AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 6(2): 1515–1537.
DOI: 10.3934/math.2021092
Received: 16 August 2020
Accepted: 02 November 2020
Published: 23 November 2020

Research article

An online conjugate gradient algorithm for large-scale data analysis in machine learning

Wei Xue[1,2,3], Pengcheng Wan[1], Qiao Li[1], Ping Zhong[2,*], Gaohang Yu[4] and Tao Tao[1,*]

[1] School of Computer Science and Technology, Anhui University of Technology, Maanshan 243032, China

[2] National Key Laboratory of Science and Technology on Automatic Target Recognition, National University of Defense Technology, Changsha 410073, China

[3] Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 210096, China

[4] School of Sciences, Hangzhou Dianzi University, Hangzhou 310018, China

* **Correspondence:** Email: zhongping@nudt.edu.cn, taotao@ahut.edu.cn.

Abstract: In recent years, the amount of available data is growing exponentially, and large-scale data is becoming ubiquitous. Machine learning is a key to deriving insight from this deluge of data. In this paper, we focus on the large-scale data analysis, especially classification data, and propose an online conjugate gradient (CG) descent algorithm. Our algorithm draws from a recent improved Fletcher-Reeves (IFR) CG method proposed in Jiang and Jian[13] as well as a recent approach to reduce variance for stochastic gradient descent from Johnson and Zhang [15]. In theory, we prove that the proposed online algorithm achieves a linear convergence rate under strong Wolfe line search when the objective function is smooth and strongly convex. Comparison results on several benchmark classification datasets demonstrate that our approach is promising in solving large-scale machine learning problems, viewed from the points of area under curve (AUC) value and convergence behavior.

Keywords: machine learning; online learning; stochastic optimization; conjugate gradient; variance reduction
Mathematics Subject Classification: 65K05, 68W27

1. Introduction

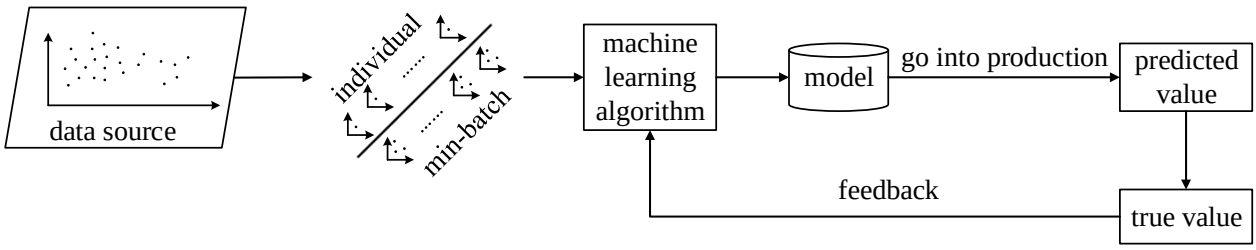
We first establish the key notations to make them consistent in this paper. Vectors are denoted by boldface letters, e.g., $\mathbf{w}$, and $\nabla f(\mathbf{w})$ denotes the gradient of function f at point $\mathbf{w}$. Superscript "T" is the transpose operation of a vector. We use $\|\mathbf{w}\|$ as a shorthand for the l_2-norm of $\mathbf{w}$. Given a dataset

$\mathcal{D} = \{(\mathbf{z}_i, y_i)_{i=1}^m\}$, where $\mathbf{z}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$, the focus of this paper is structural risk minimization [28], a fundamental subject in machine learning, which is a combination of two terms

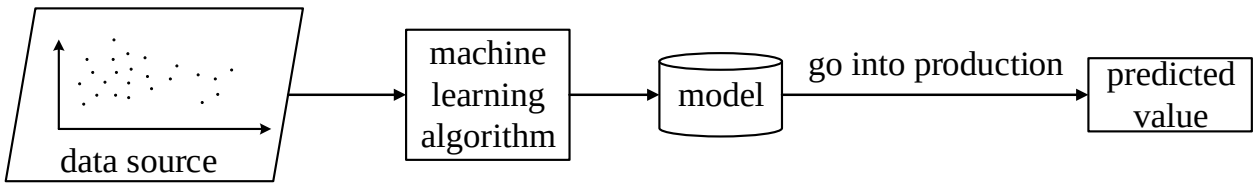
$$\min_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w}) \triangleq \frac{1}{m}\sum_{i=1}^{m} \underbrace{l(\mathbf{w}; \mathbf{z}_i, y_i) + r(\mathbf{w})}_{f_i(\mathbf{w})}. \tag{1.1}$$

Here, $\mathbf{w}$ is the optimization variable (usually called weight vector in the research area of machine learning), $l : \mathbb{R}^n \to \mathbb{R}$ is the loss function associated with sample pair $(\mathbf{z}_i, y_i)$, and the regularization function $r : \mathbb{R}^n \to \mathbb{R}$ is a penalty on the complexity of $\mathbf{w}$.

Optimization learning algorithms for solving (1.1) mainly fall into two broad categories, namely, online learning and batch learning. The two learning mechanisms are illustrated in Figure 1. In online learning, data streams (either individually or in mini-batches) into the learning algorithm and updates the prediction model. We just need to use the new (real-time) data to build a model. However, in batch learning, the model is trained using the entire dataset. This process is also called offline learning. In batch learning, if there is need to update the learning model based on new data, the model should be trained from scratch all over again using both the previous data and the new data [2].



(a) online learning



(b) batch learning

Figure 1. Online learning vs. batch learning.

The prototypical online learning algorithm is stochastic gradient descent (SGD) method [23], which is defined as

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \nabla f_{i_k}(\mathbf{w}_k), \tag{1.2}$$

where η_k is the learning rate (also known as stepsize in numerical optimization), and the index i_k, corresponding to the sample pair $(\mathbf{z}_{i_k}, y_{i_k})$, is randomly chosen from $\{1, 2, \ldots, m\}$. We describe the standard SGD method in Algorithm 1. It is observed that each iteration of SGD only involves the computation of $\nabla f_{i_k}(\mathbf{w}_k)$ corresponding to one sample, resulting in a very cheap iteration cost. The

Initial point $\mathbf{w}_0$.
for $k = 0, 1, \ldots$ **do**
  Randomly pick a sample $(\mathbf{z}_{i_k}, y_{i_k})$.
  Compute the gradient $\mathbf{g}_k \leftarrow \nabla f_{i_k}(\mathbf{w}_k)$.
  Choose a learning rate η_k.
  Set the new iteration as $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \eta_k \mathbf{g}_k$.
end

Algorithm 1: standard SGD Method

gradient information is obtained from only one sample at each iteration. Further, one can use a mini-batch method, in which a small subset of all samples is randomly selected per iteration and employ the iteration.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{\eta_k}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k \subseteq \mathcal{D}} \nabla f_i(\mathbf{w}_k). \tag{1.3}$$

where $|\mathcal{S}_k|$ denotes the number of elements in the set $\mathcal{S}_k$.

In recent years, SGD has been widely studied in machine learning and optimization community; see [5, 16, 19, 22, 27, 29, 32] for example. The standard SGD can quickly reach a proximate optimum solution during the learning process, but its convergence rate may slow down when more accurate solutions are required. Due to the variance of random sampling, with a suitably chosen $\eta_k = O(1/k)$, SGD achieves a sub-linear convergence rate of $O(1/k)$.

Initial point $\mathbf{w}_0$, update frequency T, learning rate η.
for $k = 0, 1, \ldots$ **do**
  Compute the batch gradient $\mathbf{u}_k \leftarrow \nabla f(\mathbf{w}_k)$.
  Initialize $\mathbf{x}_0 \leftarrow \mathbf{w}_k$.
  for $t = 0, 1, \ldots, T-1$ **do**
    Randomly select $i_t \in \{1, 2, \ldots, m\}$.
    Compute the gradient $\mathbf{g}_t \leftarrow \nabla f_{i_t}(\mathbf{x}_t) - \nabla f_{i_t}(\mathbf{x}_0) + \mathbf{u}_k$.
    Set the new iteration as $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \mathbf{g}_t$.
  end
  Option 1: $\mathbf{w}_{k+1} \leftarrow \mathbf{x}_T$.
  Option 2: $\mathbf{w}_{k+1} \leftarrow \mathbf{x}_t$ for randomly chosen $t \in \{1, \ldots, T\}$.
end

Algorithm 2: SVRG Method

In order to improve the convergence of SGD, Johnson and Zhang [15] proposed a stochastic variance reduced gradient (SVRG) method to reduce the variance of random sampling. The description of SVRG is in Algorithm 2. As argued by Bottou et al. [3], for the first epochs, SGD is more efficient, but once the iterations approach the solution the benefits of the fast convergence rate of SVRG can be observed. In [21], Nguyen et al. introduced a stochastic recursive gradient algorithm (SARAH). Different from SVRG, SARAH employs a simple recursive approach to update stochastic gradient and uses more stable gradient estimates than SVRG. But the choice of the learning rate η depends on a Lipschitz constant of the objective function. Without explicit knowledge of this

constant, η is typically chosen by experimentation in both SVRG and SARAH. By incorporating the idea of variance reduction, Moritz et al. [20] proposed a stochastic L-BFGS method and proved a linear rate of convergence. In [26], Tan et al. used the well-known Barzilai-Borwein (BB) method [1] to automatically compute the learning rate for SVRG, and established a linear convergence rate. Recently, based on the BB method, an improved SVRG, named stochastic two-point stepsize gradient method, has been proposed by Shao et al. [25], which also achieves a linear convergence rate for smooth and strongly convex functions. Liu et al. [17] used the BB approach to adaptively compute the learning rate for SARAH. In [14], Jin et al. proposed a stochastic CG method with variance reduction, which converges more quickly than SVRG. Actually, the idea of stochastic CG has been studied before. Schraudolph and Graepel [24] combined the idea of CG with stochastic setting to optimize unconstrained quadratic problems, and the resulting algorithms converge orders of magnitude faster than ordinary SGD. In [30], Xu and Dai employed CG to accelerate the convergence of stochastic approximation algorithm, Robbins-Monro method. Jiang and Wilford [12] proposed a stochastic CG method for the approximation of functions, which performs the CG steps by using an inner product that is based on stochastic sampling.

Motivated by this, we propose in this paper a new online variance reduced CG method and prove that the proposed method converges linearly for strongly convex and smooth objective functions. The remainder of this paper is organized as follows. In Section 2, we give a brief introduction of the IFR CG method used in batch optimization. And then, we present the proposed method along with the theoretical analysis in Section 3. Numerical experiments are reported in Section 4. Finally, we conclude this paper in Section 5.

