Neurocomputing 314 (2018) 177–185

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Mini-batch algorithms with Barzilai–Borwein update step

Zhuang Yang[a], Cheng Wang[a,*], Yu Zang[a], Jonathan Li[a,b]

[a] Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, Xiamen FJ 361005, China

[b] Department of Geography and Environmental Management, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 28 July 2017
Revised 9 May 2018
Accepted 4 June 2018
Available online 26 June 2018

Communicated by Prof. Tianbao Yang

Keywords:
Stochastic optimization
Mini batches
Barzilai–Borwein method
Variance reduction
Convex optimization

ABSTRACT

As a way to accelerate stochastic schemes, mini-batch optimization has been a popular choice for large scale learning due to its good general performance and ease of parallel computing. However, the performance of mini-batch algorithms can vary significantly based on the choice of the step size sequence, and, in general, there is a paucity of guidance for making good choices. In this paper, we propose to use the Barzilai–Borwein (BB) update step to automatically compute step sizes for the state of the art mini-batch method (mini-batch semi-stochastic gradient descent (mS2GD) method), thereby obtaining a new optimization method: mS2GD-BB. We prove that mS2GD-BB converges linearly in expectation for non-smooth strongly convex objective functions. We analyze the complexity of mS2GD-BB and show that it achieves as fast a rate as modern stochastic gradient methods. Numerical experiments on standard data sets indicate that the performance of mS2GD-BB is superior to some state of the art methods.

1. Introduction

Machine learning poses data-driven optimization problems in which the objective function involves the summation of loss functions over a set of data to be modeled. Deterministic optimization methods must evaluate this sum in its entirety for each evaluation of the objective, respectively its gradient. Therefore, as available data sets grow ever larger and larger, such optimization methods become increasingly inefficient. They are also inapposite for the online setting, where partial data must be modeled as it arrives.

By contrast, stochastic gradient descent (SGD) methods [1–8] work with gradient estimates obtained from a single sample of training data. This greatly reduces computational costs for large and redundant data sets. However, the total number of gradient evaluations of SGD depends on the variance of the stochastic gradients, which results in very slow convergence. In order to deal with the issue of high variance, SGD [9–13] usually use a diminishing step size, but not too rapidly, i.e., the step size is proportional to $\frac{1}{k^a}$ with $\frac{1}{2} < a \leq 1$. However, diminishing step size further makes SGD converge slowly [14]. For example, the convergence rate is sublinear, even if the objective function is strongly convex and smooth.

Another effective way to reduce the high variance in SGD is mini-batching, where each iteration is updated based on the average gradient with respect to several samples at a time rather than a single sample. Compared with the classic batch algorithm, mini-batch algorithms [15–20] reduce computational cost by orders of magnitude and yield significantly better solutions than online stochastic gradient. Does using a mini-batching strategy allow the stochastic optimization methods to use a non-decreasing step size?

Actually, in mini-batch algorithms (cf. [17,18,21–25]), almost-sure convergence is guaranteed assuming the step size is diminishing. Typically, for SGD and mini-batch algorithms, there is no guidance on the specific choice of the step size sequence, and problem parameters play a very small role in refining this choice.

Some recent works that discuss the choice of step size in mini-batch algorithms are summarized as follows. Reddi et al. [26] analyzed the stochastic variance reduced gradient (SVRG) method for nonconvex finite-sum problems and proposed a mini-batch nonconvex stochastic variance reduction gradient (MSVRG) method. In MSVRG, they showed that its step size depends on the number of iterations. However, they pointed out that because, in practice, it is difficult to compute step size of MSVRG, it is typical to try multiple step sizes and choose the one with the best results, which is time consuming in practice. Li et al. [20] proposed a new way of performing mini-batch updates beyond simple parameter averaging and argued that, under certain conditions, for their proposed method, diminishing step size can develop into a constant step size. Konečnỳ et al. [27] proposed the mini-batch semi-stochastic

* Corresponding author.
E-mail address: cwang@xmu.edu.cn (C. Wang).

https://doi.org/10.1016/j.neucom.2018.06.002
0925-2312/

gradient descent (mS2GD) method for solving nonsmooth objective functions. The mS2GD method works with a constant step size. They showed that mS2GD can reach any predefined accuracy with less overall work than a method without mini-batching and is suitable for further acceleration by parallelization. We conclude that the common practice in mini-batch algorithms is either to use a diminishing step size, or a tuning step size by hand, which, in practice, can be time.

In this paper, we propose to use the Barzilai–Borwein (BB) update step to automatically compute step size for the state of the art mini-batch method: mS2GD, thereby obtaining a new method: mS2GD-BB. We prove that mS2GD-BB converges linearly in expectation for nonsmooth strongly convex objective functions. We analyze the complexity of mS2GD-BB and show that, under certain conditions, mS2GD-BB has as fast a rate as modern stochastic methods such as SAG [28], SDCA [29] and SVRG [30]. Finally, we conduct experiments using mS2GD-BB to solve logistic regression. The experimental results demonstrate that our proposed method obtains an appropriate step size sequence and achieves better performance than some state of the art methods.

The remainder of this paper is organized as follows. Section 2 introduces the details of our proposed method. Section 3 analyzes the convergence of our proposed method and shows its complexity. Section 4 presents our numerical results and Section 5 concludes the paper.

2. Algorithm

2.1. Problem statement

In many machine learning problems, the following unconstrained optimization problem is usually considered:

$$\min_{w\in\mathbb{R}^d} P(w) = F(w) + R(w), \tag{1}$$

where $F(w)$ is the average of many smooth component functions $f_i(w)$, i.e.,

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w), \tag{2}$$

and $R(w)$ is relatively simple, but possibly nonsmooth [e.g., $R(w) = \|w\|_1$ or $R(w)\equiv 0$].

A standard method to solve the above optimization problem is the proximal gradient method [31–33]. Given an initial point, $w_0 \in \mathbb{R}^d$, the proximal gradient method employs the following update rule for $t = 1, 2, \ldots$:

$$w_{t+1} = \arg\min_{w\in\mathbb{R}^d}\left\{\nabla F(w_t)^T w + \frac{1}{2\eta_t}\|w - w_t\|_2^2 + R(w)\right\},$$

where $\eta_t > 0$ is the step size at the tth iteration. With the definition of proximal mapping

$$prox_R(z) \overset{\text{def}}{=} \arg\min_{w\in\mathbb{R}^d}\left\{\frac{1}{2}\|w - z\|_2^2 + R(w)\right\},$$

the proximal gradient method is compactly written as:

$$w_{t+1} = \text{prox}_{\eta_t R}(w_t - \eta_t \nabla F(w_t)). \tag{3}$$

In a large scale setting, it is more efficient to consider instead the stochastic proximal gradient method, in which the proximal operator is applied to the following stochastic gradient step:

$$w_{t+1} = \text{prox}_{\eta_t R}(w_t - \eta_t G_t), \tag{4}$$

where G_t is a stochastic estimate of the gradient $\nabla F(w_t)$.

2.2. mS2GD

Konečnỳ et al. [27] proposed the mS2GD method for solving problem (1). They showed that mS2GD reaches a predefined accuracy with less overall work than a method without mini-batching. The mS2GD method performs a deterministic step (evaluating the gradient of the objective function at the starting point), followed by a large number of stochastic steps.

