Stochastic Conjugate Gradient Algorithm With Variance Reduction

Xiao-Bo Jin, Xu-Yao Zhang, Kaizhu Huang, and Guang-Gang Geng

***Abstract*— Conjugate gradient (CG) methods are a class of important methods for solving linear equations and nonlinear optimization problems. In this paper, we propose a new stochastic CG algorithm with variance reduction[1] and we prove its linear convergence with the Fletcher and Reeves method for strongly convex and smooth functions. We experimentally demonstrate that the CG with variance reduction algorithm converges faster than its counterparts for four learning models, which may be convex, nonconvex or nonsmooth. In addition, its area under the curve performance on six large-scale data sets is comparable to that of the LIBLINEAR solver for the $L2$-regularized $L2$-loss but with a significant improvement in computational efficiency.**

***Index Terms*— Computational efficiency, covariance reduction, empirical risk minimization (ERM), linear convergence, stochastic conjugate gradient (CG).**

I. Introduction

EMPIRICAL risk minimization (ERM) is a principle in statistical learning theory for providing theoretical bounds on the performance of learning algorithms. ERM is defined as

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}) \tag{1}$$

where $\boldsymbol{w} \in R^d$ is the parameter of a machine learning model, n is the sample size, and each $f_i(\boldsymbol{w}) : R^d \to R$ estimates how well parameter $\boldsymbol{w}$ fits the data of the ith sample. This approach has been widely used to solve classification [1], regression [2], clustering [3], and ranking [4], [5], among others.

Stochastic gradient descent (SGD) [6] and its variants [7], [8] are the most widely used algorithms for minimizing the empirical risk (1) in many large-scale machine learning problems. Kingma and Ba [9] introduced the Adam method, which computes the adaptive learning rates for each parameter. Sutskever *et al.* [10] showed that SGD with momentum, using a well-designed random initialization and a slowly increasing schedule for the momentum parameter, could train both deep neural networks and recurrent neural networks. However, the success of these SGD variants heavily relies on the setting of the initial learning rate and the decay strategy of the learning rate.

The disadvantage of SGD is that the randomness introduces a variance that slows the convergence. Roux *et al.* [11] proposed the stochastic average gradient to achieve a variance reduction effect for SGD. Shalev-Shwartz and Zhang [12] introduced stochastic dual coordinate ascent to train convex linear prediction problems with a linear convergence rate. However, both methods require storing all gradients (or dual variables), thus making them unsuitable for complex applications where storing all gradients is impractical. The stochastic variance reduced gradient (SVRG) method proposed by Johnson and Zhang [13] accelerates the convergence of stochastic first-order methods by reducing the variance of the gradient estimates.

Another promising line of work is devoted to the stochasticization of the second-order quasi-Newton method, particularly the L-BFGS algorithm. Wang *et al.* [14] studied stochastic quasi-Newton methods for the nonconvex stochastic optimization. Mokhtari and Ribeiro [15] used stochastic gradients in *lieu* of deterministic gradients to determine the descent directions and approximate the objective function's curvature. Moritz *et al.* [16] introduced a stochastic variant of L-BFGS [stochastic limited-memory Broyden–Fletcher–Goldfarb–Shanno (SLBFGS)] that incorporated the idea of variance reduction. Gower *et al.* [17] proposed a new limited-memory stochastic block BFGS update with the variance reduction approach SVRG. However, the limited-memory stochastic quasi-Newton methods often require m vector pairs to efficiently compute product $H\nabla f$ (H is the Hessian), which may be prohibitive in the case of limited memory for large-scale machine learning problems.

Fletcher and Reeves (FR) [18] first showed how to extend the linear conjugate gradient (CG) method to nonlinear functions, which is called the FR method. Polak and Ribiere (PR) [19] proposed another CG method known as the PR method. Gilbert and Nocedal proved that the modified PR method $\beta_k^{\mathrm{PR}+} = \max\{0, \beta_k^{\mathrm{PR}}\}$ with Wolfe-Powell linear search is globally convergent under a sufficient descent condition. In practical computation, the PR method, the Hestenes and Stiefel [20] method, and the Liu and Storey [21] method are generally believed to be the most efficient CG methods

Manuscript received September 20, 2017; revised May 26, 2018 and August 3, 2018; accepted September 2, 2018. Date of publication September 27, 2018; date of current version April 16, 2019. This work was partially supported by the Fundamental Research Funds for the Henan Provincial Colleges and Universities in Henan University of Technology (2016RCJH06), the National Natural Science Foundation of China (61103138, 61602154, 61375039, 61473236, U1804326), National Key Research & Development Program (2016YFD0400104-5) and the National Basic Research Program of China (2012CB316301). *(Corresponding author: Guang-Gang Geng.)*

X.-B. Jin is with the Department of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China (email: xbjin9801@gmail.com).

X.-Y. Zhang is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.

K. Huang is with the Department of Electrical and Electronic Engineering, Xian Jiaotong-Liverpool University, Suzhou 215123, China.

G.-G. Geng is with the Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China (e-mail: gengguanggang@cnnic.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2868835

[1]CGVR algorithm is available on github: https://github.com/xbjin/cgvr

because they essentially restart if a bad direction occurs. Although the convergence of the conjugate descent [22] method, the Dai and Yuan [23] method, and the FR method has been established, their numerical results are not good.

In this paper, we propose a stochastic variant of the CG method called CG with variance reduction (CGVR) that integrates the variance reduction method. The proposed method has the following advantages.

1) It only requires a few iterations to quickly converge because of the idea of SVRG, but it converges more quickly than SVRG because of the use of the CG rather than the general gradient.
2) The parameters of CGVR are not sensitive to the data sets, and the empirical settings always work well; in particular, its step size is determined through a Wolfe line search.
3) It only stores the last gradient vector similar to CG; in contrast, the quasi-Newton variants often store a set of vector pairs.
4) CGVR with $L2$-regularized $L2$-loss achieves a generalization performance comparable to that of the LIBLINEAR solver [24] in less running time for large-scale machine learning problems.

Our contributions are as follows.

1) We propose a stochastic variant of the CG method with variance reduction, where both the Wolfe line search and the gradient computation are built on subsamples.
2) We prove the linear convergence of CGVR with the FR method for strongly convex and smooth functions.
3) We conduct a series of experiments on six large-scale data sets with four state-of-the-art learning models, which may be convex, nonconvex, or nonsmooth. The experimental results show that CGVR converges faster on large-scale data sets than several other algorithms, and its area under the curve (AUC) performance with $L2$-regularized $L2$-loss is comparable to that of the LIBLINEAR solver with a significant improvement in the computational efficiency.

The remainder of this paper is organized as follows. In Section II, we briefly introduce the SVRG and SLBFGS algorithms. In Section III, we propose our CGVR algorithm and prove its linear convergence for strongly convex and smooth functions. In Section IV, we conduct experiments on convergence and generalization to compare CGVR with its counterparts. Finally, we draw some conclusions in Section V.

II. SVRG and SLBFGS Algorithms

A. SVRG Algorithm

The SVRG algorithm was proposed by Johnson and Zhang [13] for optimizing (1) and is depicted in Algorithm 1.

There are two loops in Algorithm 1. In the outer loop, the full gradient $\boldsymbol{u}_k$ is computed. We retain a snapshot $\boldsymbol{x}_0$ of $\boldsymbol{w}$ after every m SGD iterations. In the inner loop, we randomly select an example from the data set X to produce a variance-reduced gradient estimate (see [13, Th. 1]). There are two options to select the next $\boldsymbol{w}$, e.g., $\boldsymbol{w}_{k+1}$. Although Option I is a better choice than Option II because it takes more iterations to obtain the next $\boldsymbol{w}$, the convergence analysis is only available for Option II [13].