2. Brief introduction of IFR CG

Due to the low memory requirement and strong convergence property, CG method is a very efficient batch algorithm, and it is still one of the most active research fields of optimization [6, 7, 11, 13, 18, 31]. Take (1.1) as an example. Suppose that f is differentiable, then given an initial point $\mathbf{w}_0$, a CG method generates a sequence $\{\mathbf{w}_k\}$ by using the recurrence

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{d}_k, \tag{2.1}$$

where α_k is the stepsize usually obtained by a line search, and the search direction $\mathbf{d}_k$ is computed by

$$\mathbf{d}_k = -\nabla f(\mathbf{w}_k) + \beta_k \mathbf{d}_{k-1}, \mathbf{d}_0 = -\nabla f(\mathbf{w}_0). \tag{2.2}$$

β_k is the CG update parameter. Different choices of β_k yield different CG methods. Some well-known formulas for β_k are β_k^{HS}, β_k^{FR}, β_k^{PR}, β_k^{CD}, β_k^{LS}, and β_k^{DY}, see the comprehensive review [10] for details. Here, we focus on β_k^{FR} proposed by Fletcher and Reeves [8], which takes the form of

$$\beta_k^{FR} = \frac{\|\nabla f(\mathbf{w}_k)\|^2}{\|\nabla f(\mathbf{w}_{k-1})\|^2}. \tag{2.3}$$

Another important issue related to the performance of CG is the line search, which requires sufficient accuracy to ensure $\mathbf{d}_k$ satisfies the descent condition $\nabla f(\mathbf{w}_k)^T \mathbf{d}_k < 0$, $\forall\ k$. A common criteria for line search is the Wolfe line search

$$\begin{cases} f(\mathbf{w}_k + \alpha_k \mathbf{d}_k) - f(\mathbf{w}_k) \leq c_1 \alpha_k \nabla f(\mathbf{w}_k)^T \mathbf{d}_k, \\ \nabla f(\mathbf{w}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k \geq c_2 \nabla f(\mathbf{w}_k)^T \mathbf{d}_k, \end{cases} \tag{2.4}$$

where $0 < c_1 < c_2 < 1$. In the strong Wolfe line search, the second condition in (2.4) is replaced by

$$|\nabla f(\mathbf{w}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k| \leq -c_2 \nabla f(\mathbf{w}_k)^T \mathbf{d}_k. \tag{2.5}$$

It has been shown that the strong Wolfe line search for β_k^{FR} may not yield a descent direction unless $c_2 \leq 1/2$ [4]. In practice, it is often most efficient to choose c_2 close to 1. Hence, the constraint $c_2 \leq 1/2$, needed to ensure descent, is a significant restriction in the choice of parameter c_2 in the Wolfe line search. Note that Gilbert and Nocedal [9] gave the convergent scope $\tau_k \in [-1, 1]$ for $\tau_k \beta_k^{FR}$. Based on this, Jiang and Jian[13] considered $\tau_k = -\frac{|\nabla f(\mathbf{w}_k)^T \mathbf{d}_{k-1}|}{\nabla f(\mathbf{w}_{k-1})^T \mathbf{d}_{k-1}}$ and introduced an improved FR (IFR) formula

$$\beta_k^{IFR} = -\frac{|\nabla f(\mathbf{w}_k)^T \mathbf{d}_{k-1}|}{\nabla f(\mathbf{w}_{k-1})^T \mathbf{d}_{k-1}} \times \beta_k^{FR}. \tag{2.6}$$

The following lemma shows that the search directions yielded by IFR CG are all sufficient descent, and the IFR CG method is globally convergent under the strong Wolfe line search.

Lemma 1 (Theorem 3 [13])**.** *Suppose that $f(\boldsymbol{w})$ is differentiable, $\nabla f(\boldsymbol{w})$ is Lipschitz continuous, and $0 < c_2 < \sqrt{2}/2$, then it holds that*

$$\frac{-1}{1 - c_2^2} \leq \frac{\nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k}{\|\nabla f(\boldsymbol{w}_k)\|^2} \leq \frac{2c_2^2 - 1}{1 - c_2^2}. \tag{2.7}$$

Further, IFR CG is globally convergent by the way of $\liminf_{k \to \infty} \|\nabla f(\boldsymbol{w}_k)\| = 0$.

3. Proposed method and theoretical analysis

3.1. SIFR CG method

Based on the theories mentioned above, we now describe the core of the proposed online CG algorithm, called stochastic improved Fletcher-Reeves conjugate gradient (SIFR CG for short) method, for solving (1.1) in detail and provide the pseudo-code.

SIFR CG operates in cycles. At the beginning of each cycle, an iteration point $\mathbf{w}_k$ is available at which the method computes a batch gradient $\mathbf{u}_k \leftarrow \nabla f(\mathbf{w}_k)$. Then, after initializing $\mathbf{x}_0 \leftarrow \mathbf{w}_k$, $\mathbf{g}_0 \leftarrow \mathbf{h}_k$, and $\mathbf{d}_0 \leftarrow -\mathbf{g}_0$, a set of T inner iterations indexed by t with an update $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \alpha_t \mathbf{d}_t$ are performed, where α_t is chosen according to the strong Wolfe line search. We define the subsampled function $f_{\mathcal{S}}(\mathbf{w})$ as:

$$f_{\mathcal{S}}(\mathbf{w}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}_k} f_i(\mathbf{w}). \tag{3.1}$$

where $|\mathcal{S}_k|$ denotes the number of elements in the set S_k.The search direction $\mathbf{d}_t$ is computed by $-\mathbf{g}_t + \beta_t \mathbf{d}_{t-1}$, where $\mathbf{g}_{t+1} \leftarrow \nabla f_{\mathcal{S}}(\mathbf{x}_{t+1}) - \nabla f_{\mathcal{S}}(\mathbf{w}_k) + \mathbf{u}_k$, and $\mathcal{S}$ is chosen randomly. The pseudo-code of SIFR CG is given in Algorithm 3.

There are a couple of things to note about this algorithm.

Remark 1. *To mitigate the effects of some outliers in the training data, we restrict α_t and β_t by adding some additional constraints.*

Remark 2. *For convenience, we use the condition $|\boldsymbol{g}_{t+1}^T \boldsymbol{d}_t| \leq -c_2 \boldsymbol{g}_t^T \boldsymbol{d}_t$ in the theoretical analysis.*

Remark 3. *The following experiments are carried out using $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{x}_T$.*

Initial point $\mathbf{w}_0$, update frequency T, strong Wolfe line search parameters c_1 and c_2, $\alpha_{min} > 0$, $\alpha_{max} > 0$, threshold parameter $\epsilon > 0$.
Compute the initial batch gradient and initialize $\mathbf{h}_0 \leftarrow \nabla f(\mathbf{w}_0)$.
for $k = 0, 1, \ldots$ **do**
  Compute the batch gradient $\mathbf{u}_k \leftarrow \nabla f(\mathbf{w}_k)$.
  Initialize $\mathbf{x}_0 \leftarrow \mathbf{w}_k$.
  Initialize $\mathbf{g}_0 \leftarrow \mathbf{h}_k$.
  Initialize $\mathbf{d}_0 \leftarrow -\mathbf{g}_0$.
  for $t = 0, 1, \ldots, T-1$ **do**
    Randomly choose a subset $\mathcal{S} \subseteq \mathcal{D}$.
    Determine the learning rate α_t satisfying the strong Wolfe line search

$$\begin{cases} f_{\mathcal{S}}(\mathbf{x}_t + \alpha_t \mathbf{d}_t) - f_{\mathcal{S}}(\mathbf{x}_t) \le c_1 \alpha_t \nabla f_{\mathcal{S}}(\mathbf{x}_t)^T \mathbf{d}_t, \\ |\nabla f_{\mathcal{S}}(\mathbf{x}_t + \alpha_t \mathbf{d}_t)^T \mathbf{d}_t| \le -c_2 \nabla f_{\mathcal{S}}(\mathbf{x}_t)^T \mathbf{d}_t. \end{cases} \tag{3.2}$$

    Set $\alpha_t \leftarrow \min(\alpha_{max}, \max(\alpha_t, \alpha_{min}))$.
    Set the new iteration as $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \alpha_t \mathbf{d}_t$.
    Compute the variance reduced gradient $\mathbf{g}_{t+1} \leftarrow \nabla f_{\mathcal{S}}(\mathbf{x}_{t+1}) - \nabla f_{\mathcal{S}}(\mathbf{w}_k) + \mathbf{u}_k$.
    Compute conjugate parameter $\beta_{t+1} \leftarrow -\frac{|\mathbf{g}_{t+1}^T \mathbf{d}_t|}{\mathbf{g}_t^T \mathbf{d}_t} \times \frac{\|\mathbf{g}_{t+1}\|^2}{\|\mathbf{g}_t\|^2}$.
    if $\beta_{t+1} > \epsilon$ **then**
      set $\beta_{t+1} \leftarrow 0$.
    end
    Update the search direction $\mathbf{d}_{t+1} \leftarrow -\mathbf{g}_{t+1} + \beta_{t+1} \mathbf{d}_t$.
  end
  Set $\mathbf{h}_{k+1} \leftarrow \mathbf{g}_T$.
  Choose t randomly from $\{1, \ldots, T\}$ and set $\mathbf{w}_{k+1} \leftarrow \mathbf{x}_t$.
end

Algorithm 3: SIFR CG Method

3.2. Theoretical analysis

In this section we analyze the convergence property of the proposed method SIFR CG. Our analysis makes use of the following assumptions.

Assumption 1. *$\forall\ i$, $f_i : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.*

Assumption 2. *$\forall\ \boldsymbol{w} \in \mathbb{R}^n$ and $\mathcal{S} \subseteq \mathcal{D}$, there exist two positive constants ϕ and φ such that $\phi \boldsymbol{I} \le \nabla^2 f_{\mathcal{S}}(\boldsymbol{x}) \le \varphi \boldsymbol{I}$, where $\boldsymbol{I}$ denotes the identity matrix, and $\nabla^2 f_{\mathcal{S}}(\boldsymbol{x})$ is the Hessian matrix of f at point $\boldsymbol{x}$.*

For Algorithm 3, we immediately have the following result.

Theorem 1. *Suppose that Assumptions 1 and 2 hold, and let $\xi > 0$ be an upper bound on $-\boldsymbol{g}_t^T \boldsymbol{d}_t / |\boldsymbol{g}_{t+1}^T \boldsymbol{d}_t|$, then, $\forall\ t$, we have*

$$\|\boldsymbol{d}_t\|^2 \le \rho(t) \|\boldsymbol{g}_0\|^2, \tag{3.3}$$

where $\rho(t) = (\xi + \frac{4}{3}) \frac{(\epsilon\xi)^t - (\epsilon^2)^t}{\xi - \epsilon} + (\epsilon^2)^t$.

Proof. It follows from the lines 14–16 in Algorithm 3 that $\beta_{t+1} \leq \epsilon$, therefore,

$$\|\mathbf{g}_t\|^2/\|\mathbf{g}_{t-1}\|^2 \leq -\epsilon \mathbf{g}_t^T \mathbf{d}_t/|\mathbf{g}_{t+1}^T \mathbf{d}_t| \leq \epsilon\xi. \tag{3.4}$$

From the definition of $\mathbf{d}_t$, we have

$$\begin{aligned} \|\mathbf{d}_t\|^2 &= \| - \mathbf{g}_t + \beta_t \mathbf{d}_{t-1}\|^2 = \|\mathbf{g}_t\|^2 - 2\beta_t \mathbf{g}_t^T \mathbf{d}_{t-1} + \beta_t^2 \|\mathbf{d}_{t-1}\|^2 \\ &\leq \epsilon(\xi + \frac{2c_2}{1-c_2^2})\|\mathbf{g}_{t-1}\|^2 + \epsilon^2 \|\mathbf{d}_{t-1}\|^2. \end{aligned} \tag{3.5}$$

The last inequality follows from Lemma 1 and Remark 2. Using $0 < c_2 < 1/2$, it follows that

$$\begin{aligned} \|\mathbf{d}_t\|^2 &\leq \epsilon(\xi + \frac{4}{3})\|\mathbf{g}_{t-1}\|^2 + \epsilon^2 \|\mathbf{d}_{t-1}\|^2 \\ &\leq \epsilon(\xi + \frac{4}{3})[\|\mathbf{g}_{t-1}\|^2 + \epsilon^2 \|\mathbf{g}_{t-2}\|^2 + \ldots + (\epsilon^2)^{t-1}\|\mathbf{g}_0\|^2] + (\epsilon^2)^t \|\mathbf{g}_0\|^2. \end{aligned} \tag{3.6}$$

The second inequality follows from $\mathbf{d}_0 = \mathbf{g}_0$. Observing (3.4), we immediately get $\|\mathbf{g}_t\|^2 \leq (\epsilon\xi)^t \|\mathbf{g}_0\|^2$, and (3.6) can be rewritten as

$$\begin{aligned} \|\mathbf{d}_t\|^2 &\leq \epsilon(\xi + \frac{4}{3})[(\epsilon\xi)^{t-1}\|\mathbf{g}_0\|^2 + \epsilon^2(\epsilon\xi)^{t-2}\|\mathbf{g}_0\|^2 + \ldots + (\epsilon^2)^{t-1}\|\mathbf{g}_0\|^2] + (\epsilon^2)^t \|\mathbf{g}_0\|^2 \\ &= \Big[(\xi + \frac{4}{3})\frac{(\epsilon\xi)^t - (\epsilon^2)^t}{\xi - \epsilon} + (\epsilon^2)^t\Big]\|\mathbf{g}_0\|^2, \end{aligned} \tag{3.7}$$

which completes the proof. □

The following two lemmas show a lower bound on $\|\nabla f(\mathbf{x})\|^2$ and an upper bound on the variance reduced gradient estimates $\mathbf{g}_t$, respectively.