In the stochastic steps of mS2GD, the stochastic estimate of $\nabla F(w_t)$ takes the form

$$G_t = \nabla\Phi_I(w_t) - \nabla\Phi_I(\widetilde{w}) + \nabla F(\widetilde{w}), \tag{5}$$

where $\nabla\Phi_I(w_t) = \frac{1}{|I|}\sum_{i\in I}\nabla f_i(w_t)$, $\nabla\Phi_I(\widetilde{w}) = \frac{1}{|I|}\sum_{i\in I}\nabla f_i(\widetilde{w})$, $\nabla F(\widetilde{w}) = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\widetilde{w})$, $I \subset \{1, 2, \ldots, n\}$ is a random mini-batch of size b, and $\widetilde{w}$ is an "old" reference point for which the gradient, $\nabla F(\widetilde{w})$, has previously been evaluated.

2.3. Barzilai-Borwein update step

The BB method, proposed by Barzilai and Borwein [34] and equipped with some quasi-Newton property, has proven to be a powerful paradigm for solving nonlinear optimization problems. To solve the unconstrained minimization problem,

$$\min f(w), \tag{6}$$

where f is differentiable, a typical iteration of the BB method uses the following iteration scheme:

$$w_{t+1} = w_t - \eta_t \nabla f(w_t), \tag{7}$$

where η_t is evaluated as follows:

$$\eta_t = \|s_t\|_2^2/(s_t^T y_t), \tag{8}$$

where $s_t = w_t - w_{t-1}$ and $y_t = \nabla f(w_t) - \nabla f(w_{t-1})$ for $t \geq 1$.

The way to determine step size in Eq. (8) is in accordance with the so called quasi-Newton equation (a.k.a. secant equation):

$$H_t s_t = y_t. \tag{9}$$

In Eq. (9), H_t is an approximation of the Hessian matrix of f at the current iteration, w_t, and appears in the quasi-Newton iteration scheme: $w_{t+1} = w_t - H_t^{-1}\nabla f(w_t)$. Barzilai and Borwein [34] regarded $H_t = \eta_t^{-1} I$ as an approximation for the Hessian of f at w_t. Hence, one can find η_k such that the residual of the secant equation, i.e., $\|(1/\eta_t)s_t - y_t\|_2^2$, is minimized, which leads to Eq. (8). In practice, such a method to calculate step size in Eq. (8) effectively reduces computational cost and achieves better performance. (For detailed information, such as convergence analysis and variants of the BB method, please see [35–41] and the references therein.)

2.4. mS2GD with BB step size

According to the introduction about the mS2GD and the BB step size in Sections 2.2 and2.3, we now describe mS2GD-BB (Algorithm 1). The only difference between mS2GD and mS2GD-BB is that in the latter, instead of using a prefixed η in mS2GD, we use the BB update step to compute step size, η_s.

Note that, to deal with the nonsmooth case, we introduce the sub-gradient[1] of objective function, $P(w)$, to compute the step size in Algorithm 1. As indicated in Algorithm 1, if we always set $\eta_s = \eta$ in mS2GD-BB instead of using Eq. (10), then mS2GD-BB is reduced

[1] We adopt the definition of sub-gradient from [42]. Namely, if P is a convex function, then a vector g' is called the sub-gradient of the function, P, at the point $w_0 \in dom\, R$ if, for any $w \in dom\, P$, we have that $P(w) \geq P(w_0) + \langle g', w - w_0\rangle$.

Algorithm 1 mS2GD-BB.

Input: m (max of stochastic steps per iteration); mini-batch size $b \in [n]$; initial point $\widetilde{w}_0$; initial step size η_0
for $s = 0, 1, 2, \ldots,$ **do**
  Set $\widetilde{w} = \widetilde{w}_s$
  Compute and store $g_s = (1/n)\sum_{i=1}^{n} \nabla f_i(\widetilde{w})$, $g_s^{'} = \partial P(\widetilde{w})$
  if $s > 0$ **then**

$$\eta_s = \frac{b}{m} \|\widetilde{w}_s - \widetilde{w}_{s-1}\|_2^2 / (\widetilde{w}_s - \widetilde{w}_{s-1})^T (g_s^{'} - g_{s-1}^{'}) \tag{10}$$

  end if
  Set $w_0 = \widetilde{w}$
  Randomly choose $t_s \in \{1, 2, \ldots, m\}$
  for $t = 0$ to $t_s - 1$ **do**
    Randomly choose mini-batch $I_t \subset \{1, \ldots, n\}$ of size b
    Compute a stochastic estimate of $\nabla F(w_t)$; $G_t = \nabla \Phi_{I_t}(w_t) - \nabla \Phi_{I_t}(\widetilde{w}) + g_s$
    $w_{t+1} = prox_{\eta_s R}(w_t - \eta_s G_t)$
  end for
  Set $\widetilde{w}_{s+1} = w_{t_s}$
end for

to the original mS2GD. In addition, for the first epoch of mS2GD-BB, we set an initial step size η_0. However, we observed from numerical experiments that the performance of our proposed method is insensitive to the choice of initial step size η_0. Moreover, with a minor modification, the BB update step can also be easily incorporated in other mini-batch algorithms, such as MSVRG, etc.

3. Analysis

In this section, we state our assumptions, analyze the convergence of our proposed method, and show how our method converges linearly in expectation for strongly convex and nonsmooth functions. Finally, we analyze the complexity of mS2GD-BB.

3.1. Assumptions

Our analysis is conducted taking into account the following assumptions:

Assumption 1. Each gradient of the convex function $f_i(w)$ in Eq. (2) is differentiable and Lipschitz continuous with a positive constant, L, which means that for all w and $v \in \mathbb{R}^d$

$$\|\nabla f_i(w) - \nabla f_i(v)\|_2 \le L \| w - v \|_2 . \tag{11}$$

Assumption 1 implies that the gradient of the average function, $F(w)$, is also Lipschitz continuous, i.e., there exists a positive constant, L, such that for all w, $v \in \mathbb{R}^d$, holds as follows:

$$\|\nabla F(w) - \nabla F(v)\|_2 \le L \| w - v \|_2 . \tag{12}$$

Assumption 2. $P(w)$ is μ-strongly convex, i.e., there exists $\mu > 0$ such that for all w, $v \in \mathbb{R}^d$, holds as follows:

$$P(w) \ge P(v) + \xi^T (w - v) + \frac{\mu}{2} \| w - v \|_2^2 \qquad \forall \xi \in \partial P(w). \tag{13}$$

The convexity parameter of a function is the largest μ for which the above condition holds. Let $F(w)$ and $R(w)$ have convexity parameters μ_F and μ_R, respectively; then we take $\mu \ge \mu_F + \mu_R$. Note that, although we must have $\mu_F \le L$, it is possible for $\mu > L$.

