SVRG with Adaptive Epoch Size

Erxue Min, Yawei Zhao, Jun Long, Chengkun Wu, Kuan Li, and Jianping Yin
College of Computer, National University of Defense Technology,
Changsha, China 410073

***Abstract*—Stochastic gradient descent (SGD) is a commonly used technique in large-scale machine learning tasks, but its convergence is slow due to the inherent variance. In recent years, a popular method, Stochastic Variance Reduced Gradient (SVRG) , addresses this shortcoming via computing the full gradient of the entire dataset in each epoch. However, conventional SVRG and its variants usually need to identify a hyper-parameter - the epoch size, which is essential to the convergence performance. Few previous studies discuss how to systematically find a suitable value for that hyper-parameter, which makes it hard to gain a good convergence performance in practical machine learning applications. In this paper, we propose a new stochastic gradient descent named AESVRG, which introduces variance reduction and computes the full gradient adaptively. Its enhanced implementation, AESVRG+, has a convergence performance that can outplay existing SVRG with fine-tuned epoch sizes. An extensive evaluation illustrates the significant performance improvement of our method.**

I. Introduction

Recent years have seen the rapid development of network and various sensors [1], thus machine learning methods become popular as they are suitable for handling data with huge size. Many machine learning models like neural networks [2], logistic regression, [3] support vector machine [4] and matrix factorization [5] involves the following optimization procedure:

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

where n is the size of the training data. $F(\omega)$ is the loss function, which calculates the training loss of a machine learning model, and ω is a parameter for the loss function. $R(\omega)$ is the regularizer, which is widely used to avoid overfitting. It is worth noting that the total number of instances in the training dataset, i.e. n can be very large in the 'big data' era.

Gradient Descent (GD) [6] is a canonical method to solve the above-mentioned optimization problem. The gradient of $F(\omega)$, denoted by $\nabla F(\omega)$, can be obtained from the whole training set, which is extremely time-consuming if n is very large. Besides, GD is an iterative-convergent algorithm, that is, it usually takes thousands of iterations for the parameter ω to be converged. Since GD needs to compute the gradient of $F(\omega)$ in each iteration, when the volume of data is large, the computation cost increases sharply and impairs the convergent performance significantly.

Stochastic Gradient Descent (SGD) [7] addresses this issue 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 randomly selected from the entire training set. Thus, SGD significantly outperforms GD in terms of time efficiency . 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 $F(\omega)$ converge, a decaying learning rate is usually used to reduce the variance. However, the learning rate becomes too small after hundreds of iterations, which can prevent the loss function from converging. In conclusion, SGD with a dacaying learning rate incurs a sub-linear convergence rate.

In recent years, variance-reduced variants of SGD such as SVRG [8] were proposed to reduce the variance of stochastic gradient. Those methods can obtain a linear convergence performance with a constant learning rate. In SVRG, a full gradient is only occasionally computed during the inexpensive SGD steps, and thus splits the optimization workload into different epochs. This strategy effectively reduce the variance of stochastic gradients. On the basis of SVRG, many variants have been proposed to further improve its performance. SVRG-BB [9] uses the Barzilai and Borwein (BB) method to compute the step size before each epoch [10]. CHEAPSVRG [11] aims at reducing the expensive computational cost of a full gradient through using a surrogate represented by a subset of the training data. mS2GD [12] introduces mini-batching into the computation of stochastic steps, which exhibits a clear advantage for parallel computation. EMGD [13], SVRGHT [14], Prox-SVRG [15] and SVRG with second-order information [16] modify the update rule of stochastic steps, and show advantages to SVRG in some cases. However, most recent studies argued that the epoch size m should be constant [8, 9, 11] or increased monotonically [12], 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 a good choice of those hyper-parameters is essential for time efficiency in real-world machine learning applications.