Lemma 2 (Lemma 5 [20], Lemma 1 [14])**.** *Let $\boldsymbol{w}_*$ be the unique minimizer of f, and suppose that f is strongly convex with parameter σ, then, $\forall\, \boldsymbol{x}$, we have $\|\nabla f(\boldsymbol{x})\|^2 \geq 2\sigma[f(\boldsymbol{x}) - f(\boldsymbol{w}_*)]$.*

Lemma 3 (Lemma 6 [20], Lemma 2 [14])**.** *Let $\boldsymbol{w}_*$ be the unique minimizer of f, $\boldsymbol{u}_k = \nabla f(\boldsymbol{w}_k)$, and $\boldsymbol{g}_t = \nabla f_{\mathcal{S}}(\boldsymbol{x}_t - \nabla f_{\mathcal{S}}(\boldsymbol{w}_k) + \boldsymbol{u}_k$, then by taking an expectation w.r.t. $\mathcal{S}$, we have $\mathbb{E}[\|\boldsymbol{g}_t\|^2] \leq 4\varphi[f(\boldsymbol{x}_t) - f(\boldsymbol{w}_*) + f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)]$.*

Based on the above theoretical basis, we now state our main result in the following theorem.

Theorem 2. *Suppose that Assumptions 1 and 2 hold, and let $\boldsymbol{x}_*$ be the unique minimizer of f, then, $\forall$ k, we have*

$$\mathbb{E}[f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)] \leq \theta^k \mathbb{E}[f(\boldsymbol{w}_0) - f(\boldsymbol{w}_*)]. \tag{3.8}$$

Here, the convergence rate θ is given by

$$\theta = \frac{1 + \frac{8\varphi\epsilon\alpha_{max}T}{3} + 4\varphi^2\alpha_{max}^2\Lambda}{2\sigma\alpha_{min}T - \frac{8\varphi\epsilon\alpha_{max}T}{3}} < 1 \tag{3.9}$$

with $\Lambda = \frac{\xi+4/3}{\xi-\epsilon} \times \frac{1-(\xi\epsilon)^T}{1-\xi\epsilon} - \frac{\epsilon+4/3}{\xi-\epsilon} \times \frac{1-(\epsilon^2)^T}{1-\epsilon^2}$, assuming that we choose $8\varphi\epsilon\alpha_{max} < 3\sigma\alpha_{min}$ and that we choose T large enough to satisfy

$$T > \frac{3 + 12\varphi^2\alpha_{max}^2\Lambda}{6\sigma\alpha_{min} - 16\varphi\epsilon\alpha_{max}}. \tag{3.10}$$

Proof. Using the Lipschitz continuity of ∇f, which follows from Assumption 2, we have

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^T(\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{\varphi}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &= f(\mathbf{x}_t) + \alpha_t \nabla f(\mathbf{x}_t)^T \mathbf{d}_t + \frac{\varphi\alpha_t^2}{2}\|\mathbf{d}_t\|^2. \end{aligned} \tag{3.11}$$

Taking expectations of (3.11) and using Lemma 2 gives

$$\begin{aligned} \mathbb{E}[f(\mathbf{x}_{t+1})] &\leq \mathbb{E}[f(\mathbf{x}_t)] + \alpha_t \mathbb{E}[\nabla f(\mathbf{x}_t)^T \mathbf{d}_t] + \frac{\varphi\alpha_t^2}{2}\mathbb{E}[\|\mathbf{d}_t\|^2] \\ &\leq \mathbb{E}[f(\mathbf{x}_t)] + \alpha_t\Big(- 2\sigma\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] + \beta_t \mathbb{E}[\mathbf{g}_t^T \mathbf{d}_{t-1}]\Big) + \frac{\varphi\alpha_t^2}{2}\mathbb{E}[\|\mathbf{d}_t\|^2] \\ &\leq \mathbb{E}[f(\mathbf{x}_t)] + \alpha_t\Big(- 2\sigma\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] + \beta_t \frac{c_2}{1 - c_2^2}\mathbb{E}[\|\mathbf{g}_{t-1}\|^2]\Big) + \frac{\varphi\alpha_t^2}{2}\mathbb{E}[\|\mathbf{d}_t\|^2] \\ &\leq \mathbb{E}[f(\mathbf{x}_t)] + \alpha_t\Big(- 2\sigma\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] + \frac{2\beta_t}{3}\mathbb{E}[\|\mathbf{g}_{t-1}\|^2]\Big) + \frac{\varphi\alpha_t^2}{2}\mathbb{E}[\|\mathbf{d}_t\|^2]. \end{aligned} \tag{3.12}$$

Using Theorem 1 and Lemma 3, (3.12) becomes

$$\begin{aligned} &\mathbb{E}[f(\mathbf{x}_{t+1})] \\ &\leq \mathbb{E}[f(\mathbf{x}_t)] - 2\sigma\alpha_t\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] + \frac{8\varphi\alpha_t\beta_t}{3}\Big(\mathbb{E}[f(\mathbf{x}_{t-1}) - f(\mathbf{w}_*)] + \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]\Big) \\ &\quad + \frac{\varphi\alpha_t^2}{2} \times 4\varphi\rho(t)\Big(\mathbb{E}[f(\mathbf{x}_0 - f(\mathbf{w}_*)] + \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]\Big) \\ &\leq \mathbb{E}[f(\mathbf{x}_t)] - 2\sigma\alpha_{min}\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] \\ &\quad + \frac{8\varphi\epsilon\alpha_{max}}{3}\Big(\mathbb{E}[f(\mathbf{x}_{t-1}) - f(\mathbf{w}_*)] + \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]\Big) + 4\varphi^2\alpha_{max}^2\rho(t)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]. \end{aligned} \tag{3.13}$$

By summing (3.13) over $t = 0, 1, \ldots, T-1$, we obtain

$$\begin{aligned} \mathbb{E}[f(\mathbf{x}_T)] \leq \mathbb{E}[f(\mathbf{x}_0)] &- 2\sigma\alpha_{min}\sum_{t=0}^{T-1}\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] \\ &+ \frac{8\varphi\epsilon\alpha_{max}}{3}\sum_{t=0}^{T-2}\Big(\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{w}_*)] + \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]\Big) \\ &+ 4\varphi^2\alpha_{max}^2\sum_{t=0}^{T-1}\rho(t)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]. \end{aligned} \tag{3.14}$$

Rearranging (3.14) gives

$$\begin{aligned} 0 \leq \mathbb{E}[f(\mathbf{w}_k)] - \mathbb{E}[f(\mathbf{x}_T)] &- 2\sigma\alpha_{min}T\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}_*)] \\ &+ \frac{8\varphi\epsilon\alpha_{max}T}{3}\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}_*)] + \frac{8\varphi\epsilon\alpha_{max}T}{3}\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)] \\ &+ 4\varphi^2\alpha_{max}^2\sum_{t=0}^{T-1}\rho(t)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)]. \end{aligned} \tag{3.15}$$

Further, we have

$$\begin{aligned} 0 \leq \mathbb{E}[f(\mathbf{w}_k)] - \mathbb{E}[f(\mathbf{w}_*)] &+ \Big(\frac{8\varphi\epsilon\alpha_{max}T}{3} + 4\varphi^2\alpha_{max}^2\sum_{t=0}^{T-1}\rho(t)\Big)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)] \\ &+ \Big(\frac{8\varphi\epsilon\alpha_{max}T}{3} - 2\sigma\alpha_{min}T\Big)\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}_*)] \\ = \Big(1 + \frac{8\varphi\epsilon\alpha_{max}T}{3} &+ 4\varphi^2\alpha_{max}^2\sum_{t=0}^{T-1}\rho(t)\Big)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)] \\ &+ \Big(\frac{8\varphi\epsilon\alpha_{max}T}{3} - 2\sigma\alpha_{min}T\Big)\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}_*)]. \end{aligned} \tag{3.16}$$

The second inequality follows from $f(\mathbf{w}_*) \leq f(\mathbf{x}_T)$. Note that $\sum_{t=0}^{T-1}\rho(t) = \frac{\xi+4/3}{\xi-\epsilon} \times \frac{1-(\xi\epsilon)^T}{1-\xi\epsilon} - \frac{\epsilon+4/3}{\xi-\epsilon} \times \frac{1-(\epsilon^2)^T}{1-\epsilon^2}$ and denote this result by Λ, then we have

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}_*)] \leq \theta\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}_*)], \tag{3.17}$$

on condition that $\alpha_{max}/\alpha_{min} < 0.75\sigma/(\varphi\epsilon)$, where $\theta = \frac{1+\frac{8\varphi\epsilon\alpha_{max}T}{3}+4\varphi^2\alpha_{max}^2\Lambda}{2\sigma\alpha_{min}T-\frac{8\varphi\epsilon\alpha_{max}T}{3}}$. Since we chose T to satisfy (3.10), it follows that the rate θ is less than one. Thus, we get the result as desired. □

4. Comparative experiments

To show the efficiency and effectiveness of the proposed method, we compare it with SVRG [15] (a stochastic learning algorithm) and IFR CG [13] (a batch learning algorithm) on five classification datasets, as shown in Table 1. These datasets are obtained from the LIBSVM database *. All experiments are implemented using the Armadillo C++ library on a PC (Core i9-9900X CPU@3.50GHz, 64 GB of memory) with Ubuntu 18.04.1 operating system.

Table 1. Summary of the benchmark datasets.

datasets	m	n
a9a	32561	123
w8a	49749	300
ijcnn1	49990	22
susy	5000000	18
higgs	11000000	28

*Note: m and n represent the number of samples and features of the data set respectively.

*http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

4.1. Problem setup

We evaluate these methods on three popular machine learning problems, including

• l_2-logistic regression

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \log(1 + e^{-y_i\mathbf{w}^T\mathbf{z}_i}) + \lambda\|\mathbf{w}\|^2, \tag{4.1}$$

• l_1-SVM

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i\mathbf{w}^T\mathbf{z}_i) + \lambda\|\mathbf{w}\|^2, \tag{4.2}$$

• l_2-SVM

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i\mathbf{w}^T\mathbf{z}_i)^2 + \lambda\|\mathbf{w}\|^2, \tag{4.3}$$

where $\lambda > 0$ is the regularization parameter.

