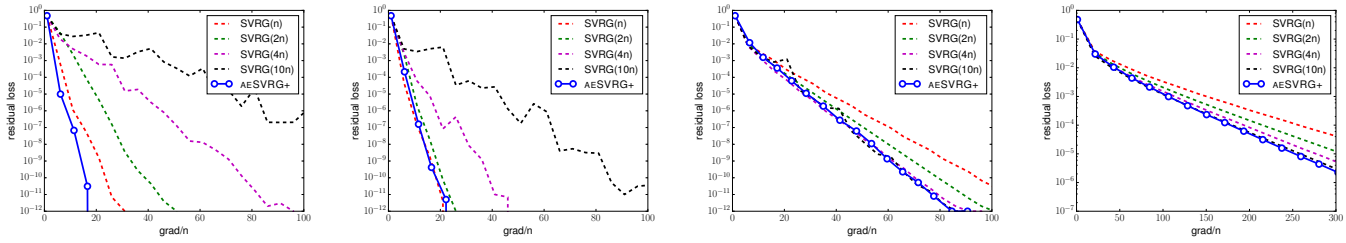
$$\min_{\omega} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i \omega^{\mathrm{T}} x_i}) + \lambda \parallel \omega \parallel^2 . \tag{9}$$

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



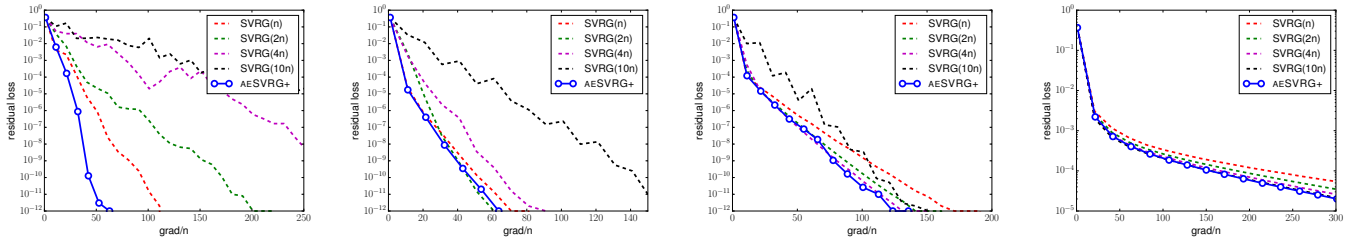
(a) w8a (b) mushrooms (c) abalone (d) mg

Fig. 1. Comparison of AESVRG, AESVRG+, SVRG++, S2GD on four datasets, with $\eta = 0.1$.



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

Fig. 2. Generally, AESVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized logistic regression on dataset ijcnn1



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

Fig. 3. Generally, AESVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized logistic regression on dataset a9a

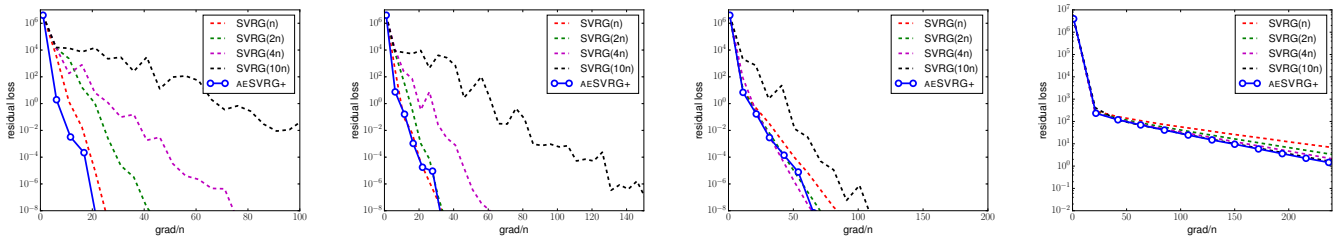
TABLE I
DETAILS OF DATASETS AND MODELS

Dataset	Size	Dimension	Model	λ
ijcnn1	49990	22	logistic	10^{-4}
a9a	32561	123	logistic	10^{-4}
w8a	49749	300	logistic	10^{-4}
mushrooms	8124	112	logistic	10^{-4}
YearPredictionMSD	463715	90	ridge	10^{-4}
cadata	20640	8	ridge	10^{-4}
abalone	4177	8	ridge	10^{-4}
mg	1385	6	ridge	10^{-4}