Generally, the epoch size m has a great impact on the convergence performance of SVRG. Specifically, if m is too small, it wastes too much time for frequent computation of the full gradient. If m is too 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 [17], the learning rate η can also have a significant impact on the convergence performance. However, those previous studies do not provide

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

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

a practical method to determine the value of those hyper-parameters.

Since SVRG applies SGD with a 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. Intuitively, if we can detect the fluctuation and break the current epoch as soon as it happens, we should be able to achieve a better performance than the original SVRG. Thus the key idea is to find an effective strategy to detect the fluctuation in real time. In this paper, we develop a novel algorithm, AESVRG, which can adjust adaptively the epoch size. AESVRG applies a new stop condition regarding the variance and the changing rate of parameters. Besides, 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 brief, our contributions are highlighted as follows:

- A novel algorithm denoted by AESVRG is proposed, which can adjust the epoch size to a suitable value dynamically.
- An improvement of AESVRG denoted by AESVRG+ is presented, which is insensitive to the selection of hyper-parameters.
- Extensive empirical studies shows the effectiveness of our proposed algorithms, which outperform their counterparts significantly on the convergence performance.

This paper is organized as follows: Section II reviews the related work. Section III describes the SVRG algorithm. Section IV presents our version of SVRG, AESVRG, and its improved method, i.e. AESVRG+. Section V demonstrates the numerical results of our algorithms. Section VI discusses the strengths and weaknesses of our work. Section VII concludes this paper.

II. Related work

Several variants of SVRG discussing the strategy of adjusting epoch size were proposed, including SVRG++ [18], S2GD [19], SVRG_Auto_Epoch [18] and so on.

SVRG++ is designed for non-strongly-convex and sum-of-non-convex objectives specifically. It adopts a simple strategy that the epoch size m doubles between every two consecutive epochs. This method is absolutely heuristic and sometimes not justified. Although [18] analyzed the convergence of SVRG++ in theory and has conducted experiments to confirm their theoretical findings, our experiments show that when η is large or moderate, the exponential growth of m will incur a greater variance and impair convergence.

S2GD designs a probability model of m and shows that a large epoch size should be used with a high probability. The motivation of designing this probability model is not indicated and it seems arbitrary. A notable drawback of S2GD is that 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 number of stochastic steps per epoch is also a sensitive parameter. Although it performs well with best-tuned parameters, it is not practical in reality.

SVRG_Auto_Epoch is introduced as an further 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. It keeps track of the average $diff_t$ in the last $n/4$ iterations and compare this quantity with the average $diff_t$ of the previous epoch. If the former is greater than half of the latter, SVRG++ will terminate the current epoch and step into a new one. Although this method is reasonable, it has too much parameters to tune. Moreover, it introduces a lot of extra computation for each iteration, which impairs the performance significantly.

Comparing with the above methods, AESVRG is intuitively more reasonable and explainable. It can adaptively adjust the epoch size without tuning extra hyper-parameters. Moreover, it requires little additional computation cost.

III. Preliminaries

In this section we review the SVRG algorithm proposed by Johnson and Zhang[8]. 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:

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

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

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

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** $\tilde{\omega}_{s+1}$

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 the inner loop. Note that there are two options for the update. Although only the convergence of SVRG with Option I is analyzed in [8], 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 a variance reducer will degenerate to the basic SGD, which results in a 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 their convergence performance outplay 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 hand, 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 a constant epoch size. A straightforward approach is to evaluate the training loss occasionally. However, the training loss computation requires to pass over the entire training set, which is rather time consuming.

When we apply gradient descent to the convex problem, as the ω_t gradually approaches to the optimal value ω^*, 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 $\|\omega_{t+1} - \omega_t\| < \|\omega_t - \omega_{t-1}\|$. 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 stop condition in each epoch. If 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 requires only two parameters: the learning rate η and the 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 stop 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 our empirical experiments.

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 frequent computation of the full gradient.
- Compared 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.