We set $\lambda = \{0.01, 0.001, 0.0001\}$ and set $|\mathcal{S}| = \sqrt{m}$ during the whole experiments. We set the number of iterations of the outer loop to 25, and set the inner loop index T to 30 and 50, respectively. For SVRG, we set $\eta = \{0.1, 0.05, 0.001, 0.005, 0.01, 0.05\}$. For IFR CG and SIFR CG, we set $c_1 = 10^{-4}$ and $c_2 = 0.1$. Additionally, in SIFR CG, we set $\alpha_{max} = 1/\alpha_{min} = 10^5$ and $\epsilon = 10$. We randomly divide each dataset into three parts, $1/3$ for test, $1/5$ for validation, and the rest for training. After finding the optimal parameters that maximize the AUC values on validation set, we employ the corresponding model to the test set.

4.2. Experimental results

We first adopt the the cross validation technique to obtain the optimal parameters for each learning model, and the cross validation results are summarize in Tables 2 and 3. We can see that for different learning models and different datasets, the optimal parameters are often data-dependent and should be tuned correspondingly.

After getting the optimal learning rates for SVRG and the optimal regularization parameters for all the methods, we report the changing curves of AUC values on each test datasets obtained by different method with increasing computational cost in Figures 2−5. In order to fairly compare the performance of different methods, all the x-axes represent the computational cost that is measured by the number of gradient computations divided by m, and all the y-axes represent the AUC value. In general, the closer the AUC is to 1, the higher the accuracy of the method is. Figures 2−5 show the changing results when $T = 30$, $T = 40$, $T = 50$, and $T = 60$, respectively. Firstly, we can see from the four figures that the proposed method SIFR CG significantly outperforms SVRG and IFR CG on the whole test datssets. That is, the classification accuracy of our method is the highest. Secondly, SVRG and IFR CG have their own advantages and disadvantages on different datasets, therefore, they are hard to differentiate. Thirdly, unlike SVRG and IFR CG, we find that the role of increasing the number of loops to improve the performance of SIFR CG is less obvious. In other words, the SIFR CG method takes fewer iterations to get the optimal solution.

Table 2. Summary of the optimal parameters for each method obtained by cross validation on different datasets and different learning problems.

l_2-logistic regression	$T = 30$			$T = 40$		
	SVRG	IFR CG	SIFR CG	SVRG	IFR CG	SIFR CG
a9a	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$
w8a	$\lambda = .1, \eta = .1$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .1$	$\lambda = .1$	$\lambda = .008$
ijcnn1	$\lambda = .05, \eta = .1$	$\lambda = .05$	$\lambda = .005$	$\lambda = .05, \eta = .1$	$\lambda = .05$	$\lambda = .005$
susy	$\lambda = .005, \eta = .1$	$\lambda = .005$	$\lambda = .005$	$\lambda = .005, \eta = .1$	$\lambda = .005$	$\lambda = .005$
higgs	$\lambda = .05, \eta = .1$	$\lambda = .05$	$\lambda = .005$	$\lambda = .05, \eta = .1$	$\lambda = .05$	$\lambda = .005$
l_1-SVM	$T = 30$			$T = 40$		
	SVRG	IFR CG	SIFR CG	SVRG	IFR CG	SIFR CG
a9a	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .008$	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .008$
w8a	$\lambda = .1, \eta = .0001$	$\lambda = .05$	$\lambda = .1$	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .1$
ijcnn1	$\lambda = .1, \eta = .001$	$\lambda = .05$	$\lambda = .008$	$\lambda = .1, \eta = .0001$	$\lambda = .05$	$\lambda = .05$
susy	$\lambda = .008, \eta = .05$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$
higgs	$\lambda = .008, \eta = .1$	$\lambda = .1$	$\lambda = .005$	$\lambda = .05, \eta = .05$	$\lambda = .1$	$\lambda = .005$
l_2-SVM	$T = 30$			$T = 40$		
	SVRG	IFR CG	SIFR CG	SVRG	IFR CG	SIFR CG
a9a	$\lambda = .01, \eta = .01$	$\lambda = .1$	$\lambda = .05$	$\lambda = .005, \eta = .01$	$\lambda = .1$	$\lambda = .05$
w8a	$\lambda = .1, \eta = .001$	$\lambda = .1$	$\lambda = .1$	$\lambda = .1, \eta = .001$	$\lambda = .1$	$\lambda = .05$
ijcnn1	$\lambda = .1, \eta = .05$	$\lambda = .1$	$\lambda = .05$	$\lambda = .1, \eta = .05$	$\lambda = .1$	$\lambda = .05$
susy	$\lambda = .005, \eta = .05$	$\lambda = .005$	$\lambda = .005$	$\lambda = .005, \eta = .05$	$\lambda = .005$	$\lambda = .005$
higgs	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$

Table 3. Summary of the optimal parameters for each method obtained by cross validation on different datasets and different learning problems (cont.).

l_2-logistic regression	$T = 50$			$T = 60$		
	SVRG	IFR CG	SIFR CG	SVRG	IFR CG	SIFR CG
a9a	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$
w8a	$\lambda = .1, \eta = .1$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .1$	$\lambda = .1$	$\lambda = .005$
ijcnn1	$\lambda = .05, \eta = .1$	$\lambda = .05$	$\lambda = .005$	$\lambda = .05, \eta = .1$	$\lambda = .05$	$\lambda = .005$
susy	$\lambda = .005, \eta = .1$	$\lambda = .005$	$\lambda = .005$	$\lambda = .005, \eta = .1$	$\lambda = .005$	$\lambda = .005$
higgs	$\lambda = .01, \eta = .1$	$\lambda = .05$	$\lambda = .005$	$\lambda = .008, \eta = .1$	$\lambda = .05$	$\lambda = .005$
l_1-SVM	$T = 50$			$T = 60$		
	SVRG	IFR CG	SIFR CG	SVRG	IFR CG	SIFR CG
a9a	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .1$	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .01$
w8a	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .01$	$\lambda = .1, \eta = .01$	$\lambda = .05$	$\lambda = .05$
ijcnn1	$\lambda = .1, \eta = .0001$	$\lambda = .05$	$\lambda = .005$	$\lambda = .1, \eta = .001$	$\lambda = .05$	$\lambda = .008$
susy	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$
higgs	$\lambda = .05, \eta = .05$	$\lambda = .1$	$\lambda = .005$	$\lambda = .008, \eta = .05$	$\lambda = .1$	$\lambda = .005$
l_2-SVM	$T = 50$			$T = 60$		
	SVRG	IFR CG	SIFR CG	SVRG	IFR CG	SIFR CG
a9a	$\lambda = .1, \eta = .005$	$\lambda = .1$	$\lambda = .1$	$\lambda = .1, \eta = .005$	$\lambda = .1$	$\lambda = .05$
w8a	$\lambda = .1, \eta = .001$	$\lambda = .1$	$\lambda = .1$	$\lambda = .1, \eta = .001$	$\lambda = .1$	$\lambda = .05$
ijcnn1	$\lambda = .1, \eta = .05$	$\lambda = .1$	$\lambda = .01$	$\lambda = .05, \eta = .05$	$\lambda = .1$	$\lambda = .1$
susy	$\lambda = .005, \eta = .05$	$\lambda = .005$	$\lambda = .005$	$\lambda = .005, \eta = .05$	$\lambda = .005$	$\lambda = .005$
higgs	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$	$\lambda = .1, \eta = .01$	$\lambda = .1$	$\lambda = .005$

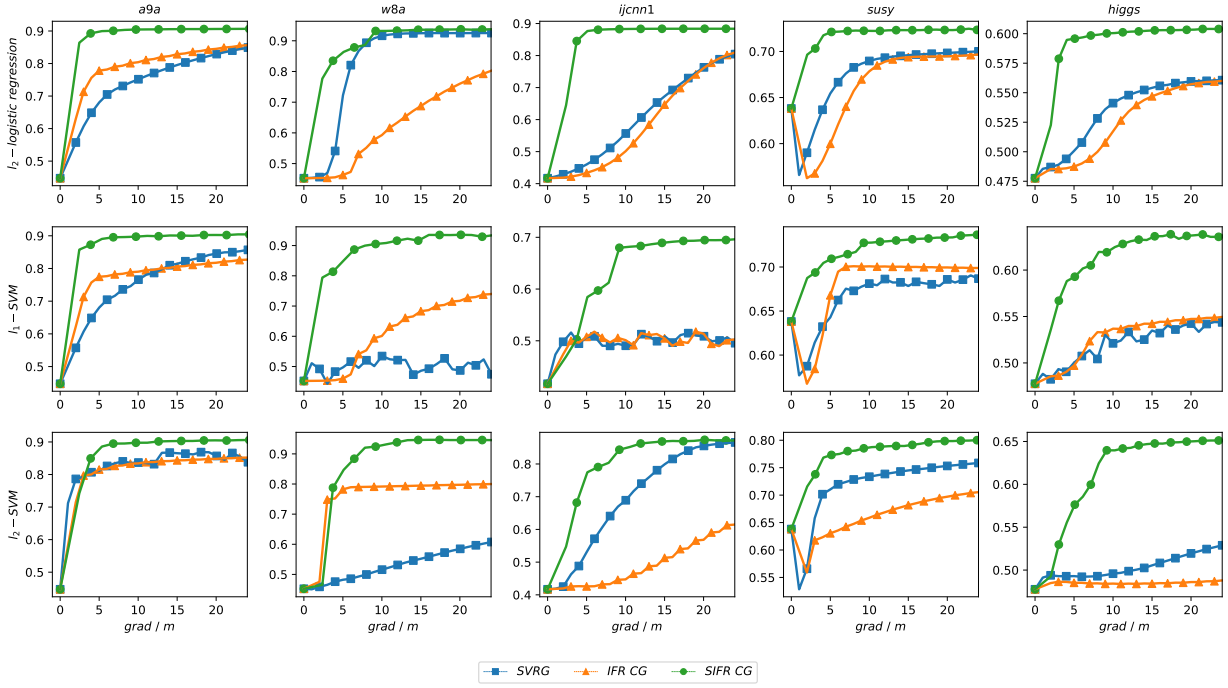


Figure 2. Changing curves of AUC values between different methods on the test data when $T = 30$.

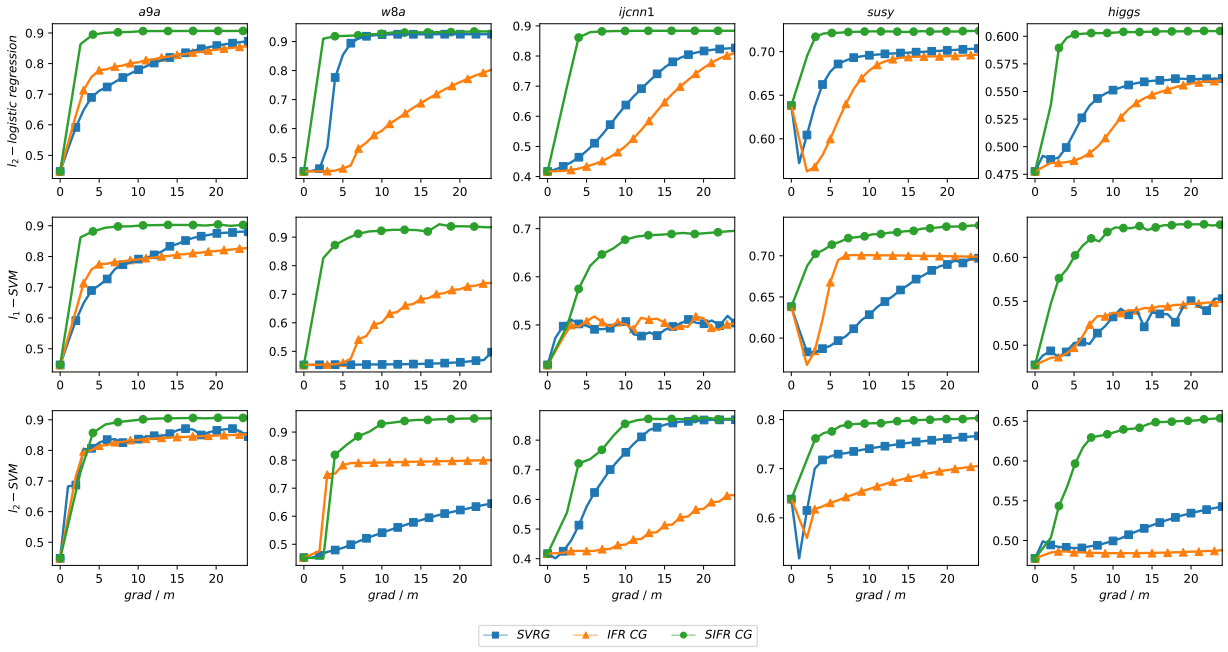


Figure 3. Changing curves of AUC values between different methods on the test data when $T = 40$.

