Speed Maintained SVRG

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678–2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—Stochastic gradient desent (SGD) is widely used for large-scale machine learning optimization but has slow convergence rate due to the high 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. Although many variants of SVRG have been proposed to improve the performance, few papers discuss how frequently should a full-gradient be computed. In our paper, we propose SMSVRG, a variant of SVRG, but instead of fixing the epoch size, it applies an real-time checking strategy to adjust the epoch size dynamically. Moreover, we provide the guidance of choosing the checking interval and propose an improved method SMSVRG+, which is comparable to and sometimes even better than SVRG with best-tuned epoch sizes for smooth and strongly convex functions.**

I. Introduction

In machine learning, it is common to encounter the following optimization problem 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, and each f_i is the cost function of the i-th training data. Note that we assume each f_i is smooth and convex, and the function F is strongly convex. $R(\omega)$ is the regularizer, which is widely used to avoid overfitting.

Gradient descent (GD) is a basic method to solve such optimization problem and is effective for small or moderate-scale datasets. However, since GD passes over the entire training data to compute the full gradient every iteration, when the volume of data is large, the computation cost increase significantly and would affect the performance.

Stochastic optimization methods mitigates this shortcomings by computing only a small subset of the input data. For example, Stochastic gradient descent (SGD) randomly selects one data sample each iteration, which improves time efficiency a lot. Nevertheless, the stochastic method introduce variance thus can hardly make any progress when the parameters are close to the optimum. Common practice to ensure convergence is decaying learning rate, which incurs a sub-linear convergence rate.

Stochastic average gradient (SAG) [9] and its variant such as SAGA [3], SDCA [11] proves to converge linearly for strongly convex problems but requires to store all the component gradients, which bring in an intolerable memory burden.

In recent years, the stochastic variance reduces gradient (SVRG) [4] method is popular and widely used, because it achieves the linear convergence and does not need to store the component gradients. As is shown in Algorithm (1), 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] method uses the BB method proposed by Barzilai and Borwein in [2] to compute the step size before each epoch, which generally achieve the same level of sub-optimality as 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] introduces mini-batching into the computation of stochastic steps to reduce the variance and shows clear advantage when parallel computation is allowed. EMGD [14], SVR-GHT [8], Prox-SVRG [13] and SVRG+second-order information [5] modify the update rule of stochastic steps, and show advantages to naive SVRG in some cases. All these methods proved to obtain improvements on SVRG to some extent. However, there are few papers discussing about how frequently should a full gradient be computed, i.e. how to set the epoch size m. Most papers set $m = 2n$ or $m = n$ regardless of the learning rate, where n denotes the size of training data. 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. It is obvious that m has a great impact on the performance of SVRG. When m is too small, it waste too much time on computing full gradient therefore converges slowly. Contrarily, when m is rather big, the variance increases a lot, decelerating the convergence of training loss. As is clearly acknowledged that the optimal m is strongly related to the learning rate η. According to the analysis of variance in [YaWei], If the update item γ_t in each SGD iteration is p-dimensional, and can be denoted by $\gamma_t = (a_{t1}, a_{t2}..., a_{tp})$, after m iterations, the variance d_m holds that $d_m{=}\eta^2 \sum\limits_{j=1}^{p}\left(\sum\limits_{i=0}^{m-1} a_{ij}\right)^2$. On the condition that d_t

has a upper bound d^b to guarantee convergence, we must set m to be small to avoid high variance if η is big. And we could choose a relatively big value for m with a small η, for purpose of reducing the frequency of time-consuming full gradient computation. Our experiments show that the loss function begins to fluctuate after only $n/10$ iterations with very large η, and keeps reducing for even $20n$ iterations with very small η.

In this paper, we propose a novel algorithm: SMSVRG, which can adjust the epoch size dynamically. It applies a strategy that the amount of change in parameters are computed at the same interval, if the amount of change of current interval is greater than that of former, the algorithm will finish the current epoch and begin the next epoch. Besides, we analyze and give guidance of how to set the interval size. Based on the analysis, we modify our algorithm as SMSVRG+, which show better performance in our experiments. This paper is organized as follows: Section 2 review related work about setting the epoch size of SVRG. Section 3 present the new variant of SVRG, i.e. SMSVRG. Section 4 demonstrates the extensive performance of our algorithm. Finally, we conclude this paper.

II. Related work

Several variants of SVRG focusing on 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 big 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.

