# Barzilai-Borwein Step Size for Stochastic Gradient Descent

Conghui Tan[†] Shiqian Ma[†] Yu-Hong Dai[‡] Yuqiu Qian[§]

May 16, 2016

**Abstract**

One of the major issues in stochastic gradient descent (SGD) methods is how to choose an appropriate step size while running the algorithm. Since the traditional line search technique does not apply for stochastic optimization algorithms, the common practice in SGD is either to use a diminishing step size, or to tune a fixed step size by hand, which can be time consuming in practice. In this paper, we propose to use the Barzilai-Borwein (BB) method to automatically compute step sizes for SGD and its variant: stochastic variance reduced gradient (SVRG) method, which leads to two algorithms: SGD-BB and SVRG-BB. We prove that SVRG-BB converges linearly for strongly convex objective functions. As a by-product, we prove the linear convergence result of SVRG with *Option I* proposed in [10], whose convergence result is missing in the literature. Numerical experiments on standard data sets show that the performance of SGD-BB and SVRG-BB is comparable to and sometimes even better than SGD and SVRG with best-tuned step sizes, and is superior to some advanced SGD variants.

## 1 Introduction

The following optimization problem, which minimizes the sum of cost functions over samples from a finite training set, appears frequently in machine learning:

$$\min\ F(x) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1.1}$$

where $n$ is the sample size, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is the cost function corresponding to the $i$-th sample data. Throughout this paper, we assume that each $f_i$ is convex and differentiable, and the function $F$ is strongly convex. Problem (1.1) is challenging when $n$ is extremely large so that computing $F(x)$ and $\nabla F(x)$ for given $x$ is prohibited. Stochastic gradient descent (SGD) method and its variants have been the main approaches for solving (1.1). In the $t$-th iteration of SGD, a random training sample $i_t$ is chosen from $\{1, 2, \ldots, n\}$ and the iterate $x_t$ is updated by

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t), \tag{1.2}$$

where $\nabla f_{i_t}(x_t)$ denotes the gradient of the $i_t$-th component function at $x_t$, and $\eta_t > 0$ is the step size (a.k.a. learning rate). In (1.2), it is usually assumed that $\nabla f_{i_t}$ is an unbiased estimation to $\nabla F$, i.e.,

$$\mathbb{E}[\nabla f_{i_t}(x_t) \mid x_t] = \nabla F(x_t). \tag{1.3}$$

[†]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong. Email: chtan@se.cuhk.edu.hk, sqma@se.cuhk.edu.hk.

[‡]Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. Email: dyh@lsec.cc.ac.cn.

[§]Department of Computer Science, The University of Hong Kong, Hong Kong. Email: qyq79@connect.hku.hk.

However, it is known that the total number of gradient evaluations of SGD depends on the variance of the stochastic gradients and it is of sublinear convergence rate for the strongly convex and smooth problem (1.1), which is inferior to the full gradient method. As a result, many works along this line have been focusing on designing variants of SGD that can reduce the variance and improve the complexity. Some popular methods along this line are briefly summarized as follows. The stochastic average gradient (SAG) method proposed by Le Roux et al. [22] updates the iterates by

$$x_{t+1} = x_t - \frac{\eta_t}{n} \sum_{i=1}^{n} y_i^t, \tag{1.4}$$

where at each iteration a random training sample $i_t$ is chosen and $y_i^t$ is defined as

$$y_i^t = \begin{cases} \nabla f_i(x_t) & \text{if } i = i_t, \\ y_i^{t-1}, & \text{otherwise.} \end{cases}$$

It is shown in [22] that SAG converges linearly for strongly convex problems. The SAGA method proposed by Defazio et al. [7] is an improved version of SAG, and it does not require the strong convexity assumption. It is noted that SAG and SAGA need to store the latest gradients for the $n$ component functions $f_i$. The SDCA method proposed by Shalev-Shwartz and Zhang [24] also requires to store all the component gradients. The stochastic variance reduced gradient (SVRG) method proposed by Johnson and Zhang [10] is now widely used in the machine learning community for solving (1.1), because it achieves the variance reduction effect for SGD, and it does not need to store the $n$ component gradients.

As pointed out by Le Roux et al. [22], one important issue regarding to stochastic algorithms (SGD and its variants) that has not been fully addressed in the literature, is how to choose an appropriate step size $\eta_t$ while running the algorithm. In classical gradient descent method, the step size is usually obtained by employing line search techniques. However, line search is computationally prohibited in stochastic gradient methods because one only has sub-sampled information of function value and gradient. As a result, for SGD and its variants used in practice, people usually use a diminishing step size $\eta_t$, or use a best-tuned fixed step size. Neither of these two approaches can be efficient.

Some recent works that discuss the choice of step size in SGD are summarized as follows. AdaGrad [8] scales the gradient by the square root of the accumulated magnitudes of the gradients in the past iterations, but it still requires a fixed step size $\eta$. [22] suggests a line search technique on the component function $f_{i_k}(x)$ selected in each iteration, to estimate step size for SAG. [13] suggests performing line search for an estimated function, which is evaluated by a Gaussian process with samples $f_{i_t}(x_t)$. [14] suggests to generate the step sizes by a given function with an unknown parameter, and to use the online SGD to update this unknown parameter.

**Our contributions** in this paper are in several folds.

(i). We propose to use the Barzilai-Borwein (BB) method to compute the step size for SGD and SVRG. The two new methods are named as SGD-BB and SVRG-BB, respectively. The per-iteration computational cost of SGD-BB and SVRG-BB is almost the same as SGD and SVRG, respectively.

(ii). We prove the linear convergence of SVRG-BB for strongly convex functions. As a by-product, we show the linear convergence of SVRG with Option I (SVRG-I) proposed in [10]. Note that in [10] only convergence of SVRG with Option II (SVRG-II) was given, and the proof for SVRG-I has been missing in the literature. However, SVRG-I is numerically a better choice than SVRG-II, as demonstrated in [10].

(iii). We conduct numerical experiments for SGD-BB and SVRG-BB on solving logistic regression and SVM problems. The numerical results show that SGD-BB and SVRG-BB are comparable to and sometimes even better than SGD and SVRG with best-tuned step sizes. We also compare SGD-BB with some advanced SGD variants, and demonstrate that our method is superior.

The rest of this paper is organized as follows. In Section 2 we briefly introduce the BB method in the deterministic setting. In Section 3 we propose our SVRG-BB method, and prove its linear convergence for strongly convex functions. As a by-product, we also prove the linear convergence of SVRG-I. In Section 4 we propose our SGD-BB method. A smoothing technique is also implemented to improve the performance of SGD-BB. We conduct numerical experiments for SVRG-BB and SGD-BB in Section 5. Finally, we draw some conclusions in Section 6.

