EUI: A General Framework of the Reduced Variance Stochastic Gradient Descent and the Accelerated Implementation

Abstract

Stochastic gradient descent (SGD) with the variance reduction technique is efficient to train parameters of many machine learning algorithms. Although many variants of SGD with the variance reduction technique have been proposed, the analysis of variance does not gain enough discussion, and thus the power of the variance reduction technique will not be exploited. In this paper, we provide a fine-grained analysis of the variance from a new prospective, and propose an unified framework, i.e. EUI to illustrate the existing versions of SGD with the variance reduction technique. Moreover, since the variance reduction technique needs a snapshot of the full gradient for every epoch, which costs much time during the train of parameter, a new implementation of the framework, i.e. EstimateVR has been proposed. EstimateVR avoids the time-consuming calculations of the full gradient, instead uses the estimation of the full gradient, and gains the geometry convergence rate. Both the theoretical and the empirical studies have verified that EstimateVR outperforms other previous work.

Introduction

Many machine learning problems such as classification and regression can be presented like the optimisation problem in Equation 1. ω is the parameter of a machine learning model. $F(\omega)$ is the loss function which is the sum of the finite functions, i.e. $f_i(\omega)$. $R(\omega)$ is the regulariser which is used to prevent overfitting.

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

.

Gradient descent (GD) is often used to train the parameters in such underlying machine learning problems. Since GD computes the full gradient in each iteration, it leads to extremely large amount of derivative calculations unavoidably. With the proliferation of training data, GD is not efficient to train parameters for a large scale machine learning task. The stochastic gradient descent (SGD) overcomes the disadvantage of GD by using a stochastic gradient instead of the full gradient to train parameters. However, the variance of the optimisation objective due to the stochastic gradient usually impairs the convergence of parameters during the training process. Specifically, when the parameter is close to the optimum, it is more and more difficult to made a further progress due to the variance.

Conventional studies show that a decaying learning rate can be used to decrease the variance. However, the decaying learning rate usually slows the convergence of the loss function, when the learning rate has been decreased to a tiny value. Recently, Johnson et al. improves SGD with the variance reduction technique denoted by SVRG which uses a constant learning rate to train the parameter (**?**). SVRG is organised by epochs, and each of which consists of a number of iterations. Based on the variance reduction technique adopted by SVRG, many variants of SVRG such as S2GD (**?**), mS2GD(**?**), EMGD (**?**), SVR-GHT (**?**), Prox-SVRG (**?**), SVRG++ (**?**), Katyusha (**?**) have been proposed. Those achievements use the variance reduction technique, but accelerate SVRG by using a weak assumption, or apply it in different scenarios. However, the variance and the potential of the variance reduction technique lack of enough discussion. Since the variance reduction technique is designed to reduce the variance caused by the stochastic gradient, it is meaningful to measure and give the quantitative analysis of the variance so that the potential of the variance reduction technique can be exploited. Additionally, the variance reduction technique needs a snapshot of the full gradient at the beginning of an epoch, which consumes much time. If the snapshot of the full gradient can be avoided, those versions of the reduced variance will be accelerated a lot, and achieve a better performance of the convergence.

Comparing with SGD and GD, we understand the reason of the underlying variance reduction technique from a new prospective. The update gradient by using variance reduction technique can be divided into three ingredients, including the variance source, the variance reducer and the progressive direction. Specifically, the variance reducer is the real reason to reduce the variance. From the analysis of the variance reducer, we find that the source of the variance is introduced from the distance between the current and the expected state of the parameter. The analysis is applied to the existing versions of SGD with the variance reduction technique in order to provide an entire understand of these achievements. Moreover, we design a new version SGD with the variance reduction technique, which replaces the snap-

shot of the full gradient by an unbiased estimate of the full gradient, and gains the linear convergence. Such the unbiased estimate of the full gradient sharply decreases the time consumption caused by the full gradient.