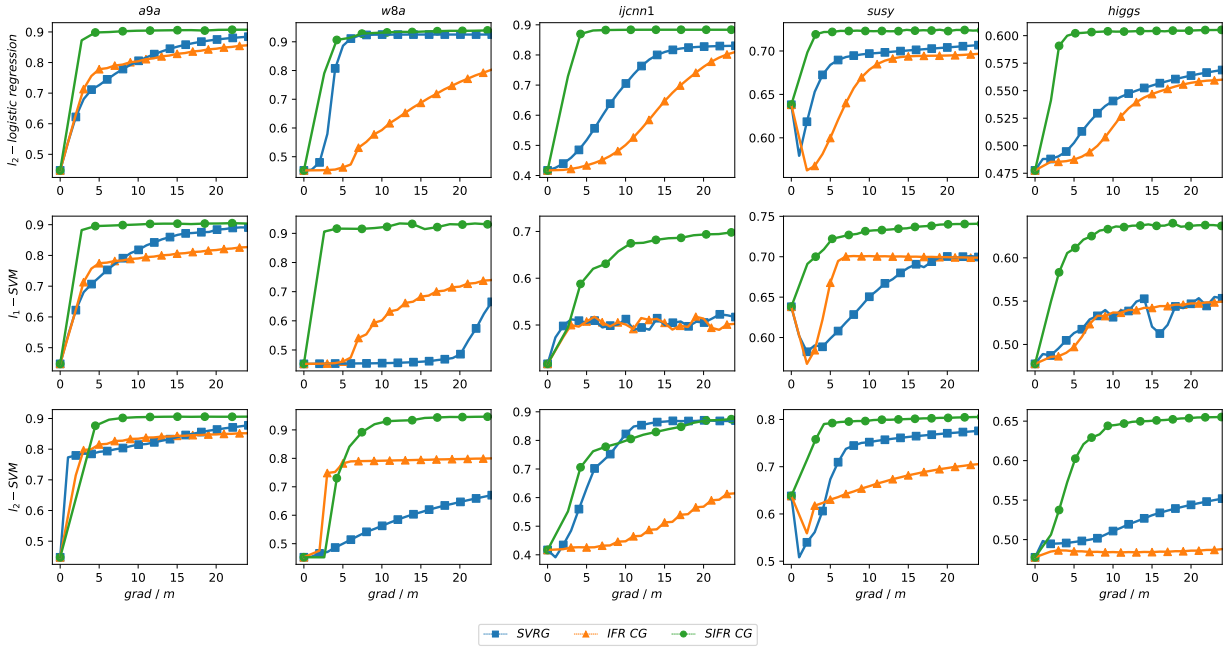


Figure 4. Changing curves of AUC values between different methods on the test data when $T = 50$.

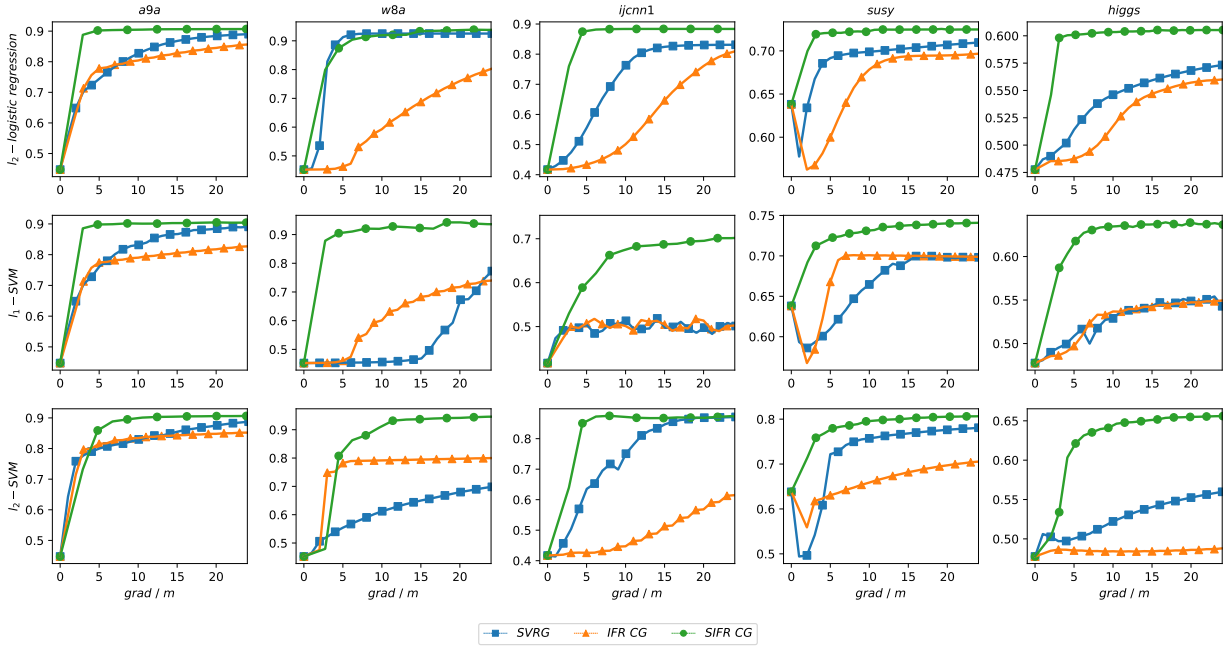


Figure 5. Changing curves of AUC values between different methods on the test data when $T = 60$.

Additionally, Figures 6−17 show the convergence behaviors and the corresponding average computing time (the numbers of iterations of the outer loop is set to 25) of the three methods on each training dataset. In the convergence behavior figures (see Figures 6, 7, 9, 10, 12, 13, 15 and 16), all the x-axes also represent the computational cost that is measured by the number of gradient computations divided by m, and all the y-axes represent the logarithm of objective function value (base 10). We see that the convergence rate of the SVRG is not as fast as the other two methods because SVRG is sensitive to the learning rate. However, for all test environments, the proposed method SIFR CG has the fastest convergence speed and reaches the smallest objective function value, which is consistent with the AUC results shown in Figures 2−5. In the average computing time figures (see Figures 8, 11, 14, and 17), all the x-axes are in logarithmic time (base 10). We can see that SVRG has the least computing time on these datasets. On the contrary, IFR CG employs the entire dataset to conduct the line search to select the learning rate at each iteration, resulting in a heavy computing burden. The computing time of SIFR CG is between SVRG and IFR CG, and it can be observed that the computing time of SIFR CG is close to IFR CG when the dataset is small and is close to SVRG when the data is large.

Overall, preliminary experiments show that the proposed method SIFR CG not only has better learning performance, but converges speedily. And The advantage of SIFR CG is more obvious on large-scale datasets.

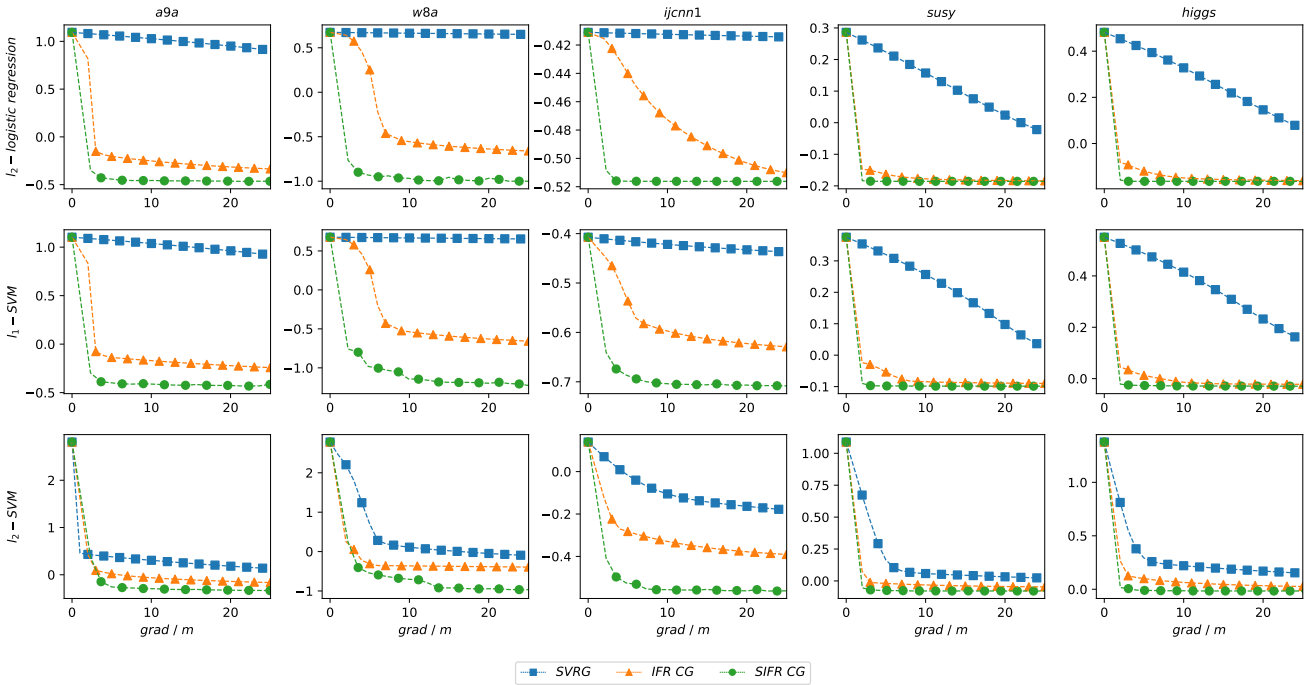


Figure 6. Convergence behaviors of different methods on the training data when $\lambda = 10^{-2}$ and $T = 30$.