Algorithm 1 SVRG

Given $\boldsymbol{w}_0$, update frequency m, step size α
for $k = 0, 1, \cdots, T$ **do**
  $\boldsymbol{u}_k = \nabla f\ (\boldsymbol{w}_k)$
  $\boldsymbol{x}_0 = \boldsymbol{w}_k$
  for $t = 0, 1, \cdots, m-1$ **do**
    Randomly select $i_t \in \{1, 2, \cdots, n\}$
    $\boldsymbol{g}_t = \nabla f_{i_t}(\boldsymbol{x}_t) - \nabla f_{i_t}(\boldsymbol{x}_0) + \boldsymbol{u}_k$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha \boldsymbol{g}_t$
  end for
  Option I: $\boldsymbol{w}_{k+1} = \boldsymbol{x}_m$
  Option II: $\boldsymbol{w}_{k+1}$ for randomly chosen $t \in \{1, 2, \cdots, m\}$
end for

B. SLBFGS

For a subset $\mathcal{S} \subset \{1, 2, \ldots, n\}$, we define the subsampled function $f_{\mathcal{S}}(\boldsymbol{w})$ as

$$f_{\mathcal{S}}(\boldsymbol{w}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} f_i(\boldsymbol{w}) \tag{2}$$

where $|\mathcal{S}|$ denotes the number of elements in the set $\mathcal{S}$. Correspondingly, our algorithm uses the stochastic estimates of the gradient $\nabla f_{\mathcal{S}}$. In addition, we use stochastic approximations for the inverse Hessian $\nabla^2 f_{\mathcal{T}}$, where $\mathcal{T} \subset \{1, 2, \ldots, n\}$ is different from $\mathcal{S}$ to decouple the estimation of the gradient from the estimation of the Hessian.

The algorithm performs a full gradient computation every m iterations and updates the inverse Hessian approximation every L iterations. The vector $\boldsymbol{y}_r$ is computed by the product of the stochastic approximation of the Hessian and the vector $\boldsymbol{s}_r$, where $\boldsymbol{s}_r$ is the difference of two consecutive sequences with the length L. The product $H_r \boldsymbol{g}_t$ is directly obtained from the two-loop recursion, whose inputs are the most recent M vector pairs $\{(\boldsymbol{s}_j, \boldsymbol{y}_j)\}_{j=r-M+1}^{r}$.

III. Stochastic Conjugate Gradient With Variance Reduction

Although SVRG accelerates the convergence of SGD by reducing the variance of the gradient estimates, which is sensitive to the learning rate. SLBFGS requires M vector pairs to compute the product $H \nabla f$, and it needs to calculate the Hessian matrix H. In the following, we propose a new algorithm called stochastic CGVR to overcome the above-mentioned disadvantages.

A. Framework of Algorithm

We adapt the CG algorithm [25] from SVRG to obtain the CGVR algorithm in Algorithm 3. We compute a variance-reduced gradient $\boldsymbol{g}_{t+1}$ on the set $\mathcal{S}_{k,t} \subset \{1, 2, \ldots, n\}$, which is randomly generated in the tth loop of the kth iteration

$$\boldsymbol{g}_{t+1} = \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_{t+1}) - \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_0) + \boldsymbol{u}_k \tag{3}$$

Algorithm 2 Stochastic L-BFGS

Initialize $r = 0$
$H_0 = I$
for $k = 0, 1, \cdots, T$ **do**
  $\boldsymbol{u}_k = \nabla f(\boldsymbol{w}_k)$
  $\boldsymbol{x}_0 = \boldsymbol{w}_k$
  for $t = 0, 1, \cdots, m-1$ **do**
    Sample a minibatch $\mathcal{S}_{k,t} \subset \{1, 2, \cdots, n\}$
    $\boldsymbol{g}_t = \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t) - \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_0) + \boldsymbol{u}_k$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha H_r \boldsymbol{g}_t$
    if t mod $L == 0$ **then**
      $r = r + 1$
      $\boldsymbol{v}_r = \frac{1}{L}\sum_{j=t-L}^{t-1} \boldsymbol{x}_j$
      Sample a minibatch $\mathcal{T}_r \subset \{1, 2, \cdots, n\}$
      $\boldsymbol{s}_r = \boldsymbol{v}_r - \boldsymbol{v}_{r-1}$
      $\boldsymbol{y}_r = \nabla^2 f_{\mathcal{T}_r}(\boldsymbol{v}_r)\boldsymbol{s}_r$
      Compute $H_r \boldsymbol{g}_t$ with $\boldsymbol{g}_t$ and $\{(\boldsymbol{s}_j, \boldsymbol{y}_j)\}_{j=r-M+1}^{r}$ by two-loop recursion
    end if
  end for
  Option I: $\boldsymbol{w}_{k+1} = \boldsymbol{x}_m$
  Option II: $\boldsymbol{w}_{k+1}$ for randomly chosen $t \in \{1, 2, \cdots, m\}$
end for

where $\boldsymbol{g}_{t+1}$ corresponds to $\nabla f(\boldsymbol{x}_{t+1})$ in the CG algorithm, and $\nabla f_{\mathcal{S}_{k,t}}(\cdot)$ is computed in (2).

PR is a popular and important method in CG, which defines the parameter β_{t+1} as follows:

$$\beta_{t+1}^{\mathrm{PR}} = \frac{\boldsymbol{g}_{t+1}^T(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)}{\boldsymbol{g}_t^T \boldsymbol{g}_t}. \tag{4}$$

Simultaneously, the Fletcher–Reeves method uses another approach to compute β_{t+1}

$$\beta_{t+1}^{\mathrm{FR}} = \frac{\|\boldsymbol{g}_{t+1}\|^2}{\|\boldsymbol{g}_t\|^2}. \tag{5}$$

We use a trick to set $\beta_{t+1} = 0$ and restart [26] the iteration with the steepest descent step at the beginning of each iteration. Restarting will periodically refresh the algorithm and erase the old information that may not be beneficial. Nocedal *et al.* (see [25, eq. (5.51), p. 124]) provide a strong theoretical result about restarting, it leads to m-step quadratic convergence, that is,

$$\|\boldsymbol{x}_{t+m} - \boldsymbol{x}^*\| = O(\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2) \tag{6}$$

where $\boldsymbol{x}^*$ is a local minimizer of the function.

The search direction $\boldsymbol{p}_t$ may fail to be a descent direction unless α_t satisfies certain conditions. We can avoid this situation by requiring the step length α_t to satisfy the strong Wolfe conditions, which are

$$f(\boldsymbol{x}_t + \alpha_t \boldsymbol{p}_t) \le f(\boldsymbol{x}_t) + c_1 \alpha_t \nabla f(\boldsymbol{x}_t)^T \boldsymbol{p}_t \tag{7}$$

$$|\nabla f(\boldsymbol{x}_t + \alpha_t \boldsymbol{p}_t)^T \boldsymbol{p}_t| \le -c_2 \nabla f(\boldsymbol{x}_t)^T \boldsymbol{p}_t \tag{8}$$

where $0 < c_1 < c_2 < 1$. In our experiments, we computed parameter β as

$$\beta_{t+1}^{\mathrm{PR}+} = \max\left\{\beta_{t+1}^{\mathrm{PR}}, 0\right\} \tag{9}$$

Algorithm 3 Stochastic CG With Variance Reduction

Given $\boldsymbol{w}_0$
$\boldsymbol{h}_0 = \nabla f(\boldsymbol{w}_0)$
for $k = 0, 1, \cdots, T-1$ **do**
  $\boldsymbol{u}_k = \nabla f(\boldsymbol{w}_k)$
  $\boldsymbol{x}_0 = \boldsymbol{w}_k$
  $\boldsymbol{g}_0 = \boldsymbol{h}_k$
  $\boldsymbol{p}_0 = -\boldsymbol{g}_0$
  for $t = 0, \cdots, m-1$ **do**
    Sample a minibatch $\mathcal{S}_{k,t} \subset \{1, 2, \cdots, N\}$
    Call the line search algorithm to find α_t approximately optimize:
    $$\min_{\alpha} f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t + \alpha \boldsymbol{p}_t).$$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \alpha_t \boldsymbol{p}_t$
    Compute $\boldsymbol{g}_{t+1} = \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_{t+1}) - \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_0) + \boldsymbol{u}_k$
    Compute β by
    Option I:
    $$\beta_{t+1}^{\mathrm{PR}+} = \max\left\{\frac{\boldsymbol{g}_{t+1}^T(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)}{\boldsymbol{g}_t^T \boldsymbol{g}_t}, 0\right\}$$
    Option II:
    $$\beta_{t+1}^{\mathrm{FR}} = \frac{\|\boldsymbol{g}_{t+1}\|^2}{\|\boldsymbol{g}_t\|^2}$$
    $\boldsymbol{p}_{t+1} = -\boldsymbol{g}_{t+1} + \beta_{t+1}\boldsymbol{p}_t$
  end for
  $\boldsymbol{h}_{k+1} = \boldsymbol{g}_m$
  Option I: $\boldsymbol{w}_{k+1} = \boldsymbol{x}_m$
  Option II: $\boldsymbol{w}_{k+1} = \boldsymbol{x}_t$ for randomly chosen $t \in \{0, 1, \cdots, m-1\}$
end for

which leads to the PR$^+$ method; then, the strong Wolfe condition ensures that the descent property holds.

