An Adaptive Incremental Gradient Method With Support for Non-Euclidean Norms

Binghui Xie**, Chenhan Jin*, Kaiwen Zhou, Wei Meng, James Cheng
The Chinese University of Hong Kong

Abstract

Stochastic variance reduced methods have shown strong performance in solving finite-sum problems. However, these methods usually require the users to manually tune the step-size, which is time-consuming or even infeasible for some large-scale optimization tasks. To overcome the problem, we propose and analyze several novel adaptive variants of the popular SAGA algorithm. Eventually, we design a variant of Barzilai-Borwein step-size which is tailored for the incremental gradient method to ensure memory efficiency and fast convergence. We establish its convergence guarantees under general settings that allow non-Euclidean norms in the definition of smoothness and the composite objectives, which cover a broad range of applications in machine learning. The theoretical results supporting non-Euclidean norms fill the void of existing work. Numerical experiments on standard datasets demonstrate a competitive performance of the proposed algorithm compared with existing variance-reduced methods and their adaptive variants.

1 Introduction

Many machine learning tasks involve solving the following optimization problem with a finite-sum structure:

\[
\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i \in [n]} f_i(x),
\]

(1)

where \( x \in \mathbb{R}^d \) is the model parameter, each \( f_i : \mathbb{R}^d \rightarrow \mathbb{R} \) is a loss function for the \( i \)-th training sample and \([n] := \{1, \ldots, n\}\). More generally, the “composite” (or proximal) case covers a larger range of applications in machine learning:

\[
\min_{x \in \mathbb{R}^d} F(x) := f(x) + h(x),
\]

(2)

where \( h(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R} \) is a simple and convex (but possibly non-differentiable) function, and the proximal operation of \( h(\cdot) \) is easy to compute. Here, we also define \( F_i(x) = f_i(x) + h(x) \) and \( \nabla F_i(x) = \nabla f_i(x) + \partial h(x) \) where \( \partial h(x) \)
denotes a sub-gradient of \( h(\cdot) \) at \( x \). Throughout the paper, we denote \( x^\ast \) as an optimal solution of Problem (2).

However, when \( n \) is very large, which is common in the modern learning tasks, gradient descent (GD) has a prohibitively high per-iteration cost. To mitigate the issue, stochastic gradient descent (SGD) (Robbins and Monro, 1951; Nemirovski et al., 2009), as an alternative, has been widely adopted to solve Problem (2) in modern machine learning tasks. Its simplest update process can be written as:

\[
x_{k+1} = \arg \min_x \left\{ \langle \nabla_k, x \rangle + \frac{1}{2\eta} \|x - x_k\|^2 + h(x) \right\},
\]

where \( \eta \) is the step-size, and the gradient estimator \( \nabla_k \) satisfies \( \mathbb{E}[\nabla_k] = \nabla f(x_k), \forall x \in \mathbb{R}^d \) to ensure the convergence. The choice in SGD is to set \( \nabla_k = \nabla f_i(x) \). However, due to the undiminishing variance \( \mathbb{E} \|\nabla f_i(x_k) - \nabla f(x_k)\|^2 \), SGD only converges at a sub-linear rate even if \( F(\cdot) \) is strongly convex and smooth.

Variance reduction. Recently, some stochastic variance reduced (VR) methods have been proposed to solve the variance issue of SGD, such as SAG (Roux et al., 2012), SAGA (Defazio et al., 2014), SVRG (Johnson and Zhang, 2013), SARAH (Nguyen et al., 2017) and SVRG-Loopless (Kovalev et al., 2020). The methods use better choices of the gradient estimator \( \nabla_k \) with its variance reducing as the algorithm converges. Based on this property, in theory, VR methods only need \( \mathcal{O}((n + \kappa) \log(\frac{1}{\epsilon})) \) stochastic gradient evaluations, while GD typically needs \( \mathcal{O}(n \kappa \log(\frac{1}{\epsilon})) \) evaluations and SGD requires \( \mathcal{O}(\frac{1}{\epsilon}) \) complexity.

Consequently, VR methods are commonly used in practice, especially for convex problems such as logistic regression (Dubois-Taine et al., 2021). However, all the VR methods above require a constant step-size, which depends on the characteristics of Problem (2), such as the smoothness constant. In practice, researchers adopt a computationally expensive grid-search method to find a step-size. Yet such constant step-sizes might lead to poor empirical performance.

Adaptive VR methods. Consequently, researchers have also proposed some adaptive VR methods to address the issue in recent years. Tan et al. (2016) proposed SVRG-BB based on Barzilai-Borwein (BB) step-size (Barzilai and Borwein, 1988) and derived SGD-BB and SAG-BB as heuristics. Liu et
al. (2019) developed SVRG-AS and SARAH-AS. Although the methods mentioned above are capable of automatically changing step-size, they still introduce additional hard-to-tune hyperparameters. Li et al. (2020) designed another variants of SVRG-BB and SARAH-BB given the knowledge of some problem-dependent constants. More recently, Dubois-Taine et al. (2021) proposed AdaSVRG for convex objectives, which extends the adaptive step-size in AdaGrad to SVRG’s inner loop. However, the theoretical analysis of AdaSVRG requires a bounded domain, and thus the algorithm needs to perform projection in each iteration, which is rarely the case in machine learning tasks. Shi et al. (2021) emphasized the importance of adapting to the local geometry and proposed an implicit adaptive strategy for SARAH (AI-SARAH). However, AI-SARAH requires solving a sub-problem in each iteration, which could be expensive for general objective functions other than linear regressions.

Although a considerable amount of work has been done for adaptive SVRG and SARAH, another stochastic variance reduction method, SAGA, does not have any adaptive variant. SAGA, which is free of inner loop length tuning and usually has a better practical performance compared with SVRG, has been implemented in the scikit-learn package (Pedregosa et al., 2011) (while SVRG and SARAH have not been) to solve linear regression tasks. Therefore, an adaptive variant of SAGA is of great interest.

However, unlike SVRG and SARAH, which conduct a full gradient computation in each outer iteration and match the adaptive step-size frameworks like BB step-size (Tan et al., 2016), SAGA does not compute full gradients. Instead, SAGA maintains a table of the component function gradients $\nabla f_i(\cdot)$ that were lastly evaluated. Naively using the strategies of the adaptive SVRG or SARAH will not make full use of incremental gradients, hence will introduce a high computation cost for SAGA, resulting in a poor memory overhead. In addition, there exists a discrepancy in the theory between VR methods and their adaptive variants. VR methods are known to be able to solve Problem 1 and 2 both in convex and strongly convex settings. However, most of the adaptive VR methods are limited to solving Problem 1 either in the convex setting or in the strongly convex setting. Moreover, although Euclidean norm space is common and widely considered in both academia and industry, there exist some problems related to non-Euclidean norms that are also crucial to study both in theory and industry, see Appendix C for details. To the best of our knowledge, all the known adaptive VR methods and variants of SAGA do not work with a non-Euclidean norm. Such challenges lead us to the natural question: is it possible to design an efficient adaptive variant for SAGA that can address the above problems under theoretical guarantees?

