Journal of Global Optimization (2023) 87:1009–1025
https://doi.org/10.1007/s10898-022-01205-4

A mini-batch stochastic conjugate gradient algorithm with variance reduction

Caixia Kou[1] · **Han Yang**[1]

Received: 1 September 2021 / Accepted: 16 June 2022 / Published online: 1 July 2022

Abstract
Stochastic gradient descent method is popular for large scale optimization but has slow convergence asymptotically due to the inherent variance. To remedy this problem, there have been many explicit variance reduction methods for stochastic descent, such as SVRG Johnson and Zhang [*Advances in neural information processing systems*, (2013), pp. 315–323], SAG Roux et al. [*Advances in neural information processing systems*, (2012), pp. 2663–2671], SAGA Defazio et al. [*Advances in neural information processing systems*, (2014), pp. 1646–1654] and so on. Conjugate gradient method, which has the same computation cost with gradient descent method, is considered. In this paper, in the spirit of SAGA, we propose a stochastic conjugate gradient algorithm which we call SCGA. With the Fletcher and Reeves type choices, we prove a linear convergence rate for smooth and strongly convex functions. We experimentally demonstrate that SCGA converges faster than the popular SGD type algorithms for four machine learning models, which may be convex, nonconvex or nonsmooth. Solving regression problems, SCGA is competitive with CGVR, which is the only one stochastic conjugate gradient algorithm with variance reduction so far, as we know.

Keywords Deep learning · Empirical risk minimization · Stochastic conjugate gradient · Linear convergence

1 Introduction

Nowadays, deep learning has been widely applied in various fields, and it performs well in such scenarios as computer vision [1], speech recognition [2, 3], word processing [4], and malware detection [5]. In supervised learning, we assume that there are n input-output samples $\{(x_i, y_i)\}_{i=1}^n$ and $P(x, y)$ is the true relationship between inputs and outputs. Ideally,

✉ Caixia Kou
koucx@bupt.edu.cn

Han Yang
yanghan1@bupt.edu.cn

[1] School of Science, Beijing University of Posts and Telecommunications, No.10, XiTuCheng Road, Beijing 100876, China

the expected risk is defined as:

$$F(\omega) = \int l(\omega) dP(x, y) = \mathbb{E}[l(\omega)],$$

where $\omega \in \mathbb{R}^d$ and $l(\omega)$ is the loss function that measures the distance between the prediction and the real value y. We aim to minimize $F(\omega)$. While the information about P is not complete, in practice, there is a formula that involves an estimate of the expected risk F. It is defined as the empirical risk function:

$$f(\omega) = \frac{1}{n} \sum_{i=1}^{n} f_i(\omega),$$

where each $f_i(\omega) : \mathbb{R}^d \to \mathbb{R}$ is the loss function corresponding to the i-th sample. Hence, the following optimization problem ERM that minimizes the sum of loss functions over samples from a finite training set appears frequently in deep learning:

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

The full gradient descent algorithm [6] is a classical algorithm to solve (1), and the update rule for $k = 0, 1, 2, \cdots$ can be described as:

$$\omega_{k+1} = \omega_k - \alpha_k \nabla f(\omega_k) = \omega_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_i(\omega_k).$$

Because of the structure of $f(\omega)$, $\nabla f(\omega)$ is the average sum of every loss function gradient $\nabla f_i(\omega)$, which is corresponding to i-th sample. However, the calculation of $\nabla f(\omega)$ is challenging when n is extremely large. A modification of the full gradient descent is the stochastic gradient descent method (SGD) [7–9] with the iteration update:

$$\omega_{k+1} = \omega_k - \alpha_k g_k,$$

where g_k covers the choices

$$g_k = \begin{cases} \nabla f_{i_k}(\omega_k), \, i_k \text{ is randomly selected from } \{1, 2, \cdots, n\}, \\ \dfrac{1}{|S|} \displaystyle\sum_{i \in S} \nabla f_i(\omega_k), \, S \text{ is a mini-batch of } n \text{ samples.} \end{cases}$$

The calculation of g_k, as the estimate of full gradient $\nabla f(\omega_k)$, is much cheaper than $\nabla f(\omega_k)$. Based on the above basic framework, there are two main classes of SGD variants. One is the accelerated methods [10–12]. The other one is adaptive learning rate methods like AdaGrad [13], AdaDelta [14], RMSProp [15] and Adam [16].

Though the expectation of g_k equals to full gradient $\nabla f(\omega_k)$, randomly different choices of g_k may yield the variance, which causes the slow convergence rate of SGD. In fact, the convergence rate of SGD is sublinear under certain conditions, which is slower than full gradient descent methods. Hence, another important class, variance reduction SGD, is proposed to improve the convergence rate. Le Roux et al. [18] proposed the stochastic average gradient (SAG) that gets a reduction of variance for SGD, which leads to a linear convergence rate when each $f_i(\omega)$ is smooth and strongly convex, but the estimation of the gradient is biased. Johnson and Zhang [19] proposed the stochastic variance reduced gradient (SVRG) which also accelerates the convergence rate while it needs to compute

the gradient of all samples after every m SGD iterations. The SAGA method proposed by Defazio, Bach and Lacoste-Julien [20] makes a trade-off between time and space. It needs to store the gradients of all samples in a table and consequently only updates the gradient of one sample at each iteration. Nguyen, Liu, Scheinberg and Taká et al. [21] proposed the stochastic recursive gradient algorithm(SARAH) which is a new variance reducing stochastic gradient algorithm. For strongly convex case, it has the linear convergence rate. Though it doesn't require a storage of the past gradients, the estimation of gradient is not unbiased.

As is known to all, the conjugate gradient method (CG) [22, 23], as another important method in classical optimization, often performs better than full gradient descent methods. Moreover, the calculation of CG methods is similar to full gradient descent methods. The four classical nonlinear CG methods are FR [24], PRP [25, 26], HS [27] and DY [28]. In recent years, other efficient CG algorithms [29, 30] are proposed. More details about the CG can be found in [17, 31].