III. Speed Maintained SVRG

In this section we describe two novel algorithms: SMSVRG and SMSVRG+, which can set the appropriate iteration number in each epoch automatically 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.

Algorithm 1 SVRG

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

$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega})$
for $s = 0, 1, \ldots$ **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 $i_t \in \{1, 2, ..., n\}$
        $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t} - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
    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. smSVRG

When we apply gradient descent to convex problem, as the ω_t will gradually approach the optimal value ω^*, thus the gradient $\nabla F(\omega_t)$ keeps decreasing. Then we have $\|\omega_{t+1} - \omega_t\| < \|\omega_t - \omega_{t-1}\|$. Considering several iterations, we can also have $\|\omega_{t+m_0} - \omega_t\| < \|\omega_t - \omega_{t-m_0}\|$. Inspired by this, algorithm SMSVRG set $\|\omega_{t+m_0} - \omega_t\| > \|\omega_t - \omega_{t-m_0}\|$ as a stop condition in each epoch. Algorithm SMSVRG just require two parameters: learning rate η, checking interval m_0. Note that the difference between SVRG and SMSVRG is that in the latter we adjust the epoch size dynamically, instead of using a prefixed m as in SVRG. In each epoch, SMSVRG compute the inequality

$$\|\omega_t - \omega_{t-m_0}\| > \|\omega_{t-m_0} - \omega_{t-2m_0}\|$$

every m_0 iterations, if the inequality holds the algorithm will break the inner loop and begin the next epoch.

B. Optimal Choice of checking interval

In SMSVRG, we use the $\Delta\omega$ of several iterations to detect when loss function begin to fluctuate and fail to converge. However, how to choose a suitable value of checking interval 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 big thus the confidence of the inequality is low. As a result, the inequality may holds because of 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.

When we set m_0 to be relatively big, the variance becomes small and the confidence of inequality holds is high.

However, it will be too late to detect the fluctuation of objective function 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 big $C(m_0)$ 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:

$$\begin{aligned} m_0 &= \max_{m_0} C(m_0) - D_r(m_0) \\ &= \max_{m_0} C(m_0) - \frac{D_s(m_0)}{es + n} \\ &\text{s.t.} 0 < m_0 < n \end{aligned} \tag{2}$$

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 (2) 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 big. 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 begin 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+

On the basis of the analysis of section III-B, we improve our algorithm SMSVRG through dynamically adjusting the checking interval, instead of a constant value. According to the two conclusions, we initialize the m_0 to be $0.1n$ and set it to be $(int(es/n)+1)*(0.1n)$ in each epoch, where es denote the iteration number of previous epoch. With this strategy , when each epoch seems to be capable of enduring more iterations, SMSVRG+ will expand the checking interval to improve the confidence, avoiding stopping the epoch ahead of time due to variance.

IV. Numerical Experiments

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

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

Algorithm 2 SMSVRG

Require: learning rate η, checking interval m_0, initial point $\tilde{\omega}$

$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega})$
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 $i_t \in \{1, 2, ..., n\}$
    $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t} - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
  end for
  $\tilde{\omega}_{s+1} = \omega_m$
end for
return $\tilde{\omega}_{s+1}$

Algorithm 3 SMSVRG+

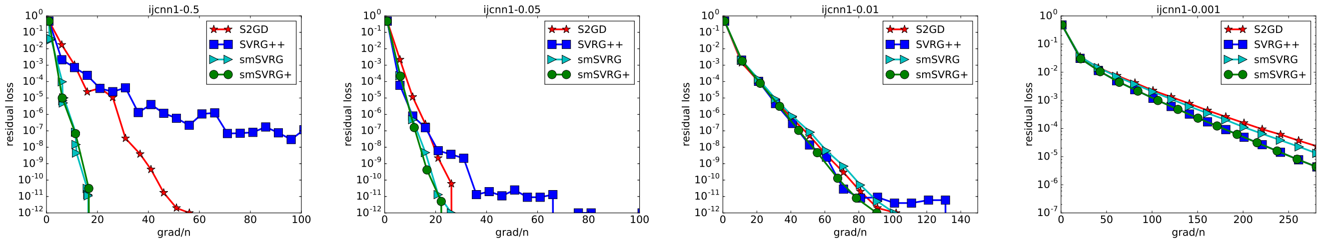
Require: learning rate η, checking interval m_0, initial point $\tilde{\omega}$
Initialize: $m_0 = 0.1n$