1.1 Contribution

In this paper, we give a positive answer to above question. We propose the first adaptive framework of SAGA, which we call SAGA-BB, to solve Problem 1 and 2 both in (strongly) convex and non-Euclidean norm space settings. Specifically, we summarize our contributions as follows.

- We propose a novel adaptive stochastic incremental gradient framework based on SAGA. With this framework, we design some novel variants of the existing adaptive step-size, which make full use of the incremental gradients maintained by SAGA. These variants can automatically adjust to the local geometry.
- We also propose a variant of BB step-size. This leads to SAGA-BB, a simple yet powerful adaptive VR method. SAGA-BB is almost tune-free and presents more robust and consistent practical performance than its counterparts. Moreover, SAGA-BB enjoys the same efficient implementation for linear models as the original SAGA method, while allowing both single sample and mini-batch settings.
- We conduct convergence analysis on the composite problem which supports a proximal operator. We prove that SAGA-BB achieves a linear convergence rate $O((n + \kappa) \log(\frac{1}{\epsilon}))$ for strongly convex problems, and a sub-linear rate $O(\frac{1}{\epsilon})$ for general convex objectives. Both rates match the oracle complexity achieved by SAGA (Defazio et al., 2014). Our theoretical results allow non-Euclidean norms in the definition of smoothness. This fills the void of the existing analyses of SAGA.
- We conduct experiments on common machine learning tasks to verify the robustness and effectiveness of SAGA-BB. SAGA-BB demonstrates a strongly competitive performance compared with fine-tuned SAGA, SVRG, SARAH, SVRG-BB and other related methods, across different standard datasets.

2 Algorithm

In this section, we introduce our adaptive variant of SAGA, named SAGA-BB. Before presenting our results, we first discuss the existing adaptive step-sizes in the deterministic setting (Section 2.1), which inspire our work. We next propose several stochastic adaptive step-sizes for SAGA and demonstrate SAGA-BB is the best among them (Section 2.2).

2.1 Deterministic Adaptive Step-size

We summarize several successful adaptive step-sizes for gradient methods in the deterministic setting, which inspired us.

Barzilai-Borwein step-size. Barzilai-Borwein step-size is inspired by quasi-Newton methods, which solves Problem (1) by choosing $\eta$ as an approximation of $H_f^{-1}$ for GD, where $H$ is the Hessian matrix of $f$. However, one needs to solve a linear system to obtain such an approximation, which is time consuming in some cases. To solve the problem, Barzilai and Borwein (1988) proposed BB step-size to satisfy the residual of the secant equation, which can be formulated as: $\min ||\eta_k s_k - y_k||$ or $\min ||s_k - \eta_k y_k||$, where $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$. The corresponding solutions are

$$\eta_k^{BB1} = \frac{s_k^T s_k}{s_k^T y_k} \quad \text{or} \quad \eta_k^{BB2} = \frac{s_k^T y_k}{y_k^T y_k},$$

respectively.

ALS step-size. It is well known that with the constant step-size $\eta \leq 1/L$, GD has a sub-linear rate for convex and
smoothness objectives, where $L$ is the smoothness constant. To obtain an estimation of the local smoothness constant, Armijo (1966); Vrahatis et al. (2000); Liu et al. (2019) proposed the ALS step-size. That is,

$$
\eta_{k}^{ALS} = \frac{1}{2}\left\| \nabla f(x_k) - \nabla f(x_{k-1}) \right\|.
$$

With the scheme above, ALS step-size can obtain the local estimation of the smoothness constant $1/L$ adaptively without additional information.

**YK step-size.** Malitsky and Mishchenko (2020) proposed another $L$ estimation method—YK step-size:

$$
\eta_{k}^{YK} = \min \left\{ \sqrt{1 + \theta_{k-1}\eta_{k-1}}, \frac{\left\| x_k - x_{k-1} \right\|}{2\left\| \nabla f(x_k) - \nabla f(x_{k-1}) \right\|} \right\}.
$$

The $\theta_k$ in YK step-size and its variants introduced below (Section 2.2) is defined as $\eta_k/\eta_k^1$. This adaptive step-size leads to a novel analysis technique for gradient descent. From the perspective of the scheme, we can regard the YK step-size as a bounded variant of ALS step-size.

Note that all the above methods require a full gradient evaluation per epoch or per iteration. Directly extending them to the incremental gradient method is impractical. To tackle these issues, we next provide several novel variants to incorporate these methods to SAGA.

### 2.2 Adaptive Step-sizes for SAGA

We propose a tailor-made adaptive step-size scheme for SAGA, presented in Algorithm 1. Since we want to adjust the step-sizes based on the local geometry of our objective, it is natural to capture the change in the points and gradients with $s_k$ and $y_k$ respectively, in each iteration. With the local information, the step-size will be adjusted automatically every $m$ iterations. Since there are $n$ points, we set the frequency $m$ as $n$ in practice. Although the scheme requires an initial step-size, we observe from numerical experiments that the performance of our method is not sensitive to the choice of the initial step-size.

Based on our framework (Algorithm 1), it is natural to update the step-size by the arithmetic mean of the information from the last $m$ iterations. We propose the following variants of the deterministic adaptive step-sizes in Section 2.1 for SAGA.

**The Stochastic Variant of BB1 step-size.**

$$
\frac{1}{m} \sum_{i=k-m}^{k} \frac{1}{\alpha} \frac{s_i^T y_i}{s_i^T y_i}.
$$

**The Stochastic Variant of BB2 step-size.**

$$
\frac{1}{m} \sum_{i=k-m}^{k} \frac{1}{\alpha} \frac{s_i^T y_i}{y_i^T y_i}.
$$

**The Stochastic Variant of ALS step-size.**

$$
\frac{1}{m} \sum_{i=k-m}^{k} \frac{1}{\alpha} \left\| s_i \right\| / \left\| y_i \right\|.
$$

---

**Algorithm 1 SAGA with adaptive step-size scheme**

**Input:** Max epochs $K$, initial point $x_1$, initial step size $\eta_1$, update frequency $m$.

**Initialize:** Initialize the gradient table $\nabla f_i(\phi_i)$ with $\phi_i^1 = x_1$, and the average $\mu_k = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\phi_i)$.