Here, x_i is the instances in the training dataset, and y_i is the label of x_i. λ is the coefficient of regularizer. Additionally, the $l2$-regularized ridge regression task is conducted on datasets: YearPredictMSD, cadata, mg and abalone. The loss function of $l2$-regularized ridge regression task is formulated:

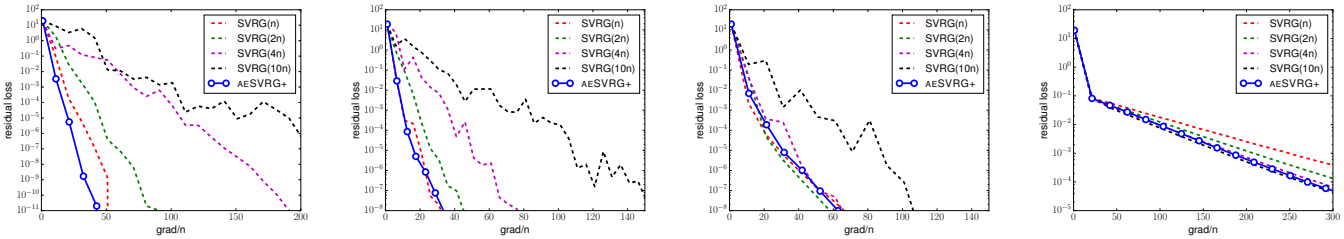
$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} \left(\omega^{\mathrm{T}} x_i - y_i\right)^2 + \lambda \parallel \omega \parallel^2 . \qquad (10)$$

We scale the value of all features to $[-1, 1]$ and set λ to be 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 $F(\omega^*)$ is estimated by running the gradient descent for a long time. The x-axis in all figures means that the number of gradient calculations divided by the size of training data, i.e. $\mathrm{grad/n}$. This is a metric for measuring gradient complexity in previous work, which is a similar to measure the time cost during train of parameters. Our numerical experiments include three parts: comparison of convergence performance with previous methods, comparison of convergence performance with SVRG by varying learning rate and sensitivity test varying m_0. The experimental results report the superior performance of our methods.



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

Fig. 4. Generally, AESVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized ridge regression on dataset YearPredictionMSD



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

Fig. 5. Generally, AESVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized ridge regression on dataset cadata

B. Comparison of convergence performance with previous methods

In this section, we compare our AESVRG and AESVRG+ with two aforementioned existing methods: SVRG++ and S2GD. We do not compare with SVRG_Auto_Epoch in that we find that its termination condition of epoch is never satisfied and thus SVRG_Auto_Epoch keeps doing SGD iteration, resulting in non-convegent. For SVRG++, we initialize $m = n$. For S2GD, we set the maximum of m to be $4n$. For both AESVRG and AESVRG+, we set the window size, i.e. m_0 to be $0.1n$. The learning rate, i.e. η, is fixed as 0.1 throughout this part of experiments. We evaluate these methods by running logistic regression on the 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 AESVRG and AESVRG+ always outperform SVRG++ and S2GD and converge rapidly. Besides, the performance of AESVRG+ is superior to that of AESVRG. The main reason is that AESVRG+ can adjust the windows size to a suitable value adaptively.

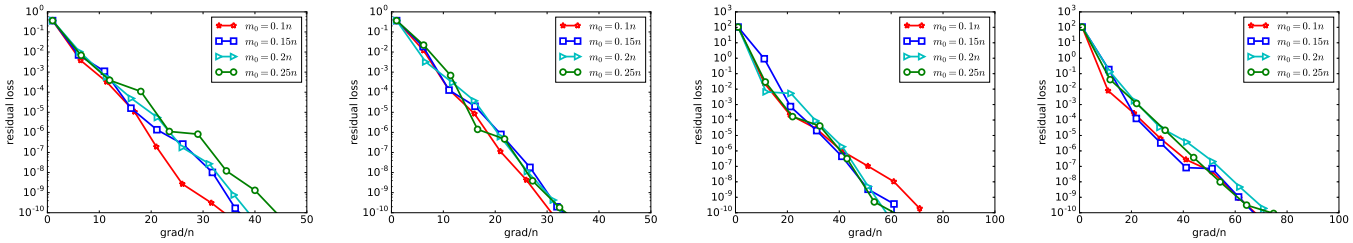
C. Comparison of convergence performance with SVRG by varying learning rate

To demonstrate that our algorithm is capable of adjusting epoch size adaptively regarding to the learning rate, we compare our algorithm with SVRG for different η. Since AESVRG+ performs better than AESVRG, only AESVRG+ is used to conduct the comparison with SVRG. For SVRG, we increase the epoch size, i.e. m as four different values: n, $2n$, $4n$, $10n$. Those values are used to conduct performance evaluation in SVRG [6]. Besides, the performance with the epoch size larger than $10n$ is similar to the performance with an extremely large epoch size. It is because that the full gradient is rare to be computed due to such extremely epoch size. The dashed lines represents SVRG with a fixed epoch size; while the green solid lines stands for AESVRG+.