Note that

$$\nabla f(\boldsymbol{x}_t) = \mathbb{E}[\boldsymbol{g}_t]. \tag{10}$$

CGVR uses $f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t + \alpha \boldsymbol{p}_t)$ rather than $f(\boldsymbol{x}_t + \alpha \boldsymbol{p}_t)$ to search for the steps that satisfy the following conditions:

$$f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t + \alpha_t \boldsymbol{p}_t) \le f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t) + c_1 \alpha_t \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t)^T \boldsymbol{p}_t \tag{11}$$

$$|\nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t + \alpha_t \boldsymbol{p}_t)^T \boldsymbol{p}_t| \le -c_2 \nabla_{\mathcal{S}_{k,t}} f(\boldsymbol{x}_t)^T \boldsymbol{p}_t. \tag{12}$$

Although $f(\boldsymbol{x}_t + \alpha \boldsymbol{p}_t)$ is also possible, using $f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t + \alpha \boldsymbol{p}_t)$ for the linear search will clearly be faster. The values of $c_1 = 10^{-4}$ and $c_2 = 0.1$ are commonly used in the CG algorithm. The initial search step is set to 1.

Since $\boldsymbol{g}_t$ in the CGVR algorithm replaces the role of $\nabla f(\boldsymbol{x}_t)$ in the classical CG algorithm, to borrow some conclusions of the CG algorithm, we also use the following condition in the convergence analysis:

$$\left|\boldsymbol{g}_{t+1}^T \boldsymbol{p}_t\right| \le -c_2 \boldsymbol{g}_t^T \boldsymbol{p}_t. \tag{13}$$

The ideal step length would be the global minimizer of the univariate function $\phi(\cdot)$ defined as

$$\phi(\alpha) = f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t + \alpha \boldsymbol{p}_t). \tag{14}$$

We can perform an inexact line search to identify a step length. The line search algorithm [25] is divided into two steps. The first step begins from an initial estimate α_1 and constantly increases this estimate until it finds a suitable step length or a range that includes the desired step length. The second step is invoked by calling a function called *zoom* that successively decreases the size of the interval until an acceptable step length is identified.

We stop the line search and ***zoom*** procedure if it cannot obtain a lower function value after 20 iterations. In the ***zoom*** procedure, we use the middle point of the interval as a new candidate rather than complex quadratic, cubic, or bisection interpolation. These tricks work well in practice.

Finally, we use Option I to obtain the next $\boldsymbol{w}$, but the convergence analysis is only available for Option II.

B. Convergence Analysis

For the convenience of discussion, we define

$$\delta_f(\boldsymbol{x}) = f(\boldsymbol{x}) - f(\boldsymbol{w}_*), \quad f_t = f(\boldsymbol{x}_t), \quad \nabla f_t = \nabla f(\boldsymbol{x}_t). \tag{15}$$

We can now investigate the convergence of the CGVR algorithm by updating β_t with Fletcher–Reeves update (5) (Option II).

In the following discussion, we use β_t to represent β_t^{FR} unless otherwise specified. Our analysis uses the following assumptions.

Assumption 1: The CGVR algorithm is implemented with a step length α_t that satisfies $\alpha_t \in [\alpha_{lo}, \alpha_{hi}]$ $(0 < \alpha_{lo} < \alpha_{hi})$ and condition (13) with $c_2 < 1/5$.

Assumption 2: The function f_i is twice continuously differentiable for each $1 \le i \le n$, and there exist constants $0 < \lambda \le \Lambda$ such that

$$\lambda I \preceq \nabla^2 \ f_{\mathcal{S}}(\boldsymbol{x}) \preceq \Lambda I \tag{16}$$

for all $\boldsymbol{x} \in \mathbb{R}^d$ and $\mathcal{S} \subset \{1, 2, \ldots, N\}$.

Assumption 3: There exists $\hat{\beta} < 1$ such that

$$\beta_{t+1} = \frac{\|\boldsymbol{g}_{t+1}\|^2}{\|\boldsymbol{g}_t\|^2} \le \hat{\beta}. \tag{17}$$

In the following Lemmas 1 and 2, we estimate the lower bound of $\|\nabla f(\boldsymbol{x})\|$ and the upper bound of $\mathbb{E}[\|\boldsymbol{g}_t\|^2]$, whose proofs are provided in [16, Lemmas 5 and 6].

Lemma 1: Suppose that f is continuously differentiable and strongly convex with parameter λ. Let $\boldsymbol{w}_*$ be the unique minimizer of f. Then, for any $\boldsymbol{x} \in \mathbb{R}^d$, we have

$$\|\nabla f(\boldsymbol{x})\|^2 \ge 2\lambda \delta_f(\boldsymbol{x}). \tag{18}$$

Lemma 2: Let $\boldsymbol{w}_*$ be the unique minimizer of f. Let $\boldsymbol{u}_k = \nabla f(\boldsymbol{w}_k)$, and let $\boldsymbol{g}_t = \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{x}_t) - \nabla f_{\mathcal{S}_{k,t}}(\boldsymbol{w}_k) + \boldsymbol{u}_k$ be the variance-reduced stochastic gradient. Taking an expectation with respect to $\mathcal{S}_{k,t}$, we obtain

$$\mathbb{E}[\|\boldsymbol{g}_t\|^2] \le 4\Lambda \ (\delta_f(\boldsymbol{x}_t) + \delta_f(\boldsymbol{w}_k)). \tag{19}$$

The following Theorem 1 is used to estimate the upper and lower bounds of $\boldsymbol{g}_t^T \boldsymbol{p}_t / \|\boldsymbol{g}_t\|^2$, which are proven using the mathematical induction method [25, Lemma 5.6].

Theorem 1: Suppose that the CGVR (or CG) algorithm is implemented with the step length α_t that satisfies condition (13) with $0 < c_2 < 1/2$; then, the FR method [25], [27] generates descent directions $\boldsymbol{p}_k$ that satisfy

$$-\frac{1}{1-c_2} \le \frac{\boldsymbol{g}_t^T \boldsymbol{p}_t}{\|\boldsymbol{g}_t\|^2} \le \frac{2c_2 - 1}{1 - c_2}. \tag{20}$$

The following two theorems state our main results.

Theorem 2: Suppose that Assumptions 1 and 3 hold for Algorithm 3. Then, for any t, we have

$$\mathbb{E}[\|\boldsymbol{p}_t\|^2] \le \eta(t)\mathbb{E}[\|\boldsymbol{g}_0\|^2] \tag{21}$$

where

$$\eta(t) = \frac{2}{1-\hat{\beta}}\hat{\beta}^t - \frac{1+\hat{\beta}}{1-\hat{\beta}}\hat{\beta}^{2t}. \tag{22}$$

Proof: Combining condition (13) with Theorem (1), we have

$$\begin{aligned} -\boldsymbol{g}_t^T \boldsymbol{p}_{t-1} &\le -c_2 \boldsymbol{g}_{t-1}^T \boldsymbol{p}_{t-1} \\ &\le \frac{c_2}{1-c_2}\|\boldsymbol{g}_{t-1}\|^2. \end{aligned} \tag{23}$$

According to Assumption 3, we obtain

$$\mathbb{E}[\|\boldsymbol{g}_t\|^2] \le \hat{\beta}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2]. \tag{24}$$