1: for $k = 1, 2, \ldots, K$ do
2: 
3: \text{if } k \mod m == 0 then
4: \text{ update } \eta_k
5: \text{ else }
6: \eta_k = \eta_{k-1}
7: \text{ end if }
8: \text{ Randomly choose a sample } i \text{ and take } \phi_i^{k+1} = x_k, \text{ store } \nabla f_i(\phi_i^{k+1}) \text{ in the table and keep other entries unchanged.}
9: \text{ end for }

10: \text{ arg min } \left\{ \left\{ \langle \nabla_k, x \rangle + \frac{1}{2\eta_k} \left\| x - x_k \right\|^2 + h(x) \right\} \right\}
11: \text{return } x_{K+1}

---

**The Stochastic Variant of YK step-size.**

$$
\min \left\{ \sqrt{1 + \theta_{k-1}\eta_{k-1}}, \frac{1}{m} \sum_{i=k-m}^{k} \frac{1}{\alpha} \left\| s_i \right\| \right\}.
$$

The constant $\alpha$ in the above stochastic variants is a common hyperparameter to control the convergence, which is similarly required in previous works (Li et al., 2020; Loizou et al., 2021; Tan et al., 2016).

To demonstrate how the above adaptive step-sizes are related to local information, we conduct a heuristic experiment on the jection dataset. Inspired by Shi et al. (2021), we consider Problem (1) with each $f_i(x) = \log(1 + \exp(-b_i^Ta_i^T x)) + \lambda\left\| x \right\|^2$ as an example, where $a_i \in \mathbb{R}^d$ is the data vector, and $b_i \in \{-1, +1\}$ is the label. The local curvature of $f(x)$ can be denoted as its Hessian $\nabla^2 f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{\exp(-b_i^Ta_i^T x)}{[1 + \exp(-b_i^Ta_i^T x)]^2} a_i a_i^T + \lambda I$. It is easy to derive that the local curvature is highly related to the $p_i(x) = \exp(-b_i^Ta_i^T x) / [1 + \exp(-b_i^Ta_i^T x)]$. We plot the distribution of $p_i$ in Figure 1(a).

Correspondingly, we plot the evolution of step-sizes generated by these variants in Figure 1(b). The constant $\alpha$ is set to 1 here. We could see that $p_i$ is decreasing as the number of passes increases, and stabilizes after around 20 passes. Thus, the local curvature will be smaller as well and a larger step-size can be used. Correspondingly, it is reasonable that the step-sizes increase as the number of passes grows, as shown in Figure 1(b).

However, the trajectories of these step-sizes are much steeper than that of $p_i$, which can lead to unstable performance of algorithms and divergence. The reason is that arithmetic mean scheme is highly affected by the outliers and sensitive to the difference of $f_i$. Take the variant of BB1 step-
size as an example. \( s_i^T y_i \) can be much smaller for some indexes, which is also found in Fletcher (2005); Burdakov et al. (2019). This causes the \( s_i^T s_i \) to have a very large value, which consequently leads to a large step-size.

To reduce the impact of the “outliers”, we use the median instead of arithmetic mean to gather the information. The median can be regarded as weighted arithmetic mean, where the weight lies in the denominator. Therefore, for the “outliers” with small denominators, their weights would also be small. They will contribute less to the update of step-size. Based on these considerations, we propose the stable adaptive step-size for SAGA as follows:

**Stable BB1 step-size.**

\[
\frac{\sum_{i=k-m}^{k} s_i^T y_i}{\alpha \sum_{i=k-m}^{k} s_i^T s_i}.
\]

**Stable BB2 step-size.**

\[
\frac{\sum_{i=k-m}^{k} y_i^T y_i}{\alpha \sum_{i=k-m}^{k} y_i^T s_i}.
\]

**Stable ALS step-size.**

\[
\frac{\sum_{i=k-m}^{k} \|s_i\|}{\alpha \sum_{i=k-m}^{k} \|y_i\|}.
\]

**Stable YK step-size.**

\[
\min \left\{ \sqrt{1 + \theta_{k-1} \eta_{k-1}}, \frac{\sum_{i=k-m}^{k} \|s_i\|}{\alpha \sum_{i=k-m}^{k} \|y_i\|} \right\}.
\]

We now refer the previous stochastic adaptive step-sizes as the unstable ones. We plot the evolution of these stable variants in Figure 2(a). From the result, we can see the stable step-sizes show a much smaller and smoother performance than those unstable ones in Figure 1(b). For illustration, we also compare the performance of the stable methods with the unstable methods on the ijcnn dataset. Since \( \|s_i\|/\|y_i\| \) is the geometrical mean of \( s_i^T s_i/\|y_i\|^2 \) and \( s_i^T y_i/\|y_i\|^2 \), we set \( \alpha = 2 \) for stable BB2 step-size, \( \alpha = \sqrt{m}/2 \) for stable YK and ALS methods, and \( \alpha = m \) for stable BB1 step-size across different datasets and losses. For the ijcnn dataset, we tuned \( \alpha = 20/m, \alpha = 5/m, \alpha = 4/m \) for unstable BB2, YK, ALS, BB1 step-sizes, respectively. Note that different from unstable step-sizes, these choices for stable methods also work well on other datasets (refer to Appendix G). As shown in Figure 2(b), the stable methods all enjoy a faster convergence.

Among these stable methods, BB2 shows a more consistent performance across different datasets (refer to the Appendix G). In addition, as implemented in Appendix A, BB2 step-size only requires \( O(n) \) memory consumption for problems with a linear predictor, while other methods need \( O(nd) \) memory overhead. Therefore, the stable BB2 step-size is the sensible choice for SAGA.

### 3 Theory in General Norms

In this section, we theoretically analyze the performance of SAGA-BB in general norm spaces and composite cases. We first introduce some preliminaries about general norms.

#### 3.1 Preliminary

**Norm Space and Notations.** Let \( E \) be a real vector space with finite dimension \( d \) and \( E^* \) is its dual space. \( \langle y, x \rangle \) represents the value of a linear function \( y \in E^* \) at \( x \in E \). \( \| \cdot \| \) denotes an arbitrary norm over \( E \). The dual norm \( \| \cdot \|_* \) over the dual space is defined in the standard way: \( \|y\|_* := \max \{ \langle y, x \rangle \mid \|x\| \leq 1 \} \). For instance, \( \ell_p \) norm is the dual of \( \ell_q \) norm if \( \frac{1}{p} + \frac{1}{q} = 1 \).

**Non-Euclidean Norm Smoothness.** We consider the smoothness with respect to an arbitrary norm \( \| \cdot \| \) over \( E \). Symbolically, we say the differentiable function \( f(\cdot) \) is \( L \)-smooth w.r.t. \( \| \cdot \| \) if \( \forall x, y \in E \), it satisfies: \( \| \nabla f(x) - \nabla f(y) \|_* \leq L \|x - y\| \). Additionally, if \( f(\cdot) \) is convex, the following holds (Nesterov, 2004): \( \| \nabla f(x) - \nabla f(y) \|_* \leq 2L(f(x) - f(y) - \langle \nabla f(y), x - y \rangle) \).