## 2 The Barzilai-Borwein Step Size

The BB method, proposed by Barzilai and Borwein in [3], has been proved to be very successful in solving nonlinear optimization problems. The key idea behind the BB method is motivated by quasi-Newton methods. Suppose we want to solve the unconstrained minimization problem

$$\min_x \; f(x), \tag{2.1}$$

where $f$ is differentiable. A typical iteration of quasi-Newton methods for solving (2.1) takes the following form:

$$x_{t+1} = x_t - B_t^{-1}\nabla f(x_t), \tag{2.2}$$

where $B_t$ is an approximation of the Hessian matrix of $f$ at the current iterate $x_t$. Different choices of $B_t$ give different quasi-Newton methods. The most important feature of $B_t$ is that it must satisfy the so-called secant equation:

$$B_t s_t = y_t, \tag{2.3}$$

where $s_t = x_t - x_{t-1}$ and $y_t = \nabla f(x_t) - \nabla f(x_{t-1})$ for $t \geq 1$. It is noted that in (2.2) one needs to solve a linear system, which may be time consuming when $B_t$ is large and dense. One way to alleviate this burden is to use the BB method, which replaces $B_t$ by a scalar matrix $\frac{1}{\eta_t}I$. However, one cannot choose a scalar $\eta_t$ such that the secant equation (2.3) holds with $B_t = \frac{1}{\eta_t}I$. Instead, one can find $\eta_t$ such that the residual of the secant equation is minimized, i.e.,

$$\min_{\eta_t} \left\| \frac{1}{\eta_t} s_t - y_t \right\|_2^2,$$

which leads to the following choice of $\eta_t$:

$$\eta_t = \frac{\|s_t\|_2^2}{s_t^\top y_t}. \tag{2.4}$$

Therefore, a typical iteration of the BB method for solving (2.1) is

$$x_{t+1} = x_t - \eta_t \nabla f(x_t), \tag{2.5}$$

where $\eta_t$ is computed by (2.4).

**Remark 2.1.** *Another choice of $\eta_t$ is obtained by solving*

$$\min_{\eta_t} \|s_t - \eta_t y_t\|_2^2,$$

*which leads to*

$$\eta_t = \frac{s_t^\top y_t}{\|y_t\|_2^2}. \tag{2.6}$$

*In this paper, we will focus on the choice in* (2.4)*, because the practical performance of* (2.4) *and* (2.6) *are similar.*

For convergence analysis, generalizations and variants of the BB method, we refer the interested readers to [19, 20, 9, 5, 6, 4] and references therein. Recently, BB method has been successfully applied for solving problems arising from emerging applications, such as compressed sensing [28], sparse reconstruction [27] and image processing [26].

## 3 Barzilai-Borwein Step Size for SVRG

We see from (2.5) and (2.4) that the BB method does not need any parameter and the step size is computed while running the algorithm. This has been the main motivation for us to work out a black-box stochastic gradient descent method that can compute the step size automatically without requiring any parameters. In this section, we propose to incorporate the BB step size to SVRG which leads to the SVRG-BB method.

The following assumption is made throughout this section.

**Assumption 3.1.** *We assume that* (1.3) *holds for any $x_t$. We assume that the objective function $F(x)$ is $\mu$-strongly convex, i.e.,*

$$F(y) \geq F(x) + \nabla F(x)^\top (y-x) + \frac{\mu}{2}\|x-y\|_2^2, \quad \forall x, y \in \mathbb{R}^d.$$

*We also assume that the gradient of each component function $f_i(x)$ is $L$-Lipschitz continuous, i.e.,*

$$\|\nabla f_i(x) - \nabla f_i(y)\|_2 \leq L\|x-y\|_2, \ \forall x, y \in \mathbb{R}^d.$$

*Under this assumption, it is easy to see that $\nabla F(x)$ is also $L$-Lipschitz continuous:*

$$\|\nabla F(x) - \nabla F(y)\|_2 \leq L\|x-y\|_2, \ \forall x, y \in \mathbb{R}^d.$$

### 3.1 SVRG Method

The SVRG method proposed by Johnson and Zhang [10] for solving (1.1) is described as in Algorithm 1.

**Algorithm 1** Stochastic Variance Reduced Gradient (SVRG) Method

```
Parameters: update frequency $m$, step size $\eta$, initial point $\tilde{x}_0$
for $k = 0, 1, \cdots$ do
  $g_k = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x}_k)$
  $x_0 = \tilde{x}_k$
  $\eta_k = \eta$
  for $t = 0, \cdots, m-1$ do
    Randomly pick $i_t \in \{1, \ldots, n\}$
    $x_{t+1} = x_t - \eta_k(\nabla f_{i_t}(x_t) - \nabla f_{i_t}(\tilde{x}_k) + g_k)$
  end for
  Option I: $\tilde{x}_{k+1} = x_m$
  Option II: $\tilde{x}_{k+1} = x_t$ for randomly chosen $t \in \{1, \ldots, m\}$
end for
```

There are two loops in SVRG (Algorithm 1). In the outer loop (each outer iteration is called an *epoch*), a full gradient $g_k$ is computed, which is used in the inner loop for generating stochastic gradients with lower variance. $\tilde{x}$ is then chosen, based on the outputs of inner loop, for the next outer loop. Note that two options for choosing $\tilde{x}$ are suggested in SVRG. Intuitively, Option I in SVRG (denoted as SVRG-I) is a better choice than Option II (denoted as SVRG-II), because the former used the latest information from the inner loop. This has been confirmed numerically in [10] where SVRG-I was applied to solve real applications. However, the convergence analysis is only available for SVRG-II (see, e.g., [10], [12] and [2]), and the convergence for SVRG-I has been missing in the literature. We now cite the convergence analysis of SVRG-II given in [10] as follows.

**Theorem 3.2** ([10])**.** *Consider SVRG in Algorithm 1 with Optioin II. Let $x^*$ be the optimal solution to problem* (1.1)*. Assume that $m$ is sufficiently large so that*

$$\alpha := \frac{1}{\mu\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta} < 1, \tag{3.1}$$

*then we have linear convergence in expectation for SVRG:*

$$\mathbb{E}\left[F(\tilde{x}_k) - F(x^*)\right] \le \alpha^k [F(\tilde{x}_0) - F(x^*)].$$

There has been a series of follow-up works on SVRG and its variants. Xiao and Zhang [29] developed a proximal SVRG method for minimizing the finite sum function plus a nonsmooth regularizer. [17] applied Nesterov's acceleration technique to SVRG to improve the convergence rate that depends on the condition number $L/\mu$. [2] proved if the full gradient computation $g_k$ was replaced by a growing-batch estimation, the linear convergence rate can be preserved. [1] and [21] showed that SVRG with minor modifications can converge to a stationary point for nonconvex optimization problems.

## 3.2 SVRG-BB Method

It is noted that in SVRG, the step size $\eta$ needs to be provided by the user. According to (3.1), the choice of $\eta$ is dependent on $L$, which may be difficult to estimate in practice. In this section, we propose the SVRG-BB method that computes the step size using the BB method. Our SVRG-BB algorithm is described in Algorithm 2. Note that the only difference between SVRG and SVRG-BB is that in the latter we use BB method to compute the step size $\eta_k$, instead of using a prefixed $\eta$ as in SVRG.