To present our analysis about the variance, we first present an unified framework, i.e. EUI which can express various of the existing variants. EUI mainly consists of three ingredients which should be considered when designing a good variant of SGD with the variance reduction technique. After analysing the variance in quantitative, the guide to design a good version of SGD has been provided. Furthermore, a new version of reduced variance SGD, i.e. EstimateVR has been proposed, and the convergence performance of EstimateVR has been analysed in theroetical. Finally, the extensive empirical studies verify that EstimateVR outperforms than the existing methods. The contributions of the paper are outlined as follows:

- The variance caused by the stochastic gradient in the underlying framework is analysed in quantitative. A lower bound of the variance is presented.
- A general framework, i.e. EUI is proposed to unify the existing variants of reduced variance SGD. The variance of the previous work has been analysed by using this framework.
- A new version of SGD with the variance reduction technique, i.e. EstimateVR is been proposed, which replaces the snapshot of the full gradient by using an estimate via a sampling strategy.
- The theoretical and extensive numerical evaluations verify that EstimateVR outperforms than other previous work significantly.

The paper is organised as follows. Section **??** reviews the recent work about SGD with the variance reduction technique. Section presents the general framework, i.e. EUI which unifies almost all the existing variants of SGD with the variance reduction technique. Section **??** analyses the variance of the previous work under the framework. Section **??** presents a new reduced variance SGD. Section demonstrates the extensive performance evaluations to verify our theoretical analysis.

Related work

EUI: the general framework of reduced variance SGD

Framework

We present a general framework denoted by EUI which contains a loop of the epochs. As illustrated in Algo. 1, an epoch consists of a number of iterations. EUI first identifies the sampling strategy by a probability distribution. Generally, the probability distribution is uniform so that the training data will be picked uniformly. The epoch size then is required to be identified, which is represented by the function $\mathcal{E}$. The size of an epoch will be shown to have a great impact on the variance, but it is non-trivial to identify a good value. Additionally, the function $\mathcal{U}$ stands for the update rule during the train of the parameter. A sequence of the local parameters, i.e. $\omega_{i_j}^t$ with $j \in \{1, 2, ..., m_s\}$ is obtained by performing the update of the parameter in an epoch. When the inner loop is completed, the global parameter ω^s will be updated by the sequence of the parameter $\omega_{i_j}^t$ with $j \in \{1, 2, ..., m_s\}$, which is shown by the function $\mathcal{I}$.

Algorithm 1 The general framework of variance reduced SGD: EUI

Require: $\omega^0 \in \mathbb{R}^d$. $\forall i \in [n]$, and $[n]$ represents $\{1, 2, ..., n\}$.

1: $P(i_t) \leftarrow \frac{1}{n}$ where $i_t \in \{1, 2, ..., n\}$. t is a positive integer;
2: **Epoch:** the sequence of the epoch size $\{m^0, m^1, ..., m^S\} \leftarrow \mathcal{E}([i_t])$;
3: **for** $s = 0, 1, 2, ..., S$ **do**
4: $\quad \omega_0^s = \tilde{\omega}^s$;
5: $\quad g = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{\omega}^s)$;
6: $\quad$ **for** $t = 0, 1, ..., m^s$ **do**
7: $\quad\quad v = \nabla f_{i_t}(\omega_{i_t}^t) - \nabla f_{i_t}(\tilde{\omega}^s)$;
8: $\quad\quad \gamma_t^s = v + g$;
9: $\quad\quad$ **Update:** $\omega_{t+1}^s = \mathcal{U}(\eta_t, \omega_t^s, \gamma_t^s)$;
$\quad$ **Identification:** $\tilde{w}^{s+1} \leftarrow \mathcal{I}([\omega_j^s])$ with $j \in \{1, 2, ..., m^s\}$;
return $\tilde{\omega}^S$.

The general framework is powerful so that almost all the existing versions of SGD with the variance reduction technique except for SAG (**?**) and SAGA (**?**). SAG and SAGA can be expressed by EUI with a little change. To keep the simplicity, we do not unify them into the framework, but the analysis of the variance in the following part will contain them. The Fig. 1 illustrates that existing versions of reduced variance SGD can be expressed by EUI by implementing the specific functions. Compared to the basic version, i.e. SVRG, the following algorithms implement those functions by using different strategies in order to improve SVRG. For example, SVRG achieves the linear convergence when the loss function $F(\omega)$ is Lipschitz-continuous, and the functions $f_i(\omega)$ with $i \in \{1, 2, ..., n\}$ are strongly convex. However, these assumptions can be relaxed to solve the non-strongly convex objective by designing different strategies and implement those functions.