Some famous problems have better smoothness parameters when non-Euclidean norms are adopted. See the discussions in Allen-Zhu and Orecchia (2014). Moreover, We point out in Appendix C that it is important for SAGA to support non-Euclidean norms.
Bregman Divergence

We introduce the proximal setting, which generalizes the usual Euclidean setting (Zhou et al., 2020). The distance generating function (DGF) \(d : \mathbb{E} \to \mathbb{R}\) is required to be continuously differentiable and 1-strongly convex with respect to \(||\cdot||\), i.e., \(d(x) - d(y) - \langle \nabla d(y), x - y \rangle \geq \frac{1}{2}||x - y||^2\), \(\forall x, y \in \mathbb{E}\). Accordingly, the prox-term (Bregman divergence) is given as

\[ V_d(x, y) \overset{\text{def}}{=} d(x) - d(y) - \langle \nabla d(y), x - y \rangle, \quad \forall x, y \in \mathbb{E}. \]

The property of \(V_d(\cdot, \cdot)\) ensures that \(V_d(x, x) = 0\) and \(V_d(x, y) \geq \frac{1}{2}||x - y||^2 \geq 0, \forall x, y \in \mathbb{E}\). One of the key benefits that comes from such a setting is: by adjusting \(||\cdot||\) and \(d(\cdot)\) to the geometry of the problem, mirror descent achieves a smaller problem-dependent constant than the Euclidean algorithms (Nemirovskii and Yudin, 1983).

We first present two instances of Bregman Divergences:

- With \(||\cdot||\) being \(||\cdot||_2\), \(\mathbb{E} = \mathbb{R}^d\), \(d(x) = \frac{1}{2}||x||^2\), this setting can be taken as the standard Euclidean setting:
  \[ V_d(x, y) = \frac{1}{2}||x - y||_2^2. \]

- With \(||\cdot||\) being \(||\cdot||_1\), \(d(x) = \sum_i x_i \log x_i\), this setting can be taken over simplex \(\mathbb{E} \subseteq \mathbb{R}^d : \sum_{i=1}^d x_i = 1\), known as the entropy function:
  \[ V_d(x, y) = \sum_i x_i \log x_i / y_i \geq \frac{1}{2}||x - y||_1^2. \]

We select the \(V_d\) such that the proximal operator, \(\text{Prox}_\eta(x, \mathcal{G}) \overset{\text{def}}{=} \arg \min_{u \in \mathbb{E}} \{ V_d(u, x) + \langle \mathcal{G}, u \rangle + h(u) \}\) is easy to compute for any \(x \in \mathbb{E}, \mathcal{G} \in \mathbb{E}'\). Examples can be found in Parikh and Boyd (2014).

Algorithm Changes. Suppose \(f(x)\) is \(L\)-smooth w.r.t. \(||\cdot||\) and a bregman divergence \(V_d(x, y)\) is given, we perform the following change to the algorithms. In line 12 of Algorithm 1, we change the \(\arg\min\) (proximal operation) to be its non-Euclidean norm variant (Allen-Zhu, 2017): \(x_{k+1} = \text{Prox}_\eta(x_k, \eta_k \nabla f_k)\).

Generalized Strong Convexity of \(f\). A convex function \(f\) is strongly convex with respect to \(V_d(x, y)\) rather than the \(||\cdot||\), i.e., \(f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \mu V_d(x, y), \forall x, y \in \mathbb{E}\).

This can be satisfied if \(d(\cdot) \overset{\text{def}}{=} \frac{1}{\mu} f(\cdot)\). This is known as the “generalized strong convexity” (Shalev-Shwartz and Singer, 2007) and is necessary for the existing linear convergence results in the strongly convex setting. Note that in Euclidean setting, “generalized strong convexity” is equivalent to strong convexity that is with respect to \(\ell_2\) norm.

3.2 Convergence Analysis

In this section, we analyze the convergence of SAGA-BB. Note that the analysis below can be extended to the minibatch version with a random sample size \(|B|\) (see Appendix B for details). First, we present the important lemma which provides a variance bound of the stochastic gradient estimator used by SAGA-BB.

Lemma 1. (Variance Bound). For the stochastic gradient estimator \(\nabla_k\) in Algorithm 1, we have the following variance bound:

\[ \mathbb{E} \left[ \left\| \nabla f(x_k) - \nabla_k \right\|_2^2 \right] \leq 4L \left[ f(x_k) - f(x^*) - \langle \nabla f(x^*), x_k - x^* \rangle \right] + 4L \left[ \frac{1}{n} \sum_{i=1}^n f_i(x_k^i) - f(x^*) - \frac{1}{n} \sum_{i=1}^n \langle \nabla f_i(x^i), \phi_k^i - x^* \rangle \right]. \]

Now we can formally present the main theorem of SAGA-BB below. Compared with other analyses of incremental methods (Defazio et al., 2014; Zhou et al., 2019) that were built only in Euclidean space, we show that SAGA-BB can operate in non-Euclidean norms. Explicitly, we design a new Lyapunov function, which differs from Defazio et al. (2014) and helps us extend our theorems into general norms. For the general convex case, we establish the convergence result of SAGA-BB in terms of averaged iterate: \(\bar{x}_K = \frac{1}{K} \sum_{i=1}^K x_i\).

Theorem 1. (General Convex). Define the Lyapunov function \(T\), which modifies the one in Defazio et al. (2014):

\[ T_k := \frac{1}{\omega} \left[ \frac{1}{n} \sum_{i=1}^n F_i(x_k^i) - F(x^*) - \frac{1}{n} \sum_{i=1}^n \langle \nabla F_i(x^i), \phi_k^i - x^* \rangle \right]. \]

Then we have that for \(\eta_k \leq \frac{1}{\omega}\) and a constant \(\alpha > 9\):

\[ \mathbb{E}[T_k - T_{k+1}] / (\mathbb{E}[F(x_K)] - F(x^*)) \leq 2 \left[ \frac{n}{4} F(x_1) - F(x^*) \right] + \frac{1}{2} F(x_1) + \alpha L V_d(x^*, x_1). \]

Thus, for general convex objectives, SAGA-BB yields a sub-linear convergence rate \(\mathcal{O}(\frac{1}{K})\), which keeps up with the oracle complexity achieved by SAGA (Defazio et al., 2014). Moreover, unlike SAGA, we do not need to assume that Problem (2) is considered on Euclidean norm space, which makes our analysis more general. Compared with other adaptive methods, SAGA-BB can support composite problems where the proximal operator is used.

3.3 Strongly Convex Case

We consider \(f\) satisfies the generalized strongly convexity. For this case, we modify Algorithm 1 to a new method called R-SAGA-BB. In each iteration, R-SAGA-BB runs Algorithm 1 for \(K\) iterations from the current point \(\bar{x}\) and uses its output \(\bar{x}_{K+1}\) as the starting point for the next iteration. This strategy is known as the restarting technique (Dvurechensky and Gasnikov, 2016; Gorbunov et al., 2020). By choosing \(K\) properly, we get an accelerated method for the generalized strongly convexity of \(f\).