**Remark 3.3.** *A few remarks are in demand for the SVRG-BB algorithm.*

1. *One may notice that $\eta_k$ is equal to the step size computed by the BB formula* (2.4) *divided by $m$. This is because in the inner loop for updating $x_t$, $m$ unbiased gradient estimators are added to $x_0$ to get $x_m$.*

2. *If we always set $\eta_k = \eta$ in SVRG-BB instead of using* (3.2)*, then it reduces to SVRG-I.*

3. *For the first outer loop of SVRG-BB, a step size $\eta_0$ needs to be specified, because we are not able to compute the BB step size for the first outer loop. However, we observed from our numerical experiments that the performance of SVRG-BB is not sensitive to the choice of $\eta_0$.*

4. *The BB step size can also be naturally incorporated to other SVRG variants, such as SVRG with batching [2].*

**Algorithm 2** SVRG with BB step size (SVRG-BB)

```
**Parameters**: update frequency $m$, initial point $\tilde{x}_0$, initial step size $\eta_0$ (only used in the first epoch)
for $k = 0, 1, \cdots$ do
  $g_k = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x}_k)$
  if $k > 0$ then
      $\eta_k = \frac{1}{m} \cdot \|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2 / (\tilde{x}_k - \tilde{x}_{k-1})^\top (g_k - g_{k-1})$    (3.2)
  end if
  $x_0 = \tilde{x}_k$
  for $t = 0, \cdots, m-1$ do
    Randomly pick $i_t \in \{1, \dots, n\}$
    $x_{t+1} = x_t - \eta_k(\nabla f_{i_t}(x_t) - \nabla f_{i_t}(\tilde{x}_k) + g_k)$
  end for
  $\tilde{x}_{k+1} = x_m$
end for
```

## 3.3 Linear Convergence Analysis

In this section, we analyze the linear convergence of SVRG-BB (Algorithm 2) for solving (1.1) with strongly convex objective $F(x)$, and as a by-product, our analysis also proves the linear convergence of SVRG-I.

The following lemma, which is from [16], is useful in our analysis.

**Lemma 3.4** (co-coercivity)**.** *If $f(x) : \mathbb{R}^d \to \mathbb{R}$ is convex and its gradient is $L$-Lipschitz continuous, then*

$$\|\nabla f(x) - \nabla f(y)\|_2^2 \le L(x-y)^\top(\nabla f(x) - \nabla f(y)), \quad \forall x, y \in \mathbb{R}^d.$$

In the following, we first prove the following lemma, which reveals the relationship between the distances of two consecutive iterates to the optimal point.

**Lemma 3.5.** *Define*

$$\alpha_k := (1 - 2\eta_k\mu(1-\eta_k L))^m + \frac{4\eta_k L^2}{\mu(1-\eta_k L)}. \tag{3.3}$$

*For both SVRG-I and SVRG-BB, we have the following inequality for the $k$-th epoch:*

$$\mathbb{E}\,\|\tilde{x}_{k+1} - x^*\|_2^2 < \alpha_k \|\tilde{x}_k - x^*\|_2^2,$$

*where $x^*$ is the optimal solution to* (1.1)*.*

*Proof.* Let $v_{i_t}^t = \nabla f_{i_t}(x_t) - \nabla f_{i_t}(\tilde{x}_k) + \nabla F(\tilde{x}_k)$ for the $k$-th epoch of SVRG-I or SVRG-BB. Then,

$$\begin{aligned}
\mathbb{E}\|v_{i_t}^t\|_2^2 =& \mathbb{E}\,\|(\nabla f_{i_t}(x_t) - \nabla f_{i_t}(x^*)) - (\nabla f_{i_t}(\tilde{x}_k) - \nabla f_{i_t}(x^*)) + \nabla F(\tilde{x}_k)\|_2^2 \\
\le& 2\mathbb{E}\,\|\nabla f_{i_t}(x_t) - \nabla f_{i_t}(x^*)\|_2^2 + 4\mathbb{E}\,\|\nabla f_{i_t}(\tilde{x}_k) - \nabla f_{i_t}(x^*)\|_2^2 + 4\|\nabla F(\tilde{x}_k)\|_2^2 \\
\le& 2L\mathbb{E}\left[(x_t - x^*)^\top(\nabla f_i(x_t) - \nabla f_i(x^*))\right] + 4L^2\|\tilde{x}_k - x^*\|_2^2 + 4L^2\|\tilde{x}_k - x^*\|_2^2 \\
=& 2L(x_t - x^*)^\top \nabla F(x_t) + 8L^2\|\tilde{x}_k - x^*\|_2^2,
\end{aligned}$$

where in the first inequality we used the inequality $(a-b)^2 \le 2a^2 + 2b^2$ twice, in the second inequality we applied Lemma 3.4 to $f_{i_t}(x)$ and used the Lipschitz continuity of $\nabla f_{i_t}$ and $\nabla F$, and in the last equality we used the facts that $\mathbb{E}[\nabla f_{i_t}(x)] = \nabla F(x)$ and $\nabla F(x^*) = 0$.

In the next, we bound the distance of $x_{t+1}$ to $x^*$ conditioned on $x_t$ and $\tilde{x}_k$.

$$
\begin{aligned}
&\mathbb{E}\|x_{t+1} - x^*\|_2^2 \\
=&\mathbb{E}\|x_t - \eta_k v_{i_t}^t - x^*\|_2^2 \\
=&\|x_t - x^*\|_2^2 - 2\eta_k \mathbb{E}[(x_t - x^*)^\top v_{i_t}^t] + \eta_k^2 \mathbb{E}\|v_{i_t}^t\|_2^2 \\
=&\|x_t - x^*\|_2^2 - 2\eta_k (x_t - x^*)^\top \nabla F(x_t) + \eta_k^2 \mathbb{E}\|v_{i_t}^t\|_2^2 \\
\leq&\|x_t - x^*\|_2^2 - 2\eta_k (x_t - x^*)^\top \nabla F(x_t) + 2\eta_k^2 L (x_t - x^*)^\top \nabla F(x_t) + 8\eta_k^2 L^2 \|\tilde{x}_k - x^*\|_2^2 \\
=&\|x_t - x^*\|_2^2 - 2\eta_k (1 - \eta_k L)(x_t - x^*)^\top \nabla F(x_t) + 8\eta_k^2 L^2 \|\tilde{x}_k - x^*\|_2^2 \\
\leq&\|x_t - x^*\|_2^2 - 2\eta_k \mu (1 - \eta L)\|x_t - x^*\|^2 + 8\eta_k^2 L^2 \|\tilde{x}_k - x^*\|_2^2 \\
=&[1 - 2\eta_k \mu (1 - \eta_k L)]\|x_t - x^*\|_2^2 + 8\eta_k^2 L^2 \|\tilde{x}_k - x^*\|_2^2,
\end{aligned}
$$

where in the third equality we used the fact that $\mathbb{E}[v_{i_t}^t] = \nabla F(x_t)$, and in the second inequality we used the strong convexity of $F(x)$.

By recursively applying the above inequality over $t$, and noting that $\tilde{x}_k = x_0$ and $\tilde{x}_{k+1} = x_m$, we can obtain

$$
\begin{aligned}
&\mathbb{E}\|\tilde{x}_{k+1} - x^*\|_2^2 \\
\leq& [1 - 2\eta_k \mu (1 - \eta L)]^m \|\tilde{x}_k - x^*\|_2^2 + 8\eta_k^2 L^2 \sum_{j=0}^{m-1} [1 - 2\eta_k \mu (1 - \eta L)]^j \|\tilde{x}_k - x^*\|_2^2 \\
<& \left[(1 - 2\eta_k \mu (1 - \eta L))^m + \frac{4\eta_k L^2}{\mu (1 - \eta_k L)}\right] \|\tilde{x}_k - x^*\|_2^2 \\
=&\alpha_k \|\tilde{x}_k - x^*\|_2^2.
\end{aligned}
$$

□

The linear convergence of SVRG-I follows immediately.