It is natural to adapt the CG method in deep learning because of its advantages. Adaptations of conjugate gradients specifically for neural networks have been proposed earlier, such as the scaled conjugate gradient algorithm [32]. The CG method with mini-batch version has been used successfully for training of neural networks [33]. Recently, a kind of stochastic conjugate gradient algorithm with variance reduction (CGVR) [35] is proposed. The main feature of CGVR is that it calculates a stochastic gradient g together with FR conjugate parameter to compose the search direction in each iteration. But after every m iterations, it requires to calculate the full gradient to correct the stochastic gradient. To get an efficient performance, it needs a huge computational consumption. Inspired by SAGA, in this paper we aim to propose a new variance reduction stochastic conjugate gradient algorithm named as SCGA. It is expected that it has satisfactory numerical performance and less computational cost.

The remainder of this paper is organized as follows. In Sect. 2, we briefly review the variance reduction stochastic gradient descent algorithm, named SAGA. In Sect. 3, a new stochastic conjugate gradient algorithm, called SCGA, is introduced in detail and its linear convergence rate is proved. In Sect. 4, a series of experiments are conducted to compare SCGA with other algorithms. Then, it is summarized in Sect. 5.

2 Brief review of the SAGA algorithm

The SAGA algorithm [20] is a stochastic gradient descent method with variance reduction like SVRG. But compared with SVRG, SAGA doesn't need to compute the full gradients after every m SGD iterations. It only needs to restore the full gradient. To some sense, it makes to a trade-off between time and space. At each iteration, SAGA computes only the gradient of one randomly chosen sample j and then updates j-th entry of the restored full gradient while all other entries remains unchanged. Then SAGA uses the following stochastic vector g_k to approximate full gradients.

$$g_k = \nabla f_j(\omega_k) - \nabla f_j(\omega_{[j]}) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\omega_{[i]}),$$

where $\omega_{[j]}$ represents the latest iterate at which ∇f_j was evaluated. And $\nabla f_j(\omega_{[j]})$ is the gradient of the j-th sample at iterate $\omega_{[j]}$.

From taking the expectation of g_k above, with respect to all choices of random index $j \in \{1, 2, \cdots, n\}$, it follows that the expectation of g_k is exactly $\nabla f(\omega_k)$, which means this

approximation g_k is an unbiased estimate of full gradients. Also, such unbiased estimate of gradient in SAGA [20] is proved obtaining a reduced variance. Benefiting from the variance reduction, SAGA obtains a linear rate of convergence for strongly convex functions. While its computation cost of each iteration is the same as the basic SGD algorithm.

3 A new stochastic conjugate gradient algorithm

As a stochastic conjugate gradient algorithm, although CGVR accelerates the convergence rate of SGD by reducing the variance of the gradient estimates. It requires to calculate the full gradient to correct the stochastic gradient after every m SGD iterations, which leads to high computation cost. Inspired by SAGA, we propose a new stochastic conjugate gradient algorithm called SCGA to overcome the above-mentioned disadvantages.

3.1 The framework of the new algorithm

We adapt the CG algorithm from SAGA to obtain the SCGA in Algorithm 1. At the initialization step, we compute full gradient at initial iterate and store it into a matrix, named $\nabla f(\omega_{[0]}) = [\nabla f_1(\omega_0), \nabla f_2(\omega_0), \cdots, \nabla f_n(\omega_0)]$. Consequently, at each iteration, we randomly choose a subset $S \in \{1, 2, \cdots, n\}$, called a mini-batch of samples, and define the subsampled function $f_S(\omega)$ as

$$f_S(\omega) = \frac{1}{|S|} \sum_{i \in S} f_i(\omega),$$

where $|S|$ denotes the number of elements in the mini-batch S. After S_k is randomly chosen, at the current iterate ω_k, we don't need to compute the full gradient, but every gradient of the samples in S_k, i.e. $\nabla f_j(\omega_j), \forall j \in S_k$, then get the average of them, named as $\nabla f_{S_k}(\omega_k)$. Also, at the last stored iterate, we compute the average gradient on S_k

$$\mu_{S_k} = \frac{1}{|S_k|} \sum_{j \in S_k} \nabla f_j(\omega_{[k-1]}), \tag{2}$$

and the full gradient

$$\mu_{k-1} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\omega_{[k-1]}). \tag{3}$$

Then using the two gradients μ_{S_k} and μ_{k-1} at the last restored iterate, we correct ∇f_{S_k} at the current iterate to obtain the new stochastic gradient

$$g_k = \nabla f_{S_k}(\omega_k) - \mu_{S_k} + \mu_{k-1}. \tag{4}$$

It is tempting to conclude $\mathbb{E}[g_k] = \nabla f(\omega_k)$, which means that (4) is an unbiased estimate of gradient $\nabla f(\omega_k)$.

In addition, such estimate of gradient can be proved have a reducing variance. In fact, considering the variance of gradient

$$V = \mathbb{E}\left[\|g_k - \nabla f(w_k)\|^2\right] = \mathbb{E}[\|g_k\|^2] - \|\nabla f(\omega_k)\|^2, \tag{5}$$

from Lemma 4, we see that

$$\mathbb{E}[\|g_k\|^2] \leq 4\Lambda[f(\omega_k) - f(\omega_*) + f(\omega_{[k]}) - f(\omega_*)].$$

Intuitively, as $\omega_k \to w_*$ and $\omega_{[k]} \to \omega_*$ the variance goes to zero asymptotically.

After obtaining the gradients of samples $\nabla f_j(\omega_j), \forall j \in S_k$, we update the corresponding entries of the stored matrix $\nabla f(\omega_{[k]}) = \left[\nabla f_1(\omega_{[k]}), \nabla f_2(\omega_{[k]}), \cdots, \nabla f_n(\omega_{[k]})\right]$, while other entries remain unchanged. That is $\nabla f_j(\omega_{[k]}) \leftarrow \nabla f_j(\omega_k), \forall j \in S_k$. SCGA has the similar way of determining stochastic gradients with SAGA. But compared with CGVR, it does not need to compute the full gradient in each iteration instead to compute a mini-batch gradients each time. Based on this, it is reasonable to expect that SCGA consumes less computation cost than CGVR.

Algorithm 1 SCGA

Initialization: Given $\omega_0 \in \mathbb{R}^d$, compute the full gradient at initial iterate ω_0 and store it:
for $i = 1, 2, \cdots, n$ **do**
  Compute $\nabla f_i(\omega_0)$
  Store $\nabla f_i(\omega_{[0]}) \leftarrow \nabla f_i(\omega_0)$
end for
$\mu_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\omega_{[0]})$
Set the initial stochastic gradient $g_0 = \mu_0$.
Set the initial direction $p_0 = -g_0$.