3.2. Main result

The following theorem establishes the linear convergence in expectation of the mS2GD-BB method:

Theorem 1. *Suppose that Assumptions 1 and 2 hold. Let* $w_* \overset{\text{def}}{=} \arg\min_w P(w)$ *and choose* $b \in \{1, \ldots, n\}$. *Additionally, assume that* $4L\alpha(b)b/m\mu < 1$, *and* m *is sufficiently large, so that*

$$\rho \overset{\text{def}}{=} \frac{1}{b[1 - 4L\alpha(b)b/m\mu]} + \frac{4L\alpha(b)b(m+1)}{m[\mu m - 4L\alpha(b)b]} < 1, \tag{14}$$

where $\alpha(b) = ((n-b)/b(n-1))$. *Then, mS2GD-BB has linear convergence in expectation with rate* ρ:

$$\mathbb{E}[P(\widetilde{w}_s) - P(w_*)] \le \rho^s [P(\widetilde{w}_0) - P(w_*)].$$

Proof. Proofs and lemmas related to Thereom 1 are given in Appendix A. □

As seen from Theorem 1, for any fixed b, by choosing m large enough, we force ρ to be arbitrarily small. It seems that for the mS2GD-BB method to achieve a solution of any prescribed accuracy, only single outer loop ($s = 1$) is required. This is indeed the case. However, such a choice of the parameters, (m, b, s), for the method are not optimal. To satisfy $\mathbb{E}[P(\widetilde{w}_s) - P(w_*)] \le \epsilon$ and achieve linear convergence, one needs to perform $s = O(\log(1/\epsilon))$ outer loops and choose the parameters b and m to an appropriate value (generally, $m = O(\kappa)$).

3.3. Complexity analysis

For a clean comparison, we first show the complexity of the modern stochastic gradient methods and proximal gradient descent methods. Then, we analyze the complexity of our mS2GD-BB method.

To achieve ϵ-accuracy, i.e.,

$$\mathbb{E}[P(\widetilde{w}_s) - P(w_*)] \le \epsilon [P(\widetilde{w}_0) - P(w_*)], \tag{15}$$

the complexity of the modern stochastic gradient methods, such as SDCA, SAG, and SVRG is

$$O\left((n + \kappa)\log\left(\frac{1}{\epsilon}\right)\right), \tag{16}$$

where $\kappa = L/\mu$ is a condition number.

The complexity bound (16) should be compared with that of the proximal gradient descent methods (e.g., ISTA [43]), which is $O(n\kappa\log(1/\epsilon))$, or FISTA [31], which is $O(n\sqrt{\kappa}\log(1/\epsilon))$. Note that while all these methods enjoy linear convergence rate, the modern stochastic gradient methods are many orders of magnitude faster than classical deterministic methods. Indeed, we have $n + \kappa \le n\sqrt{\kappa} \le n\kappa$.

Now, we analyze the complexity of mS2GD-BB. When $m \gg 1$, from (14), we have

$$\rho \approx \frac{1}{b[1 - 4L\alpha(b)b/m\mu]} + \frac{4L\alpha(b)b}{\mu m - 4L\alpha(b)b}. \tag{17}$$

When setting $m = 10L\alpha(b)b/\mu$, i.e., $m = 10\kappa\alpha(b)b$, from (17), we have

$$\rho \approx \frac{5}{3b} + \frac{2}{3}.$$

Therefore, with appropriate choice of parameter b, we can make ρ less than 1. Specifically, if we set the number of outer iterations to $s = \lceil \log(1/\varepsilon) \rceil$ and choose $m = 10\alpha(b)b\kappa$, the mS2GD-BB method needs $(n + 10\alpha(b)b\kappa)\log(1/\varepsilon)$ units of work for achieving (15). Note that this recovers the fast rate (16).

In addition, when choosing $b = n$, we have $\alpha(b) = 0$ and hence $\rho = 1/n$. We obtain the complexity of mS2GD-BB as $O(nm\log(1/\epsilon))$. When we set $m = O(\kappa)$, this rate approximates the proximal gradient descent methods.

Hence, by modifying the mini-batch size, b, in mS2GD-BB, the rate of mS2GD-BB varies between the fast rate of the modern

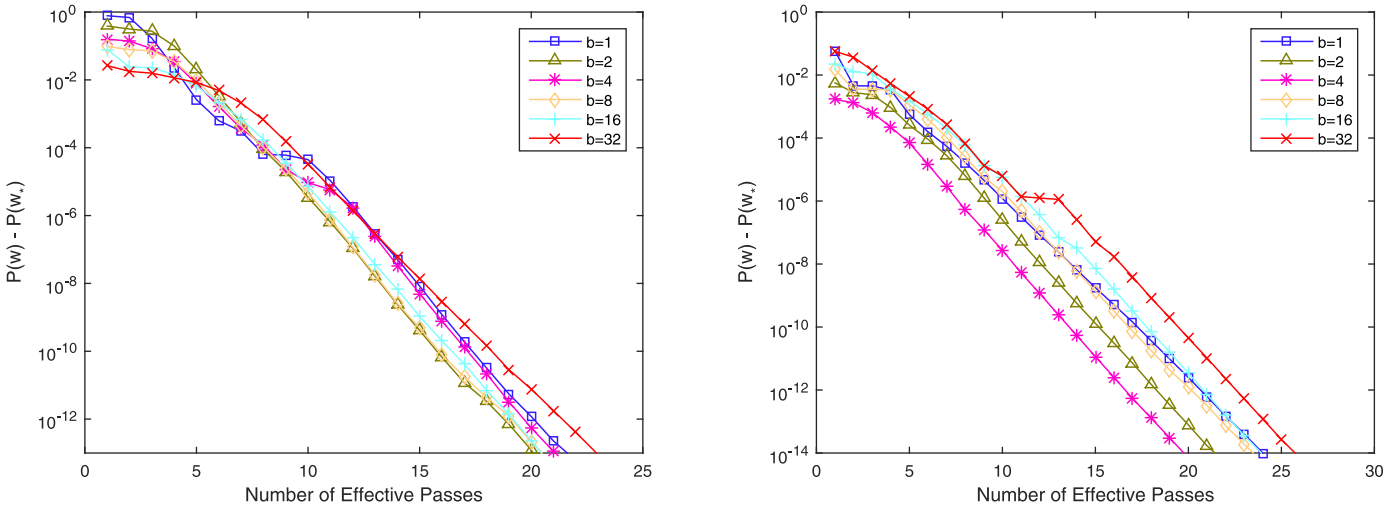


Fig. 1. Comparison of mS2GD-BB with different mini-batch sizes on *w8a* (left) and *cina0* (right).

stochastic gradient methods and the slow rates of the proximal gradient descent methods.

Comparing the analysis of mS2GD in [27], we see that mS2GD-BB and mS2GD have similar properties. Actually, the benefit of our mS2GD-BB method is that it automatically computes step size by using the BB method for mS2GD, but does not affect the advantage of mS2GD in practical applications and even can improve the performance of mS2GD.

4. Numerical experiments

In this section, we present results of several numerical experiments to illustrate the properties of our mS2GD-BB method and compare its performance with several state of the art methods. In Section 4.1, we show the properties of mS2GD-BB that are involved in solving standard testing problems for different values of b. In Section 4.2, we compare mS2GD-BB with mS2GD in regard to standard testing problems. Finally, in Section 4.3 we compare our proposed method with other state of the art methods.