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 $\tilde{\omega}_{s+1}$

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 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 reliability of the stop condition of AESVRG which is denoted by $C(m_0)$, is relatively low. As a result, 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 a lot of 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 also wastes a lot of time, and fails 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, depending 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 choose a suitable m_0 which should ensure that $C(m_0)$ is large enough and $D_r(m_0)$ is small enough. 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 the 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 in order 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 experiments 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 a large or 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 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. 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 assign a larger value to m_0.

3) Advantages:

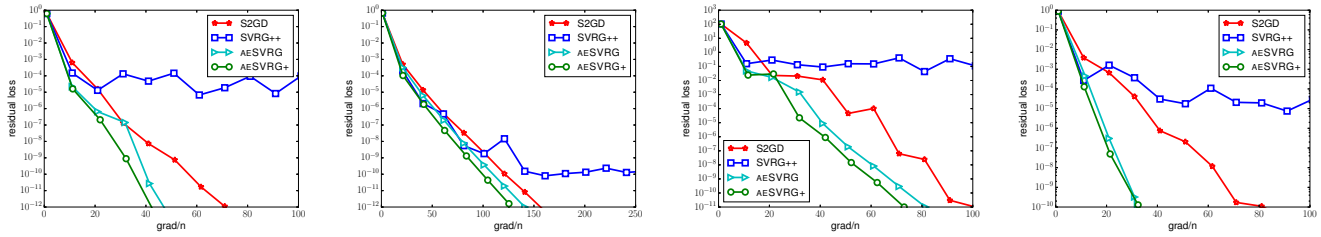
- AESVRG+ outperforms AESVRG as it will enlarge the window size to reduce variance when necessary, avoiding to stop an epoch early. It shows good performance regardless the learning rate and datasets in our experiments.
- AESVRG+ is not sensitive to the initialized m_0, as it tunes m_0 to an appropriate 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 different training datasets respectively, which are publicly available from the LIBSVM website[1]. The

[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 $\eta = 0.1$.

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}

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 the regularization method, 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. The label of each instance in these datasets is set to be 1 or -1. Thus, the loss function of $l2$-regularized logistic regression task is formulated:

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

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:

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

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 means that the number of gradient calculations divided by the size of training data, i.e. grad/n. This is a metric for measuring gradient complexity in previous work, which is similar to the time cost during train of parameters. 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. Our numerical experiments include three parts: comparison of convergence performance with previous methods, comparison of convergence performance with SVRG by varying the learning rate and sensitivity tests regarding to m_0. The experimental results report the superior performance of our methods.

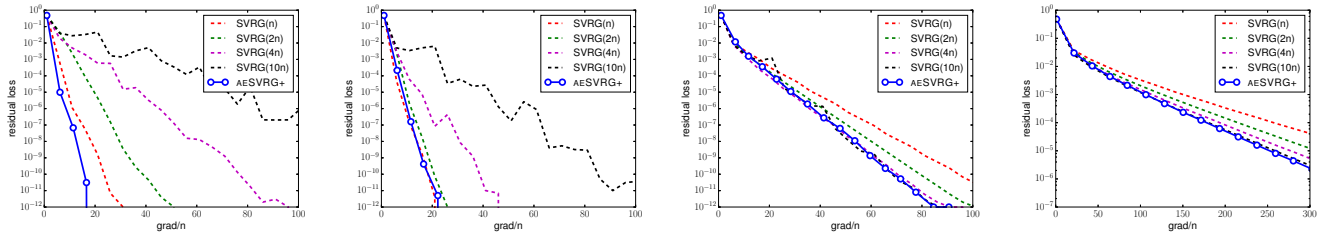
B. Comparison of convergence performance with previous methods

In this section, we compare our AESVRG and AESVRG+ with two aforementioned methods: SVRG++ and S2GD. We do not compare with SVRG_Auto_Epoch because we find that its termination condition of epoch is never satisfied and thus SVRG_Auto_Epoch keeps doing SGD iteration, resulting in no convergence. For SVRG++, we initialize $m = n$. For S2GD, we set the upper bound of m to be $4n$. For both AESVRG and AESVRG+, we set the window size m_0 to be $0.1n$. The learning rate 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