**Corollary 3.6.** *In SVRG-I, if $m$ and $\eta$ are chosen such that*

$$
\alpha := (1 - 2\eta\mu(1 - \eta L))^m + \frac{4\eta L^2}{\mu(1 - \eta L)} < 1, \tag{3.4}
$$

*then SVRG-I (Algorithm 1 with Option I) converges linearly in expectation:*

$$
\mathbb{E}\,\|\tilde{x}_k - x^*\|_2^2 < \alpha^k \|\tilde{x}_0 - x^*\|_2^2.
$$

**Remark 3.7.** *We now give some remarks on this convergence result.*

1. *To the best of our knowledge, this is the first time that the linear convergence of SVRG-I is established.*

2. *The condition required in* (3.3) *is different from the condition required in* (3.1) *for SVRG-II. As $m \to +\infty$, the first term in* (3.1) *converges to 0 sublinearly, while the first term in* (3.3) *converges to 0 linearly. On the other hand, the second term in* (3.1) *reveals that $m$ depends on the condition number $L/\mu$ linearly, while the second term in* (3.3) *suggests that $m$ depends on condition number $L/\mu$ quadratically. As a result, if the problem is ill-conditioned, then the convergence rate given in Corollary 3.6 might be slow.*

3. *The convergence result given in Corollary 3.6 is for the iterates $\tilde{x}_k$, while the one given in Theorem 3.2 is for the objective function values $F(\tilde{x}_k)$.*

The following theorem establishes the linear convergence of SVRG-BB (Algorithm 2).

**Theorem 3.8.** *Denote $\theta = (1 - e^{-2\mu/L})/2$. It is easy to see that $\theta \in (0, 1/2)$. In SVRG-BB, if $m$ is chosen such that*

$$m > \max\left\{\frac{2}{\log(1-2\theta) + 2\mu/L}, \frac{4L^2}{\theta\mu^2} + \frac{L}{\mu}\right\}, \tag{3.5}$$

*then SVRG-BB (Algorithm 2) converges linearly in expectation:*

$$\mathbb{E}\,\|\tilde{x}_k - x^*\|_2^2 < (1-\theta)^k \|\tilde{x}_0 - x^*\|_2^2.$$

*Proof.* Using the strong convexity of function $F(x)$, it is easy to obtain the following upper bound for the BB step size computed in Algorithm 2.

$$\begin{aligned}\eta_k &= \frac{1}{m}\cdot\frac{\|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2}{(\tilde{x}_k - \tilde{x}_{k-1})^\top (g_k - g_{k-1})} \\ &\le \frac{1}{m}\cdot\frac{\|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2}{\mu\|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2} = \frac{1}{m\mu}.\end{aligned}$$

Similarly, by the $L$-Lipschitz continuity of $\nabla F(x)$, it is easy to obtain that $\eta_k$ is uniformly lower bounded by $1/(mL)$. Therefore, $\alpha_k$ in (3.3) can be bounded as:

$$\begin{aligned}\alpha_k &\le \left[1 - \frac{2\mu}{mL}\left(1 - \frac{L}{m\mu}\right)\right]^m + \frac{4L^2}{m\mu^2[1 - L/(m\mu)]} \\ &\le \exp\left\{-\frac{2\mu}{mL}\left(1 - \frac{L}{m\mu}\right)\cdot m\right\} + \frac{4L^2}{m\mu^2[1 - L/(m\mu)]} \\ &= \exp\left\{-\frac{2\mu}{L} + \frac{2}{m}\right\} + \frac{4L^2}{m\mu^2 - L\mu},\end{aligned}$$

Substituting (3.5) into the above inequality yields

$$\alpha_k < \exp\left\{-\frac{2\mu}{L} + \log(1-2\theta) + \frac{2\mu}{L}\right\} + \frac{4L^2}{4L^2/\theta + L\mu - L\mu} = (1-2\theta) + \theta = 1 - \theta.$$

The desired result follows by applying Lemma 3.5. □

## 4 Barzilai-Borwein Step Size for SGD

In this section, we propose to incorporate the BB method to SGD (1.2). The BB method does not apply to SGD directly, because SGD never computes the full gradient $\nabla F(x)$. In SGD, $\nabla f_{i_t}(x_t)$ is an unbiased estimation for $\nabla F(x_t)$ when $i_t$ is uniformly sampled (see [15, 30] for studies on importance sampling, which does not sample $i_t$ uniformly). Therefore, one may suggest to use $\nabla f_{i_{t+1}}(x_{t+1}) - \nabla f_{i_t}(x_t)$ to estimate $\nabla F(x_{t+1}) - \nabla F(x_t)$ when computing the BB step size using formula (2.4). However, this approach does not work well because of the variance of the stochastic gradient estimates. The recent work by Sopyła and Drozda [25] suggested several variants of this idea to compute an estimated BB step size using the stochastic gradients. However, these ideas lack theoretical justifications and the numerical results in [25] show that these approaches are inferior to existing methods such as averaged SGD [18].

The SGD-BB algorithm we propose in this paper works in the following manner. We call every $m$ iterations of SGD as one epoch. Following the idea of SVRG-BB, SGD-BB also uses the same step size computed by the BB formula in every epoch. Our SGD-BB algorithm is described as in Algorithm 3.

**Remark 4.1.** *We have a few remarks about SGD-BB (Algorithm 3).*

**Algorithm 3** SGD with BB step size (SGD-BB)

```
Parameters: update frequency $m$, initial step sizes $\eta_0$ and $\eta_1$ (only used in the first two epochs), weighting parameter $\beta \in (0,1)$, initial point $\tilde{x}_0$
for $k = 0, 1, \cdots$ do
  if $k > 0$ then
    $\eta_k = \frac{1}{m} \cdot \|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2 / |(\tilde{x}_k - \tilde{x}_{k-1})^\top (\hat{g}_k - \hat{g}_{k-1})|$
  end if
  $x_0 = \tilde{x}_k$
  $\hat{g}_{k+1} = 0$
  for $t = 0, \cdots, m-1$ do
    Randomly pick $i_t \in \{1, \ldots, n\}$
    $x_{t+1} = x_t - \eta_k \nabla f_{i_t}(x_t)$                                   $(*)$
    $\hat{g}_{k+1} = \beta \nabla f_{i_t}(x_t) + (1-\beta)\hat{g}_{k+1}$
  end for
  $\tilde{x}_{k+1} = x_m$
end for
```

1. *SGD-BB takes the average of the stochastic gradients in one epoch as an estimation of the full gradient.*