Of all the experiments, we discuss those that were performed with $R(w) = \frac{\lambda_2}{2}\|w\|_2^2 + \lambda_1\|w\|_1$. Each $f_i(w)$ in Eq. (2) is the logistic loss function given as follows:

$$f_i(w) = \log(1 + \exp(-b_i a_i^T w)). \tag{18}$$

These functions are usually used in machine learning, with $(a_i, b_i) \in \mathbb{R}^d \times \{+1, -1\}$, $i = 1, \ldots, n$, being a training data set of example-label pairs. The resulting optimization problem (1), employed in machine learning for binary classification, takes the following form:

$$\min_{w\in\mathbb{R}^d} \quad P(w) := \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-b_i a_i^T w)) + \frac{\lambda_2}{2}\|w\|_2^2 + \lambda_1\|w\|_1. \tag{19}$$

For the experiments discussed in this work, the following four publicly available data sets were used: *rcv*1, *ijcnn*1, *w8a*,[2] and *cina*0.[3]

In logistic regression, the Lipschitz constant of function f_i is equal to $L_i = \|a_i\|_2^2/4$. Although unnecessary, for convenience, we normalize the *rcv*1 and *cina*0 data sets so that $\|a_i\|_2 = 1$ for all $i = 1, 2, \ldots, n$, which leads to the same upper boundary hold on the Lipschitz constants $L = \|a_i\|_2^2/4$. In addition, our analysis assumes (Assumption 1) the same constant, L, for all functions. Hence, for the other two data sets, we have $L = max_{i\in[n]}L_i$. Table 1 presents basic information, such as size (n), dimension (d), and Lipschitz constant (L). Also in Table 1 are the values of λ_1 and λ_2 used in our experiments. (λ_1 and λ_2 are typical benchmarks used in machine learning to obtain good classification performance for (19).)

Table 1
Summary of data sets used in the numerical experiments.

Data sets	n	d	λ_1	λ_2	L
rcv1	20,242	47236	10^{-5}	10^{-4}	0.2500
ijcnn1	49,990	22	10^{-5}	10^{-4}	0.9841
w8a	49,749	300	10^{-5}	10^{-4}	28.5000
cina0	16,033	132	10^{-4}	10^{-4}	0.2500

4.1. Properties of mS2GD-BB

Fig. 1 illustrates the results of mS2GD-BB under various values of b. In Fig. 1, the x-axis represents the number of effective passes over the data, where each effective pass evaluates n component gradients. The y-axis denotes the sub-optimality $P(w) - P(w_*)$. Here, w_* is obtained by running mS2GD with the best-tuned step size until mS2GD converges.

As seen in Fig. 1, when the mini-batch size increases to $b = 2, 4, 8, 16$, the performance of mS2GD-BB is the same or better than that of mS2GD-BB with $b = 1$.

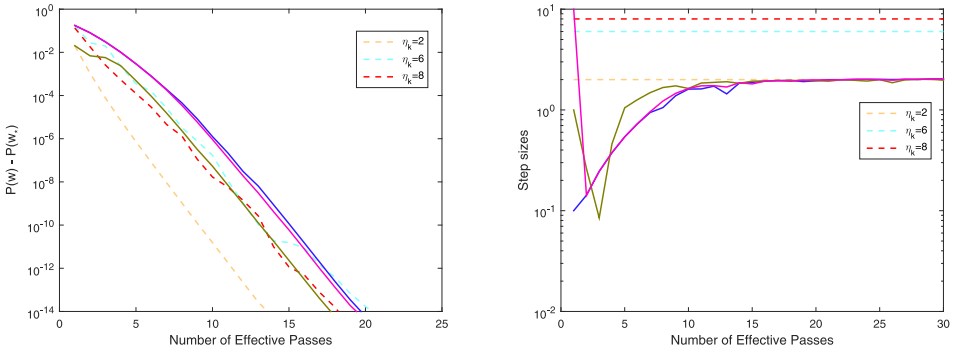
4.2. Comparison with mS2GD

In this sub-section, we compare mS2GD-BB and mS2GD with fixed step size for solving Eq. (19) using the four data sets listed in Table 1. We employed the best-tuned step size for mS2GD, and, for mS2GD-BB, three different initial step sizes, η_0. (η_0 was chosen from {0.1, 1, 10}.) For mS2GD and mS2GD-BB, using the *rcv*1 and *cina*0 data sets, we set $b = 4$; using the *ijcnn*1 data set, we set $b = 16$; using the *w8a* data set, we set $b = 2$.

The results of the comparing mS2GD with mS2GD-BB are shown in Fig. 2. In all the sub-figures, the x-axis represents the number of effective passes over the data. In Fig. 2(a), (c), (e) and (g), the y-axis denotes the sub-optimality $P(w) - P(w_*)$; in Fig. 2(b), (d), (f) and (h), the y-axis denotes the corresponding step sizes, η_s. In all the sub-figures, the dashed lines (shown in the legends of the figures) stand for mS2GD with different fixed step sizes. The solid lines represent mS2GD-BB with different initial step sizes, η_0.

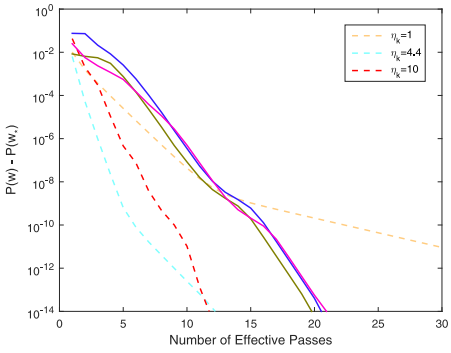
[2] *rcv*1, *covertype* and *w8a* are available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

[3] *cina*0 are available at http://www.causality.inf.ethz.ch/home.php.