Then, we use (23) and (24) to bound $\mathbb{E}[\|\boldsymbol{p}_t\|^2]$

$$\begin{aligned} \mathbb{E}[\|\boldsymbol{p}_t\|^2] &= \mathbb{E}[\|\beta_t \boldsymbol{p}_{t-1} - \boldsymbol{g}_t\|^2] \\ &= \mathbb{E}[\|\boldsymbol{g}_t\|^2] - 2\beta_t E\left[\boldsymbol{g}_t^T \boldsymbol{p}_{t-1}\right] + \beta_t^2 \mathbb{E}[\|\boldsymbol{p}_{t-1}\|^2] \\ &\le \hat{\beta}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] + \frac{2\hat{\beta}c_2}{1-c_2}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] \\ &\quad + \hat{\beta}^2\mathbb{E}[\|\boldsymbol{p}_{t-1}\|^2] \\ &= \hat{\beta}\frac{1+c_2}{1-c_2}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] + \hat{\beta}^2\mathbb{E}[\|\boldsymbol{p}_{t-1}\|^2]. \end{aligned} \tag{25}$$

According to the monotonically increasing property of the function $(1+x)/(1-x)$ with $0 < x < 1$ and $c_2 < 1/5 < 1/3$, we can conclude that

$$\frac{1+c_2}{1-c_2} < \frac{1+1/3}{1-1/3} = 2. \tag{26}$$

Thus, we can immediately obtain the following inequality:

$$\mathbb{E}[\|\boldsymbol{p}_t\|^2] \le 2\hat{\beta}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] + \hat{\beta}^2\mathbb{E}[\|\boldsymbol{p}_{t-1}\|^2]. \tag{27}$$

At the beginning of the kth iteration, we have

$$\boldsymbol{p}_0 = -\boldsymbol{g}_0. \tag{28}$$

Furthermore, we unfold (24) until we reach $\boldsymbol{g}_0$

$$\mathbb{E}[\|\boldsymbol{g}_t\|] \le \hat{\beta}^t \mathbb{E}[\|\boldsymbol{g}_0\|]. \tag{29}$$

According to (27) and (29), we further obtain

$$\begin{aligned} \mathbb{E}[\|\boldsymbol{p}_t\|^2] &\le 2\hat{\beta}\Big(\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] + \hat{\beta}^2\mathbb{E}[\|\boldsymbol{g}_{t-2}\|^2] \\ &\qquad + \cdots + (\hat{\beta}^2)^{t-1}\mathbb{E}[\|\boldsymbol{g}_0\|^2]\Big) + (\hat{\beta}^2)^t\mathbb{E}[\|\boldsymbol{p}_0\|^2] \\ &= 2\hat{\beta}\Big(\hat{\beta}^{t-1}\mathbb{E}[\|\boldsymbol{g}_0\|^2] + (\hat{\beta}^2)\hat{\beta}^{t-2}\mathbb{E}[\|\boldsymbol{g}_0\|^2] \\ &\qquad + \cdots + (\hat{\beta}^2)^{t-1}\mathbb{E}[\|\boldsymbol{g}_0\|^2]\Big) + (\hat{\beta}^2)^t\mathbb{E}[\|\boldsymbol{g}_0\|^2] \end{aligned}$$

$$
\begin{aligned}
&= 2\hat{\beta}\mathbb{E}[\|\boldsymbol{g}_0\|^2]\sum_{j=0}^{t-1}(\hat{\beta}^2)^j\hat{\beta}^{t-1-j} + (\hat{\beta}^2)^t\mathbb{E}[\|\boldsymbol{g}_0\|^2] \\
&= 2\hat{\beta}^t\mathbb{E}[\|\boldsymbol{g}_0\|^2]\sum_{j=0}^{t-1}\hat{\beta}^j + (\hat{\beta}^2)^t\mathbb{E}[\|\boldsymbol{g}_0\|^2] \\
&= \left(2\hat{\beta}^t\frac{1-\hat{\beta}^t}{1-\hat{\beta}} + \hat{\beta}^{2t}\right)\mathbb{E}[\|\boldsymbol{g}_0\|^2] \\
&= \frac{1}{1-\hat{\beta}}(2\hat{\beta}^t - (\hat{\beta}+1)\hat{\beta}^{2t})\mathbb{E}[\|\boldsymbol{g}_0\|^2] \\
&= \eta(t)\mathbb{E}[\|\boldsymbol{g}_0\|^2] \qquad (30)
\end{aligned}
$$

where

$$\eta(t) = \frac{2}{1-\hat{\beta}}\hat{\beta}^t - \frac{1+\hat{\beta}}{1-\hat{\beta}}\hat{\beta}^{2t}. \qquad (31)$$

□

Theorem 3: Suppose that Assumptions 1–3 hold. Let $\boldsymbol{w}_*$ be the unique minimizer of f. Then, for all $k \geq 0$, we have

$$\mathbb{E}[f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)] \leq \xi^k\mathbb{E}[f(\boldsymbol{w}_0) - f(\boldsymbol{w}_*)] \qquad (32)$$

where the convergence rate is given as

$$\xi = \frac{1 + \hat{\beta}\alpha_{hi}\Lambda\ (m-1) + 4\Lambda^2\alpha_{hi}^2/(1-\hat{\beta})^2}{(2\alpha_{lo}\lambda - \hat{\beta}\alpha_{hi}\Lambda)m + \hat{\beta}\alpha_{hi}\Lambda} < 1 \qquad (33)$$

assuming that we choose $\alpha_{hi}/\alpha_{lo} < \lambda/(\hat{\beta}\Lambda)$ and a sufficiently large m to satisfy

$$m \geq \frac{1 - 2\hat{\beta}\alpha_{hi}\Lambda + 4\Lambda^2\alpha_{hi}^2/(1-\hat{\beta})^2}{2\alpha_{lo}\lambda - 2\hat{\beta}\alpha_{hi}\Lambda}. \qquad (34)$$

Proof: Using the Lipschitz continuity of ∇f and Assumption 2, we have

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \nabla f(\boldsymbol{x}_t)^T(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t) + \frac{\Lambda}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2. \qquad (35)$$

Note that $\boldsymbol{p}_0 = -\boldsymbol{g}_0$; then, we have $\nabla f_0^T E[\boldsymbol{p}_0] = -\|\nabla f_0\|^2$. Since $\boldsymbol{p}_t = -\boldsymbol{g}_t + \beta_t\boldsymbol{p}_{t-1} (t \geq 1)$ and the random variables $\boldsymbol{g}_t$ and $\boldsymbol{p}_{t-1}$ are independent, with (10), (13), and (20), we have

$$
\begin{aligned}
\nabla f_t^T\mathbb{E}[\boldsymbol{p}_t] &= \mathbb{E}[\boldsymbol{g}_t]^T\mathbb{E}[-\boldsymbol{g}_t + \beta_t\boldsymbol{p}_{t-1}] \\
&= -\|\nabla f_t\|^2 + \beta_t\mathbb{E}\left[\boldsymbol{g}_t^T\boldsymbol{p}_{t-1}\right] \\
&\leq -\|\nabla f_t\|^2 - \beta_t c_2\mathbb{E}\left[\boldsymbol{g}_{t-1}^T\boldsymbol{p}_{t-1}\right] \\
&\leq -\|\nabla f_t\|^2 + \beta_t\frac{c_2}{1-c_2}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] \\
&\leq -\|\nabla f_t\|^2 + \frac{1}{4}\hat{\beta}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2]. \qquad (36)
\end{aligned}
$$