Iteration:
for $k = 1, 2, \cdots$ **do**
  Find the step size α_{k-1} satisfying (9) and (10).
  Update iterate $\omega_k = \omega_{k-1} + \alpha_{k-1} p_{k-1}$.
  Randomly sample a mini-batch S_k.
  $\forall j \in S_k$, compute $\nabla f_j(\omega_k)$ to obtain $\nabla f_{S_k}(\omega_k)$.
  Compute μ_{S_k} using (2) on S_k.
  Set $g_k = \nabla f_{S_k}(\omega_k) - \mu_{S_k} + \mu_{k-1}$
  Compute β_k by
  Option I:

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}$$

Option II:

$$\beta_k^{FR+PR} = \begin{cases} -\beta_k^{FR} & if \ \ \beta_k^{PR} < -\beta_k^{FR} \\ \beta_k^{PR} & if \ \ |\beta_k^{PR}| \le \beta_k^{FR} \\ \beta_k^{FR} & if \ \ \beta_k^{PR} > \beta_k^{FR}. \end{cases}$$

  Determine $p_k = -g_k + \beta_k p_{k-1}$.
  Update $\nabla f_j(\omega_{[k]}) \leftarrow \nabla f_j(\omega_k), \forall j \in S_k$, while other entries of the stored full gradient remain unchanged.
  Update $\mu_k = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\omega_{[k]})$.
end for

To get the stochastic conjugate gradient direction, the conjugate parameter β can be chosen as FR

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}. \tag{6}$$

Though the convergence of FR method has been well established, their numerical results are not good. And PR method, which defines the parameter β_k as follows:

$$\beta_k^{PR} = \frac{g_k^T(g_k - g_{k-1})}{g_{k-1}^T g_{k-1}}, \tag{7}$$

is generally believed to be one of the most efficient CG method in practical computation because it essentially restarts if a bad direction occurs. But in theory it may not converge. To combine the advantages of these CG methods, our stochastic conjugate gradient algorithm SCGA chooses a hybrid version between FR and PR:

$$\beta_k^{FR+PR} = \begin{cases} -\beta_k^{FR} & if \ \ \beta_k^{PR} < -\beta_k^{FR}, \\ \beta_k^{PR} & if \ \ |\beta_k^{PR}| \le \beta_k^{FR}, \\ \beta_k^{FR} & if \ \ \beta_k^{PR} > \beta_k^{FR}. \end{cases} \tag{8}$$

Note that $|\beta_k^{FR+PR}| \le \beta_k^{FR}$. Moreover, for any β_k satisfying $|\beta_k| \le \beta_k^{FR}$ we prove the convergence of SCGA in the following subsection.

In order to get an appropriate step size α_k, we introduce the inexact line search and require α_k satisfy the stochastic version of the strong Wolfe conditions

$$f_{S_k}(\omega_k + \alpha_k p_k) \le f_{S_k}(\omega_k) + c_1 \alpha_k \nabla f_{S_k}(\omega_k)^T p_k, \tag{9}$$

$$|g_{k+1}^T p_k| \le -c_2 g_k^T p_k. \tag{10}$$

where $0 < c_1 < c_2 < 1$. In addition, The SCGA algorithm is implemented with step size α_k that satisfies condition (10) with $0 < c_2 < 1/2$. It can be shown that SCGA with FR generates the descent direction p_k satisfying

$$-\frac{1}{1-c_2} \le \frac{g_k^T p_k}{\|g_k\|^2} \le \frac{2c_2 - 1}{1 - c_2}.$$

Besides, the above bounds inequality also holds for any β_k satisfying $|\beta_k| \le \beta_k^{FR}$. The similar proof details can be referred to Lemma 3.1 of [22].

3.2 The convergence analysis of SCGA

We analyze the convergence of SCGA in Alogrithm 1 with any β_k satisfying $|\beta_k| \le \beta_k^{FR}$. This convergence result leads to SCGA with the hybrid of FR and PR preserves its efficiency and assures its convergence.

The convergence analysis uses the same assumptions in CGVR as follows:

Assumption 1 The SCGA algorithm is implemented with a step size α_k that satisfies $\alpha_k \in [\alpha_1, \alpha_2]$, $0 < \alpha_1 < \alpha_2$ and condition (10) with $c_2 \le 1/5$.

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

$$\lambda I \preceq \nabla^2 f_i(\omega) \preceq \Lambda I$$

for all $\omega \in \mathbb{R}^d$.

The Assumption 2 indicates that f is also strongly convex and ∇f is Lipschitz continuous.

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

$$\beta_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2} \leq \hat{\beta}.$$

Using these assumptions, the following lemmas are directly derived. Lemma 1 and Lemma 2 estimate the lower bound of $\|\nabla f(\omega)\|$ and the upper bound of $\mathbb{E}[\|p_k\|^2]$. They are the same as Lemma 5 [34] and Theorem 2 of [35].

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

$$\|\nabla f(\omega)\|^2 \geq 2\lambda(f(\omega) - f(\omega_*)).$$

Lemma 2 *Suppose that Assumptions 1 and 3 hold for Algorithm 1. Then, for any k, we have*

$$\mathbb{E}[\|p_k\|^2] \leq \eta(k)\mathbb{E}[\|g_0\|^2], \tag{11}$$

where

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

Lemma 3 *According to Algorithm 1, for any k, we have*

$$\mathbb{E}[\alpha_k g_k^T p_k] \leq -\alpha_1 \|\nabla f(\omega_k)\|^2 + \frac{1}{4}\alpha_2\hat{\beta}^k\mathbb{E}[\|g_0\|^2].$$

Proof By the definition of $p_k = -g_k + \beta_k p_{k-1}$, we obtain

$$\begin{aligned}
\mathbb{E}[\alpha_k g_k^T p_k] &= \mathbb{E}[-\alpha_k \|g_k\|^2] + \mathbb{E}[\alpha_k\beta_k g_k^T p_{k-1}] \\
&\leq \mathbb{E}[-\alpha_k \|g_k\|^2] + \mathbb{E}[-\alpha_k\beta_k c_2 g_{k-1}^T p_{k-1}] \\
&\leq \mathbb{E}[-\alpha_k \|g_k\|^2] + \frac{c_2}{1-c_2}\mathbb{E}[\alpha_k\beta_k \|g_{k-1}\|^2] \\
&\leq \mathbb{E}[-\alpha_1 \|g_k\|^2] + \frac{c_2}{1-c_2}\mathbb{E}[\alpha_2\hat{\beta} \|g_{k-1}\|^2] \\
&\leq -\alpha_1\mathbb{E}[\|g_k\|^2] + \frac{1}{4}\alpha_2\hat{\beta}\mathbb{E}[\|g_{k-1}\|^2] \\
&\leq -\alpha_1 \|\mathbb{E}[g_k]\|^2 + \frac{1}{4}\alpha_2\hat{\beta}\mathbb{E}[\|g_{k-1}\|^2] \\
&= -\alpha_1 \|\nabla f(\omega_k)\|^2 + \frac{1}{4}\alpha_2\hat{\beta}\mathbb{E}[\|g_{k-1}\|^2].
\end{aligned}$$