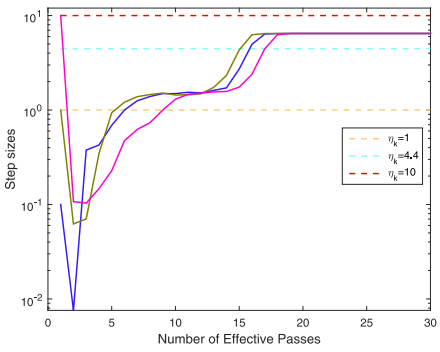


(a) Sub-optimality on *rcv*1

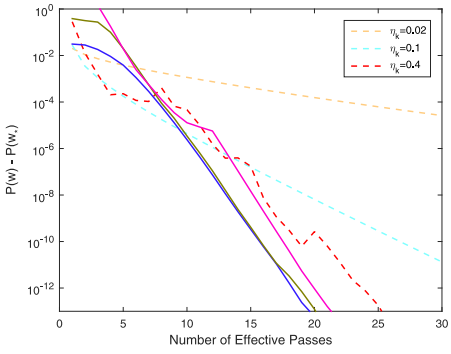
(b) Step sizes on *rcv*1



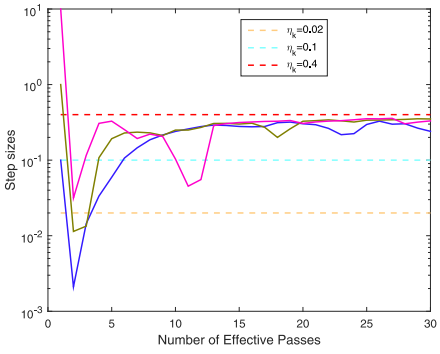
(c) Sub-optimality on *ijcnn*1



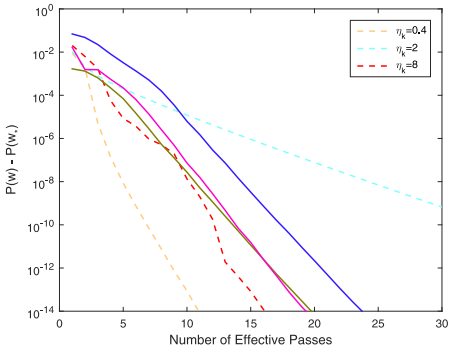
(d) Step sizes on *ijcnn*1



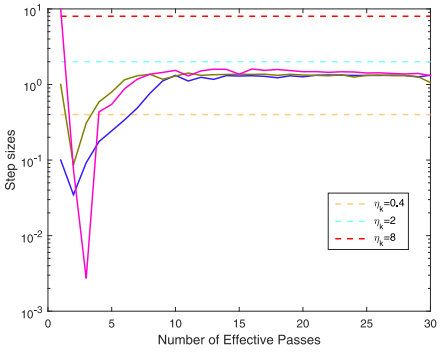
(e) Sub-optimality on *w*8*a*



(f) Step sizes on *w*8*a*



(g) Sub-optimality on *cina*0



(h) Step sizes on *cina*0

Fig. 2. Comparison of mS2GD-BB and mS2GD with fixed step sizes on different data sets. The dashed lines represent mS2GD with different fixed step sizes η_s given in the legend. The solid lines stand for mS2GD-BB with different initial step size η_0; for example, the solid lines in Fig. 2(a) and (b) correspond to mS2GD-BB with $\eta_0 = 10, 1, 0.1$, respectively.

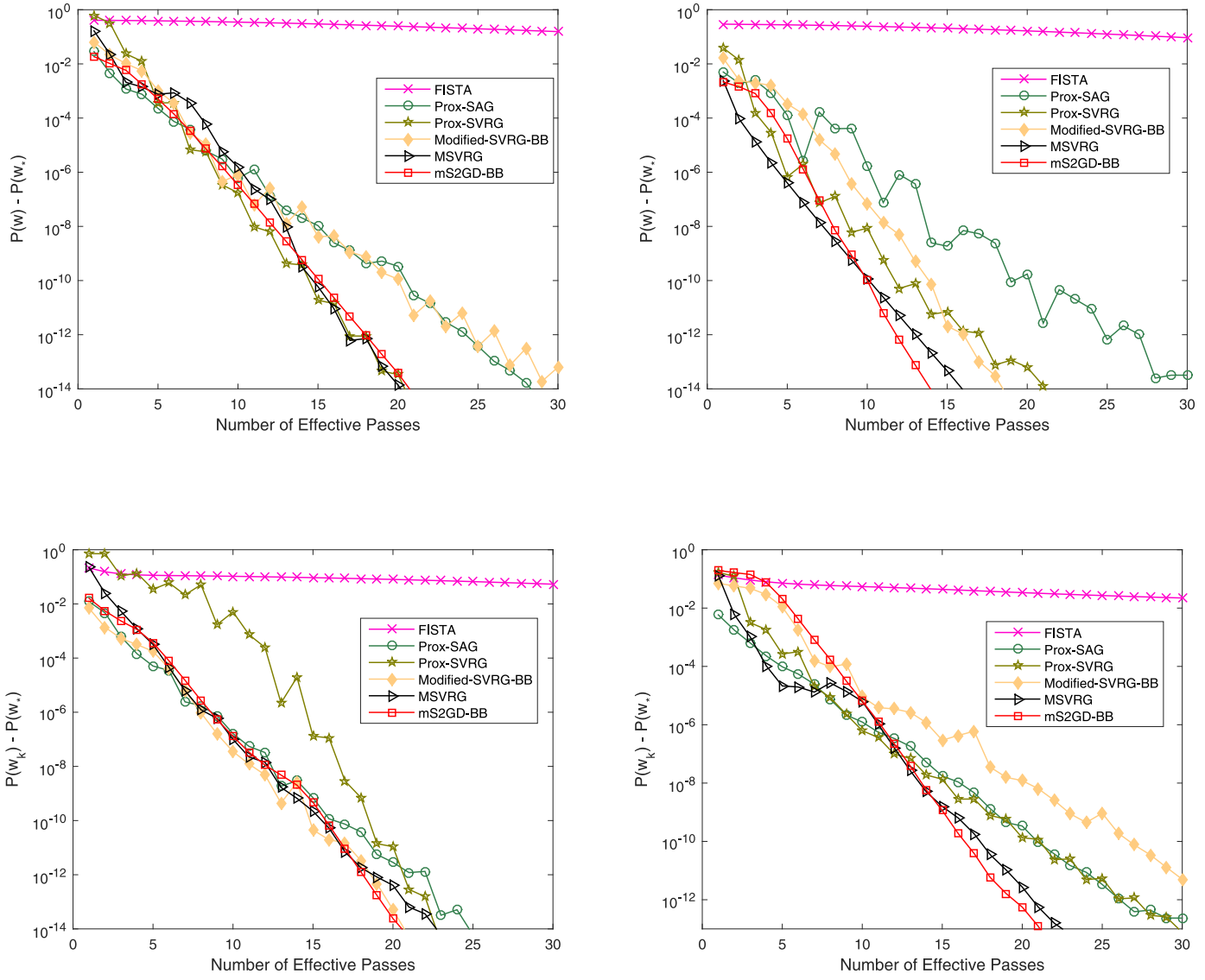


Fig. 3. Comparison of different methods on four data sets: *rcv*1 (top left), *cina*0 (top right), *ijcnn*1 (bottom left) and *w8a* (bottom right). We have used mS2GD-BB with $b = 4$.

Fig. 2(a), (c), (e) and (g) shows that mS2GD-BB always achieves the same level of sub-optimality as mS2GD with the best-tuned step size and achieves even better performance than mS2GD with the best-tuned step size. As seen from Fig. 2(a), (c), (e) and (g), the performance of mS2GD-BB is better than that of mS2GD with the other choice of step size. Moreover, as shown in Fig. 2(b), (d), (f) and (h), the step sizes computed by mS2GD-BB converge to the best-tuned step sizes after about ten or fifteen epochs. Also, as seen from Fig. 2, mS2GD-BB is insensitive to the choice of the initial step size, η_0.

4.3. Comparison with other algorithms

To further confirm the effectiveness of our proposed method, we experimented on mS2GD-BB in comparison with deterministic (FISTA) and stochastic gradient (Modified-SVRG-BB, Prox-SAG, Prox-SVRG, and MSVRG) methods for Eq. (19). The full gradient and stochastic gradient methods are specified as follows:

(1) *FISTA:* FISTA is a version of the fast iterative shrinkage-thresholding algorithms [31] for solving linear inverse problems.

(2) *Prox-SAG:* Prox-SAG is a proximal version of the SAG method [28]. In Prox-SAG, as suggested in [44], we used the standard backtracking line search to obtain the step size.