As illustrated in Figures 2, 3, 4, 5, AESVRG+ 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 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 significantly 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, AESVRG+ performs slightly superior 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 AESVRG+ with rather small η is acceptable.

D. Sensitivity test by varying m_0

In this section, we conduct the sensitivity test on both AESVRG and AESVRG+ regarding window size m_0. As analyzed in IV-B, we vary the initial m_0 ranges from 0.1 to 0.25 in order to test the sensitivity of AESVRG and AESVRG+. We conduct the experiments by using logistic



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

Fig. 6. Sensitivity test on m_0 for AESVRG and AESVRG+. The four numbers on the legends means four different choices of m_0. The η of a9a and abalone are 0.2 and 0.1, respectively.

regression and ridge regression on two datasets: a9a and abalone, respectively. As illustrated in Figures 6(a) and 6(c), the performance of AESVRG obviously varies with the m_0. And we can see in Figures 6(b) and 6(d) that the performance of AESVRG+ is not sensitive to the choice of m_0. The main reason is that AESVRG uses the prefixed m_0 all the time, while AESVRG+ will adjust the m_0 adaptively regardless of the initialization. If the initial value is too large, the stop condition holds just after a few windows of iterations, leading to a smaller epoch size than expected. Thus AESVRG+ will decrease m_0 to a suitable value gradually. On the contrary, AESVRG+ will adjust m_0 to bigger according to the size of current epoch. Hence AESVRG+ is more practical than AESVRG in reality.

VI. Discussion

Optimization problems are complex as there are numerous application scenarios, algorithms and datasets. It is impossible for SVRG to always achieve an excellent performance with the same epoch size. Besides, searching for the optimal epoch size is time-consuming and not feasible for massive data. Several existing methods provide some strategies for how to set the epoch size, but they are not well adapted to the various real problems. Hence our proposed algorithms which can adjust epoch size adaptively are great improvements of SVRG. Although we have not rigorously conducted the convergence analysis of AESVRG and AESVRG+, both of them prove to be efficient and have a significant improvement on the original SVRG experimentally. Thus the novel stop condition is a practical strategy for detecting the fluctuation of training loss. We leave it as an open question to prove the rationality theoretically.

VII. Conclusion

In this paper we propose a novel stop condition for each epoch in SVRG, leading to a new variant of SVRG: AESVRG, which can adjust the epoch size adaptively. We analyze how to choose the optimal value of parameters, and thus develops an improved method called AESVRG+. We conduct numerical experiments on real datasets to demonstrate the advantage of convergence performance of AESVRG and AESVRG+. The experiments show that both AESVRG and AESVRG+ are superior to existing methods. Moreover, the AESVRG+ is comparable to and sometimes even better than SVRG with best-tuned epoch size, but does not need to tune its epoch size.

References

[1] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-Convex optimization. In International Conference on Machine Learning, New York, USA, June 2016.

[2] 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.

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

[4] Boyd, Vandenberghe, and Faybusovich. Convex optimization. IEEE Transactions on Automatic Control, 51(11):1859–1859, 2013.

[5] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Z Mao, Aurelio Ranzato, Andrew Senior, and Paul Tucker. Large scale distributed deep networks. Advances in Neural Information Processing Systems, pages 1232–1240, 2012.

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

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

[8] 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.

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

[10] 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.

[11] Herbert Robbins and Sutton Monro. A stochastic approximation method. Annals of Mathematical Statistics, 22(3):400–407, 1951.

[12] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. Advances in Neural Information Processing Systems, 4:2663–2671, 2012.

[13] Christopher De Sa, Kunle Olukotun, and Christopher R. Global convergence of stochastic gradient descent for some non-convex matrix problems. Mathematics, pages 2332–2341, 2015.

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

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

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

[17] 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.