Theorem 2. (Strongly Convex). Assuming \(f\) is generalized strongly convex, if we choose \(K = \frac{\mu + 6\alpha L}{\mu} \) with a constant \(\alpha > 9\), then we have after \(\tau\) times running Algorithm 1,

\[ \mathbb{E}[F(x_{\tau+1})] - F(x^*) \leq \frac{1}{2} (F(x_1) - F(x^*)). \]

Therefore, for the generalized strongly convex objective, although we adopt a restarting technique, SAGA-BB yields a fast linear convergence rate \(\mathcal{O}(\left(\frac{n + \frac{1}{\omega}}{\omega}\right) \log \frac{1}{\epsilon})\) which keeps up with the oracle complexity achieved by SAGA.
4 Experiments

In this section, we conduct extensive numerical experiments to verify the efficacy of SAGA-BB. We apply our method and other variance-reduced methods for solving $\ell_2$-regularized logistic regression problems on binary classification tasks with $a_i \in \mathbb{R}^d$, $b_i \in \{-1, +1\}$, indexed by $i \in [n]$: $f(x) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-b_i a_i^T x)) + \frac{\lambda}{2} \|x\|^2$, where $(a_i, b_i)$ is the training sample. $\lambda$ is the regularization parameter which is set to $1/n$.

The datasets we choose are mushrooms, ijcnn1, w8a and covtype.\footnote{All the datasets we used are available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/} We compare SAGA-BB with SAGA, SVRG, SVRG-BB, SVRG-Loopless and SARAH. For a fair comparison, we search over the initial step-size from $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]$ and batch-size from $[1, 8, 16, 64]$ for each algorithm and each experiment. As is common, we set $m = n/batch\_size$ for these methods.

We present our results in Figure 3, where we denote the average value by colored lines and standard deviation by filled areas, across 5 independent runs. The vertical axis is the training objective minus the best loss, and the horizontal axis denotes the number of gradient evaluations normalized by $n$. We compared all the methods with their best-tuning parameters. We can see that SAGA-BB shows superior performance over other methods in all cases. In addition, SAGA-BB significantly outperforms the original SAGA method, even on the dataset where SAGA performs well.

Figure 4 shows the final gradient norm with respect to the initial step-size, presenting the robustness of SAGA-BB. In some cases, the training objective minus the best loss can be zero at the final iteration. Thus, we use the final gradient norm here. We run all the methods with 120 effective gradient passes and different step-sizes. To alleviate other parameters’ impact, we choose a batch-size of 8, with which most of the other methods perform reasonably well. In some cases, SARAH diverged, and we remove the curves accordingly so as not to clutter the plots. We can see that most of the vanilla VR methods are sensitive to the initial step-size on some datasets. For example, SARAH has a big oscillatory behavior on mushrooms, and so does SVRG on covtype. Despite being equipped with BB step-size as well, SVRG-BB is inconsistent across the datasets. However, SAGA-BB shows a more consistent behavior that it always finds a good solution for all the datasets. Thus, the experiment results demonstrate that SAGA-BB shows a strongly competitive performance. Additional figures can be found in Appendix G for further confirming our claim.

5 Conclusion

In this paper, we propose SAGA-BB, a variant of SAGA with a novel adaptive step-size. With a simple and clear construction, it exhibits a robust performance across different settings and random seeds. SAGA-BB can automatically adjust the step-size by approximating the local geometry with stochastic information. Theoretically, we prove that SAGA-BB keeps up with the oracle complexity achieved by SAGA. For future work, it would be interesting to incorporate our adaptive step-size to other stochastic methods in various settings, such as the accelerated methods.
References

Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. arXiv preprint arXiv:1407.1537, 2014.

Zeyuan Allen-Zhu. Katyusha: the first direct acceleration of stochastic gradient methods. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pages 1200–1205, 2017.

Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of Mathematics, 16(1):1 – 3, 1966.

Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. IMA journal of numerical analysis, 8(1):141–148, 1988.

A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. SIAM J. Optim., 12:79–108, 2001.

Oleg Burdakov, Yu-Hong Dai, and Na Huang. Stabilized barzilai-borwein method. arXiv preprint arXiv:1907.06409, 2019.

Aaron Defazio, Francis R. Bach, and Simon 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.

Benjamin Dubois-Taine, Sharan Vaswani, Reza Babanezhad, Mark Schmidt, and Simon Lacoste-Julien. SVRG meets adagrad: Painless variance reduction. CoRR, abs/2102.09645, 2021.

Pavel Dvurechensky and Alexander Gasnikov. Stochastic intermediate gradient method for convex problems with stochastic inexact oracle. Journal of Optimization Theory and Applications, 171(1):121–145, 2016.

Roger Fletcher. On the barzilai-borwein method. In Optimization and control with applications, pages 235–256, Springer, 2005.

Eduard Gorbunov, Marina Danilova, and Alexander Gasnikov. Stochastic optimization with heavy-tailed noise via accelerated gradient clipping. Advances in Neural Information Processing Systems, 33:15042–15053, 2020.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In Advances in Neural Information Processing Systems, pages 315–323, 2013.

Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don’t jump through hoops and remove those loops: SVRG and katyusha are better without the outer loop. In Algorithmic Learning Theory, 2020.

Bingcong Li, Lingda Wang, and Georgios B. Giannakis. Almost tune-free variance reduction. In Proceedings of the 37th International Conference on Machine Learning, 2020.

Yan Liu, Congying Han, and Tiande Guo. A class of stochastic variance reduced methods with an adaptive stepsize. Forthcoming, 2019.

Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for SGD: an adaptive learning rate for fast convergence. In The 24th International Conference on Artificial Intelligence and Statistics, 2021.

Yura Malitsky and Konstantin Mishchenko. Adaptive gradient descent without descent. In Proceedings of the 37th International Conference on Machine Learning, 2020.

Arkadi Nemirovski, Anatoli B. Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. SIAM J. Optim., 19(4):1574–1609, 2009.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Yurii E. Nesterov. Introductory Lectures on Convex Optimization - A Basic Course, volume 87 of Applied Optimization. Springer, 2004.

Lam M. Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In Proceedings of the 34th International Conference on Machine Learning, 2017.

Neal Parikh and Stephen Boyd. Proximal algorithms. Foundations and Trends in optimization, 1(3):127–239, 2014.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. The Annals of Mathematical Statistics, 22(3):400 – 407, 1951.

Nicolas Le Roux, Mark Schmidt, and Francis R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2012.

Shai Shalev-Shwartz and Yoram Singer. Online learning: Theory, algorithms, and applications. 2007.

Zheng Shi, Nicolas Loizou, Peter Richtárik, and Martin Takáč. AI-SARAH: adaptive and implicit stochastic recursive gradient methods. CoRR, abs/2102.09700, 2021.

Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent. In Advances in Neural Information Processing Systems, pages 685–693, 2016.