(3) *Prox-SVRG:* Prox-SVRG is a proximal version of the stochastic variance-reduction gradient method. Xiao and Zhang [33] reported that, in theory, the algorithm achieves better performance when a step size of $\eta = 0.1/L$ is adopted. In Prox-SVRG, we employed $m = 2n$ between full gradient evaluations.

(4) *Modified-SVRG-BB:* The SVRG-BB [45] method is used to deal with the smooth case of Eq. (2). To solve the nonsmooth case, we enhance SVRG-BB, and call the resulting method Modified-SVRG-BB. Modified-SVRG-BB uses the subgradient of the nonsmooth objective function to determine the step size sequence $\{\eta_k\}$. Moreover, to update the solution sequence, $\{w_k\}$, the proximal operation is introduced into Modified-SVRG-BB. In addition, as suggested in [45], we set $m = 2n$.

(5) *MSVRG:* MSVRG is a mini-batch version of SVRG proposed in [26] for nonconvex finite-sum problems. In MSVRG, a constant step size was used. Same as in [26], we use mini-batches of size 10 for all data sets.

Fig. 3 shows that, with most data sets listed in Table 1, mS2GD-BB achieves a better performance than the competing methods. In our experiments, for mS2GD-BB, we set $b = 4$. The best performance was given when we employ $m = \{0.14n, 0.13n, 0.11n, 0.08n\}$ on different data sets listed in Table 1, respectively.

5. Conclusion

In this paper, we have proposed a new class of iterative methods, mS2GD-BB, which employs the BB update step to automatically compute step size in mini-batch settings. We proved that our proposed method converges linearly in expectation for nonsmooth strongly convex objective functions. We analyze the complexity of mS2GD-BB and show that it achieves as fast a rate as the modern stochastic gradient methods. The results obtained on standard data sets indicate that the appropriate step size is obtained by running mS2GD-BB. Finally, comparative experiments in logistic regression show that mS2GD-BB converges more rapidly than the competing methods.

Acknowledgments

This work was supported by grants from the National Natural Science Foundation of China under Grant U1605254. We are very grateful to the anonymous reviewers for their valuable comments and suggestions.

Appendix A.

A1. Technical results

In the convergence analysis, we also employ the non-expansiveness of proximal mapping, which can be found in the optimization literature [46].

Lemma 1. *Let R be a closed convex function on $\mathbb{R}^d$ and w, $v \in \mathbb{R}^d$, then $\|prox_R(w) - prox_R(v)\|_2 \le \|w - v\|_2$.*

In addition, before giving the proof of Thereom 1, the bound variance of the modified gradient G_k is given in the following lemma, where it can be found in [27]:

Lemma 2 (Bounding variance)**.** *Let $\alpha(b) \overset{\text{def}}{=} ((n-b)/b(n-1))$. Considering the definition of G_t in Thereom 1, conditioned on w_t, we have the $\mathbb{E}[G_t] = \nabla F(w_t)$ and the variance satisfies,*

$$\mathbb{E}\left[\|G_t - \nabla F(w_t)\|_2^2\right] \le 4L\alpha(b)[P(w_t) - P(w_*) + P(\widetilde{w}) - P(w_*)]. \tag{A.20}$$

A2. Proof of Theorem 1

Now, we give the proof of Theorem 1. The proof follows the steps in [33]. For convenience, we define the stochastic mapping as follows:

$$d_t = \frac{1}{\eta_s}(w_t - w_{t+1}) = \frac{1}{\eta_s}(w_t - prox_{\eta_s R}(w_t - \eta_s G_t)),$$

so that the updated iterations are written as $w_{t+1} = w_t - \eta_s d_t$. Let us estimate the change of $\|w_{t+1} - w_*\|_2^2$. It holds that

$$\begin{aligned}\|w_{t+1} - w_*\|_2^2 &= \|w_t - \eta_s d_t - w_*\|_2^2 \\ &= \|w_t - w^*\|_2^2 - 2\eta_s d_t^T(w_t - w^*) + \eta_s^2\|d_t\|_2^2.\end{aligned} \tag{A.21}$$

Applying Lemma 3.7[4] in [33] with $w = w_t$, $v = G_t$, $w^+ = w_{t+1}$, $g = d_t$, $y = w_*$ and $\Delta = \Delta_t = G_t - \nabla F(w_t)$, we obtain

$$\begin{aligned}-d_t^T(w_t - w_*) + \frac{\eta_s}{2}\|d_t\|_2^2 \le P(w_*) - P(w_{t+1}) - \frac{\mu_F}{2}\|w_t - w_*\|_2^2 \\ -\frac{\mu_R}{2}\|w_{t+1} - w_*\|_2^2 - \Delta_t^T(w_{t+1} - w_*),\end{aligned} \tag{A.22}$$

Taking into account (A.21) and (A.22) we get:

$$\begin{aligned}\|w_{t+1} - w_*\|_2^2 \le \|w_t - w_*\|_2^2 - 2\eta_s[P(w_{t+1}) - P(w_*)] \\ -2\eta_s\Delta_t^T(w_{t+1} - w_*).\end{aligned} \tag{A.23}$$

To bound $-2\Delta_t^T(w_{t+1} - w_*)$, we define the proximal full gradient update as[5] $\overline{w}_{t+1} = prox_{\eta_s R}(w_t - \eta_s \nabla F(w_t))$. Then, we have

$$\begin{aligned}&-2\eta_s\Delta_t^T(w_{t+1} - w_*) \\ &= -2\eta_s\Delta_t^T(w_{t+1} - \overline{w}_{t+1}) - 2\eta_s\Delta_t^T(\overline{w}_{t+1} - w_*) \\ &\le 2\eta_s\|\Delta_t\|\|w_{t+1} - \overline{w}_{t+1}\| - 2\eta_s\Delta_t^T(\overline{w}_{t+1} - w_*) \\ &= 2\eta_s\|\Delta_t\|\|prox_{\eta_s R}(w_t - \eta_s G_t) - prox_{\eta_s R}(w_t - \eta_s\nabla F(w_t))\| \\ &\quad -2\eta_s\Delta_t^T(\overline{w}_{t+1} - w_*) \\ &\le 2\eta_s^2\|\Delta_t\|^2 - 2\eta_s\Delta_t^T(\overline{w}_{t+1} - w_*)\end{aligned}$$

In the first inequality, we used the Cauchy–Schwarz inequality, and in the second inequality we used Lemma 1. Combining with (A.23), we have

$$\begin{aligned}\|w_{t+1} - w_*\|_2^2 \le \|w_t - w_*\|_2^2 - 2\eta_s[P(w_{k+1}) - P(w^*)] \\ +2\eta_s^2\|\Delta_t\|^2 - 2\eta_s\Delta_t^T(\overline{w}_{t+1} - w_*).\end{aligned}$$

By taking expectation on both sides of the above inequality with respect to i_t, we obtain

$$\begin{aligned}\mathbb{E}\|w_{t+1} - w_*\|_2^2 \le \|w_t - w_*\|_2^2 - 2\eta_s[\mathbb{E}P(w_{k+1}) - P(w_*)] \\ +2\eta_s^2\mathbb{E}\|\Delta_t\|^2 - 2\eta_s\mathbb{E}[\Delta_t^T(\overline{w}_{t+1} - w_*)].\end{aligned}$$