The analysis of the variance

It is obvious that the research about the variance reduction technique is really active. Not limited in the Fig. 1, there exist various variants of SGD which using the variance reduction technique. But they usually adopts the same reduced variance stochastic gradient, i.e. Equ. 3, to update the parameter. We take SVRG as an example to analyse the improvement of other following work.

As illustrated in Equ. 3, the first item of v_t is the stochastic gradient which is denoted by the "variance source". The second item of v_t is denoted by the "variance reducer" which is used to reduce the variance. The third item of v_t is denoted by the " progressive direction" which makes sure that v_t will

Name	Algorithm						
	SVRG	S2GD	EMGD	SVR-GHT	Prox-SVRG	SVRG++	Katyusha
$\mathcal{E}$	$m^s = C$	$p(m^s) = \frac{(1-\breve{\mu}\eta)^{M-t}}{\sum_{t=1}^{M}(1-\breve{\mu}\eta)^{M-t}}$	$m^s = C$	$m^s = C$	$m^s = C$	$m^s = 2^s C$	$m^s = C$
$\mathcal{U}$	$\omega_t^s - \eta_t \gamma_t^s$	$\omega_t^s - \eta_t \gamma_t^s$	$\omega_t^s - \mathbb{B}_{\Delta_t} \eta_t \gamma_t^s$	$\mathbb{O}_k \omega_t^s - \eta_t \gamma_t^s$	$\omega_t^s - \eta_t \gamma_t^s$	$\omega_t^s - \eta_t \gamma_t^s$	$\omega_t^s - \eta_t \gamma_t^s$
$\mathcal{I}$	pick ω_j^s from {1,2,...,m^s} randomly $\omega_j^{m^s}$	$\omega_j^{m^s}$	$\omega_j^{m^s}$	$\omega_j^{m^s}$	$\sum_{j=1}^{m^s} \omega_j^s p(j)$	$\sum_{j=1}^{m^s} \omega_j^s p(j)$	$\begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} z_0 - \alpha \sum_{i=1}^{m^s} \tilde{\gamma}_t^s \\ \omega^{s-1} \\ x_0 - \frac{1}{3L} \sum_{j=1}^{m^s} \tilde{\gamma}_t^s \end{pmatrix}$

Figure 1: Many existing variants of SGD with the variance reduction technique can be described by the general framework EUI.

not be too far away from the full gradient when updating parameter in an epoch. To better understand the potential of the variance reduction technique, we illustrate another two situations i.e. SGD and GD whose update gradient are denoted by γ_t^{SGD} and γ_t^{GD}, respectively. As illustrated in Equ. 2, the variance of SGD is not reduced, and we refer its variance to the maximum. On the contrary, as illustrated in Equ. 4, the update gradient GD will not lead to the variance, and we thus refer the variance of GD as the minimum.

$$\gamma_t^{\mathrm{SGD}} = \nabla f_{i_t}(\omega_{i_t}^s) - \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}^s) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}^s); \quad (2)$$

$$\gamma_t = \nabla f_{i_t}(\omega_{i_t}^s) - \nabla f_{i_t}(\tilde{\omega}^s) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}^s); \quad (3)$$

$$\gamma_t^{\mathrm{GD}} = \nabla f_{i_t}(\omega_{i_t}^s) - \nabla f_{i_t}(\omega_{i_t}^t) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}^s); \quad (4)$$

It is obvious that the difference among the update gradients of SGD, GD and the variance reduction technique is the variance reducer. SGD has the maximal variance because its variance reducer is a constant which does not help to reduce the variance. GD has no variance because that its variance reducer reduce all the variance caused by the variance source. The variance reducer of the variance reduction technique is a tradeoff between those of SGD and GD, which consists of a constant input parameter, i.e. $\tilde{\omega}^s$ and a stochastic gradient, i.e. $\nabla f_{i_t}(\cdot)$. It does not reduce all the variance like that of GD because that the constant input parameter, i.e. $\tilde{\omega}^s$ is stale for the iterations in an epoch. In specific, the staleness of $\tilde{\omega}^s$ against $\omega_{i_t}^s$ causes the variance during the update in an epoch. Therefore, the variance caused by the stale parameter in the variance reducer will be accumulated with the iterative updates in an epoch. Such the staleness of the parameter can be measured by the distance d_t between $\omega_{i_t}^s$ and $\tilde{\omega}^s$ with $d_t = \| \omega_{i_t}^s - \tilde{\omega}^s \|^2$. $d_0 = \| \omega_{i_0} - \tilde{\omega}^s \|^2 = 0$ according to the framework. Therefore, we obtain Theorem 1 as follows.