The first inequality uses the strong Wolfe condition (10), the second one uses the lower bound $-\frac{1}{1-c_2}$ of $\frac{g_k^T p_k}{\|g_k\|^2}$. Then by using Assumption 1 ($\alpha_k \in [\alpha_1, \alpha_2]$) and Assumption 3 ($\beta_k \leq \hat{\beta}$), it yields the third inequality. Note that the monotonically increasing property of the function $\frac{x}{1-x}$ with $x \neq 1$ which implies $\frac{c_2}{1-c_2} \leq \frac{1}{4}$ with $c_2 \leq \frac{1}{5}$, so the fourth inequality holds. It is easy to know $\mathbb{E}[\|g_k\|^2] \geq \|\mathbb{E}[g_k]\|^2$, which deduces the last inequality.

Furthermore, according to Assumption 3 and taking expectation, it holds that $\mathbb{E}[\|g_k\|^2] \leq \hat{\beta}\mathbb{E}[\|g_{k-1}\|^2]$, and consequently, $\mathbb{E}[\|g_k\|^2] \leq \hat{\beta}^k\mathbb{E}[\|g_0\|^2]$, which implies the conclusion. □

In the following Lemma 4, we estimates the upper bound of $\mathbb{E}[\|g_k\|^2]$.

Lemma 4 *Let ω_* be the unique minimizer of f. Taking expectation with respect to S_k of $\|g_k\|^2$ in (4), we obtain*

$$\mathbb{E}[\|g_k\|^2] \leq 4\Lambda[f(\omega_k) - f(\omega_*) + f(\omega_{[k]}) - f(\omega_*)]. \tag{12}$$

Proof Given any mini-batch S , considering the function

$$h_S(\omega) = f_S(\omega) - f_S(\omega_*) - \nabla f_S(\omega_*)^T(\omega - \omega_*),$$

we know that $h_S(\omega_*) = \min_{\omega} h_S(\omega)$ since $\nabla h_S(\omega_*) = 0$. Therefore,

$$\begin{aligned} 0 = h_S(\omega_*) &\leq \min_{\eta}[h_S(\omega - \eta\nabla h_S(\omega))] \\ &\leq \min_{\eta}\left[h_S(\omega) - \eta\|\nabla h_S(\omega)\|^2 + \frac{1}{2}\Lambda\eta^2\|\nabla h_S(\omega)\|^2\right] \\ &= h_S(\omega) - \frac{1}{2\Lambda}\|\nabla h_S(\omega)\|^2. \end{aligned}$$

That is,

$$\|\nabla f_S(\omega) - \nabla f_S(\omega_*)\|^2 \leq 2\Lambda\left[f_S(\omega) - f_S(\omega_*) - \nabla f_S(\omega_*)^T(\omega - \omega_*)\right].$$

By taking expectation with respect to S on the above inequality, we obtain

$$\mathbb{E}\left[\|\nabla f_S(\omega) - \nabla f_S(\omega_*)\|^2\right] \leq 2\Lambda[f(\omega) - f(\omega_*)]. \tag{13}$$

In case of that $k \neq 0$, taking expectation with respect to S_k on norm of g_k in (4), we obtain

$$\begin{aligned} &\mathbb{E}\left[\|g_k\|^2\right] \\ &= \mathbb{E}\left[\|\nabla f_S(\omega_k) - \mu_S + \mu\|^2\right] \\ &= \mathbb{E}\left[\|\nabla f_S(\omega_k) - \nabla f_S(\omega_*) + \nabla f_S(\omega_*) - \mu_S + \mu\|^2\right] \\ &\leq 2\mathbb{E}\left[\|\nabla f_S(\omega_k) - \nabla f_S(\omega_*)\|^2\right] + 2\mathbb{E}\left[\|\mu_S - \nabla f_S(\omega_*) - \mu\|^2\right] \\ &= 2\mathbb{E}\left[\|\nabla f_S(\omega_k) - \nabla f_S(\omega_*)\|^2\right] + 2\mathbb{E}\left[\|\mu_S - \nabla f_S(\omega_*) - \mathbb{E}\left[\mu_S - \nabla f_S(\omega_*)\right]\|^2\right] \\ &\leq 2\mathbb{E}\left[\|\nabla f_S(\omega_k) - \nabla f_S(\omega_*)\|^2\right] + 2\mathbb{E}\left[\|\mu_S - \nabla f_S(\omega_*)\|^2\right] \\ &\leq 4\Lambda\left[f(\omega_k) - f(\omega_*) + f(\tilde{\omega}_k) - f(\omega_*)\right], \end{aligned} \tag{14}$$

where Λ is the positive constant in Assumption 2. The first inequality uses $\|a - b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. The second inequality uses $\mathbb{E}\|\xi - \mathbb{E}\xi\|^2 = \mathbb{E}\|\xi\|^2 - \|\mathbb{E}\xi\|^2 \leq \mathbb{E}\|\xi\|^2$ for any random vector ξ. The third inequality uses (13).

In case of that $k = 0$, then $g_0 = \mu_0 = \nabla f(\omega_0)$.

$$\mathbb{E}\left[\|g_0\|^2\right] = \|\nabla f(\omega_0)\|^2 = \|\nabla f(\omega_0) - \nabla f(\omega_*)\|^2 \leq 2\Lambda[f(\omega_0) - f(\omega_*)]. \tag{15}$$

The above inequality uses Assumption 2. Note that $\omega_{[0]} = \omega_0$. Thus, for $k = 0$ also satisfies (12), which together with (14) comes to the conclusion. □

Finally, we present the convergence rate of SCGA. It can achieve a linear convergence rate for strongly convex function.