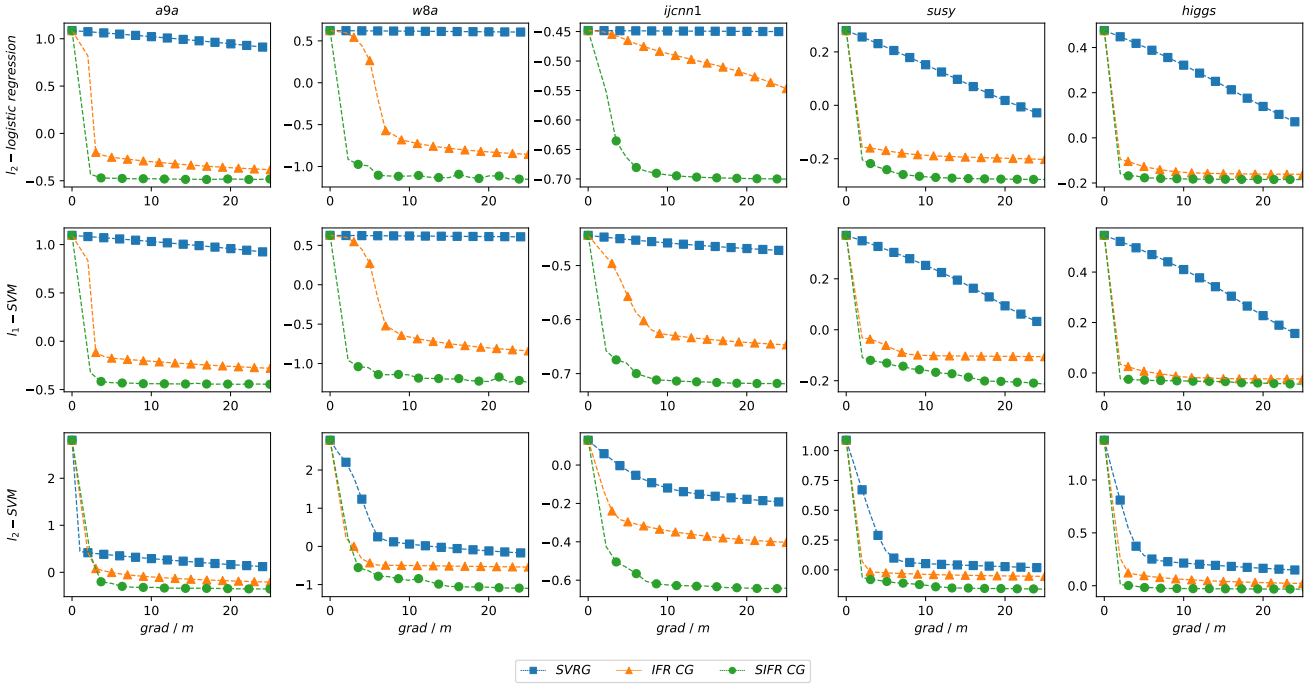


Figure 7. Convergence behaviors of different methods on the training data when $\lambda = 10^{-4}$ and $T = 30$.

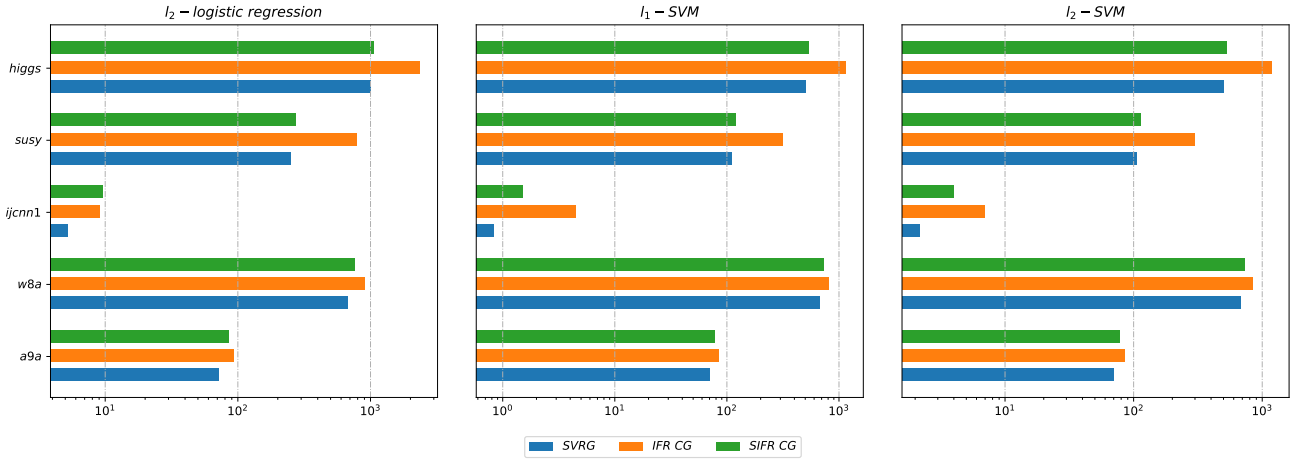


Figure 8. Average computing time of different methods on the test data when $T = 30$.

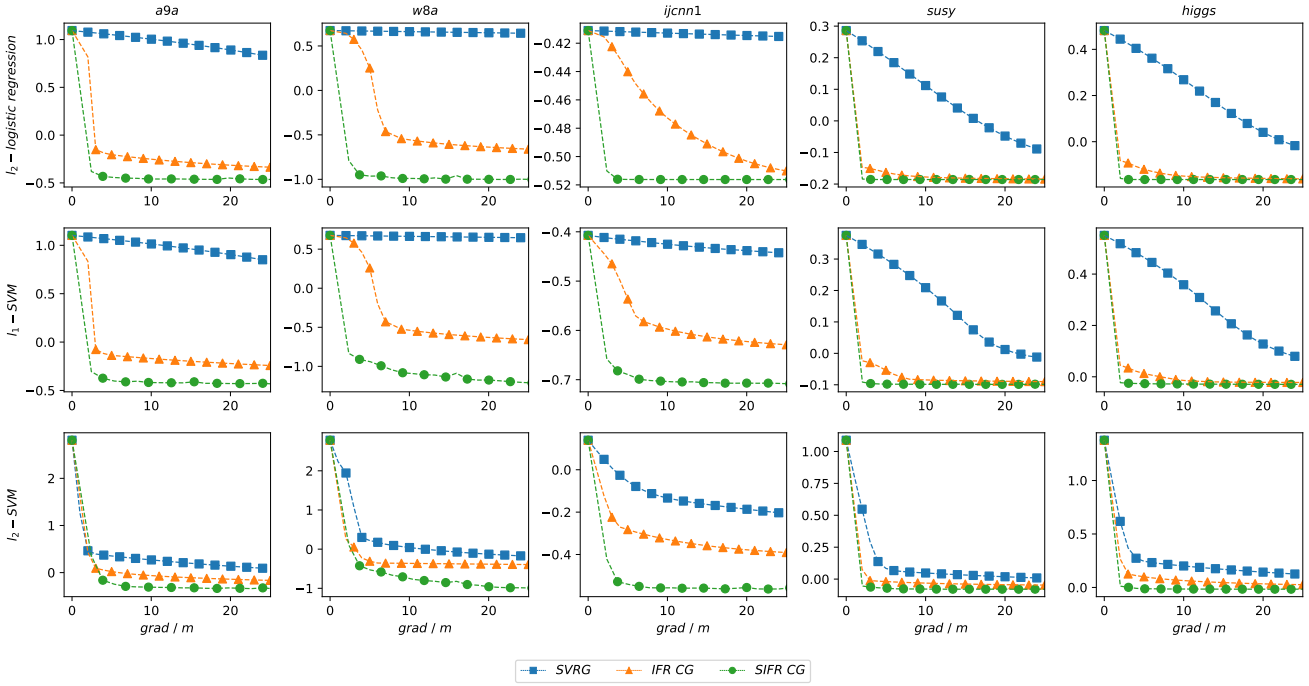


Figure 9. Convergence behaviors of different methods on the training data when $\lambda = 10^{-2}$ and $T = 40$.

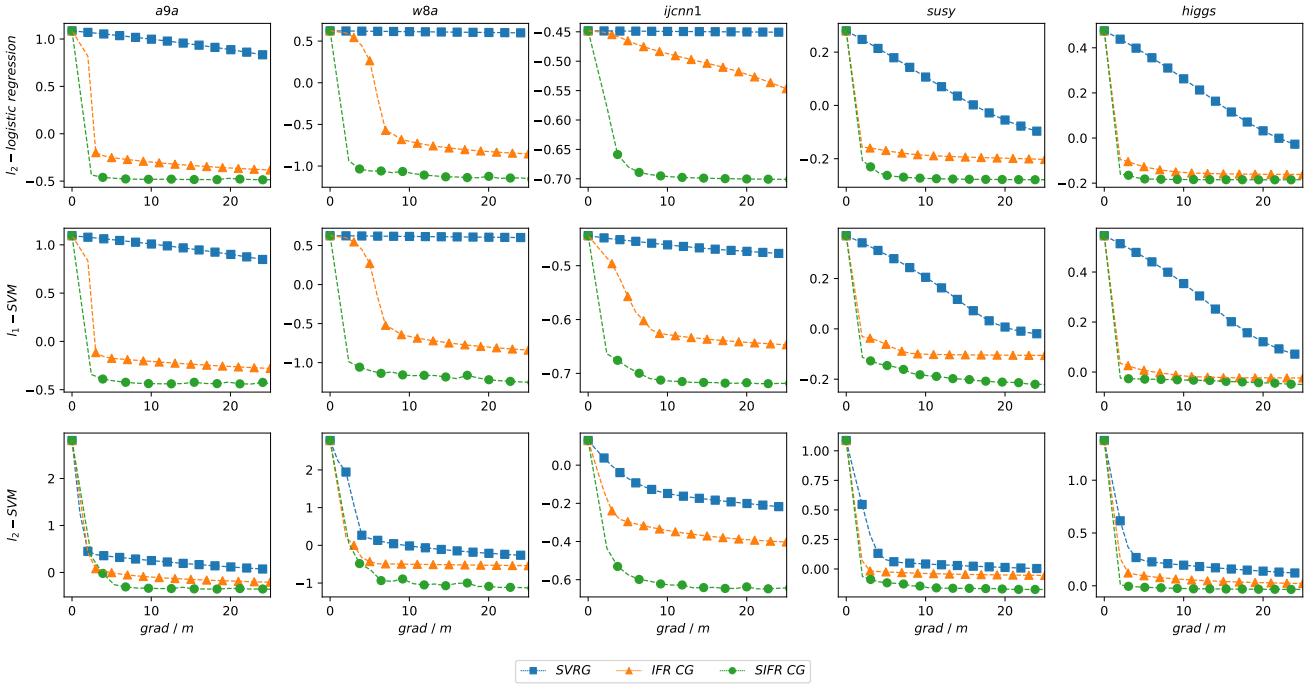


Figure 10. Convergence behaviors of different methods on the training data when $\lambda = 10^{-4}$ and $T = 40$.

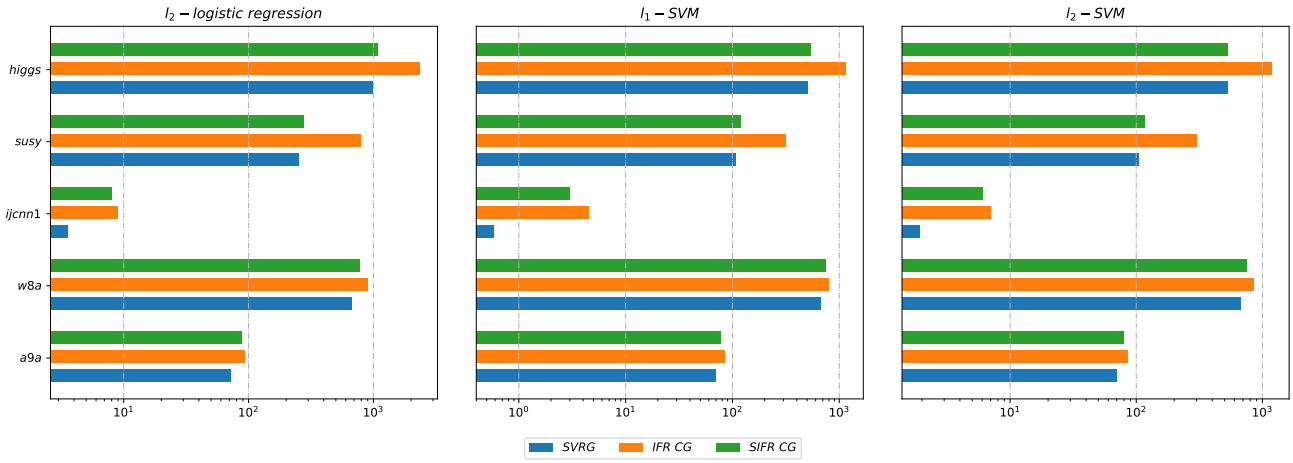


Figure 11. Average computing time of different methods on the test data when $T = 40$.