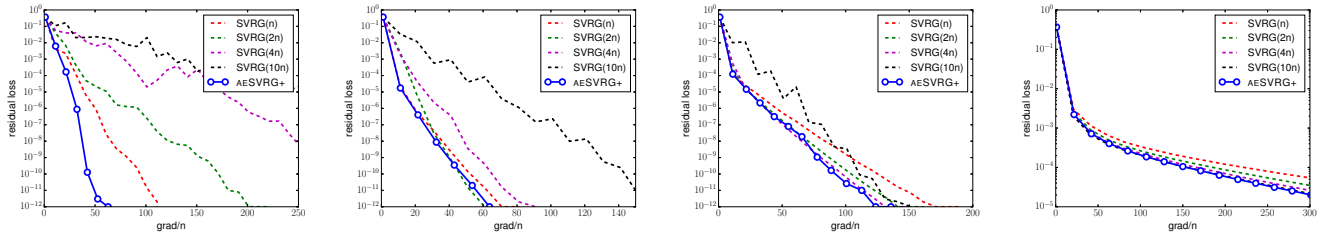
To demonstrate that our algorithm is capable of adjusting epoch size adaptively regarding to the learning rate, we compare our algorithm with SVRG for different η. Since AESVRG+ performs better than AESVRG, only AESVRG+ is used to conduct the comparison with SVRG. For SVRG, we increase the epoch size m as four different values: n, $2n$, $4n$, $10n$. Those values are used to conduct performance evaluation in SVRG [8]. Besides, the performance with an epoch size larger than $10n$ is similar to the performance with an extremely large epoch size. It is because that the full gradient is rarely computed due to such extremely epoch size. The dashed lines represent 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 best-tuned epoch size. We observe that when η is large, and m is set to be



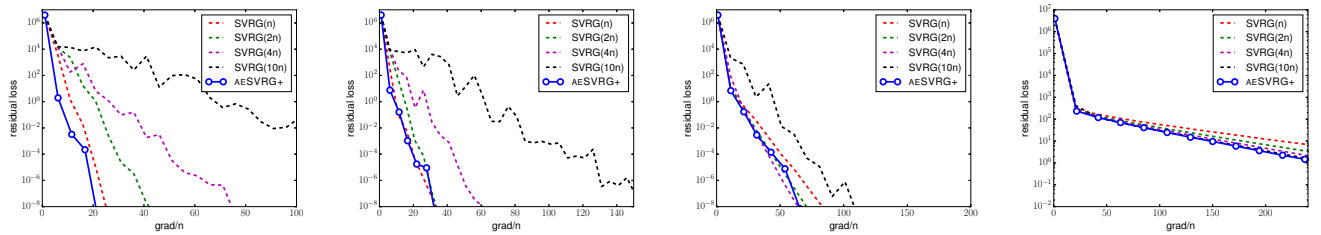
(a) ijcnn1 $\eta = 0.5$ (b) ijcnn1 $\eta = 0.05$ (c) ijcnn1 $\eta = 0.01$ (d) ijcnn1 $\eta = 0.001$

Fig. 2. Generally, AESVRG can automatically set an appropriate m with different learning rates for the $l2$-regularized logistic regression on the 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 the dataset a9a