Theorem 1 *Suppose that Assumptions 1, 2 and 3 hold. Let ω_* be the unique minimizer of f. Then, for all $k \geq 0$, we have*

$$\mathbb{E}[f(\omega_{k+1}) - f(\omega_*)] \leq C\xi^{k+1}\mathbb{E}[f(\omega_0) - f(\omega_*)],$$

where parameters ξ and C are given by

$$\xi = 1 - 2\alpha_1\lambda < 1,$$
$$C = 1 + \frac{\alpha_2\Lambda(1-\hat{\beta}) + 4\alpha_2^2\Lambda^2}{2(\xi-\hat{\beta})(1-\hat{\beta})},$$

assuming that we choose $\alpha_1 < \dfrac{1-\hat{\beta}}{2\lambda}$.

Proof By Assumption 2, the function $f_{S_k}(\omega_k)$ satisfies

$$\lambda I \preceq \nabla^2 f_{S_k}(\omega_k) \preceq \Lambda I.$$

That is for all $k \in \mathbb{N}$, the following inequality holds

$$f_{S_k}(\omega_{k+1}) \leq f_{S_k}(\omega_k) + \nabla f_{S_k}(\omega_k)^T(\omega_{k+1}-\omega_k) + \frac{\Lambda}{2}\|\omega_{k+1}-\omega_k\|^2.$$

With $w_{k+1} = w_k + \alpha_k p_k$, the above inequality also can be written as

$$f_{S_k}(\omega_{k+1}) - f_{S_k}(\omega_k) \leq \alpha_k g_k^T p_k + \frac{1}{2}\alpha_k^2\Lambda\|p_k\|^2.$$

Taking expectation in this relation conditioned on S_k, this yields

$$\begin{aligned}
&\mathbb{E}[f(\omega_{k+1})] - f(\omega_k)\\
&\leq \mathbb{E}[\alpha_k g_k^T p_k] + \frac{1}{2}\Lambda\mathbb{E}[\alpha_k^2\|p_k\|^2]\\
&\leq -\alpha_1\|\nabla f(\omega_k)\|^2 + \frac{1}{4}\alpha_2\hat{\beta}^k\mathbb{E}[\|g_0\|^2] + \frac{1}{2}\Lambda\mathbb{E}[\alpha_2^2\|p_k\|^2]\\
&\leq -\alpha_1\|\nabla f(\omega_k)\|^2 + (\frac{1}{4}\alpha_2\hat{\beta}^k + \frac{1}{2}\alpha_2^2\Lambda\eta(k))\mathbb{E}[\|g_0\|^2]\\
&\leq -2\alpha_1\lambda[f(\omega_k) - f(\omega_*)] + (\frac{1}{2}\alpha_2\Lambda\hat{\beta}^k + \alpha_2^2\Lambda^2\eta(k))[f(\omega_0) - f(\omega_*)].
\end{aligned}$$

The second inequality uses Assumptions 1 and Lemma 3, the third inequality uses Lemma 2, and the last one uses Lemma 1 and (15).

Subtracting $f(\omega_*)$ from both sides of the above inequality, taking total expectations, and rearranging, this yields

$$\begin{aligned}
\mathbb{E}[f(\omega_{k+1}) - f(\omega_*)] \leq{}& (1-2\alpha_1\lambda)\mathbb{E}[f(\omega_k) - f(\omega_*)]\\
&+ \left(\frac{1}{2}\alpha_2\Lambda\hat{\beta}^k + \alpha_2^2\Lambda^2\eta(k)\right)\mathbb{E}[f(\omega_0) - f(\omega_*)].
\end{aligned}$$

For the convenience of discussion, we define

$$\xi = 1 - 2\alpha_1\lambda, \quad \varphi(k) = \frac{1}{2}\alpha_2\Lambda\hat{\beta}^k + \alpha_2^2\Lambda^2\eta(k), \quad \Delta_k = \mathbb{E}[f(\omega_k) - f(\omega_*)].$$

Then the above inequality can be written as

$$\Delta_{k+1} \leq \xi\Delta_k + \varphi(k)\Delta_0.$$

We further obtain

$$\begin{aligned}\Delta_{k+1} &\leq \xi\left(\xi\Delta_{k-1} + \varphi(k-1)\Delta_0\right) + \varphi(k)\Delta_0 \\ &= \xi^2\Delta_{k-1} + [\xi\varphi(k-1) + \varphi(k)]\Delta_0 \\ &\vdots \\ &\leq [\xi^{k+1} + \sum_{i=0}^{k}\xi^{k-i}\varphi(i)]\Delta_0.\end{aligned}$$

We now compute $\sum_{i=0}^{k}\xi^{k-i}\varphi(i)$,

$$\begin{aligned}\sum_{i=0}^{k}\xi^{k-i}\varphi(i) &= \xi^k\frac{1}{2}\alpha_2\Lambda\sum_{i=0}^{k}\left(\frac{\hat{\beta}}{\xi}\right)^i + \xi^k\frac{2\alpha_2^2\Lambda^2}{1-\hat{\beta}}\sum_{i=0}^{k}\left(\frac{\hat{\beta}}{\xi}\right)^i - \xi^k\frac{\alpha_2^2\Lambda^2(1+\hat{\beta})}{1-\hat{\beta}}\sum_{i=0}^{k}\left(\frac{\hat{\beta}^2}{\xi}\right)^i \\ &= \xi^k\frac{1}{2}\alpha_2\Lambda\frac{1-\left(\frac{\hat{\beta}}{\xi}\right)^{k+1}}{1-\frac{\hat{\beta}}{\xi}} + \frac{\xi^k 2\alpha_2^2\Lambda^2}{1-\hat{\beta}}\frac{1-\left(\frac{\hat{\beta}}{\xi}\right)^{k+1}}{1-\frac{\hat{\beta}}{\xi}} \\ &\quad - \xi^k\frac{\alpha_2^2\Lambda^2(1+\hat{\beta})}{1-\hat{\beta}}\frac{1-\left(\frac{\hat{\beta}^2}{\xi}\right)^{k+1}}{1-\frac{\hat{\beta}^2}{\xi}} \\ &\leq \xi^k\frac{1-\left(\frac{\hat{\beta}}{\xi}\right)^{k+1}}{1-\frac{\hat{\beta}}{\xi}}\left(\frac{1}{2}\alpha_2\Lambda + \frac{2\alpha_2^2\Lambda^2}{1-\hat{\beta}}\right) \\ &\leq \xi^k\frac{1}{1-\frac{\hat{\beta}}{\xi}}\left(\frac{1}{2}\alpha_2\Lambda + \frac{2\alpha_2^2\Lambda^2}{1-\hat{\beta}}\right) \\ &= \xi^{k+1}\frac{\alpha_2\Lambda(1-\hat{\beta}) + 4\alpha_2^2\Lambda^2}{2(\xi-\hat{\beta})(1-\hat{\beta})}\end{aligned}$$