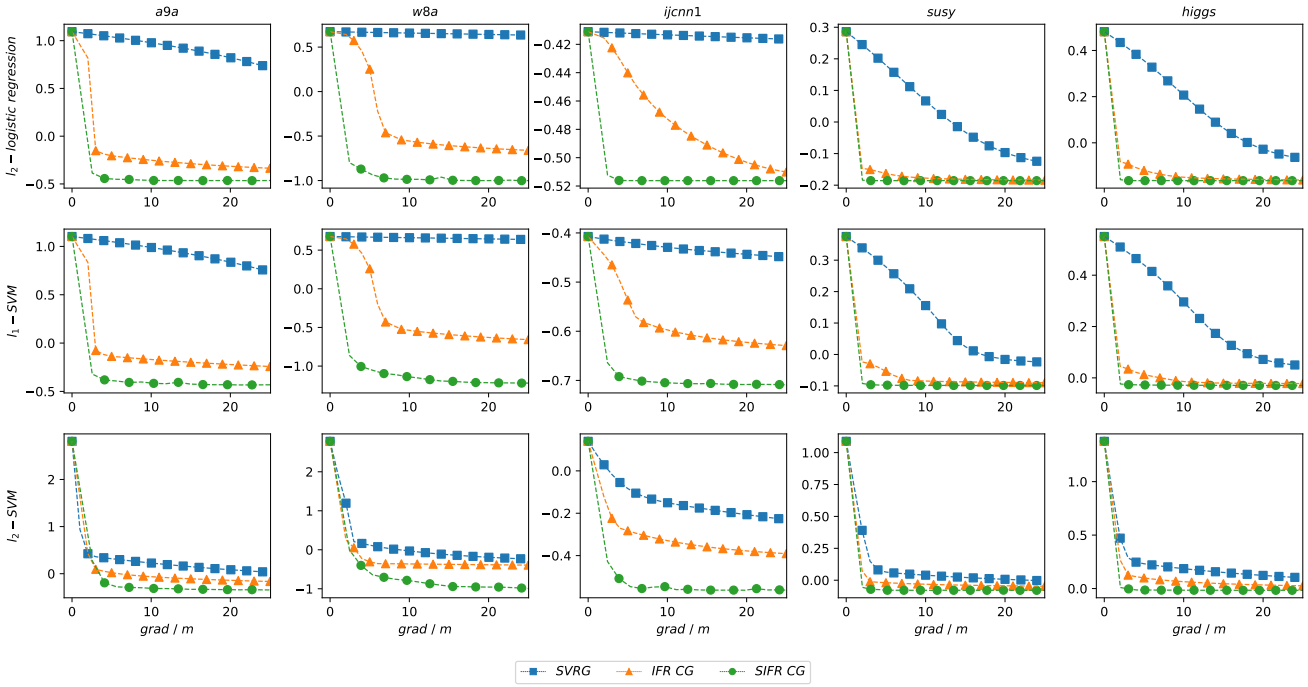


Figure 12. Convergence behaviors of different methods on the training data when $\lambda = 10^{-2}$ and $T = 50$.

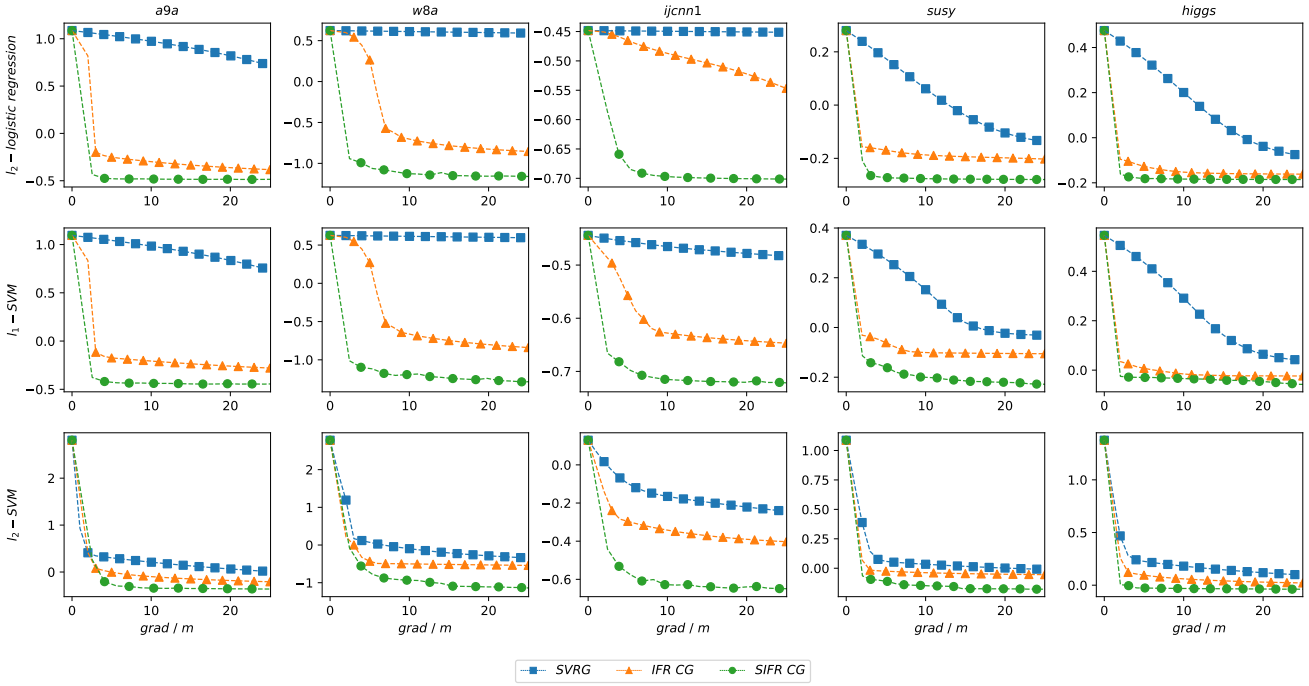


Figure 13. Convergence behaviors of different methods on the training data when $\lambda = 10^{-4}$ and $T = 50$.

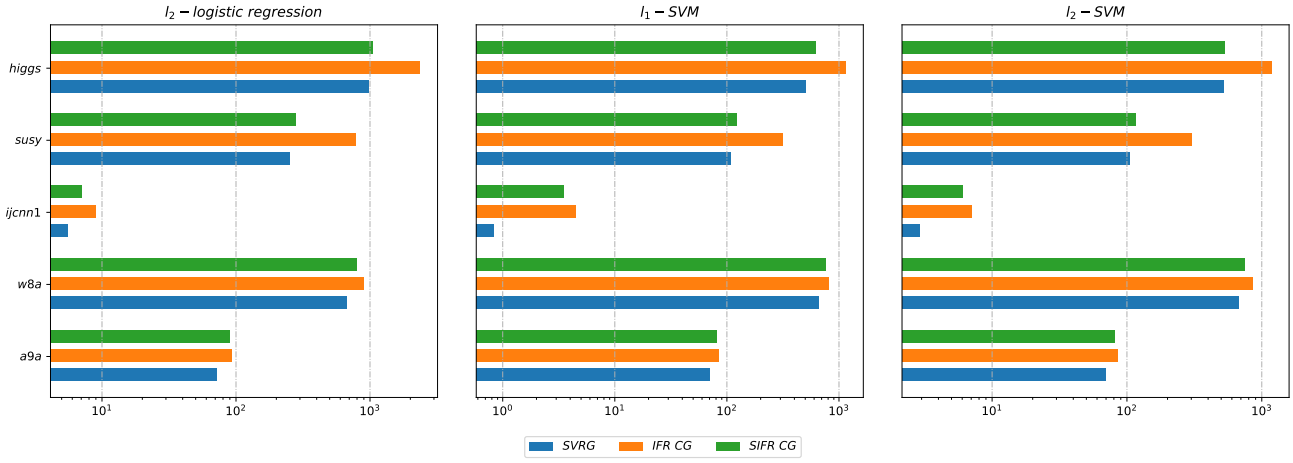


Figure 14. Average computing time of different methods on the test data when $T = 50$.

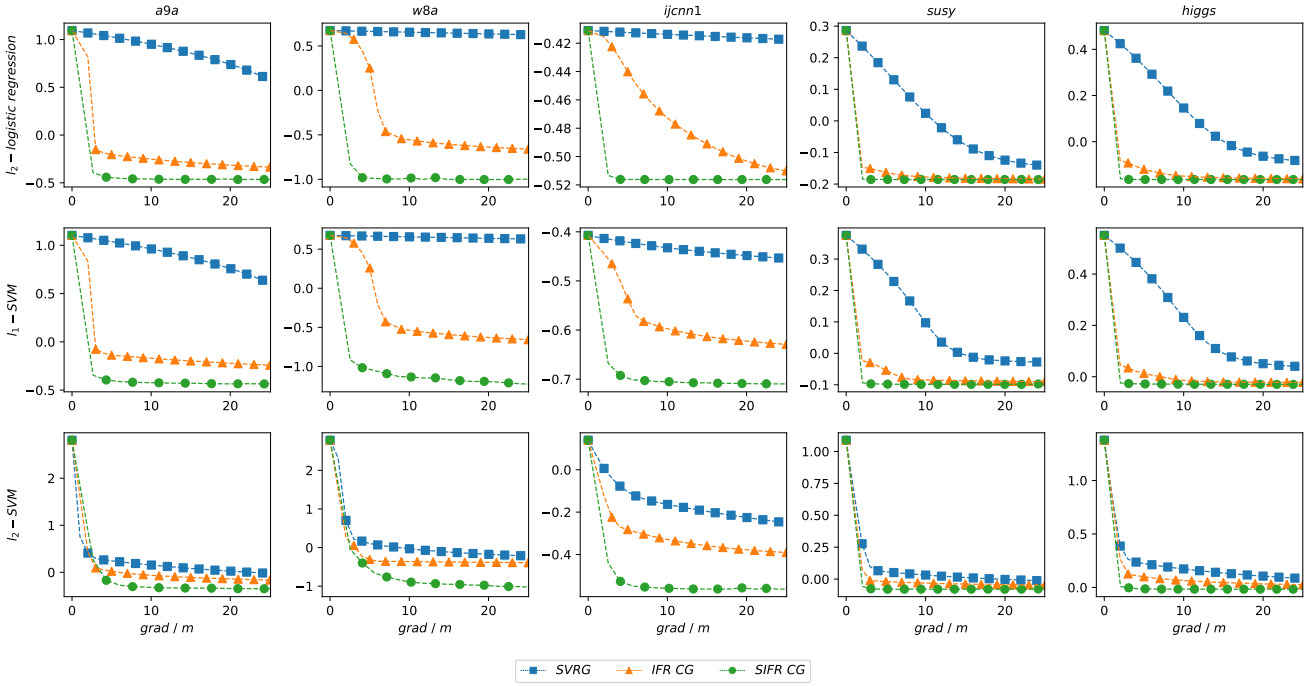


Figure 15. Convergence behaviors of different methods on the training data when $\lambda = 10^{-2}$ and $T = 60$.