2. *Note that for computing $\eta_k$ in Algorithm 3, we actually take the absolute value for the BB formula* (2.4). *This is because that unlike SVRG-BB, $\hat{g}_k$ in Algorithm 3 is the average of $m$ stochastic gradients at different iterates, not an exact full gradient. As a result, the step size generated by* (2.4) *can be negative. This can be seen from the following argument. Suppose $\beta$ is chosen such that*

$$\hat{g}_k = \frac{1}{m} \sum_{t=0}^{m-1} \nabla f_{i_t}(x_t), \tag{4.1}$$

*where we use the same notation as in Algorithm 2 and $x_t$ $(t = 0, 1, \ldots, m-1)$ denote the iterates in the $(k-1)$-st epoch. From* (4.1), *it is easy to see that*

$$\tilde{x}_k - \tilde{x}_{k-1} = -m\eta_{k-1}\hat{g}_k.$$

*By substituting this equality into the equation for computing $\eta_k$ in Algorithm 3, we have*

$$\begin{aligned} \eta_k =& \frac{1}{m} \cdot \frac{\|\tilde{x}_k - \tilde{x}_{k-1}\|^2}{|(\tilde{x}_k - \tilde{x}_{k-1})^\top (\hat{g}_k - \hat{g}_{k-1})|} \\ =& \frac{1}{m} \cdot \frac{\|-m\eta_{k-1}\hat{g}_k\|^2}{|(-m\eta_{k-1}\hat{g}_k)^\top (\hat{g}_k - \hat{g}_{k-1})|} \\ =& \frac{\eta_{k-1}}{\left|1 - \hat{g}_k^\top \hat{g}_{k-1} / \|\hat{g}_k\|_2^2\right|}. \end{aligned} \tag{4.2}$$

*Without taking the absolute value, the denominator of* (4.2) *is $\hat{g}_k^\top \hat{g}_{k-1} / \|\hat{g}_k\|_2^2 - 1$, which can be negative in stochastic settings.*

3. *Moreover, from* (4.2) *we have the following observations. If $\hat{g}_k^\top \hat{g}_{k-1} < 0$, then $\eta_k$ is smaller than $\eta_{k-1}$. This is reasonable because $\hat{g}_k^\top \hat{g}_{k-1} < 0$ indicates that the step size is too large and we need to shrink it. If $\hat{g}_k^\top \hat{g}_{k-1} > 0$, then it indicates that we should be more aggressive to take larger step size. We found from our numerical experiments that when the iterates are close to optimum, the size of $\hat{g}_k$ and $\hat{g}_{k-1}$ do not differentiate much. As a result, $\eta_k$ is usually increased from $\eta_{k-1}$ by using* (4.2). *Hence, the way we compute $\eta_k$ in Algorithm 3 is in a sense to dynamically adjust the step size, by evaluating whether we are moving the iterates along the right direction. This kind of idea can be traced back to [11].*

4. *Furthermore, in order to make sure the averaged stochastic gradients $\hat{g}_k$ in* (4.1) *is close to $\nabla F(\tilde{x}_k)$, it is natural to emphasize more on the latest sample gradients. Therefore, in Algorithm 3 we update $\hat{g}_k$ recursively using*

$$\hat{g}_{k+1} = \beta \nabla f_{i_t}(x_t) + (1-\beta)\hat{g}_{k+1},$$

*starting from $\hat{g}_{k+1} = 0$, where $\beta \in (0,1)$ is a weighting parameter.*

Note that SGD-BB requires the averaged gradients in two epochs to compute the BB step size, which can only be done starting from the third epoch. Therefore, we need to specify the step sizes $\eta_0$ and $\eta_1$ for the first two epochs. From our numerical experiments, we found that the performance of SGD-BB is not sensitive to choices of $\eta_0$ and $\eta_1$.

## 4.1 Smoothing Technique for the Step Sizes

Due to the randomness of the stochastic gradients, the step size computed in SGD-BB may vibrate drastically sometimes and this may cause instability of the algorithm. Inspired by [14], we propose the following smoothing technique to stabilize the step size.

It is known that in order to guarantee the convergence of SGD, the step sizes are required to be diminishing. Similar as in [14], we assume the step sizes are in the form of $C/\phi(k)$, where $C > 0$ is an unknown constant that needs to be estimated, $\phi(k)$ is a pre-specified function that controls the decreasing rate of the step size, and a typical choice of function $\phi$ is $\phi(k) = k+1$. In the $k$-th epoch of Algorithm 3, we have all the previous step sizes $\eta_2, \eta_3, \ldots, \eta_k$ generated by the BB method, while the step sizes generated by the function $C/\phi(k)$ are given by $C/\phi(2), C/\phi(3), \ldots, C/\phi(k)$. In order to ensure that these two sets of step sizes are close to each other, we solve the following optimization problem to determine the unknown parameter $C$:

$$\hat{C}_k := \operatorname*{argmin}_C \sum_{j=2}^{k} \left[\log \frac{C}{\phi(j)} - \log \eta_j\right]^2. \tag{4.3}$$

Here we take the logarithms of the step sizes to ensure that the estimation is not dominated by those $\eta_j$'s with large magnitudes. It is easy to verify that the solution to (4.3) is given by

$$\hat{C}_k = \prod_{j=2}^{k} \left[\eta_j \phi(j)\right]^{1/(k-1)}.$$

Therefore, the smoothed step size for the $k$-th epoch of Algorithm 3 is:

$$\tilde{\eta}_k = \hat{C}_k/\phi(k) = \prod_{j=2}^{k} \left[\eta_j \phi(j)\right]^{1/(k-1)} / \phi(k). \tag{4.4}$$

That is, we replace the $\eta_k$ in equation $(*)$ of Algorithm 3 by $\tilde{\eta}_k$ in (4.4).

In practice, we do not need to store all the $\eta_j$'s and $\hat{C}_k$ can be computed recursively by

$$\hat{C}_k = \hat{C}_{k-1}^{(k-2)/(k-1)} \cdot \left[\eta_k \phi(k)\right]^{1/(k-1)}.$$

## 4.2 Incorporating BB Step Size to SGD Variants

The BB step size and the smoothing technique we used in SGD-BB (Algorithm 3) can also be used in other variants of SGD. In this section, we use SAG as an example to illustrate how to incorporate the BB step size. SAG with BB step size (denoted as SAG-BB) is described as in Algorithm 4. Because SAG does not need diminishing step sizes to ensure convergence, in the smoothing technique we just choose $\phi(k) \equiv 1$. In this case, the smoothed step size $\tilde{\eta}_k$ is equal to the geometric mean of all previous BB step sizes.