The first and second inequality uses $\alpha_1 < \frac{1-\hat{\beta}}{2\lambda}$ which implies that $\xi > \hat{\beta} > \hat{\beta}^2$. Then, we obtain

$$\mathbb{E}[f(\omega_{k+1}) - f(\omega_*)] \leq C\xi^{k+1}\mathbb{E}[f(\omega_0) - f(\omega_*)].$$

□

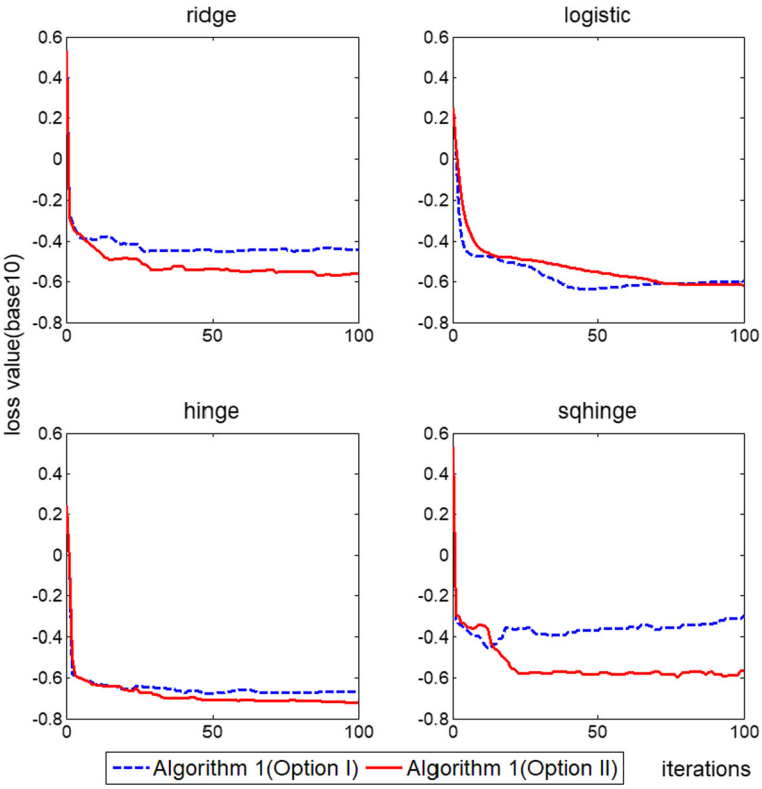

Fig. 1 Performance comparison of Algorithm 1 with Option I and Option II on *ijcnn1* data set using four learning models shown in (16) (17) (18) and (19). (The x-axis represents the number of iterations. The y-axis represents base-10 logarithm of loss values.)

4 Experiments

In this section we perform a series of experiments to validate the effectiveness of SCGA presented in Algorithm 1.

First, we conduct Algorithm 1 with different β_k options: Option I and Option II. The comparison results on data set *ijcnn1* are shown in Fig. 1, where the four sub-figures are corresponding to the four different models introduced in Sect. 4.1. We can see that Algorithm 1 with Option II performs much faster than that with Option I. This is consistent with our theoretical analysis before. Also, similar performance can be found on other data sets of Table 3.

Because of the better performance of Option II, we choose Option II for Algorithm 1, named as SCGA. Table 1 lists all the compared algorithms in the following experiments.

For a fair comparison, we use the same code base for each algorithm, just changing the main update rule. Each algorithm has its step size parameter chosen so as to give the fastest convergence. The parameters of compared algorithms and loss functions are listed in Table 2.

Table 1 Algorithms in the comparison

Algorithm	Description
SCGA	Proposed Algorithm 1 with Option II
SGD	Mini-batch SGD with momentum acceleration [8]
SAGA	Variance reduction SGD [20]
Mini-SAGA	By modifying SAGA [20] to a mini-batch version
CGVR	Variance reduction SCG [35]

Table 2 Parameters in algorithms

Parameter	Description	Value
η	The step size in SGD, SAGA and mini-SAGA	10^{-2}
$\|S\|$	The mini-batch size in SGD, mini-SAGA, CGVR and SCGA	$\sqrt{n}$
c_1	The parameter of line search in CGVR and SCGA	10^{-4}
c_2	The parameter of line search in CGVR and SCGA	0.1
λ	The regularization parameter in loss functions	10^{-4}
m	The number of inner loop iterations in CGVR	50

4.1 Machine learning models and data sets

We evaluate algorithms on the following popular machine learning models including regression and classification problems.

(1) ridge regression (ridge)

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \omega)^2 + \lambda \|\omega\|_2^2 \tag{16}$$

(2) logistic regression (logistic)

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(-y_i x_i^T \omega)) + \lambda \|\omega\|_2^2 \tag{17}$$

(3) L2-regularized L1-loss SVM (hinge)

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i x_i^T \omega)_+ + \lambda \|\omega\|_2^2 \tag{18}$$

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

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} ((1 - y_i x_i^T \omega)_+)^2 + \lambda \|\omega\|_2^2 \tag{19}$$

where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ is the i-th sample data. And the data matrix X for all dimensions is scaled into the range of $[-1, +1]$ by the max-min scaled in the preprocessing stage. These four models are convex, nonconvex or nonsmooth.

Table 3 Data sets used in comparison

Dataset	n	d
a9a	32561	123
cod-rna	59535	8
ijcnn1	49990	22
quantum	50000	78
protein	145751	74
bodyfat	252	24
housing	506	13
pyrim	74	27
space_ga	3170	6
triazines	180	60
Average Localization Error (ALE) in sensor node localization process in WSNs Data Set	107	6

The data sets are presented in Table 3. The first five are the binary classification of large-scale data sets, where the first three,*a9a*, *cod-rna* and *ijcnn1*, are from the LIBSVM data website[1], the next two, *quantum* and *protein*, are from the KDD Cup 2004 website[2]. The remaining six data sets are regression problems. The details of data sets $bodyfat$, $housing$, $pyrim$, $space\_ga$ and $triazines$ can be also found in the LIBSVM data website. The last one, *Average Localization Error (ALE) in sensor node localization process in WSNs* data set, can be found in UCI Machine Learning Repository website[3].