Note that with $\mathbb{E}(\Delta_t) = 0$ and combining Lemma 2, we have

$$\begin{aligned}\mathbb{E}\|w_{t+1} - w_*\|_2^2 \le \|w_t - w_*\|_2^2 - 2\eta_s[\mathbb{E}P(w_{t+1}) - P(w^*)] \\ +8L\alpha(b)\eta_s^2[P(w_t) - P(w_*) + P(\widetilde{w}) - P(w_*)].\end{aligned}$$

Using the strong convexity of $P(w)$, we obtain the following upper bound on the BB step size evaluated in Algorithm 1:

$$\begin{aligned}\eta_s &= \frac{b}{m}\cdot\frac{\|\widetilde{w}_s - \widetilde{w}_{s-1}\|_2^2}{(\widetilde{w}_s - \widetilde{w}_{s-1})^T(g_s' - g_{s-1}')} \\ &\le \frac{b}{m}\cdot\frac{\|\widetilde{w}_s - \widetilde{w}_{s-1}\|_2^2}{\mu\|\widetilde{w}_s - \widetilde{w}_{s-1}\|_2^2} = \frac{b}{\mu m}\end{aligned}$$

By using the last two inequalities, we ascertain that

$$\begin{aligned}\mathbb{E}\|w_{t+1} - w_*\|_2^2 \le \|w_t - w_*\|_2^2 - 2\frac{b}{\mu m}[\mathbb{E}P(w_{t+1}) - P(w_*)] \\ +8L\alpha(b)\frac{b^2}{\mu^2 m^2}[P(w_t) - P(w_*) + P(\widetilde{w}) - P(w_*)].\end{aligned}$$

We consider a fixed stage s, so that $w_0 = \widetilde{w} = \widetilde{w}_s$. In addition, by the definition of $\widetilde{w}_{s+1}$ in Algorithm 1 we have

$$\mathbb{E}[P(\widetilde{w}_{s+1})] = (1/m)\sum_{t=1}^{m} E[P(w_t)]. \tag{A.24}$$

By summing the previous inequality for $1 \le t \le m$, we have

$$\mathbb{E}\|w_{t+1} - w_*\|_2^2 + \frac{2b}{\mu m}[\mathbb{E}P(w_m) - P(w_*)] + \frac{2b}{\mu m}\left(1 - 4L\alpha(b)\frac{b}{\mu m}\right)$$

[4] We note that in Lemma 3.7 it need $0 < \eta \le 1/L$. In the following, we will give the upper bound of η_s, i.e., $\eta_s \le b/m\mu$. Obviously, if let m large enough, we can make η_s no more than $1/L$.

[5] Although we not use in the mS2GD-BB algorithm, we can still use it in the analysis nevertheless.

$$\times \sum_{t=1}^{m-1}[\mathbb{E}P(w_t) - P(w_*)]$$

$$\leq \|w_0 - w_*\|_2^2 + 8L\alpha(b)\frac{b^2}{\mu^2 m^2}$$

$$\times[P(w_0) - P(w_*) + m(P(\widetilde{w}) - P(w_*))].$$

Further, we have

$$\frac{2b}{\mu m}\left(1 - 4L\alpha(b)\frac{b}{\mu m}\right)\sum_{t=1}^{m}[\mathbb{E}P(w_t) - P(w_*)]$$

$$\leq \|\widetilde{w} - w_*\|_2^2 + 8L\alpha(b)\frac{b^2}{\mu^2 m^2}(m+1)[P(\widetilde{w}) - P(w_*)].$$

By using Eq. (A.24) and strong convexity of P implies $\|\widetilde{w} - w_*\|_2^2 \leq \frac{2}{\mu}[P(\widetilde{w}) - P(w_*)]$, we obtain

$$\frac{2b}{\mu m}\left(1 - 4L\alpha(b)\frac{b}{\mu m}\right)m[\mathbb{E}P(\widetilde{w}_{s+1}) - P(w_*)]$$

$$\leq \left(\frac{2}{\mu} + 8L\alpha(b)\frac{b^2(m+1)}{\mu^2 m^2}\right)[P(\widetilde{w}_s) - P(w_*)].$$

Dividing both sides of the above inequality by $2b[\mu m - 4L\alpha(b)b]/\mu^2 m$, we arrive at

$$\mathbb{E}P(\widetilde{w}_{s+1}) - P(w_*) \leq \left(\frac{1}{b[1 - 4L\alpha(b)b/m\mu]} + \frac{4L\alpha(b)b(m+1)}{m[\mu m - 4L\alpha(b)b]}\right)$$

$$\times[P(\widetilde{w}_s) - P(w_*)].$$

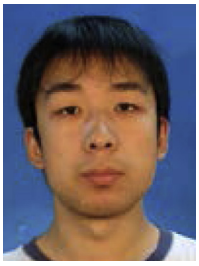
Finally using the definition of ρ in Eq. (13), and applying the above inequality recursively, we obtain the desired results.