Theorem 1. *If γ_t is p-dimensional, and can be denoted by $\gamma_t = (a_t^1, a_t^2 ..., a_t^p)$. After t iterations in the epoch, the distance d_t holds that $d_t \geq p\eta^2 \sum_{j=1}^{t} a_j^2$ when the learning rate, i.e. η_j with $j = 1, 2, ...m^s$ is a constant.*

As demonstrated in Theorem 1, the distance becomes large with a large learning rate η, a high dimension p, and the increase of the number of the iterations in an epoch, i.e. m^s. Given a specific optimisation objective where p is determined, if a constant learning rate is adopted, the variance is controlled by the design of the epoch size m^s and the average of the element of every dimension in the update gradient, i.e. $\frac{1}{p}\sum_{i=1}^{p} a_j^i$. Therefore, when the optimisation objective is determined, the variance is impacted by three aspects: the epoch size, the update rule and the parameter identification for the next iteration.

As illustrated in 1, the epoch size, i.e. m^s is designed as a constant in EMGD, SVR-GHT, Prox-SVRG, Katyusha, and designed as an ascending variable for SVRG++ and S2GD. It is noting that the design of an appropriate epoch size is not trivial. A large size of an epoch causes much variance due to the large distance according to Theorem 1. However, a small epoch size means that more epochs are required to keep the loss function converge within a given bound. Since the full gradient is required to be computed in an epoch, a large amount of epochs thus lead to many gradient calculations which are very time-consuming.

The update rule has an impact on the variance via a_j^i according to Theorem 1. Most variants of reduced variance SGD including S2GD, Prox-SVRG, SVRG++, and Katyusha adopt the same update rule with SVRG. EMGD updates the parameter within a bound in the high dimensional space. The bound is decaying so that the value of parameters will not be changed sharply, which decreases the variance. SVR-GHT introduces a hard-thresholding mechanism. Such the mechanism keeps the k largest entries in the parameter, and sets other entries to zero when updating the parameter. Since SVR-GHT is designed for a class of optimisation objective where the optimal parameter has at most k non-zero values, the value of a_j^i for an iteration in an epoch is thus not large for an iteration.

The identification of the parameters is to provide the initial parameters, i.e. ω^{s+1} for the next epoch. The variance

Algorithm 2 EstimateVR

Require: $\omega^0 \in \mathbb{R}^d$. $\forall i \in [n]$, and $[n]$ represents $\{1, 2, ..., n\}$. $\phi = 0.001$, $\alpha = 0.025$, $k^0 = n$, and $g^0 = 0$.

$P(i_t) \leftarrow \frac{1}{n}$ where $i_t \in \{1, 2, ..., n\}$. t is a positive integer;
for $s = 0, 1, 2, ..., S$ **do**
  $\omega_0^s = \tilde{\omega}^s$;
  for $t = 0, 1, ..., m-1$ **do**
    $v = \nabla f_{i_t}(\omega_t^s) - \nabla f_{i_t}(\tilde{\omega}^s)$;
    $\gamma_{t+1}^s = v + g^s$;
    $\omega_{t+1}^s = \omega_t^s - \eta\gamma_{t+1}^s$;
  $\tilde{\omega}^{s+1} = \omega_m^s$;
  $\tilde{\gamma} =$ **estimateFullGradient**$([\omega_{1:m}^s], \epsilon)$;
  $g^{s+1} = \tilde{\gamma}$;
return $\tilde{\omega}^S$.