$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega})$
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 $i_t \in \{1, 2, ..., n\}$
    $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t} - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
  end for
  $es = t$
  $m_0 = (int(es/n) + 1) * (0.1n)$
  $\tilde{\omega}_{s+1} = \omega_m$
end for
return $\tilde{\omega}_{s+1}$

The $l2$-regularized logistic regression task is conducted on the two datasets: ijcnn1, a9a. 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:

$$\min_{\omega} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i \omega^{\mathrm{T}} x_i}) + \lambda \parallel \omega \parallel^2 . \tag{3}$$

The $l2$-regularized ridge regression task is conducted on the four datasets: abalone, cadata, cpusmall, space_ga. The loss



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

Fig. 1. Comparison of SMSVRG, SMSVRG+, SVRG++, S2GD

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}
cada	20640	8	linear	10^{-4}

function of $l2$-regularized ridge regression task is:

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} \left(\omega^{\mathrm{T}} x_i - y_i\right)^2 + \lambda \parallel \omega \parallel^2 . \quad (4)$$

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 two parts: comparing with existing related methods and comparing with SVRG of different epoch sizes. Both experiments show the superior performance of our methods.

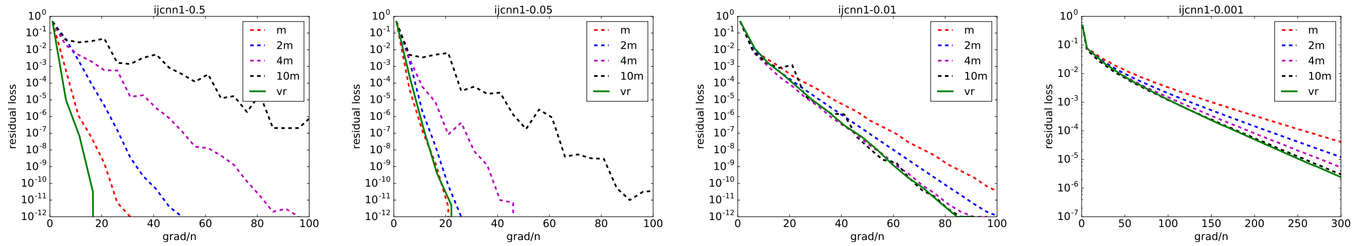
A. Comparing with existing related methods

In this section, we compare our SMSVRG(Algorithm 2) and SMSVRG+(Algorithm 3) with two aforementioned existing methods: SVRG++ and S2GD. We do not compare with SVRG_Auto_Epoch in that 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 $4n$ and set $gamma$ to be 0. For both SMSVRG and SMSVRG+, we set the checking interval m_0 to be $10/n$. We test these methods by logistic regression on dataset $ijcnn1$. From Figures III-C we can see that at most times SVRG++ fluctuates violently and fails to converge owing to the large variance caused by too excessive epoch size. It only performs well when η is quite small. Figures 1(a), 1(b) show that both SMSVRG and SMSVRG+ converge rapidly with relatively large η for they can constrain m in a small range. However, it can be seen from Figures 1(c), 1(d), when η decreases to small values, the performance of SMSVRG drop gradually and even become inferior to others, while SMSVRG+ always performs the best of all. The main reason is that when η is small, more iterations can be computed in one epoch, thus checking the inequality frequently will resulting algorithm to stop epochs too early because of variance. The strategy of increase checking interval applied by SMSVRG+ can reduce the variance. Hence it always outperform other methods.

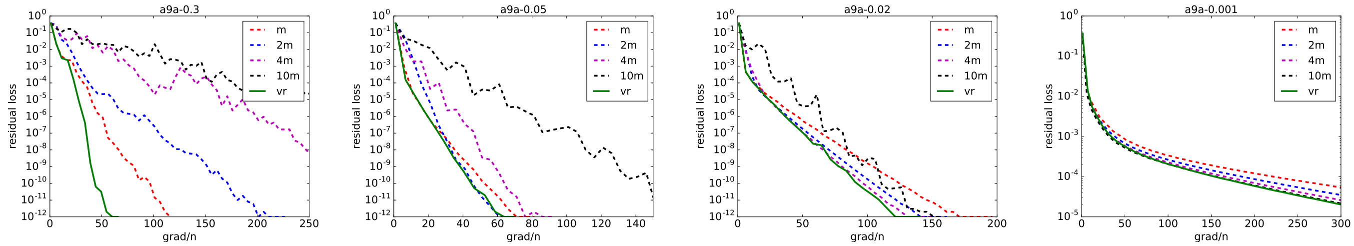
B. Comparing with SVRG of different epoch sizes