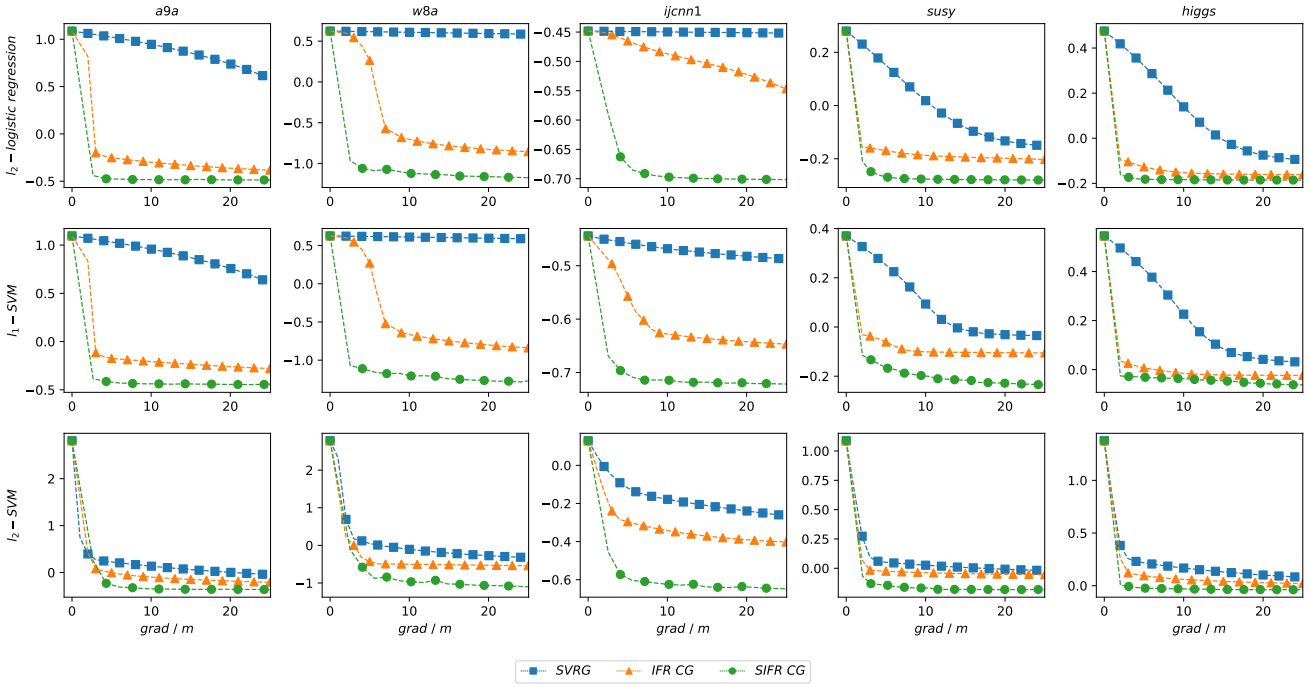


Figure 16. Convergence behaviors of different methods on the training data when $\lambda = 10^{-4}$ and $T = 60$.

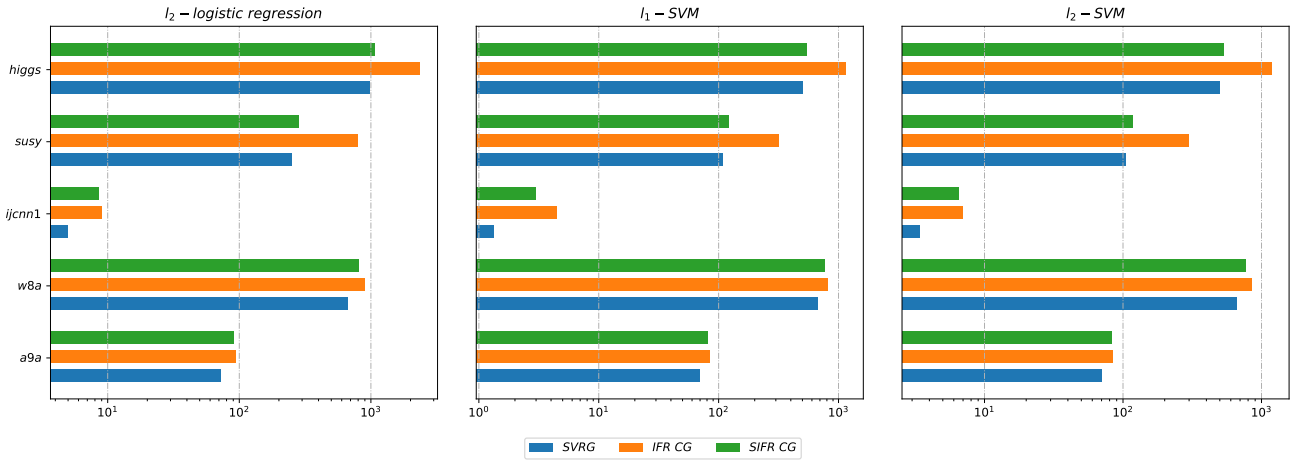


Figure 17. Average computing time of different methods on the test data when $T = 60$.

5. Conclusions

In this paper, we proposed a stochastic Fletcher-Reeves conjugate gradient algorithm and proved a linear rate of convergence in the strongly convex case. Comparative experiments on several benchmark datasets demonstrate that the proposed algorithm performs well on large-scale smooth and nonsmooth machine learning problems.

There are several interesting issues to study in future work. For example, we can replace the Wolfe line search by a nonmonotone line search to find the optimal solution faster or propose more effective conjugate gradient update parameters.

Acknowledgments

The authors thank the editor and the reviewers for their constructive comments and suggestions that greatly improved the quality and presentation of this paper. This work was partly supported by the National Natural Science Foundation of China (Grant Nos. 11661007, 61671456 and 61806004), the China Postdoctoral Science Foundation (Grant No. 2020T130767), the Natural Science Foundation of the Anhui Higher Education Institutions of China (Grant No. KJ2019A0082), and the Natural Science Foundation of Zhejiang Province, China (Grant No. LD19A010002).

Conflict of interest

The authors declare that there is no conflict of interests regarding the publication of this article.

References

1. J. Barzilai, J. M. Borwein, Two-point step size gradient methods, *IMA J. Numer. Anal.,* **8** (1988), 141–148.

2. E. Bisong, *Batch vs. online larning*, Building Machine Learning and Deep Learning Models on Google Cloud Platform, 2019.
3. L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, *SIAM Rev.,* **60** (2018), 223–311.
4. Y. H. Dai, Y. Yuan, *Nonlinear conjugate gradient methods*, Shanghai: Shanghai Scientific Technical Publishers, 2000.
5. D. Davis, B. Grimmer, Proximally guided stochastic subgradient method for nonsmooth, nonconvex problems, *SIAM J. Optim.,* **29** (2019), 1908–1930.
6. R. Dehghani, N. Bidabadi, H. Fahs, M. M. Hosseini, A conjugate gradient method based on a modified secant relation for unconstrained optimization, *Numer. Funct. Anal. Optim.,* **41** (2020), 621–634.
7. P. Faramarzi, K. Amini, A modified spectral conjugate gradient method with global convergence, *J. Optim. Theory Appl.,* **182** (2019), 667–690.
8. R. Fletcher, C. M. Reeves, Function minimization by conjugate gradients, *Comput. J.,* **7** (1964), 149–154.
9. J. C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, *SIAM J. Optim.,* **2** (1992), 21–42.
10. W. W. Hager, H. Zhang, Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent, *ACM Trans. Math. Software,* **32** (2006), 113–137.
11. A. S. Halilu, M. Y. Waziri, Y. B. Musa, Inexact double step length method for solving systems of nonlinear equations, *Stat. Optim. Inf. Comput.,* **8** (2020), 165–174.
12. H. Jiang, P. Wilford, A stochastic conjugate gradient method for the approximation of functions, *J. Comput. Appl. Math.,* **236** (2012), 2529–2544.
13. X. Jiang, J. Jian, Improved Fletcher-Reeves and Dai-Yuan conjugate gradient methods with the strong Wolfe line search, *J. Comput. Appl. Math.,* **348** (2019), 525–534.
14. X. B. Jin, X. Y. Zhang, K. Huang, G. G. Geng, Stochastic conjugate gradient algorithm with variance reduction, *IEEE Trans. Neural Networks Learn. Syst.,* **30** (2018), 1360–1369.
15. R. Johnson, T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, Advances in Neural Information Processing Systems, 2013.
16. X. L. Li, Preconditioned stochastic gradient descent, *IEEE Trans. Neural Networks Learn. Syst.,* **29** (2018), 1454–1466.
17. Y. Liu, X. Wang, T. Guo, A linearly convergent stochastic recursive gradient method for convex optimization, *Optim. Lett.,* 2020. Doi: 10.1007/s11590-020-01550-x.
18. M. Lotfi, S. M. Hosseini, An efficient Dai-Liao type conjugate gradient method by reformulating the CG parameter in the search direction equation, *J. Comput. Appl. Math.,* **371** (2020), 112708.
19. S. Mandt, M. D. Hoffman, D. M. Blei, Stochastic gradient descent as approximate Bayesian inference, *J. Mach. Learn. Res.,* **18** (2017), 4873–4907.
20. P. Moritz, R. Nishihara, M. I. Jordan, *A linearly-convergent stochastic L-BFGS algorithm*, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016.

21. L. M. Nguyen, J. Liu, K. Scheinberg, M. Takáč, *SARAH: A novel method for machine learning problems using stochastic recursive gradient,* Proceedings of the 34th International Conference on Machine Learning, 2017.
22. A. Nitanda, *Accelerated stochastic gradient descent for minimizing finite sums*, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016.
23. H. Robbins, S. Monro, A stochastic approximation method, *Ann. Math. Statist.,* **22** (1951), 400–407.
24. N. N. Schraudolph, T. Graepel, *Combining conjugate direction methods with stochastic approximation of gradients*, Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, 2003.
25. G. Shao, W. Xue, G. Yu, X. Zheng, Improved SVRG for finite sum structure optimization with application to binary classification, *J. Ind. Manage. Optim.,* **16** (2020), 2253–2266.
26. C. Tan, S. Ma, Y. H. Dai, Y. Qian, *Barzilai-Borwein step size for stochastic gradient descent*, Advances in Neural Information Processing Systems, 2016.
27. P. Toulis, E. Airoldi, J. Rennie, *Statistical analysis of stochastic gradient methods for generalized linear models*, Proceedings of the 31th International Conference on Machine Learning, 2014.
28. V. Vapnik, *The nature of statistical learning theory*, New York: Springer, 1995.
29. L. Xiao, T. Zhang, A proximal stochastic gradient method with progressive variance reduction, *SIAM J. Optim.,* **24** (2014), 2057–2075.
30. Z. Xu, Y. H. Dai, A stochastic approximation frame algorithm with adaptive directions, *Numer. Math. Theory Methods Appl.,* **1** (2008), 460–474.
31. W. Xue, J. Ren, X. Zheng, Z. Liu, Y. Ling, A new DY conjugate gradient method and applications to image denoising, *IEICE Trans. Inf. Syst.,* **101** (2018), 2984–2990.
32. Q. Zheng, X. Tian, N. Jiang, M. Yang, Layer-wise learning based stochastic gradient descent method for the optimization of deep convolutional neural network, *J. Intell. Fuzzy Syst.,* **37** (2019), 5641–5654.

AIMS Press