The last expression is obtained from the monotonically increasing characteristic of the function $x/(1-x)$. When $c_2 < 1/5$

$$\frac{c_2}{1-c_2} < \frac{1/5}{1-1/5} = \frac{1}{4}. \qquad (37)$$

Taking expectations on both sides of (35), we obtain

$$
\begin{aligned}
\mathbb{E}[f_{t+1}] &\leq f_t + \nabla f_t^T\mathbb{E}[(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t)] \\
&\quad + \frac{\Lambda}{2}\mathbb{E}[\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2] \\
&\leq f_t + \alpha_t\left(-\|\nabla f_t\|^2 + \frac{1}{4}\hat{\beta}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2]\right) \\
&\quad + \frac{\Lambda\alpha_t^2}{2}\mathbb{E}[\|\boldsymbol{p}_t\|^2] \\
&\leq f_t - \alpha_{lo}\|\nabla f_t\|^2 + \frac{\hat{\beta}\alpha_{hi}}{4}\mathbb{E}[\|\boldsymbol{g}_{t-1}\|^2] \\
&\quad + \frac{\Lambda\alpha_{hi}^2}{2}\eta(t)\cdot 4\Lambda\ (\delta_f(\boldsymbol{x}_0) + \delta_f(\boldsymbol{w}_k)) \\
&\leq f_t - 2\alpha_{lo}\lambda\delta_f(\boldsymbol{x}_t) + \hat{\beta}\alpha_{hi}\Lambda\ (\delta_f(\boldsymbol{x}_{t-1}) + \delta_f(\boldsymbol{w}_k)) \\
&\quad + \tau\eta(t)\delta_f(\boldsymbol{w}_k) \qquad (38)
\end{aligned}
$$

where

$$\tau = 4\Lambda^2\alpha_{hi}^2. \qquad (39)$$

Summing over $t = 0, 1, \ldots, m-1$ and using a telescoping sum, we obtain

$$
\begin{aligned}
\mathbb{E}[f_m] \leq \mathbb{E}[f_0] &- 2\alpha_{lo}\lambda\left(\sum_{t=0}^{m-1}\mathbb{E}[\delta_f(\boldsymbol{x}_t)]\right) \\
&+ \hat{\beta}\alpha_{hi}\Lambda\sum_{t=0}^{m-2}(\mathbb{E}[\delta_f(\boldsymbol{x}_t)] + \mathbb{E}[\delta_f(\boldsymbol{w}_k)]) \\
&+ \tau\mathbb{E}[\delta_f(\boldsymbol{w}_k)]\sum_{t=0}^{m-1}\eta(t). \qquad (40)
\end{aligned}
$$

We now compute $\sum_{t=0}^{m-1}\eta(t)$

$$
\begin{aligned}
\sum_{t=0}^{m-1}\eta(t) &= \sum_{t=0}^{m-1}\frac{2}{1-\hat{\beta}}\hat{\beta}^t - \frac{1+\hat{\beta}}{1-\hat{\beta}}(\hat{\beta}^2)^t \\
&= \frac{2}{1-\hat{\beta}}\frac{1-\hat{\beta}^m}{1-\hat{\beta}} - \frac{1+\hat{\beta}}{1-\hat{\beta}}\frac{1-\hat{\beta}^{2m}}{1-\hat{\beta}^2} \\
&= \frac{(1-\hat{\beta}^m)^2}{(1-\hat{\beta})^2} \\
&\leq \frac{1}{(1-\hat{\beta})^2}. \qquad (41)
\end{aligned}
$$

Rearranging (40) provides

$$
\begin{aligned}
0 &\leq \mathbb{E}[f_0] - \mathbb{E}[f_m] - 2\alpha_{lo}\lambda m\mathbb{E}[\delta_f(\boldsymbol{w}_{k+1})] \\
&\quad + \hat{\beta}\alpha_{hi}\Lambda\ (m-1)\mathbb{E}[\delta_f(\boldsymbol{w}_{k+1})] \\
&\quad + \hat{\beta}\alpha_{hi}\Lambda\ (m-1)\mathbb{E}[\delta_f(\boldsymbol{w}_k)] + \tau/(1-\hat{\beta})^2\mathbb{E}[\delta_f(\boldsymbol{w}_k)] \\
&\leq \mathbb{E}[(f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)] \\
&\quad + (\hat{\beta}\alpha_{hi}\Lambda\ (m-1) - 2\alpha_{lo}\lambda m)\mathbb{E}[\delta_f(\boldsymbol{w}_{k+1})] \\
&\quad + (\hat{\beta}\alpha_{hi}\Lambda\ (m-1) + \tau/(1-\hat{\beta})^2)\mathbb{E}[\delta_f(\boldsymbol{w}_k)]. \qquad (42)
\end{aligned}
$$

Furthermore, we have

$$\mathbb{E}[\delta_f(\boldsymbol{w}_{k+1})] \leq \xi\mathbb{E}[\delta_f(\boldsymbol{w}_k)] \qquad (43)$$

where

$$\xi = \frac{1 + \hat{\beta}\alpha_{hi}\Lambda\ (m-1) + 4\Lambda^2\alpha_{hi}^2/(1-\hat{\beta})^2}{(2\alpha_{lo}\lambda - \hat{\beta}\alpha_{hi}\Lambda)m + \hat{\beta}\alpha_{hi}\Lambda}. \qquad (44)$$

Let $\xi < 1$; then, it follows that:

$$m \geq \frac{1 - 2\hat{\beta}\alpha_{hi}\Lambda + 4\Lambda^2\alpha_{hi}^2/(1-\hat{\beta})^2}{2\alpha_{lo}\lambda - 2\hat{\beta}\alpha_{hi}\Lambda}. \tag{45}$$

We observe that for $\hat{\beta} < 1$, we have

$$\begin{aligned} &1 - 2\hat{\beta}\alpha_{hi}\Lambda + 4\Lambda^2\alpha_{hi}^2/(1-\hat{\beta})^2 \\ &\quad = (1 - \hat{\beta}\alpha_{hi}\Lambda)^2 + \left(\frac{4}{(1-\hat{\beta})^2} - \hat{\beta}^2\right) \cdot \Lambda^2\alpha_{hi}^2 \\ &\quad > 0. \end{aligned}$$

Assuming that we choose the step interval $[\alpha_{lo}, \alpha_{hi}]$ which satisfies

$$\frac{\alpha_{hi}}{\alpha_{lo}} < \frac{\lambda}{\hat{\beta}\Lambda} \tag{46}$$

then a sufficiently large m will ensure the linear convergence of CGVR. □

IV. Experiments

In this section, we compare our algorithm CGVR with SGD, SVRG [13], CG [25], and SLBFGS [16]. Our experiments show the effectiveness of CGVR on several popular learning models, which may be convex, nonconvex, or nonsmooth.

A. Descriptions of Models and Data Sets

We evaluate these algorithms on four state-of-the-art learning models as follows:

1) ridge regression (ridge)

$$\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i^T\boldsymbol{w}\right)^2 + \lambda\|\boldsymbol{w}\|_2^2 \tag{47}$$

2) logistic regression (logistic)

$$\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}\ln\left(1 + \exp\left(-y_i\boldsymbol{x}_i^T\boldsymbol{w}\right)\right) + \lambda\|\boldsymbol{w}\|_2^2 \tag{48}$$

3) $L2$-regularized $L1$-loss support vector machine (SVM) (hinge)

$$\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}\left(1 - y_i\boldsymbol{x}_i^T\boldsymbol{w}\right)_+ + \lambda\|\boldsymbol{w}\|_2^2 \tag{49}$$

4) $L2$-regularized $L2$-loss SVM (sqhinge)

$$\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}\left(\left(1 - y_i\boldsymbol{x}_i^T\boldsymbol{w}\right)_+\right)^2 + \lambda\|\boldsymbol{w}\|_2^2 \tag{50}$$

where $\boldsymbol{x}_i \in R^d$ and $y_i \in \{-1, +1\}$ are the feature vector and target value of the ith example, respectively, and $\lambda > 0$ is a regularization parameter. We concatenate each row $\boldsymbol{x}_i$ of data matrix X with the number 1 in (47)–(50) such that $(\boldsymbol{x}_i, 1)^T(\boldsymbol{w}, b) = \boldsymbol{x}_i^T\boldsymbol{w} + b$.

We executed all of the algorithms for the binary classification of six large-scale data sets from A Library for SVM website.[2] The information on the data sets is listed in Table I.

TABLE I
Description of Data Sets

Dataset	n	d
a9a	32,561	123
covtype	581,012	54
ijcnn1	49,990	22
w8a	49,749	300
SUSY	5,000,000	18
HIGGS	11,000,000	28

B. Implementations of Algorithms

In the preprocessing stage, each feature value for all dimensions was scaled into the range of $[-1, +1]$ by the max-min scaler. All of the algorithms were implemented in C++ using the armadillo linear algebra library [28] and Intel Math Kernel Library.[3]

To explore the convergence of the algorithms, we used the entire data set to minimize the function values of the four learning models. To compare the generalization of the algorithms in classification, we randomly divided the entire data set into three parts: 1/3 for testing, 1/5 for validation, and the remainder for training. We used the same divisions for all of the algorithms. After searching for the optimal parameters on the candidate set to maximize the AUC score on the validation set, we used a model corresponding to the best parameters to estimate the AUC scores on the test set.