4.2 Numerical comparison results

In the first experiment, we compare the convergence of SCGA with several SGD type algorithms on two kinds of data sets. Fig. 2 shows the convergence of these algorithms on five binary-classification large-scale data sets. Fig. 3 presents the results on six regression data sets. From Fig. 2 we see that SCGA has the fastest convergence on almost all of the four models, even when the loss value reaches a notably small value. In Fig. 3, SCGA makes loss drops fast at first and goes down fast to the minimum. In general, SCGA reduces the variance and smoothly converges faster than SGD, SAGA and its mini-batch version.

In the second experiment, we compare the two stochastic conjugate gradient algorithms, SCGA and CGVR [35]. Because CGVR requires computations of full gradient, while SCGA does not. Thus we measure the computational cost by the number of gradient computations divided by n instead of iterations.

We evaluate these two algorithms on the data sets in Table 3, including classification and regression. For classification data sets, *a9a*, *cod-rna*, *ijcnn1*, *quantum* and *protein*, we conduct the algorithms on four machine learning models shown in (16) (17) (18) and (19). From the results, SCGA can be seen to perform similar to CGVR, only with a slight advantage. For these data sets, we do not present the comparison plots. On the other hand, for regression

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

[2] http://osmot.cs.cornell.edu/kddcup

[3] http://archive.ics.uci.edu/ml/datasets/Average+Localization+Error+%28ALE%29+in+sensor+node+localization+process+in+WSNs

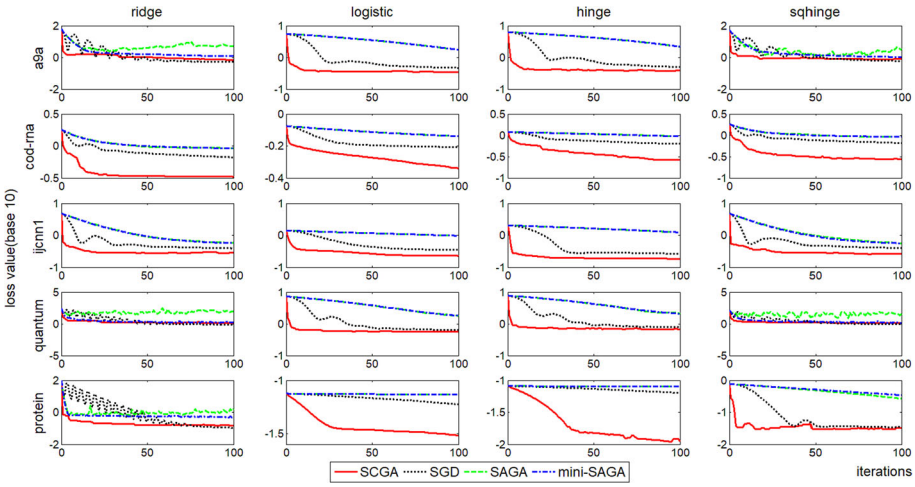

Fig. 2 Performance of SCGA compared with SGD type algorithms. On binary classification large-scale data sets *a9a*, *cod-rna*, *ijcnn1*, *quantum* and *protein* for four machine learning models shown in (16) (17) (18) and (19).(The x-axis represents numbers of iterations. The y-axis represents base-10 logarithm of loss values.)

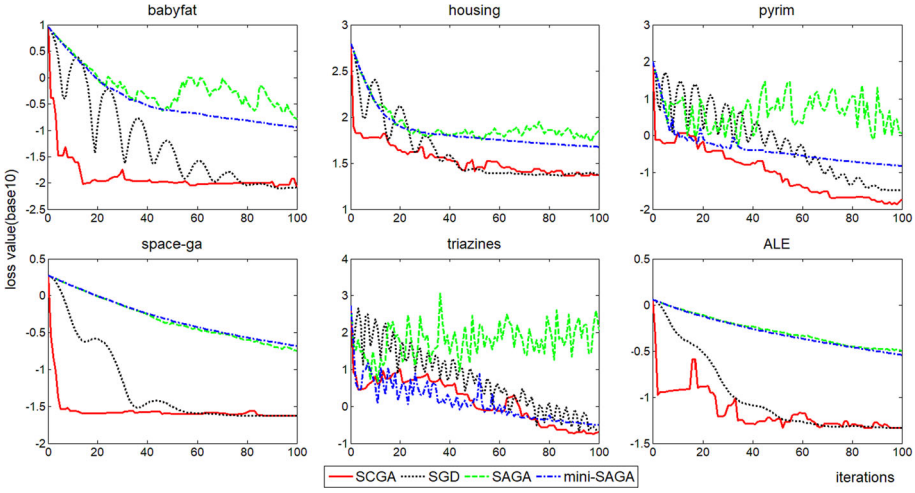

Fig. 3 Performance of SCGA compared with SGD type algorithms. On ridge regression model for data sets *bodyfat*, *housing*, *pyrim*,*space_ga*, *triazines* and *ALE in sensor node localization process in WSNs*. (The x-axis represents numbers of iterations. The y-axis represents base-10 logarithm of loss values.)

data sets, i.e., the last six data sets in Table 3, we use ridge regression model to evaluate their performance. Fig. 4 plots the logarithm of loss errors with respect to their computational cost. Both of SCGA and CGVR can rapidly goes down initially as expected, but SCGA converges to a better level in *pyrim* and *triazines* data sets.

Overall, SCGA is competitive with CGVR and clearly more advantageous than the SGD type algorithms.