in the current epoch will be introduced to the next epoch via the underlying identification of the parameter. Generally, the identification of the parameter is a tradeoff between the updates and the variance. First, we would like to use the updates which are obtained in the current epoch to train the parameters in the next epoch because those updates of the parameter help converge the loss function. Second, the variance existing in those updates is harmful to the convergence of the loss function. The more updates we use, the more variance may be introduced in the next epoch. As illustrated in Fig. 1, most of the previous work uses $\tilde{\omega}_t^s$ as the initial parameter of the next epoch, which contains all the updates of the current epoch. Prox-SVRG and SVRG++ use the average of the sequence of the parameters, i.e. $\frac{1}{m^s}\sum_{t=1}^{m^s}\omega_t^s$ which leads to less variance for the initial parameter of the next epoch, but disgards some updates of the parameter in the current epoch.

EstimateVR: a new reduced variance SGD

Although the variance reduction technique is effective to decrease the variance caused by the stochastic gradient, it has to keep a snapshot of the full gradient for every epoch. Unfortunately, the computation of the full gradient contains extensive gradient calculations which are extremely time-consuming. We design a new reduced variance SGD by using an estimate of the full gradient replacing the real computation of the full gradient to avoid the time-consuming gradient calculations.

As illustrated in Algo. 2, the new variant of SGD with the variance reduction technique is denoted by EstimateVR. EstimateVR provides an interface denoted by estimateFullGrdient which is used to implement the estimation of the full gradient. As illustrated in Fig. 2, we provide two implementations to estimate the full gradient which are denoted by sampleVR and reuseVR, respectively. sampleVR estimates the full gradient by using a subset of the instances; while reuseVR estimates the full gradient by using the average of the stale parameters during the current epoch.

sampleVR:	**reuseVR:**
$\rho_{s+1}^2 = \epsilon/s, k = -\frac{\log\frac{\alpha}{2}}{\rho_{s+1}^2}$;	$\nabla\dot{F} = \frac{1}{m}\sum_{t=1}^{m}\nabla f_{i_t}(\omega_{i_t}^s)$;
return $\frac{1}{k}\sum_{t=1}^{k}\nabla f_{i_t}(\omega_{i_m}^s)$;	**return** $\nabla\dot{F}$;

Figure 2: The function estimateFullGradient in Algo. 2 is presented with two implementations: sampleVR and reuseVR.

- sampleVR: As illustrated in Equ. 5, the k instances are sampled from the training data, and we obtain the k stochastic gradients. The average of them, i.e. $\frac{1}{k}\sum_{i=1}^{k}\nabla f_i(\tilde{\omega}^s)$ is used as the progressive direction for the next epoch. Since $\mathbb{E}\left(\frac{1}{k}\sum_{i=1}^{k}\nabla f_i(\tilde{\omega}^s)\right) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\tilde{\omega}^s)\right) = \nabla F(\tilde{\omega}^s)$ holds, the new update gradient, i.e. $\dot{\gamma_t}$ is the same with γ_t in probability, that is, $\mathbb{E}(\dot{\gamma_t}) = \mathbb{E}(\gamma_t) = \nabla F(\omega_{i_t}^s)$.

$$\dot{\gamma_t} = \nabla f_{i_t}(\omega_{i_t}^s) - \nabla f_{i_t}(\tilde{\omega}^s) + \frac{1}{k}\sum_{i=1}^{k}\nabla f_{i_k}(\tilde{\omega}^s); \quad (5)$$

  The estimate of the full gradient is denoted by $\dot{g} = \frac{1}{k}\sum_{i=1}^{k}\nabla f_i(\tilde{\omega}^s)$. Let $d^s = \dot{g} - g = (d_1^s, d_2^s, ..., d_p^s)$. Considering each element in d^s, i.e. d_i^s with $i \in \{1, 2, ..., p\}$, we obtain $P(d_i^s \geq \rho) \leq e^{-2k\rho^2}$ when sampling k instances from a large amount of the training data randomly. If $\alpha = p(d_i^s \notin [-\rho, \rho]) \leq e^{-2k\rho^2}$, then the least number of samples, i.e. k should satisfy

$$k \geq -\frac{\log\frac{\alpha}{2}}{\rho^2} \quad (6)$$

  according to Hoeffding's inequality so that we can acquire $(1-\alpha)$-confidence interval $[-\rho, \rho]$.
- reuseVR: As illustrated in Equ. 7, the average of the stochastic gradients is used to estimate the full gradient for the next epoch. The variance between the real full gradient and the estimation, i.e. $\nabla\dot{F} = \frac{1}{m}\sum_{t=0}^{m-1}\nabla f_{i_t}(\omega_t^s)$ becomes large with the increase of the epoch size m. An effective way to solve this problem is to estimate the full gradient by using the last n stochastic gradients, i.e. $\nabla\dot{F} = \frac{1}{n}\sum_{t=m-n}^{m-1}\nabla f_{i_t}(\omega_t^s)$.