SGD, SVRG, CG, and CGVR need to calculate the gradient, and SLBFGS also calculates the Hessian matrix. In our implementations, we used numerical methods to estimate the gradient $\nabla f(\boldsymbol{x})$ and the Hessian matrix $\nabla^2 f(\boldsymbol{x})$ with a small constant $\epsilon = 10^{-4}$

$$\nabla f(\boldsymbol{x})_i = \frac{f(\boldsymbol{x} + \epsilon\boldsymbol{e}_i) - f(\boldsymbol{x} - \epsilon\boldsymbol{e}_i)}{2\epsilon} \tag{51}$$

$$\begin{aligned} \nabla^2 f(\boldsymbol{x})_{i,j} = &(f(\boldsymbol{x} + \epsilon\boldsymbol{e}_i + \epsilon\boldsymbol{e}_j) - f(\boldsymbol{x} + \epsilon\boldsymbol{e}_i) \\ &- f(\boldsymbol{x} + \epsilon\boldsymbol{e}_j) + f(\boldsymbol{x}))/\epsilon^2 \end{aligned} \tag{52}$$

where the subscript represents the ith (or $\{i, j\}$th) element of the matrix on the left side of the equations, and $\boldsymbol{e}_i$ is the ith unit vector.

For fair comparisons, we used the original C++ version of the LIBLINEAR solver [24] in the discussion of generalization, which is more effective than the general LIBSVM for training SVM models on large-scale problems. We know that the measured AUC represents the area under the receiver operating characteristic curve, which is a graphical plot that demonstrates the discrimination ability of a binary classification model when its discrimination threshold is varied. The discrimination threshold is a real value, but the output of LIBLINEAR is a discrete class label $\{-1, +1\}$. For a given threshold, many identical output values result in a large uncertainty in sorting, which was used to calculate the AUC value. Thus, we modified the predict function of LIBLINEAR to directly output the discrimination value $\boldsymbol{x}^T\boldsymbol{w} + b$. By default, LIBLINEAR optimizes the dual form of $L2$-regularized $L2$-loss SVM in model (50).

[2] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/data sets/

[3] https://software.intel.com/en-us/mkl

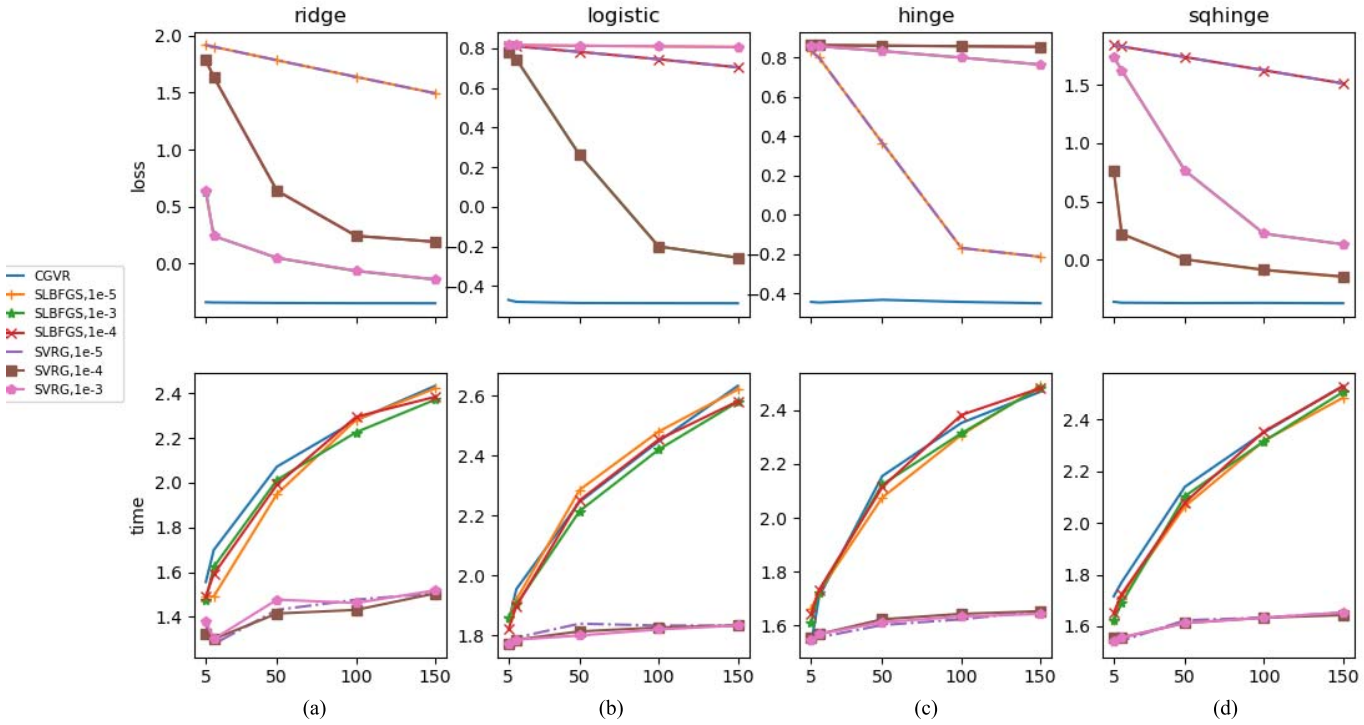


Fig. 1. Convergence on four loss functions with $\lambda = 10^{-4}$ for three variance reduction algorithms with different learning rates on a9a data sets. (a) x-axis represents parameter m. (b) y-axis is a logarithm of the value (base 10). (c) Number in the legends is the learning rate. (d) Upper and lower rows correspond to the loss and the time, respectively.

C. Parameter Investigation

The CGVR algorithm has two main parameters: the number of iterations of the inner loop m and the number of iterations of the outer loop T. Parameter T will be discussed in Section IV-D.

We selected data set **a9a** as our research object and reported the AUC measures after 25 outer loops ($T = 25$). We used an identical random seed to initialize vector $\boldsymbol{w}_0$, where $\boldsymbol{w}_0$ is uniformly distributed over the interval $[0, 1]$. In CGVR and SLBGFS, we set the sampling size $|\mathcal{S}_{k,t}|$ to $\sqrt{n}$, which was used to calculate the gradient vector and the Hessian matrix of the function. For CG and CGVR, we set $c_1 = 10^{-4}$ and $c_2 = 0.1$. In SLBFGS, we set the memory size M to 10 and the Hessian update interval L to 10. SGD is accelerated in the relevant direction and dampens oscillations using the momentum method, where the momentum coefficient is commonly set to 0.9. For SGD, SVRG, and SLBFGS, we attempted three different constant step sizes: 10^{-3}, 10^{-4}, and 10^{-5}.

Fig. 1 shows the function values and the running time on CGVR and the other two algorithms (SLBFGS and SVRG) at different learning rates as parameter m increases. The parameter λ is set to 10^{-4}, and the learning rate was obtained from the set $\{10^{-3}, 10^{-4}, \text{and} 10^{-5}\}$. The four columns of the subfigures demonstrate that different algorithms optimize the ridge, logistic, hinge, and sqhinge losses.

We observe that SVRG and SLBFGS reduce the losses to some extent with increasing m when setting an appropriate learning rate. The CGVR algorithm can quickly approach the minimum value of a function with only five inner loops, which slowly decreases when the m value increases. Thus, CGVR is insensitive to the parameter m, and it requires only a few inner loops to quickly converge.

It is clear that the running time of all algorithms increases with increasing m, but the running time of the SLBFGS and SVRG algorithms varies with the learning rate. Although the running time of CGVR is comparable to that of the other algorithms when parameter m is identical, it still has a great advantage in terms of time efficiency because CGVR only requires a few iterations of inner loops to quickly converge.