Michael N Vrahatis, George S Androulakis, JN Lambrinos, and George D Magoulas. A class of gradient unconstrained minimization algorithms with adaptive stepsize. Journal of Computational and Applied Mathematics, 114(2):367–386, 2000.
Kaiwen Zhou, Qinghua Ding, Fanhua Shang, James Cheng, Danli Li, and Zhi-Quan Luo. Direct acceleration of saga using sampled negative momentum. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1602–1610, 2019.

Kaiwen Zhou, Yanghua Jin, Qinghua Ding, and James Cheng. Amortized nesterov’s momentum: A robust momentum and its application to deep learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 211–220, 2020.
Algorithm 2 Implementation of SAGA-BB

Stored: Scalar table $\Phi^k$ where $\Phi^k = (a_i, \phi_j^k)$, and $\mu^k$.

1: At iteration $k$
2: if $k \mod m == 0$ then
3: $\eta_k = \frac{\sum_{i=k-m}^k s_i^k y_i}{\alpha \sum_{i=k-m}^k y_i^i y_i}$
4: end if
5: Randomly choose a sample $i$ and compute the gradient estimator with the scalar table. Store $\phi^{k+1}_i = (a_i, x_k)$ with other entries unchanged.
6: $\nabla_k = (\nabla \psi_i(\Phi^{k+1}) - \nabla \psi_i(\Phi^k)) \cdot d_i + \mu^k$
7: $\mu_{k+1} = \mu_k + \frac{1}{n} (\nabla \psi_i(\Phi^{k+1}) - \nabla \psi_i(\Phi^k)) \cdot d_i$
8: $s^k_k y_k = (\Phi^{k+1}_i \cdot \nabla \psi_i(\Phi^{k+1}) + \Phi^k_i \cdot \nabla \psi_i(\Phi^k_i) + \Phi_k \cdot \nabla \psi_i(\Phi^k + 1) + \Phi_k \cdot \nabla \psi_i(\Phi^k))$
9: $y^T_k y_k = \|\nabla \psi_i(\Phi^{k+1}) \cdot d_i \|^2 + 2(\nabla \psi_i(\Phi^{k+1}) \cdot d_i + \nabla \psi_i(\Phi^k) \cdot d_i \|^2$
10: $x_{k+1} = \arg \min_x \{ \langle \nabla_k, x \rangle + \frac{1}{4\eta_k} \| x - x_k \|^2 + h(x) \}$

A Implementation of SAGA-BB

The memory-efficient implementation of SAGA-BB at iteration $k$ is presented in Algorithm 2. It is inspired by the fact that the objectives $f_j(x)$ in many ML tasks can be interpreted as a function of scalars (Zhou et al., 2019), i.e., $\psi_i((a_i, x_i))$. We only store the scalar $(a_i, \phi_j^k)$ instead of the vector $\nabla f_j(\phi_j^k)$ for each $i$. Based on this, we can have the value of $s^k_k y_k$ and $y^T_k y_k$ without extra storage requirement as shown in Algorithm 2, whereas other adaptive step-sizes need to store the points vector $\phi^k_i$. Thus, we achieve the $O(n)$ storage requirement for SAGA-BB in practice.

B ALGORITHM FOR MINI-BATCH

Let $|B|$ denote the batch size. At each iteration $k$, we will pick a mini-batch $B$ from 1,.., $n$ of size $|B|$ and the mini-batch gradients can be represented as:

$$\nabla f_B(\phi_B^k) = \frac{1}{|B|} \sum_{j \in B} f_j(\phi_j^k).$$

We also use $\phi_B$ to denote the the set of points $\phi_j$, for each $j \in B$.

Algorithm 3 SAGA-BB

Input: Max epochs $K$, batch-size $|B|$, initial point $x_1$, initial step size $\eta_1$, update frequency $m$.

Initialize: Initial the gradient table $\nabla f_i(\phi_i^1)$ table with $\phi_i^1 = x_1$, and the average $\mu = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\phi_i^1)$.

1: for $k = 1, 2, \ldots, K$ do
2: if $k \mod m == 0$ then
3: $\eta_k = \frac{\sum_{i=k-m}^k s_i^k y_i}{\alpha \sum_{i=k-m}^k y_i^i y_i}$
4: end if
5: Randomly pick a batch $B$, take $\phi_j^{k+1} = x_k$, store $\nabla f_j(\phi_j^{k+1})$ with other entries unchanged for every $j \in B$.
6: $\nabla_k = \nabla f_B(\phi_B^{k+1}) - \nabla f_B(\phi_B^k) + \mu_k$
7: $y_k = \sum_{j \in B} (\nabla f_j(\phi_j^{k+1}) - \nabla f_j(\phi_j^k))$
8: $s_k = \sum_{j \in B} (\phi_j^{k+1} - \phi_j^k)$
9: $\mu_{k+1} = \mu_k + \frac{1}{n} \sum_{j \in B} (\phi_j^{k+1} - \phi_j^k)$
10: $x_{k+1} = \arg \min_x \{ \langle \eta_k \nabla_k, x \rangle + V_d(x, x_k) + \eta_k h(x) \}$
12: return $x_{K+1}$
It is direct to upper bound the first term on the right side using smoothness argument, i.e.,

Similarly, we start with per-iteration analysis in mini-batch setting. Using convexity of \( f \)

E PROOF OF THEOREM 1

Algorithm 4 Restarted SAGA-BB (R-SAGA-BB)

**Input:** Max epochs \( \tau \), number of iterations \( K \) of SAGA-BB, initial point \( x_1 \), initial step size \( \eta_1 \).

**Initialize:** Initial the gradient table \( \nabla f_i(\phi_i^1) \) table with \( \phi_i^1 = x_1 \), and the average \( \mu = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\phi_i^1) \).

1: for \( t = 1, 2, \ldots, \tau \) do
2: Run SAGA-BB (Algorithm 3) for \( K \) iterations with batch-size \( |B| \), step size \( \gamma \), starting point \( \bar{x}_t \), and update frequency \( m \). Define the output of SAGA-BB by \( \bar{x}_{t+1} \).
3: end for
4: return \( \bar{x}_{\tau+1} \).

C Importance of Non-Euclidean Norms

Euclidean norm space is common and widely considered in both academia and industry. However, there exist some problems related to non-Euclidean norms that are also crucial to study both in theory (Allen-Zhu and Orecchia, 2014) and industry (e.g., Positron Emission Tomography (PTE) can be seen as a Maximum Likelihood Estimate problem related to non-Euclidean norms that are also crucial to study both in theory (Allen-Zhu and Orecchia, 2014) and industry (e.g., Euclidean norm space is common and widely considered in both academia and industry. However, there exist some problems related to \( \ell_1 \) norm case over the large simplex (Ben-Tal et al., 2001)). Katyusha proposed by Allen-Zhu (2017), as a variant of SVRG, can be extended to non-Euclidean norm smoothness settings. However, there is still a void in the existing analysis for SAGA and its variants. The main reason is that the theory of SAGA crucially relies on the \( \ell_2 \) norm space. To tackle this problem, we propose a new Lyapunov function which supports arbitrary norms and show that it can also derive a characteristic inequality. In the following subsections, we conduct a comprehensive study on the convergence analysis.