**Algorithm 4** SAG with BB step size (SAG-BB)

```
Parameters: update frequency $m$, initial step sizes $\eta_0$ and $\eta_1$ (only used in the first two epochs), weighting parameter $\beta \in (0,1)$, initial point $\tilde{x}_0$
$y_i = 0$ for $i = 1, \ldots, n$
for $k = 0, 1, \cdots$ do
  if $k > 0$ then
    $\eta_k = \frac{1}{m} \cdot \|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2 / |(\tilde{x}_k - \tilde{x}_{k-1})^\top (\hat{g}_k - \hat{g}_{k-1})|$
    $\tilde{\eta}_k = \left(\prod_{j=2}^{k} \eta_j\right)^{\frac{1}{k-1}}$          ▷ smoothing technique
  end if
  $x_0 = \tilde{x}_k$
  $\hat{g}_{k+1} = 0$
  for $t = 0, \cdots, m-1$ do
    Randomly pick $i_t \in \{1, \ldots, n\}$
    $y_{i_t} = \nabla f_{i_t}(x_t)$
    $x_{t+1} = x_t - \frac{\eta_k}{n} \sum_{i=1}^{n} y_i$          ▷ SAG update
    $\hat{g}_{k+1} = \beta \nabla f_{i_t}(x_t) + (1-\beta)\hat{g}_{k+1}$
  end for
  $\tilde{x}_{k+1} = x_m$
end for
```

## 5 Numerical Experiments

In this section, we conduct some numerical experiments to demonstrate the efficacy of our SVRG-BB (Algorithm 2) and SGD-BB (Algorithm 3) algorithms. In particular, we apply SVRG-BB and SGD-BB to solve two standard testing problems in machine learning: logistic regression with $\ell_2$-norm regularization

$$(LR) \qquad \min_x \; F(x) = \frac{1}{n}\sum_{i=1}^{n} \log\left[1 + \exp(-b_i a_i^\top x)\right] + \frac{\lambda}{2}\|x\|_2^2, \tag{5.1}$$

and the squared hinge loss SVM with $\ell_2$-norm regularization

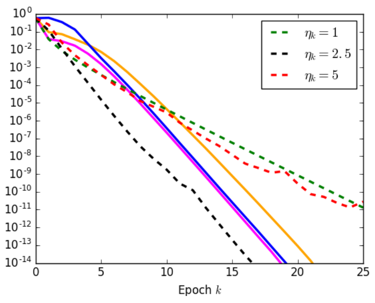
$$(SVM) \qquad \min_x \; F(x) = \frac{1}{n}\sum_{i=1}^{n} \left([1 - b_i a_i^\top x]_+\right)^2 + \frac{\lambda}{2}\|x\|_2^2, \tag{5.2}$$

where $a_i \in \mathbb{R}^d$ and $b_i \in \{\pm 1\}$ are the feature vector and class label of the $i$-th sample, respectively, and $\lambda > 0$ is a weighting parameter.

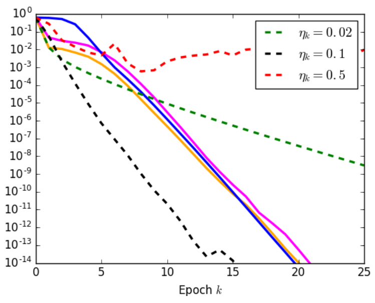
We tested SVRG-BB and SGD-BB for (5.1) and (5.2) for three standard real data sets, which were downloaded from the LIBSVM website[1]. Detailed information of these three data sets are given in Table 1.

Table 1: Data and model information of the experiments

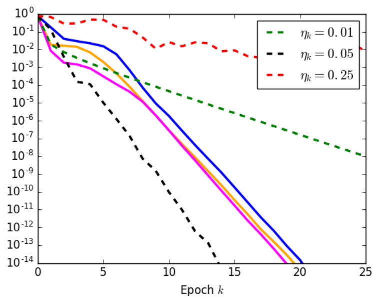
| Dataset | $n$ | $d$ | model | $\lambda$ |
|---|---|---|---|---|
| rcv1.binary | 20,242 | 47,236 | *LR* | $10^{-5}$ |
| w8a | 49,749 | 300 | *LR* | $10^{-4}$ |
| ijcnn1 | 49,990 | 22 | *SVM* | $10^{-4}$ |



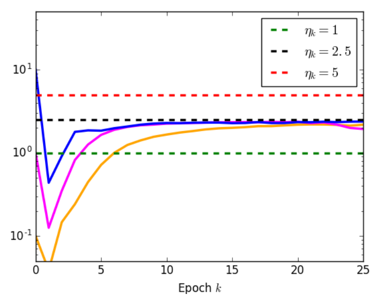
(a) Sub-optimality on rcv1.binary



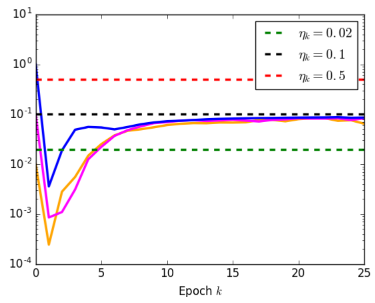
(b) Sub-optimality on w8a



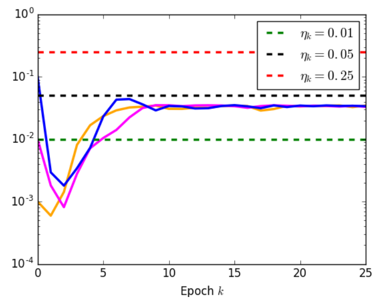
(c) Sub-optimality on ijcnn1



(d) Step sizes on rcv1.binary



(e) Step sizes on w8a



(f) Step sizes on ijcnn1

Figure 1: Comparison of SVRG-BB and SVRG with fixed step sizes on different problems. The dashed lines stand for SVRG with different fixed step sizes $\eta_k$ given in the legend. The solid lines stand for SVRG-BB with different $\eta_0$; for example, the solid lines in Sub-figures (a) and (d) correspond to SVRG-BB with $\eta_0 = 10, 1, 0.1$, respectively.

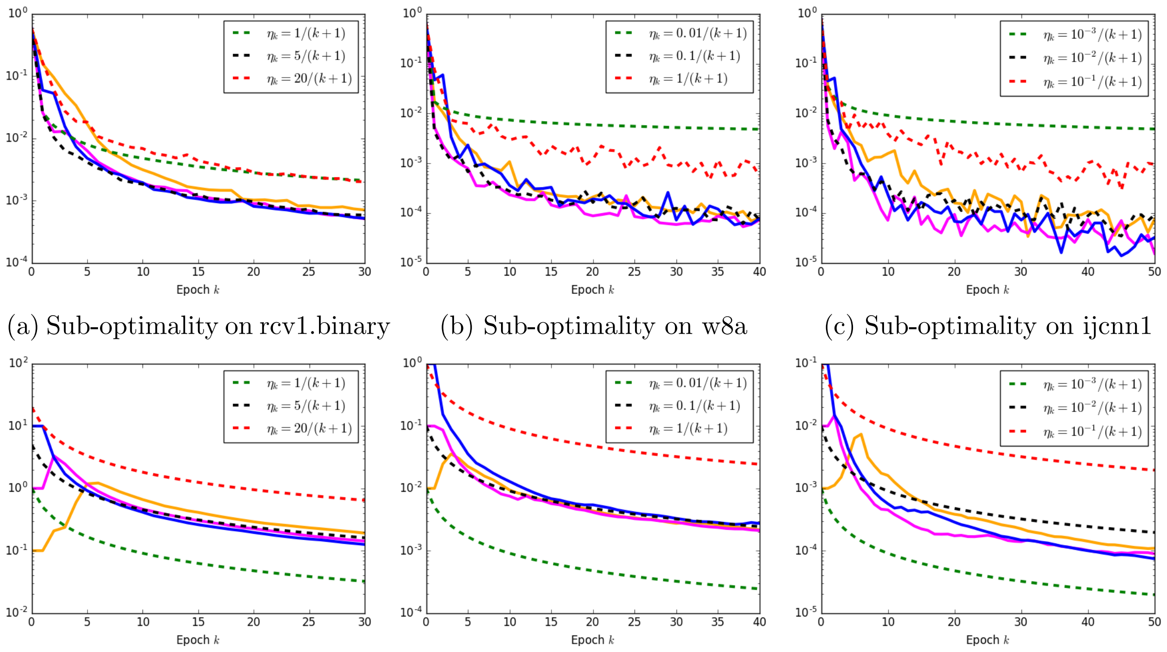
## 5.1 Numerical Results of SVRG-BB

In this section, we compare SVRG-BB (Algorithm 2) with SVRG (Algorithm 1) for solving (5.1) and (5.2). We used the best-tuned step size for SVRG, and three different initial step sizes $\eta_0$ for SVRG-BB. For both SVRG-BB and SVRG, we set $m = 2n$ as suggested in [10].