$$\dot{\gamma_t} = \nabla f_{i_t}(\omega_{i_t}^s) - \nabla f_{i_t}(\tilde{\omega}^s) + \frac{1}{m}\sum_{t=0}^{m-1}\nabla f_{i_t}(\omega_t^s); \quad (7)$$

  . Since the stochastic gradients are computed in the iterations of the current epoch, the estimation of the full gradient does not lead to any more calculations of the gradient.

Although both sampleVR and reuseVR outperform the previous work in the empirical study, we provide the theoretical analysis of the convergence and the gradient complexity for sampleVR. Such those analysis for reuseVR will be left as the future work. As illustrated in Theorem 2 and 3, sampleVR converges at a linear rate, but the linear rate is not the best when comparing with SVRG and other following work. However, its gradient complexity is extremely smaller than the previous work because α is usually very small, for example 0.01. Therefore, sampleVR converges fast, and gains the best convergence performance in the extensive empirical studies.

Theorem 2. *Let* $\delta = \frac{4L\eta^2 m^s}{\eta(1-2\eta L)m^s - \frac{1}{\gamma}} < 1$*, and* d^s *in the* s_{th} *iteration be small enough, so that* $d^s \leq F(\tilde{\omega}^s) - F(\omega_*)$ *holds. EstimateVR has the linear convergence rate:* $F(\tilde{\omega}^{s+1}) - F(\omega_*) \leq \frac{1+2L}{2L}\delta[F(\tilde{\omega}^s) - F(\omega_*)]$*. Specifically,* $\mathbb{E} \parallel d^s \parallel^2 = 0$ *always holds, we thus obtain* $\mathbb{E}(F(\tilde{\omega}^{s+1}) - F(\omega_*)) \leq \delta\mathbb{E}([F(\tilde{\omega}^s) - F(\omega_*)])$.

Theorem 3. *Assume the objective needs* s *epochs to achieve to the* ϵ*, and then the gradient complexity is* $(1-\alpha)\left(m - \frac{1}{6}\frac{\log\frac{\alpha}{2}}{\rho_0^2}s(s+1)(2s+1)\right) + \alpha n$ *atomic gradient computation.*

Performance evaluation

Experimental settings

The existing variants of SGD with the variance reduction technique, including SVRG, S2GD, SVRG++, cheapSVRG have been used to conduct the performance comparison with our proposed algorithms, i.e. sampleVR and reuseVR. Those algorithms are evaluated on eight datasets, including ijcnn1, colon-cancer, duke-brease-cancer, a9a, mg, cpusmall, yearPredictionMSD, and space-ga. All of those datasets are public on the LibSVM website [1]. First, those algorithms are compared by conducting the $l2$-regularized logistic regression tasks on the datasets: ijcnn1, colon-cancer, duke-breast-cancer, and a9a. The loss function of the $l2$-regularised logistic regression tasks is:

$$\min_{\omega} \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-y_i\omega^{\mathrm{T}}x_i}) + \lambda \parallel \omega \parallel^2 \qquad (8)$$

. Second, we compare those algorithms by conducting ridge regression tasks on the other four datasets, i.e. mg, cpusmall, yearPredictionMSD, and space-ga. The loss function of the ridge regression tasks is:

$$\min_{\omega} \frac{1}{n}\sum_{i=1}^{n}\left(\omega^{\mathrm{T}}x_i - y_i\right)^2 + \lambda \parallel \omega \parallel^2 \qquad (9)$$

. We set λ to be 10^{-5}, and the learning rate to be 10^{-4} for all the evaluation tests. The epoch size m^s is set to be the size of the training data, i.e. n defaultly. The x-axis in all the figures is the computational cost measured by the number of gradient computations divided by the size of the training data, i.e. n. The y-axis in all the figures denotes the training loss residual which is the training loss minus the optimum. Here, the optimum is estimated by running the gradient descent for a long time. The fraction in the bracket of the legend of SVRG++ represents the initial size of the epoch divided by the default size of the epoch. The value in the bracket of the legend of sampleVR represents ϵ according to Algo. 2.