D PROOF OF LEMMA 1

\[
\mathbb{E} \left[ \left\| \nabla f(x_k) - \nabla \bar{x}_k \right\|^2 \right] \leq \mathbb{E} \left[ \left\| \nabla f_B(x_k) - \nabla f_B(\phi_B) \right\|^2 \right]
\]

\[
\leq 2\mathbb{E} \left[ \left\| \nabla f_B(x_k) - \nabla f_B(x^*) \right\|^2 \right] + 2\mathbb{E} \left[ \left\| \nabla f_B(\phi_B) - \nabla f_B(x^*) \right\|^2 \right]
\]

\[
\leq 4L \left[ f(x_k) - f(x^*) - \langle \nabla f(x^*), x_k - x^* \rangle \right]
\]

\[
+ 4L \left[ \frac{1}{n} \sum_{i=1}^{n} f_i(\phi_i^k) - f(x^*) - \frac{1}{n} \sum_{i=1}^{n} \langle \nabla f_i(x^*), \phi_i^k - x^* \rangle \right],
\]

where \( a \) uses \( \mathbb{E}[\|X - \mathbb{E}X\|^2] \leq \mathbb{E}[\|X\|^2] \), \( b \) uses \( \|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2 \) and \( c \) follows from Theorem 2.1.5 in Nesterov (2004).

E PROOF OF THEOREM 1

In this case, we modify the Lyapunov function as

\[
T_k := \frac{1}{\omega} \left[ \frac{1}{n} \sum_{i=1}^{n} F_i(\phi_i^k) - F(x^*) - \frac{1}{n} \sum_{i=1}^{n} \langle \nabla F_i(x^*), \phi_i^k - x^* \rangle \right].
\]

Similarly, we start with per-iteration analysis in mini-batch setting. Using convexity of \( f(\cdot) \) at \( x_k \), we have

\[
f(x_k) - f(x^*) \leq \langle \nabla f(x_k) - \nabla \bar{x}_k, x_k - x^* \rangle + \langle \nabla \bar{x}_k, x_k - x_{k+1} \rangle + \langle \nabla \bar{x}_k, x_{k+1} - x^* \rangle.
\]

Taking expectation with respect to sample batch size \( |B| \), we obtain

\[
f(x_k) - f(x^*) \leq \mathbb{E} \left[ \langle \nabla \bar{x}_k, x_k - x_{k+1} \rangle + \langle \nabla \bar{x}_k, x_{k+1} - x^* \rangle \right].
\]

It is direct to upper bound the first term on the right side using smoothness argument, i.e.,

\[
f(x_{k+1}) - f(x_k) \leq \langle \nabla f(x_k) - \nabla \bar{x}_k, x_{k+1} - x_k \rangle + \langle \nabla \bar{x}_k, x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2.
\]

Taking expectation with respect to sample \( k \) and re-arranging, we obtain

\[
\mathbb{E} \left[ \langle \nabla \bar{x}_k, x_k - x_{k+1} \rangle \right] \leq f(x_k) - \mathbb{E}[f(x_{k+1})] + \mathbb{E} \left[ \langle \nabla f(x_k) - \nabla \bar{x}_k, x_k - x_{k+1} \rangle \right]
\]

\[
+ \frac{L}{2} \mathbb{E} \left[ \|x_{k+1} - x_k\|^2 \right].
\]
Note that
\[
\mathbb{E}[T_{k+1}] = \frac{1}{\omega} \left[ (1 - \frac{|B|}{n}) \frac{1}{n} \sum_{i=1}^{n} F_i(\phi_k^i) - F(x^*) - (1 - \frac{|B|}{n}) \frac{1}{n} \sum_{i=1}^{n} \langle \nabla F_i(x^*), \phi_k^i - x^* \rangle - \frac{|B|}{n} \langle \nabla F(x^*), x_k - x^* \rangle \right] + \frac{|B|}{n} F(x_k)
\]
\[
(\ref{eq:7})
\]
\[
(\ref{eq:8})
\]
where (d) using the optimality of \(x^*\) that \(\nabla F(x^*) = 0\). Upper-bounding (5) using (6), Lemma 2, (7) and re-arranging, we obtain
\[
\mathbb{E}[T_{k+1}] + \mathbb{E}[f(x_{k+1})] - f(x^*)
\]
\[
\leq \mathbb{E} \left[ \langle \nabla f(x_k) - \nabla_k, x_{k+1} - x_k \rangle \right] + \frac{L}{2} \mathbb{E} \left[ \|x_{k+1} - x_k\|^2 \right] + \mathbb{E}[T_{k+1}]
\]
\[
+ \frac{1}{\eta_k} (V_d(x^*, x_{k+1}) - \mathbb{E}[V_d(x^*, x_{k+1})] - \mathbb{E}[V_d(x_{k+1}, x_k)]) + h(x^*) - \mathbb{E}[h(x_{k+1})]
\]
\[
(\ref{eq:a}) \quad \frac{1}{2\beta} \mathbb{E} \left[ \|\nabla f(x_k) - \nabla_k\|^2 \right] + \left( \frac{\beta}{2} + \frac{L}{2} - \frac{1}{2\eta_k} \right) \mathbb{E} \left[ \|x_{k+1} - x_k\|^2 \right]
\]
\[
+ \frac{1}{\eta_k} (V_d(x^*, x_k) - \mathbb{E}[V_d(x^*, x_{k+1})]) + \mathbb{E}[T_{k+1}] + h(x^*) - \mathbb{E}[h(x_{k+1})]
\]
\[
(\ref{eq:b}) \quad T_k + \left( \frac{2L\omega}{\beta} - \frac{|B|}{n} \right) T_k + \left( \frac{2L}{\beta} + \frac{|B|}{n\omega} \right) [F(x_k) - F(x^*)]
\]
\[
+ \left( \frac{\beta}{2} + \frac{L}{2} - \frac{1}{2\eta_k} \right) \mathbb{E} \left[ \|x_{k+1} - x_k\|^2 \right] + \frac{1}{\eta_k} (V_d(x^*, x_k) - \mathbb{E}[V_d(x^*, x_{k+1})])
\]
\[
\quad - \frac{2L}{\beta} \left[ \frac{1}{n} \sum_{i=1}^{n} h(\phi_k^i - h(x^*) - \frac{1}{n} \sum_{i=1}^{n} \langle \partial h(x^*), \phi_k^i - x^* \rangle \right] + h(x^*) - \mathbb{E}[h(x_{k+1})]
\]
\[
- \frac{2L}{\beta} [h(x_k) - h(x^*) - \langle \partial h(x^*), x_k - x^* \rangle],
\]
where (a) uses Young’s inequality with \(\beta > 0\) and that \(d(\cdot)\) is 1-strongly convex, (b) follows from Lemma 1 and (7). Letting \(\beta = 8L, \omega = \frac{4|B|}{n}\) and \(9L \leq \frac{1}{\eta_k} \leq \alpha L\), we obtain
\[
\mathbb{E}[T_{k+1}] + \mathbb{E}[F(x_{k+1})] - F(x^*)
\]
\[
\leq T_k + \frac{1}{2}[F(x_k) - F(x^*)] + \frac{1}{\eta_k} (V_d(x^*, x_k) - \mathbb{E}[V_d(x^*, x_{k+1})])
\]
Thus,
\[
\frac{1}{2}[\mathbb{E}[F(x_{k+1})] - F(x^*)]
\]
\[
\leq T_k - \mathbb{E}[T_{k+1}] + \frac{1}{2}[F(x_k) - \mathbb{E}[F(x_{k+1})]] + \frac{1}{\eta_k} (V_d(x^*, x_k) - \mathbb{E}[V_d(x^*, x_{k+1})])
\]
Summing the above inequality from \(k = 1, \cdots, K\) and taking expectation with respect to all randomness, we obtain
\[
\frac{1}{2} \sum_{k=1}^{K} \left[ \mathbb{E}[F(x_{k+1})] \right] - K F(x^*) \leq T_1 + \frac{1}{2} F(x_1) + \frac{1}{\eta_k} V_d(x^*, x_1)
\]
Using Jensen’s inequality on the left side, we obtain the output after \(K\) iterations of SAGA-BB:
\[
\mathbb{E}[F(x_{K+1})] - F(x^*) \leq \frac{2}{K} \left[ \frac{n}{4|B|} [F(x_1) - F(x^*)] + \frac{1}{2} F(x_1) + \alpha LV_d(x^*, x_1) \right].
\]
Therefore, with \(|B| = 1\) we complete the proof of Theorem 1.
Consider the first run of SAGA-BB (Algorithm 3), i.e., \( t = 0 \). Recall the proof of Theorem 1, we sum the inequality (9) from \( k = 1, \cdots, K \) and taking expectation with respect to all randomness, we obtain