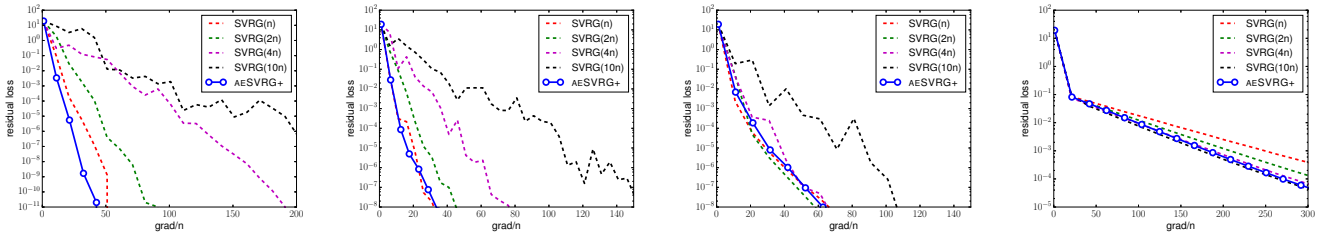
(a) YearPredictionMSD $\eta = 0.01$ (b) YearPredictionMSD $\eta = 0.005$ (c) YearPredictionMSD $\eta = 0.002$ (d) YearPredictionMSD $\eta = 0.0001$

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

a small value, e.g. n, can achieve the best performance. The main reason is that when η is large, the variance becomes significant, 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 inferior to SVRG with large epoch sizes, but outperforms SVRG with recommended epoch sizes, i.e. n and $2n$. It is worth 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 a rather small η is acceptable.

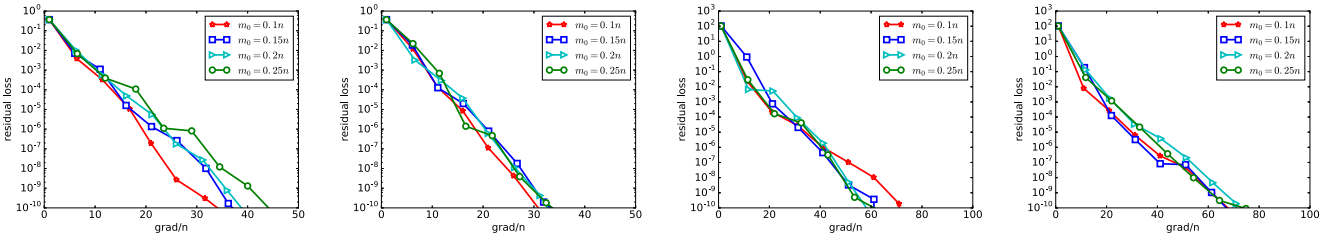
D. Sensitivity test by varying m_0

In this section, we conduct sensitivity tests on both AESVRG and AESVRG+ regarding the 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 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), 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 increase m_0 according to the size of current epoch. Hence AESVRG+ is more practical than AESVRG in reality.



(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 the dataset cadata



(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.

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, the selection for an optimal epoch size is time-consuming and not feasible for a massive dataset. Several existing methods provide some strategies for the choice of 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. Despite the fact that we have to tune the window size m_0, this hyper-parameter is not performance sensitive. First, Comparing with existing methods, our algorithms have limited the range of our hyper-parameter to a small value. Second, it can be adapted to the learning rate automatically, which is more practical than the counterparts. Although we have not rigorously conducted a convergence analysis of AESVRG and AESVRG+, both of them prove to be efficient experimentally 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+. 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 sizes, but does not need to tune its epoch size.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Project NO. 60970034, 61170287, 61232016, 61303189 and 31501073).

References

[1] Yawei Zhao, Deke Guo, Jia Xu, Pin Lv, Tao Chen, and Jianping Yin. Cats: Cooperative allocation of tasks and scheduling of sampling intervals for maximizing data sharing in wsns. *Acm Transactions on Sensor Networks*, 12(4):29, 2016.

[2] 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.

[3] 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.

[4] X. Liu, J. Yin, L. Wang, L. Liu, J. Liu, C. Hou, and J. Zhang. An adaptive approach to learning optimal neighborhood kernels. *IEEE Transactions on Systems Man and Cybernetics Part B Cybernetics A Publication*

of the IEEE Systems Man and Cybernetics Society, 43(1):371–384, 2013.

[5] 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.

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

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

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

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

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

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

[12] 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.

[13] 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.

[14] 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.

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

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

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

[18] 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.

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