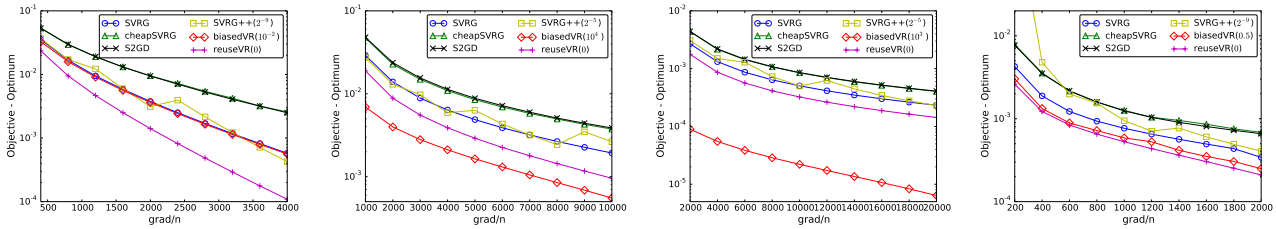
[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

$l2$-regularised logistic regression

As illustrated in Fig. 3, we compare the performance of all the algorithms by conducting the $l2$-regularised logistic regression tasks. It is obvious that our proposed algorithms, i.e. sampleVR and reuseVR make the training loss converge faster than other existing algorithms. The main reason is that both sampleVR and reuseVR replaces the computation of the full gradient with an estimation, and thus get rid of the time-consuming calculations of the gradient significantly. Additionally, the comparison of the performance of sampleVR with different settings of ϵ is shown in Fig. 4. sampleVR generally obtains a better performance with a larger ϵ. It is because that the number of the required sampled instances becomes small with a large ϵ according to Equ. 6 and Fig. 2. The computational cost is decreased with a large ϵ, which leads to the acceleration of sampleVR. However, as illustrated in Fig. 4(a) and 4(d), if ϵl is set to be too large, the performance will be impaired due to the variance between the full gradient and the estimation. In specific, a very large ϵ means that the number of the sampled instances is extremely small. Since the estimation of the full gradient is obtained by the computation of the stochastic gradients according to those instances, such the estimation has much variance which slows the convergence of the training loss.