\[
\frac{1}{2} \sum_{i=1}^{K} \left| \mathbb{E}[F(x_{k+1})] \right| - K F(x^*) \leq T_1 + \frac{1}{2} [F(x_1) - \mathbb{E}[F(x_{k+1})]] + \frac{1}{\eta_k} V_d(x^*, x_1)
\]

\[
\leq T_1 + \frac{1}{2} [F(x_1) - F(x^*)] + \frac{1}{\eta_k} V_d(x^*, x_1).
\]

Then by using Jensen’s inequality on the left side, we obtain the output after \( K \) iterations of SAGA-BB:

\[
\mathbb{E}[F(x_K)] - F(x^*) \leq \frac{2}{K} \left( \left( \frac{n}{4|B|} + \frac{1}{2} \right) (F(x_1) - F(x^*)) + \alpha L V_d(x^*, x_1) \right).
\]

Observed that the proof of Theorem 1 still holds if we substitute \( V_d(x^*, x_1) \) everywhere by its upper bound \( \frac{1}{\mu} (F(x_1) - F(x^*)) \).

By using generalized strong convexity of \( f(\cdot) \), we have

\[
\mu V_d(x^*, x_1) \leq f(x_1) - f(x^*) - \langle \nabla f(x^*), x_1 - x^* \rangle
\]

\[
\leq f(x_1) - f(x^*) - \langle \nabla f(x^*), x_1 - x^* \rangle + h(x_1) - h(x^*) - \langle \partial h(x^*), x_1 - x^* \rangle
\]

\[
\leq F(x_1) - F(x^*) - \langle \nabla F(x^*), x_1 - x^* \rangle
\]

\[
\leq F(x_1) - F(x^*)
\]

where the second inequality uses the convexity of \( h(\cdot) \) and \( \partial h(\cdot) \) denotes its sub-gradient. It implies that when \( t = 1 \), i.e., after \( K \) iterations of SAGA-BB, we have

\[
\mathbb{E}[F(x_1)] - F(x^*) \leq \frac{2}{K} \left( \left( \frac{n}{4|B|} + \frac{1}{2} + \frac{\alpha L}{\mu} \right) (F(x_1) - F(x^*)) \right).
\]

Thus taking \( K = \frac{\mu n + 6\alpha |B| L}{\mu |B|} \), we have

\[
\mathbb{E}[F(x_1)] - F(x^*) \leq \frac{1}{2} (F(x_1) - F(x^*)).
\]

Then, by induction, we can show that for arbitrary \( t = 1, 2, \ldots, \tau \), the inequality

\[
\mathbb{E}[F(x_{\tau+1})] - F(x^*) \leq \frac{1}{2^\tau} (F(x_1) - F(x^*)).
\]

Therefore, with \( |B| = 1 \) we complete the proof of Theorem 2.

**G ADDITIONAL EXPERIMENTS**

Following the experiments presented in the main paper, we further evaluate the performance of our methods with additional experiments.

**G.1 Additional experiments with stable methods**

In this subsection, we evaluate the stable methods on different datasets, presented in Figure 5. Among these stable methods, BB2 shows a more consistent performance across different datasets.

**G.2 Additional experiments for logistic loss**

This subsection presents additional numerical results to present the robustness and the competitive performance of SAGA-BB.
Figure 5: Performance of different stable methods on different datasets.

Figure 6: Performance under different initial step size and fixed batch-size of 1. For the plot to be readable, we limit the gradient norm to a maximum value of 1.

G.3 Additional experiments for Huber loss
In this subsection, We apply our method and other variance-reduced methods for solving binary $\ell_2$-regularized problems with Huber loss. These figures show that the good performance of SAGA-BB is consistent across losses, batch-sizes and datasets.
Figure 7: Performance under different initial step size and fixed batch-size of 16. To make the plot readable, we limit the gradient norm to a maximum value of 1.

Figure 8: Performance under different initial step size and fixed batch-size of 64. To make the plot readable, we limit the gradient norm to a maximum value of 1.

Figure 9: Convergence on different LibSVM datasets for Huber loss

Figure 10: Performance under different initial step size and fixed batch-size of 1. To make the plot readable, we limit the gradient norm to a maximum value of 1.
Figure 11: Performance under different initial step size and fixed batch-size of 8. To make the plot readable, we limit the gradient norm to a maximum value of 1.

Figure 12: Performance under different initial step size and fixed batch-size of 16. To make the plot readable, we limit the gradient norm to a maximum value of 1.

Figure 13: Performance under different initial step size and fixed batch-size of 64. To make the plot readable, we limit the gradient norm to a maximum value of 1.