D. Convergence of CGVR

We set the number of inner loop iterations m to 50 and compare the convergence of several algorithms on six large-scale data sets. The best model that corresponds to the optimal learning rate was chosen for SGD, SVRG, and SLBFGS, where the optimal learning rate taken from $\{10^{-3}, 10^{-4}, \text{and } 10^{-5}\}$ minimizes the value of the final iteration on the validation set.

Fig. 2 shows the convergence of the five algorithms with $\lambda = 10^{-4}$. SGD unstably converges on the ridged model, where an inappropriate learning rate causes large fluctuations in the loss value. SLBFGS does not show better convergence than SVRG because both SLBFGS and SVRG are sensitive to the learning rates. In general, CG converges faster than SLBFGS, SGD, and SVRG. CGVR has the fastest convergence on almost all of the four models, even when the loss value reaches a notably small value. We also observe that all

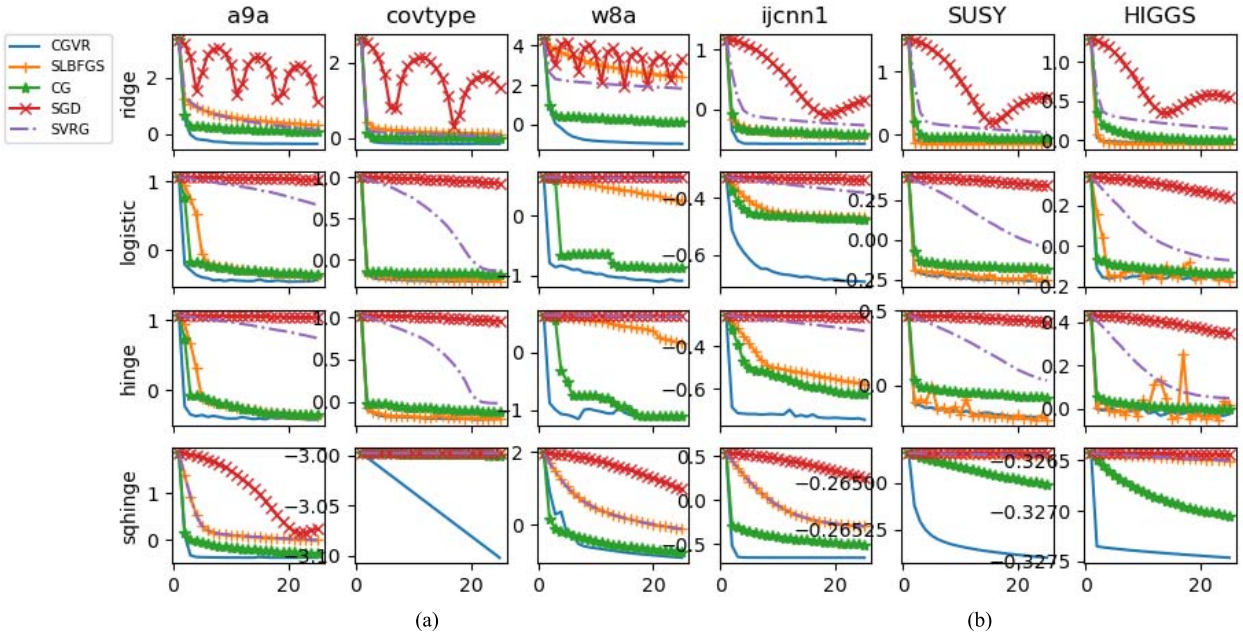


Fig. 2. Convergence of four loss functions with $\lambda = 10^{-4}$ for five algorithms on six data sets. (a) y-axis represents the logarithm of the loss value (base 10). (b) x-axis represents the number of iterations of the outer loop.

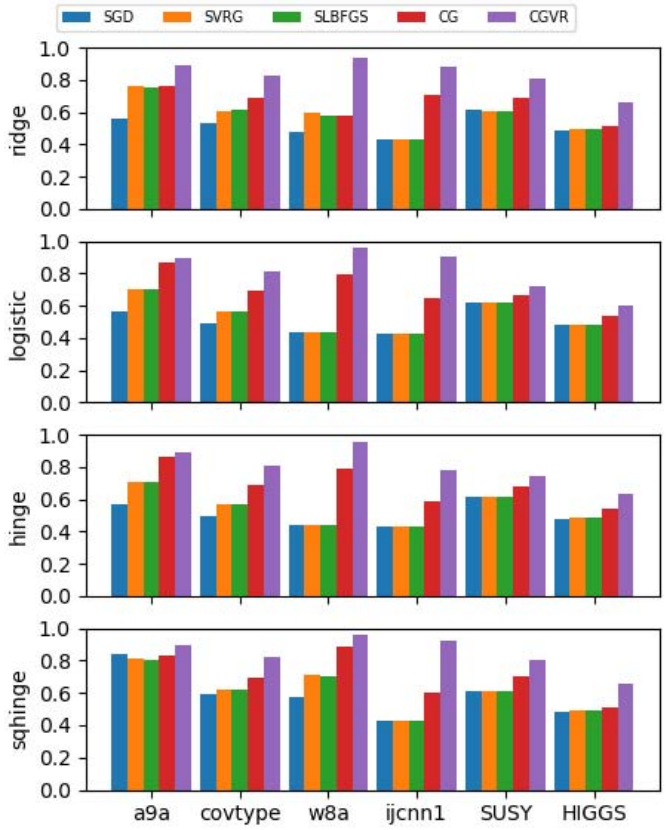


Fig. 3. Comparisons of the AUC score of CGVR and its counterpart algorithms.

of the algorithms converge faster on the sqhinge model than on the other models.

E. Generalization of CGVR

To analyze the generalization of the CGVR algorithm, we show the average AUC scores of SGD, SVRG, SLBFGS, CG, and CGVR on five random splits of data sets for each model in Fig. 3. Furthermore, we compare the average AUC performance and the execution time of CGVR and the LIBLINEAR solver on six large-scale data sets (Fig. 4). The learning rate α was selected from $\{10^{-3}, 10^{-4}, \text{and} 10^{-5}\}$, and the regularization coefficient λ was drawn from $\{1 \times 10^{-1}, 5 \times 10^{-2}, 1 \times 10^{-2}, 8 \times 10^{-3}, \text{and} 5 \times 10^{-3}\}$. The parameter C in LIBLINEAR is equal to $1/(2\lambda)$. The LIBLINEAR solver optimizes the sqhinge model from (50). The training parameters and the model parameters were optimized in the space of grid (α, λ) through the hold validation on the training and validation data sets.

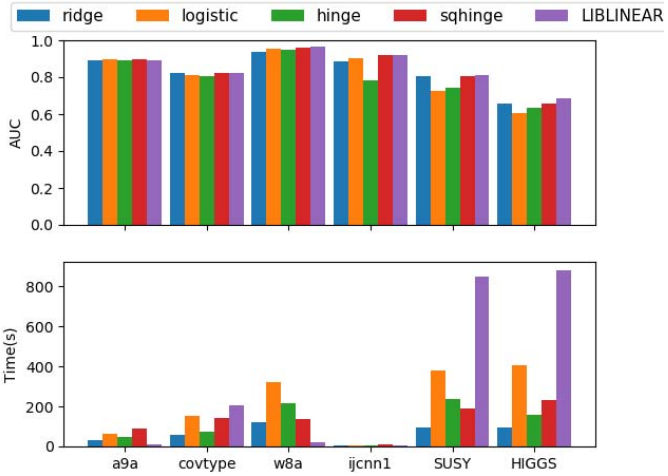


Fig. 4. Comparisons of the AUC score and the execution time between CGVR with different models (ridge, logistic, hinge, and sqhinge) and LIBLINEAR.

Fig. 3 shows that CGVR significantly outperforms the other counterparts on six data sets. SVRG and SLBFGS show the close generalization performance. The classical CG algorithm shows better generalization performance than SVRG, SGD, and SLBFGS. Combining the discussion on the convergence

of the algorithms, we can conclude that the CGVR algorithm achieves a smaller minimum value generally has better generalization performance in the case of appropriate regularization conditions.