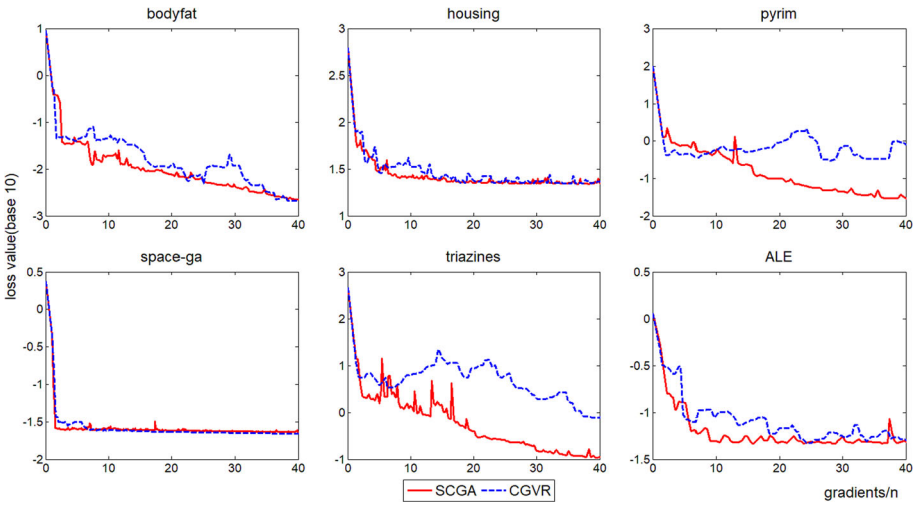

Fig. 4 Performance comparisons of SCGA and CGVR. On ridge regression model for data sets *bodyfat*, *housing*, *pyrim*,*space_ga*, *triazines* and *ALE in sensor node localization process in WSNs* data sets. (The x-axis represents computational cost measured by the number of gradient computations divided by n.The y-axis represents base-10 logarithm of loss values.)

5 Conclusion

In this paper, we propose a new stochastic conjugate gradient algorithm with variance reduction, named SCGA. At each iteration, SCGA only computes the gradients of mini-batch samples then updates them into the stored full gradient, instead of computing full gradients in CGVR. We prove that SCGA with a class of FR choices obtain a convergence rate for strongly convex function. Moreover, among the class of FR choices, we introduce a choice to SCGA, which is a hybrid of FR and PR, shown in Option II of Algorithm 1. From a series of experiments, it demonstrates that SCGA converges faster than SGD type algorithms. And compared with CGVR, SCGA is a competitive algorithm, especially for some regression problems.

Acknowledgements We would like to thank the anonymous referees for their helpful comments. We also would like to thank professor Dai, Y. H. for the valuable suggestions. This work was supported by the Chinese NSF grants (Nos. 11971073, 12171052 and 11871115).

References

1. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks, In: *Advances in neural information processing systems*, pp. 1097–1105. (2012)
2. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on audio, speech, and language processing **20**(1), 30–42 (2011)
3. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine **29**(6), 82–97 (2012)
4. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. (2008)

5. Dahl, G. E., Stokes, J. W., Deng, L., Yu, D.: Large-scale malware classification using random projections and neural networks, In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 3422–3426. (2013)
6. Cauchy, A.: Méthode générale pour la résolution des systemes d'équations simultanées. Comp. Rend. Sci. Paris. **25**(1847), 536–538 (1847)
7. Robbins, H., Monro, S.: A stochastic approximation method, *The annals of mathematical statistics*, pp. 400–407, (1951)
8. Bottou, L.: Large-scale machine learning with stochastic gradient descent, *Proc. COMPSTAT*, pp. 177–186, (2010)
9. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning. MIT press Cambridge, **1**, (2016)
10. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics **4**(5), 1–17 (1964)
11. Nesterov, Y.: A method of solving a convex programming problem with convergence rate o (1/k2), In *Soviet Mathematics Doklady*, (1983)
12. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Netw **12**(1), 145–151 (1999)
13. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Machine Learning Research, **12**(7), (2011)
14. Zeiler, M. D.: Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701, (2012)
15. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,. COURSERA: Neural networks for machine learning **4**(2), 26–31 (2012)
16. Kingma, D., Ba, J.: Adam: A method for stochastic optimization, *Computer ence*, (2014)
17. Hager, W.W., Zhang, H.: A survey of nonlinear conjugate gradient methods. Pacific Journal of Optimization, **2**(1), 35–58 (2006)
18. Roux, N. L., Schmidt, M., Bach, F. R.: A stochastic gradient method with an exponential convergence rate for finite training sets, in *Advances in neural information processing systems*, pp. 2663–2671, (2012)
19. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction, In *Advances in neural information processing systems*, pp. 315–323. (2013)
20. Defazio, A., Bach, F., Lacoste-Julien, S.: Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, in *Advances in neural information processing systems*, pp. 1646–1654. (2014)
21. Nguyen, L. M., Liu, J., Scheinberg, K., Taká, M.: Sarah: A novel method for machine learning problems using stochastic recursive gradient, (2017)
22. Gilbert, J.C., Nocedal, J.: Global convergence properties of conjugate gradient methods for optimization. SIAM J. Optim. **2**, 21–42 (1992)
23. Nocedal, J., Wright, S.: *Numerical optimization.* Springer Science & Business Media, (2006)
24. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. The computer journal **7**(2), 149–154 (1964)
25. Polak, E., Ribiere, G.: Note sur la convergence de méthodes de directions conjuguées," *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, **3**(R1), pp. 35–43, (1969)
26. Polyak, B.T.: The conjugate gradient method in extreme problem. USSR Comp. Math. Math. Phys. **9**(4), 94–112 (1969)
27. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards* **49**(6), 409 (1952)
28. Dai, Y.H., Yuan, Y.: A nonlinear conjugate gradient method with a strong global convergence property. Siam Journal on Optimization **10**(1), 177–182 (1999)
29. Hager, W.W., Zhang, H.: A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM Journal on Optimization **16**(1), 170–192 (2005)
30. Dai, Y.H., Kou, C.X.: A nonlinear conjugate gradient algorithm with an optimal property and an improved wolfe line search. Siam J Optim **23**(1), 296–320 (2013)
31. Dai, Y.H., Yuan, Y.: *Nonlinear conjugate gradient methods*. Shanghai Science and Technology Publisher, (2000)
32. Møller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural networks **6**(4), 525–533 (1993)
33. Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A. Y.: On optimization methods for deep learning, In *ICML*, (2011)
34. Moritz, P., Nishihara, R., Jordan, M. I.: A linearly convergent stochastic l-bfgs algorithm, *Mathematics*, (2015)

35. Jin, X.B., Zhang, X.Y., Huang, K., Geng, G.G.: Stochastic conjugate gradient algorithm with variance reduction. IEEE transactions on neural networks and learning systems **30**(5), 1360–1369 (2018)