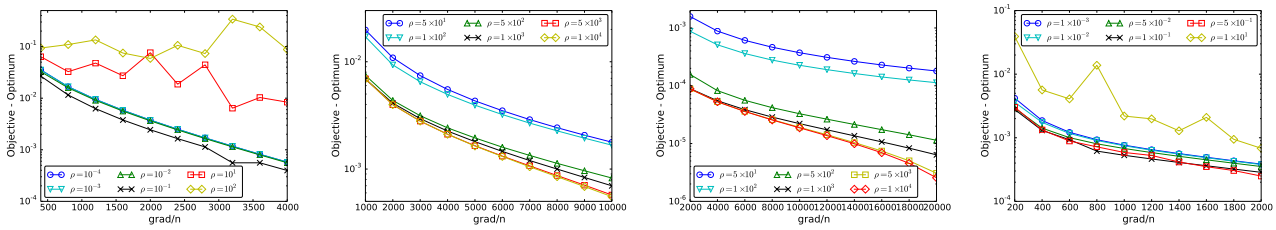
Ridge regression

As illustrated in Fig. 5, we report the comparison of the performance by using all the algorithms to conduct the ridge regression tasks. It is significant that reuseVR makes the loss function converge faster than the existing algorithms. It is because that reuseVR uses the average of the stochastic gradients during the iterations of the current epoch to estimate the full gradient for the next epoch, thus leading to no computational cost of the gradients. Additionally, sampleVR has a better performance for the datasets mg, cpusmall, and space-ga than the existing algorithms significantly. Although sampleVR does not outperform other algorithms for the dataset yearPredictionMSD at the beginning of the train process, its performance is comparable to the other algorithms, and finally shows the advantage over most of the existing algorithms. The main reason is that sampleVR takes an unbiased estimation of the full gradient, instead of costing much time to compute it. Therefore, sampleVR accelerates the convergence of the loss function than other algorithms. As illustrated in Fig. 6, the performance of sampleVR has been compared by varying the value of ϵ. It is significant that the variance becomes noticeable with the increase of ϵ. It is because that a large ϵ leads to a few instances which are sampled in the current epoch according to Equ. 6 and Fig. 2. Therefore, the estimation of the full gradient has much variance which slows the convergence of the training loss.



(a) ijcnn1 (b) colon-cancer (c) duke-breast-cancer (d) a9a

Figure 3: Both sampleVR and reuseVR make the training loss of the $l2$-regularised logistic regression tasks converge faster than the other existing algorithms.

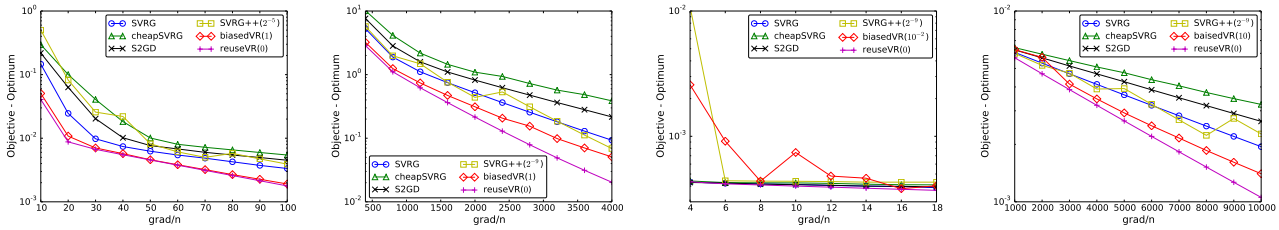


(a) ijcnn1 (b) colon-cancer (c) duke-breast-cancer (d) a9a

Figure 4: Generally, sampleVR with a large ϵ has a better performance for the the $l2$-regularised logistic regression tasks. However, the increase of the variance with a too large ϵ slows the convergence of the training loss.

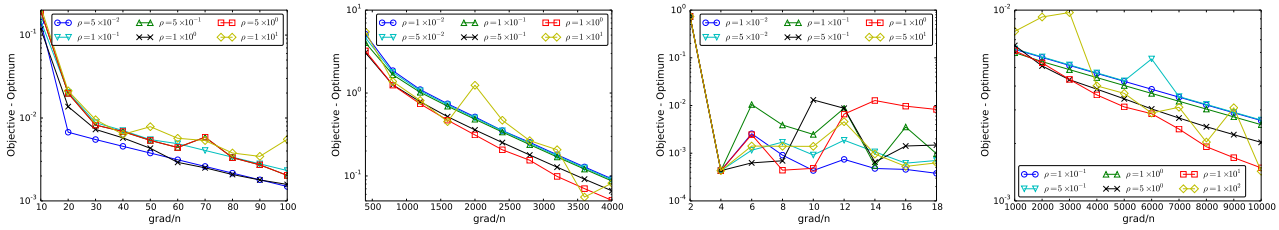
Moreover, as illustrated in Fig. 6(a), 6(b) and 6(d), an extremely small ϵ impairs the performance of sampleVR. The reason is that a small ϵ means that a large number of the instances are sampled for estimating the full gradient, and thus leads to much calculations of gradients. Such a large amount of calculations of gradients cost much time, and make the convergence of the loss function slow down.

Conclusion



(a) mg (b) cpusmall (c) yearPredictionMSD (d) space-ga

Figure 5: Generally, both sampleVR and reuseVR outperforms the other existing algorithms on the convergence of the training loss when conducting the ridge regression tasks.



(a) mg (b) cpusmall (c) yearPredictionMSD (d) space-ga

Figure 6: A large ϵ leads to the fast convergence of the training loss for sampleVR when conducting the ridge regression tasks. However, the performance of sampleVR is impaired due to the increase of variance when ϵ is set to be too large.