The comparison results of SVRG-BB and SVRG are shown in Figure 1. In all the six sub-figures, the $x$-axis denotes the number of epochs $k$, i.e., the number of outer loops in Algorithm 2. In Figures 1(a), 1(b) and 1(c), the $y$-axis denotes the sub-optimality $F(\tilde{x}_k) - F(x^*)$, and in Figures 1(d), 1(e) and 1(f), the $y$-axis denotes the step size $\eta_k$. Note that $x^*$ is obtained by running SVRG with the best-tuned step size until it converges, which is a common practice in the testing of stochastic gradient descent methods. In all the six sub-figures, the dashed lines correspond to SVRG with fixed step sizes given in the legends of the figures. Moreover, the dashed lines in black color always represent SVRG with best-tuned fixed step size, and the green dashed lines use a smaller fixed step size, and the red dashed lines use a larger fixed step size, compared with the best-tuned ones. The solid lines correspond to SVRG-BB with different initial step sizes $\eta_0$. The solid lines with blue, purple and yellow colors in Figures 1(a) and 1(d) correspond to $\eta_0 = 10$, 1, and 0.1, respectively; the solid lines with blue, purple and yellow colors in Figures 1(b) and 1(e) correspond to $\eta_0 = 1$, 0.1, and 0.01, respectively; the solid lines with blue, purple and yellow colors in Figures 1(c) and 1(f) correspond to $\eta_0 = 0.1$, 0.01, and 0.001, respectively.

It can be seen from Figures 1(a), 1(b) and 1(c) that, SVRG-BB can always achieve the same level of sub-optimality as SVRG with the best-tuned step size. Although SVRG-BB needs slightly more epochs compared with SVRG with the best-tuned step size, it clearly outperforms SVRG with the other two choices of step sizes. Moreover, from Figures 1(d), 1(e) and 1(f) we see that the step sizes computed by SVRG-BB converge to the best-tuned step sizes after about

[1]`www.csie.ntu.edu.tw/~cjlin/libsvmtools/`.



(a) Sub-optimality on rcv1.binary (b) Sub-optimality on w8a (c) Sub-optimality on ijcnn1

(d) Step sizes on rcv1.binary (e) Step sizes on w8a (f) Step sizes on ijcnn1

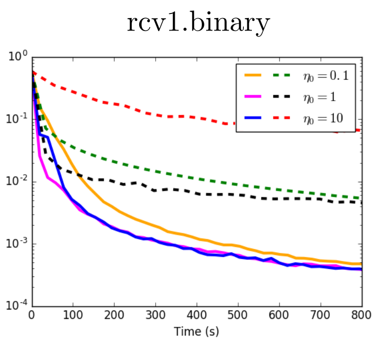
Figure 2: Comparison of SGD-BB and SGD. The dashed lines correspond to SGD with diminishing step sizes in the form $\eta/(k+1)$ with different constants $\eta$. The solid lines stand for SGD-BB with different initial step sizes $\eta_0$; for example, the solid lines in Sub-figure (a) and (d) correspond to SGD-BB with $\eta_0 = 10, 1, 0.1$, respectively.

10 to 15 epochs. From Figure 1 we also see that SVRG-BB is not sensitive to the choice of $\eta_0$. Therefore, SVRG-BB has very promising potential in practice because it generates the best step sizes automatically while running the algorithm.

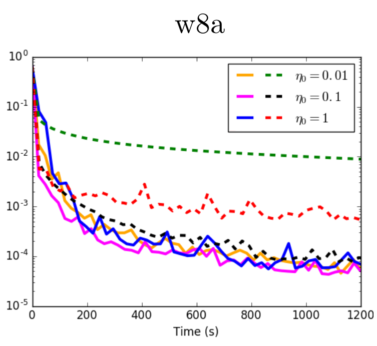
## 5.2 Numerical Results of SGD-BB

In this section, we compare SGD-BB with smoothing technique (Algorithm 3) with SGD for solving (5.1) and (5.2). We set $m = n$, $\beta = 10/m$ and $\eta_1 = \eta_0$ in our experiments. We used $\phi(k) = k + 1$ when applying the smoothing technique. Since SGD requires diminishing step size to converge, we tested SGD with diminishing step size in the form $\eta/(k+1)$ with different constants $\eta$. The comparison results are shown in Figure 2. Similar as Figure 1, the dashed line with black color represents SGD with the best-tuned $\eta$, and the green and red dashed lines correspond to the other two choices of $\eta$. The solid lines with blue, purple and yellow colors in Figures 2(a) and 2(d) correspond to $\eta_0 = 10$, 1, and 0.1, respectively; the solid lines with blue, purple and yellow colors in Figures 2(b) and 2(e) correspond to $\eta_0 = 1$, 0.1, and 0.01, respectively; the solid lines with blue, purple and yellow colors in Figures 2(c) and 2(f) correspond to $\eta_0 = 0.1$, 0.01, and 0.001, respectively.

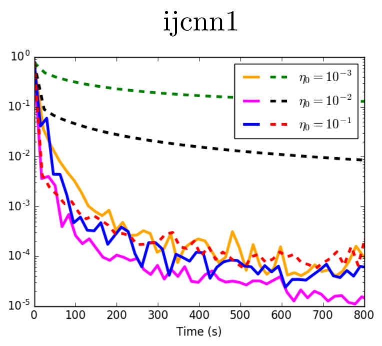
From Figures 2(a), 2(b) and 2(c) we can see that SGD-BB gives comparable or even better sub-optimality than SGD with best-tuned diminishing step size, and SGD-BB is significantly better than SGD with the other two choices of step size. From Figures 2(d), 2(e) and 2(f) we see that after only a few epochs, the step sizes generated by SGD-BB approximately coincide with the best-tuned diminishing step sizes. It can also be seen that after only a few epochs, the step sizes are stabilized by the smoothing technique and they approximately follow the same decreasing trend as the best-tuned diminishing step sizes.