Fig. 4 shows that our algorithm CGVR achieves AUC scores that are comparable to those of the LIBLINEAR solver on six data sets. In the four discussed models, CGVR performs the best on all data sets when solving the sqhinge model (see 50), which is exactly the loss function optimized by the LIBLINEAR solver in default settings. Furthermore, LIBLINEAR runs faster than CGVR on small-scale data sets, such as on data sets a9a, w8a, and ijcnn1, whose sample sizes are less than 100 000. However, on data sets with millions of data points, such as SUSY and HIGGS, our algorithm CGVR runs faster than LIBLINEAR, where it only iterates 25 times.

V. Conclusion

In this paper, we proposed a new CG algorithm based on variance reduction (CGVR). We prove the linear convergence of CGVR with Fletcher–Reeves update. The empirical results from six large-scale data sets show the power of our algorithm on four classic learning models, where these models may be convex, nonconvex, or nonsmooth. The advantages of our algorithm CGVR are as follows.

1) It only requires a few iterations to quickly converge compared with its counterpart SVRG.
2) The empirical settings for CGVR always work well: the parameters are insensitive to the data sets, most of which are related to the classic Wolfe line search subroutine.
3) It requires less storage space during running, similar to the CG algorithm; it only needs to store the last gradient vector, whereas SLBFGS must store M vector pairs.
4) CGVR achieves a generalization performance comparable to that of the LIBLINEAR solver for optimizing the $L2$-regularized $L2$-loss while providing a great improvement in computational efficiency in large-scale machine learning problems.

In the future work, with the implementation of the algorithm using a numerical gradient, we can easily apply it to other problems, such as sparse dictionary learning and low-rank matrix approximation problems.

References

[1] X.-B. Jin, C.-L. Liu, and X. Hou, "Regularized margin-based conditional log-likelihood loss for prototype learning," *Pattern Recognit.*, vol. 43, no. 7, pp. 2428–2438, 2010.

[2] X. Zhang, L. Wang, S. Xiang, and C. Liu, "Retargeted least squares regression algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 2206–2213, Sep. 2015.

[3] X.-B. Jin, G.-S. Xie, K. Huang, and A. Hussain, "Accelerating infinite ensemble of clustering by pivot features," *Cogn. Comput.*, pp. 1–9, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s12559-018-9583-8#citeas, doi: 10.1007/s12559-018-9583-8.

[4] X.-B. Jin, G.-G. Geng, M. Sun, and D. Zhang, "Combination of multiple bipartite ranking for multipartite Web content quality evaluation," *Neurocomputing*, vol. 149, pp. 1305–1314, Feb. 2015.

[5] X.-B. Jin, G.-G. Geng, G.-S. Xie, and K. Huang, "Approximately optimizing NDCG using pair-wise loss," *Inf. Sci.*, vol. 453, pp. 50–65, Jul. 2018.

[6] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, 2010, pp. 177–186.

[7] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. ICLR Workshop*, 2016, pp. 1–4.

[8] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proc. ICLR*, 2018, pp. 1–23.

[9] D. P. Kingma and J. Ba. (2015). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[10] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, vol. 28, Apr. 2013, pp. 1139–1147.

[11] N. L. Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence_rate for finite training sets," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)* New York, NY, USA: Curran Associates Inc., 2012, pp. 2663–2671.

[12] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss," *J. Mach. Learn. Res.*, vol. 14, pp. 567–599, Feb. 2013.

[13] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *NIPS*, New York, NY, USA: Curran Associates Inc., 2013, pp. 315–323.

[14] X. Wang, S. Ma, D. Goldfarb, and W. Liu. (2016). "Stochastic quasi-Newton methods for nonconvex stochastic optimization." [Online]. Available: https://arxiv.org/abs/1607.01231

[15] A. Mokhtari and A. Ribeiro, "RES: Regularized stochastic BFGS algorithm," *IEEE Trans. Signal Process.*, vol. 62, no. 23, pp. 6089–6104, Dec. 2014.

[16] P. Moritz, R. Nishihara, and M. Jordan, "A linearly-convergent stochastic L-BFGS algorithm," in *Proc. Mach. Learn. Res. (PMLR)*, A. Gretton and C. C. Robert, Eds. Cadiz, Spain, 2016, pp. 249–258.

[17] R. M. Gower, D. Goldfarb, and P. Richtárik, "Stochastic block BFGS: Squeezing more curvature out of data," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1–10.

[18] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Comput. J.*, vol. 7, no. 2, pp. 149–154, 1964.

[19] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *Revue française d'informatique recherche opérationnelle. Série rouge*, vol. 3, no. 16, pp. 35–43, 1969.

[20] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bureau Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952.

[21] Y. Liu and C. Storey, "Efficient generalized conjugate gradient algorithms, part 1: Theory," *J. Optim. Theory Appl.*, vol. 69, no. 1, pp. 129–137, 1991.

[22] J. E. Dennis, Jr., "Practical methods of optimization, vol. 1: Unconstrained optimization (R. Fletcher)," *SIAM Rev.*, vol. 24, no. 1, pp. 97–98, 1982.

[23] Y. H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM J. Optim.*, vol. 10, no. 1, pp. 177–182, 1999.

[24] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9 pp. 1871–1874, Jan. 2008.

[25] J. Nocedal and S. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2006.

[26] M. J. D. Powell, "Restart procedures for the conjugate gradient method," *Math. Program.*, vol. 12, no. 1, pp. 241–254, 1977.

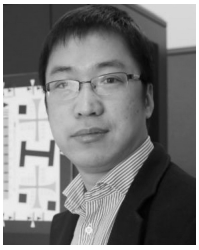
[27] M. Al-Baali, "Descent property and global convergence of the Fletcher–Reeves method with inexact line search," *IMA J. Numer. Anal.*, vol. 5, no. 1, pp. 121–124, 1985.

[28] C. Sanderson and R. Curtin, "Armadillo: A template-based C++ library for linear algebra," *J. Open Source Softw.*, vol. 1, no. 2, pp. 26–27, 2016.

Xiao-Bo Jin received the Ph.D. degree in pattern recognition and intelligent systems from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009.

He is currently an Associate Professor with the School of Information Science and Engineering, Henan University of Technology, Zhengzhou, China. His current research interests include web mining and machine learning, pattern recognition, and neurocomputing.

Xu-Yao Zhang was a Visiting Scholar with the Montreal Institute for Learning Algorithms, University of Montreal, Montreal, QC, Canada, from 2015 to 2016. He is currently an Associate Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include machine learning, pattern recognition, handwriting recognition, and deep learning.



Kaizhu Huang received the Ph.D. degree from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2004. He was an Associate Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. He was a Researcher with the Fujitsu Research and Development Centre, CUHK, and the University of Bristol, Bristol, U.K., from 2004 to 2009. He is currently the Head of the Department of Electrical and Electronic Engineering, Xian Jiaotong-Liverpool University, Suzhou, China. He has been involved in machine learning, pattern recognition, and neural information processing. He has authored or co-authored 8 books in Springer and over 120 international research papers (45+ SCI-indexed international journals and 60+ EI conference papers), e.g., in journals (*Journal of Machine Learning Research*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING) and conferences (NIPS, IJCAI, SIGIR, UAI, CIKM, ICDM, ICML, ECML, CVPR).

Dr. Huang is the Senior PC of AAAI from 2016 to 2018, the Publication Chair of ACML in 2016, the Program Co-Chair of ICONIP in 2014, the Organizing Co-Chair of DMC from 2012 to 2017, the Publication Chair of ICDAR 2011, the Publicity Chair of ACPR in 2011, the Session Chair of ICONIP2006 from 2009 to 2011. He served on the program committees in many international conferences such as ICONIP, IJCNN, IWACI, EANN, KDIR. He was a recipient of 2011 Asian Pacific Neural Network Assembly Distinguished Younger Researcher Award. He serves as an Associate Editors for *Neurocomputing*, *Cognitive Computation*, and *Springer Nature Big Data Analytics*.

Guang-Gang Geng received the Ph.D. degree from the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

He was with the Computer Network Information Center, Chinese Academy of Sciences, Beijing, in 2008. He is currently a Professor with the China Internet Network Information Center, Beijing. His current research interests include machine learning, adversarial information retrieval on the web, and web search.