In order to demonstrate the outstanding adaptability of our algorithm with different learning rate in any case, we conduct sufficient experiments. Since SMSVRG+ performs better than SMSVRG, we only compare the former with SVRG. The experiments are tested by logistic regression and ridge regression on four datasets. For SVRG, we set epoch size m as four different values: $n, 2n, 4n, 10n$. The first three values are chosen for the reason that they are commonly used in many papers. Besides, our experiments show that epoch size bigger that $10n$ perform almost the same. It is easy to comprehend as the full gradient computation becomes less important when epoch size is rather big. In all figures, the dashed lines correspond to SVRG with fixed epoch size given in the legends of the figures, while the green solid lines correspond to SMSVRG

It can be seen from Figures 2(a) to 3(h) that SMSVRG can always have the similar performance as SVRG with most suitable epoch size. We observe that when η is big, setting m to be a small value, i.e. n, can achieve better performance. The main reason is that when η is big, the variance becomes big simultaneously, so m must be set small to constrain the variance. As η diminishes, the optimal value of m increases, which means that the algorithm can tolerate more variance induced by extra iterations. As illustrated in Figures, our method is comparable to and sometimes even better than SVRG with best-tuned epoch sizes when learning rate is large or medium. However, 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 very small η is acceptable.

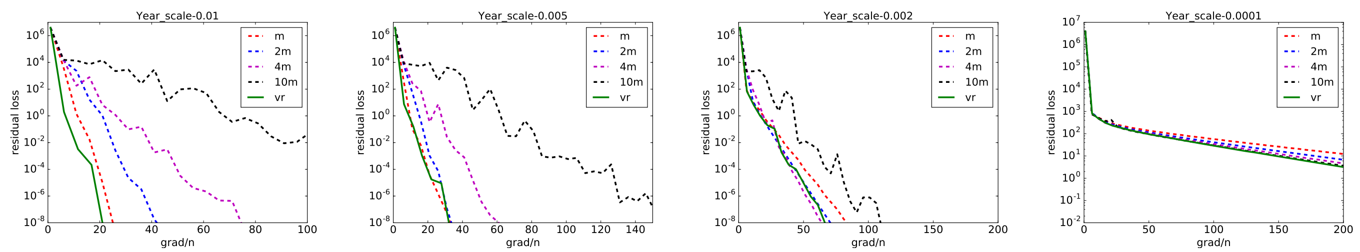


(a) ijcnn1 $\eta = 0.5$ (b) ijcnn1 $\eta = 0.05$ (c) ijcnn1 $\eta = 0.01$ (d) ijcnn1 $\eta = 0.001$

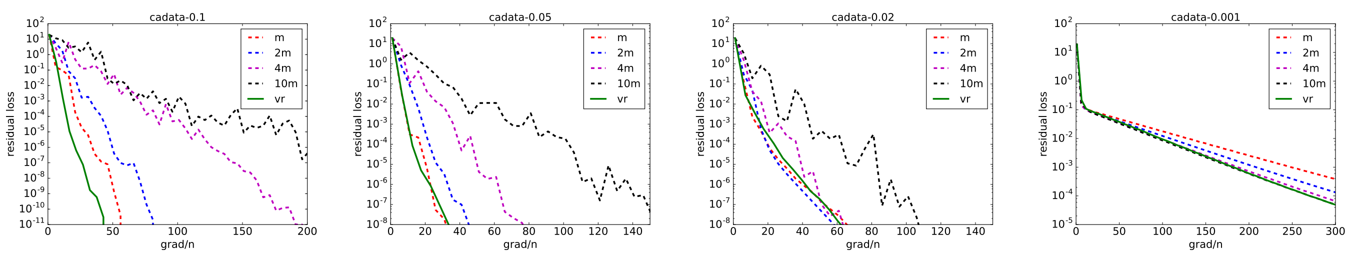


(e) a9a $\eta = 0.3$ (f) a9a $\eta = 0.05$ (g) a9a $\eta = 0.02$ (h) a9a $\eta = 0.001$

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



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



(e) cadata $\eta = 0.1$ (f) cadata $\eta = 0.05$ (g) cadata $\eta = 0.02$ (h) cadata $\eta = 0.001$

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

V. Conclusion

1111
111
111
111
111

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.