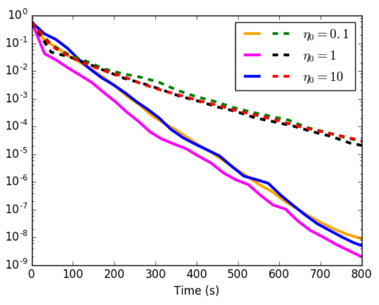
(a) AdaGrad versus SGD-BB



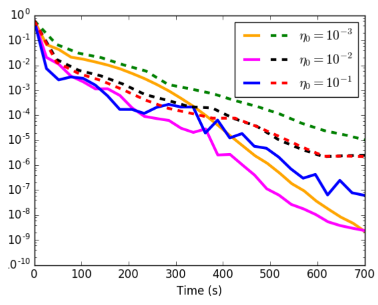
(b) AdaGrad versus SGD-BB



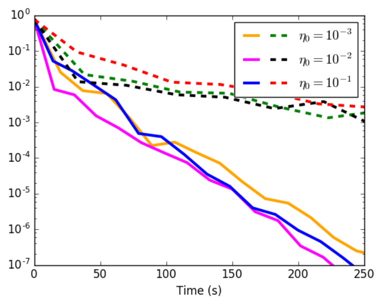
(c) AdaGrad versus SGD-BB



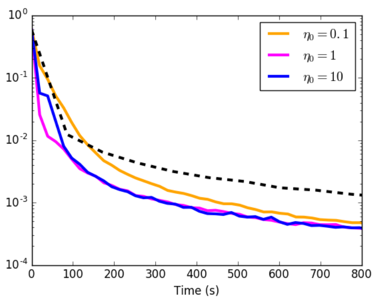
(d) SAG-L versus SAG-BB



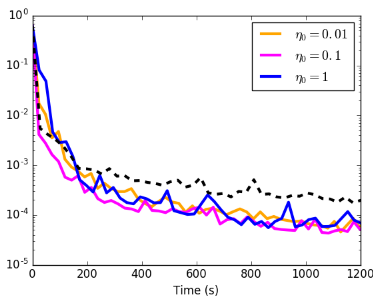
(e) SAG-L versus SAG-BB



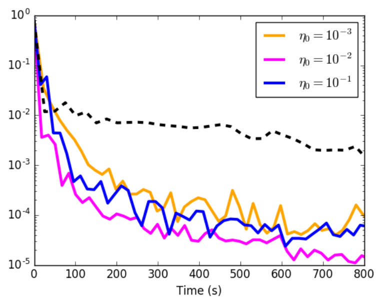
(f) SAG-L versus SAG-BB



(g) oLBFGS versus SGD-BB



(h) oLBFGS versus SGD-BB



(i) oLBFGS versus SGD-BB

Figure 3: Comparison of SGD-BB and SAG-BB with three existing methods. The $x$-axes all denote the CPU time (in seconds). The $y$-axes all denote the sub-optimality $F(x_k) - F(x^*)$. In the first row, solid lines stand for SGD-BB, and dashed lines stand for AdaGrad; In the second row, solid lines stand for SAG-BB, and dashed lines stand for SAG with line search; In the third row, solid lines stand for SGD-BB, and the dashed lines stand for oLBFGS.

## 5.3 Comparison with Other Methods

In this section, we compare our SGD-BB (Algorithm 3) and SAG-BB (Algorithm 4) with three existing methods: AdaGrad [8], SAG with line search (denoted as SAG-L) [22], and a stochastic quasi-Newton method: oLBFGS [23]. For both SGD-BB and SAG-BB, we set $m = n$ and $\beta = 10/m$. Because these methods have very different per-iteration complexity, we compare their CPU time needed to achieve the same sub-optimality.

Figures 3(a), 3(b) and 3(c) show the comparison results of SGD-BB and AdaGrad. From these figures we see that AdaGrad usually has a very quick start, but in many cases the convergence becomes slow in later iterations. Besides, AdaGrad is still somewhat sensitive to the initial step sizes. Especially, when a small initial step size is used, AdaGrad is not able to increase the step size to a suitable level. As a contrast, SGD-BB converges very fast in all three tested problems, and it is not sensitive to the initial step size $\eta_0$.

Figures 3(d), 3(e) and 3(f) show the comparison results of SAG-BB and SAG-L. From these figures we see that the SAG-L is quite robust and is not sensitive to the choice of $\eta_0$. However, SAG-BB is much faster than SAG-L to reach the same sub-optimality on the tested problems.

Figures 3(g), 3(h) and 3(i) show the comparison results of SGD-BB and oLBFGS. For oLBFGS we used a best-tuned step size. From these figures we see that oLBFGS is much slower than SGD-BB, which is mainly because oLBFGS needs more computational effort per iteration.

## 6 Conclusion

In this paper we proposed to use the BB method to compute the step sizes for SGD and SVRG, which leads to two new stochastic gradient methods: SGD-BB and SVRG-BB. We proved the linear convergence of SVRG-BB for strongly convex function, and as a by-product, we proved the linear convergence of the original SVRG with option I for strongly convex function. We also proposed a smoothing technique to stabilize the step sizes generated in SGD-BB, and we showed how to incorporate the BB method to other SGD variants such as SAG. We conducted numerical experiments on real data sets to compare the performance of SVRG-BB and SGD-BB with existing methods. The numerical results showed that the performance of our SVRG-BB and SGD-BB is comparable to and sometimes even better than the original SVRG and SGD with best-tuned step sizes, and is superior to some advanced SGD variants.

## References

[1] Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. *arXiv preprint arXiv:1603.05643*, 2016.

[2] R. Babanezhad, M. O. Ahmed, A. Virani, M. Schmidt, K. Konečnỳ, and S. Sallinen. Stop wasting my gradients: Practical SVRG. In *Advances in Neural Information Processing Systems*, pages 2242–2250, 2015.

[3] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[4] Y.-H. Dai. A new analysis on the Barzilai-Borwein gradient method. *Journal of Operations Research Society of China*, 1(2):187–198, 2013.

[5] Y.-H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.

[6] Y.-H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang. The cyclic Barzilai-Borwein method for unconstrained optimization. *IMA Journal of Numerical Analysis*, 26(3):604–627, 2006.

[7] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[9] R. Fletcher. On the Barzilai-Borwein method. In *Optimization and control with applications*, pages 235–256. Springer, 2005.

[10] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[11] H. Kesten. Accelerated stochastic approximation. *The Annals of Mathematical Statistics*, 29(1):41–59, 1958.

[12] J. Konečnỳ and P. Richtárik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[13] M. Mahsereci and P. Hennig. Probabilistic line searches for stochastic optimization. *arXiv preprint arXiv:1502.02846*, 2015.

[14] P. Y. Massé and Y. Ollivier. Speed learning on the fly. *arXiv preprint arXiv:1511.02540*, 2015.

[15] D. Needell, N. Srebro, and R. Ward. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *NIPS*, 2014.

[16] Y. Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[17] A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[18] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control and Optimization*, 30:838–855, 1992.

[19] M. Raydan. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, 13(3):321–326, 1993.

[20] M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.

[21] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. 2016.

[22] R. L. Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[23] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 436–443, 2007.

[24] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Jornal of Machine Learning Research*, 14:567–599, 2013.

[25] K. Sopyła and P. Drozda. Stochastic gradient descent with Barzilai-Borwein update step for svm. *Information Sciences*, 316:218–233, 2015.

[26] Y. Wang and S. Ma. Projected Barzilai-Borwein methods for large scale nonnegative image restorations. *Inverse Problems in Science and Engineering*, 15(6):559–583, 2007.

[27] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM J. SCI. COMPUT*, 32(4):1832–1857, 2010.

[28] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[29] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[30] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015.