References

[1] H. Robbins, S. Monro, A stochastic approximation method, Annal. Math. Stat. (1951) 400–407.
[2] T. Zhang, Solving large scale linear prediction problems using stochastic gradient descent algorithms, in: Proceedings of the Twenty-first International Conference on Machine Learning, ACM, 2004, p. 116.
[3] H. Szu, I. Kopriva, Unsupervised learning with stochastic gradient, Neurocomputing 68 (2005) 130–160.
[4] A. Nemirovski, A. Juditsky, G. Lan, A. Shapiro, Robust stochastic approximation approach to stochastic programming, SIAM J. Optim. 19 (4) (2009) 1574–1609.
[5] L. Bottou, Stochastic gradient descent tricks, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 421–436.
[6] S. Ghadimi, G. Lan, Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization I: a generic algorithmic framework, SIAM J. Optim. 22 (4) (2012) 1469–1492.
[7] D. Needell, R. Ward, N. Srebro, Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 1017–1025.
[8] V. Jumutc, J.A. Suykens, Reweighted stochastic learning, Neurocomputing 198 (2016) 135–147.
[9] Z. Luo, On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks, Neural Comput. 3 (2) (1991) 226–245.
[10] M.V. Solodov, Incremental gradient algorithms with stepsizes bounded away from zero, Comput. Optim. Appl. 11 (1) (1998) 23–35.
[11] A. Nemirovski, A. Juditsky, G. Lan, A. Shapiro, Robust stochastic approximation approach to stochastic programming, SIAM J. Optim. 19 (4) (2008) 1574–1609.
[12] V.S. Borkar, S.P. Meyn, The O.D. E. method for convergence of stochastic approximation and reinforcement learning, SIAM J. Control Optim. 38 (2) (2000) 447–469.
[13] O. Shamir, T. Zhang, Stochastic gradient descent for non-smooth optimization: convergence results and optimal averaging schemes., in: Proceedings of the International Conference on Machine Learning, 2013, pp. 71–79.
[14] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, 2010, pp. 177–186.
[15] Z.L. Sun, D.S. Huang, C.H. Zheng, L. Shang, Using batch algorithm for kernel blind source separation, Neurocomputing 69 (1) (2005) 273–278.
[16] A. Cotter, O. Shamir, N. Srebro, K. Sridharan, Better mini-batch algorithms via accelerated gradient methods, in: Proceedings of the Advances in Neural Information Processing Systems, 2011, pp. 1647–1655.
[17] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, Optimal distributed online prediction, in: Proceedings of the Twenty-eighth International Conference on Machine Learning (ICML-11), 2011, pp. 713–720.
[18] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, Optimal distributed online prediction using mini-batches, J. Mach. Learn. Res. 13 (Jan) (2012) 165–202.
[19] S. Shalev-Shwartz, T. Zhang, Accelerated mini-batch stochastic dual coordinate ascent, in: Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 378–385.
[20] M. Li, T. Zhang, Y. Chen, A.J. Smola, Efficient mini-batch training for stochastic optimization, in: Proceedings of the Twentieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 661–670.
[21] J.C. Duchi, Y. Singer, Efficient online and batch learning using forward backward splitting, J. Mach. Learn. Res. 10 (2009) 2899–2934.
[22] Y. Nesterov, Primal-dual subgradient methods for convex problems, Math. Program. 120 (1) (2009) 221–259.
[23] L. Xiao, Dual averaging methods for regularized stochastic learning and online optimization, J. Mach. Learn. Res. 11 (2010) 2543–2596.
[24] G. Lan, An optimal method for stochastic composite optimization, Math. Program. 133 (2012) 365–397.
[25] R.H. Byrd, S.L. Hansen, J. Nocedal, Y. Singer, A stochastic quasi-newton method for large-scale optimization, SIAM J. Optim. 26 (2) (2016) 1008–1031.
[26] S.J. Reddi, A. Hefny, S. Sra, B. Póczós, A.J. Smola, Stochastic variance reduction for nonconvex optimization, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 314–323.
[27] J. Konečnỳ, J. Liu, P. Richtárik, M. Takáč, Mini-batch semi-stochastic gradient descent in the proximal setting, IEEE J. Sel. Top. Signal Process. 10 (2) (2016) 242–255.
[28] N.L. Roux, M. Schmidt, F.R. Bach, A stochastic gradient method with an exponential convergence rate for finite training sets, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 2663–2671.
[29] S. Shalev-Shwartz, T. Zhang, Stochastic dual coordinate ascent methods for regularized loss minimization, J. Mach. Learn. Res. 14 (Feb) (2013) 567–599.
[30] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 315–323.
[31] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.
[32] N. Parikh, S. Boyd, et al., Proximal algorithms, Found. Trends Optim. 1 (3) (2014) 127–239.
[33] L. Xiao, T. Zhang, A proximal stochastic gradient method with progressive variance reduction, SIAM J. Optim. 24 (4) (2014) 2057–2075.
[34] J. Barzilai, J.M. Borwein, Two-point step size gradient methods, IMA J. Numer. Anal. 8 (1) (1988) 141–148.
[35] M. Raydan, On the Barzilai and Borwein choice of steplength for the gradient method, IMA J. Numer. Anal. 13 (3) (1993) 321–326.
[36] B. Molina, M. Raydan, Preconditioned Barzilai–Borwein method for the numerical solution of partial differential equations, Numer. Alg. 13 (1) (1996) 45–60.
[37] E.G. Birgin, J.M. Martínez, M. Raydan, Nonmonotone spectral projected gradient methods on convex sets, SIAM J. Optim. 10 (4) (2000) 1196–1211.
[38] Y. Dai, J. Yuan, Y.X. Yuan, Modified two-point stepsize gradient methods for unconstrained optimization, Comput. Optim. Appl. 22 (1) (2002) 103–109.
[39] Y.H. Dai, R. Fletcher, On the asymptotic behaviour of some new gradient methods, Math. Program. 103 (3) (2005) 541–559.
[40] Z. Xie, S. Chen, SCIHTBB: Sparsity constrained iterative hard thresholding with Barzilai–Borwein step size, Neurocomputing 74 (17) (2011) 3663–3676.
[41] H. Nosratipour, O.S. Fard, A.H. Borzabadi, An adaptive nonmonotone global Barzilai–Borwein gradient method for unconstrained optimization, Optimization (2017) 1–15.
[42] Y. Nesterov, Introductory Lectures on Convex Optimization: Basic Course, Kluwer Academic, 2004.
[43] M.A.T. Figueiredo, R.D. Nowak, An EM algorithm for wavelet-based image restoration, IEEE Trans. Image Process. 12 (8) (2003) 906–916.
[44] M. Schmidt, R. Babanezhad, M.O. Ahmed, A. Defazio, A. Clifton, A. Sarkar, Non-uniform stochastic average gradient method for training conditional random fields, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2015.
[45] C. Tan, S. Ma, Y.-H. Dai, Y. Qian, Barzilai–Borwein step size for stochastic gradient descent, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 685–693.
[46] R.T. Rockafellar, Convex Analysis:, Princeton University Press, 1970.

Zhuang Yang received his M.Sc. degree in numerical mathematics from the GuiLin University of Electronic Technology, China in 2014. He is currently pursuing a Ph.D. degree in information and communication engineering with the Fujian Key Laboratory of Sensing and Computing for Smart Cities and the Department of Communication Engineering, School of Information Science and Engineering, Xiamen University, Xiamen, China. His current research interests include machine learning, matrix analysis, optimization methods.

Cheng Wang received Ph.D. degrees in Communication and Signal Processing from the National University of Defense Technology (NUDT), Changsha, China, in 2002. He is currently a Professor in the School of Information Science and Engineering, and Executive Director of Fujian Key Laboratory of Sensing and Computing for Smart City, Xiamen University. He is also the Chair of Working Group I/6 on Multi-sensor Integration and Fusion, in the International Society of Remote Sensing (ISPRS). He has coauthored more than 150 papers in referred journals and top conferences including IEEE-TGRS, PR, IEEE-TITS, CVPR, AAAI, and ISPRS-JPRS. His current research interests include point cloud analysis, multi-sensor fusion, mobile mapping and geospatial bigdata. He is a Fellow of IET.



Yu Zang is currently a Research Assistant Professor at the School of Information Science and Technology, Xiamen University, China. He received his B.S. and Ph.D. degree in Xi'an Jiaotong University in 2008 and 2014. His main researches include remote sensing image processing, computer vision & graphics and mobile LiDAR data analysis.

Jonathan Li received the Ph.D. degree in geomatics engineering from the University of Cape Town, South Africa. He is currently professor with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, China. He is also professor and head of the Mobile Sensing and Geodata Science Lab at the Faculty of Environment, University of Waterloo, Canada. His current research interests include information extraction from LiDAR point clouds and from earth observation images. He has co-authored more than 300 publications, over 130 of which were published in refereed journals including IEEE-TGRS, IEEE-TITS, IEEE-GRSL, IEEE-JSTARS, ISPRS-JPRS, IJRS, PERS and RSE. He is Chair of the ISPRS Working Group I/6 on LiDAR for Airborne and Spaceborne Sensing (2016–2020), Chair of the ICA Commission on Sensor-driven Mapping (2015–2019), and Associate Editor of IEEE-TITS and IEEE-JSTARS.