Journal of Machine Learning Research 26 (2025) 1-44 Submitted 5/23; Revised 10/24; Published 3/25

Nonconvex Stochastic Bregman Proximal Gradient Method with Application to Deep Learning

Kuangyu Ding KUANGYUD@U.NUS.EDU
Department of Mathematics
National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076

Jingyang Li LI_JINGYANG@U.NUS.EDU
Department of Mathematics
National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076

Kim-Chuan Toh MATTOHKC@NUS.EDU.SG
Department of Mathematics
Institute of Operations Research and Analytics
National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076

Editor: Sanjiv Kumar

Abstract

Stochastic gradient methods for minimizing nonconvex composite objective functions typically rely on the Lipschitz smoothness of the differentiable part, but this assumption fails in many important problem classes like quadratic inverse problems and neural network training, leading to instability of the algorithms in both theory and practice. To address this, we propose a family of stochastic Bregman proximal gradient (SBPG) methods that only require smooth adaptivity. SBPG replaces the quadratic approximation in SGD with a Bregman proximity measure, offering a better approximation model that handles non-Lipschitz gradients in nonconvex objectives. We establish the convergence properties of vanilla SBPG and show it achieves optimal sample complexity in the nonconvex setting. Experimental results on quadratic inverse problems demonstrate SBPG's robustness in terms of stepsize selection and sensitivity to the initial point. Furthermore, we introduce a momentum-based variant, MSBPG, which enhances convergence by relaxing the mini-batch size requirement while preserving the optimal oracle complexity. We apply MSBPG to the training of deep neural networks, utilizing a polynomial kernel function to ensure smooth adaptivity of the loss function. Experimental results on benchmark datasets confirm the effectiveness and robustness of MSBPG in training neural networks. Given its negligible additional computational cost compared to SGD in large-scale optimization, MSBPG shows promise as a universal open-source optimizer for future applications.

Keywords: Nonconvex stochastic algorithm, Bregman distance, Smooth adaptivity, Deep neural network, Algorithmic stability

1. Introduction

In this paper, we present and analyze a family of nonconvex stochastic Bregman proximal gradient methods (SBPG) for solving the following generic stochastic minimization problem:

$$\min_{\boldsymbol{x}\in\overline{C}} \mathbb{E}_{\boldsymbol{\xi}}[f(\boldsymbol{x},\boldsymbol{\xi})] + R(\boldsymbol{x}), \tag{1}$$

where $f(\cdot,\boldsymbol{\xi})$ is a nonconvex differentiable function on $\overline{C}$, R is a proper lower-semicontinuous convex function, $\boldsymbol{\xi}$ is a random variable, and $\overline{C}$ is the closure of C, which is a nonempty convex open subset of $\mathbb{R}^d$. We denote $F(\boldsymbol{x}) \coloneqq \mathbb{E}_{\boldsymbol{\xi}}[f(\boldsymbol{x},\boldsymbol{\xi})]$, and $\Phi(\boldsymbol{x}) \coloneqq F(\boldsymbol{x}) + R(\boldsymbol{x})$. This type of stochastic minimization problem is common in machine learning and statistics (Hastie et al., 2009; Shapiro et al., 2021; Zhang, 2004), where the optimizer has limited access to the distribution of $\boldsymbol{\xi}$ and can only draw samples from it. In many instances, the smooth part of the objective function $F(\boldsymbol{x})$ can be formulated as a finite-sum structure $F(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x})$. Although the distribution of $\boldsymbol{\xi}$ is known in such cases, when n is extremely large, calculating the true gradient for the smooth part of the objective function becomes extremely expensive. As a result, stochastic first-order methods, originating from the work of Robbins and Monro (1951), have emerged as the prevailing approach for solving these large-scale optimization problems. In particular, stochastic (proximal) gradient descent and its numerous variants (Duchi et al., 2011; Duchi and Singer, 2009; Gu et al., 2020; Kingma and Ba, 2014; Allen-Zhu, 2018; Wang et al., 2022) have been widely used in large-scale stochastic optimization for machine learning (LeCun et al., 2015; Shapiro et al., 2021; Zhang, 2004). From a modeling perspective, stochastic (proximal) gradient descent can be viewed as minimizing a sequence of upper quadratic approximations of the nonconvex objective $\Phi(\boldsymbol{x})$:

$$\boldsymbol{x}^{k+1} = \operatorname*{argmin}_{\boldsymbol{x}\in\overline{C}} \left\{ \underbrace{F(\boldsymbol{x}^k,\boldsymbol{\Xi}_k) + \langle \widetilde{\nabla}_k,\, \boldsymbol{x}-\boldsymbol{x}^k\rangle + \frac{1}{2\alpha_k}\|\boldsymbol{x}-\boldsymbol{x}^k\|^2}_{F_{\boldsymbol{x}^k}(\boldsymbol{x})\text{: model of } F \text{ at } \boldsymbol{x}^k} + R(\boldsymbol{x}) \right\}, \tag{2}$$

where $F(\boldsymbol{x}^k,\boldsymbol{\Xi}_k) \coloneqq \frac{1}{|\boldsymbol{\Xi}_k|}\sum_{\boldsymbol{\xi}\in\boldsymbol{\Xi}_k} f(\boldsymbol{x}^k,\boldsymbol{\xi})$, $\boldsymbol{\Xi}_k$ is the set of samples of $\boldsymbol{\xi}$ at the k-th iteration, and $\widetilde{\nabla}_k$ is an estimator of the exact gradient $\nabla F(\boldsymbol{x}^k)$. This modeling perspective is well-established in deterministic optimization and has been used in methods such as Newton method, Gauss-Newton method, bundle method, and trust-region method, as discussed in various sources such as Hiriart-Urruty and Lemaréchal (1993); Nesterov (2003); Lin et al. (2007); Paren et al. (2022).

Despite their widespread use, stochastic gradient methods (2) face several well-known challenges both in theory and practice. One of the key assumptions underlying the analysis of these methods is the Lipschitz continuity of the gradient of the differentiable part, which is critical for ensuring convergence and establishing complexity results. However, this assumption is not always valid. For instance, even a simple function like $F(x) = x^4$ does not admit a globally Lipschitz gradient over $\mathbb{R}$, illustrating the limitations in analyzing stochastic gradient methods in more general settings. In addition, choosing the appropriate stepsize is another challenge in both the theoretical analysis and the practical usage

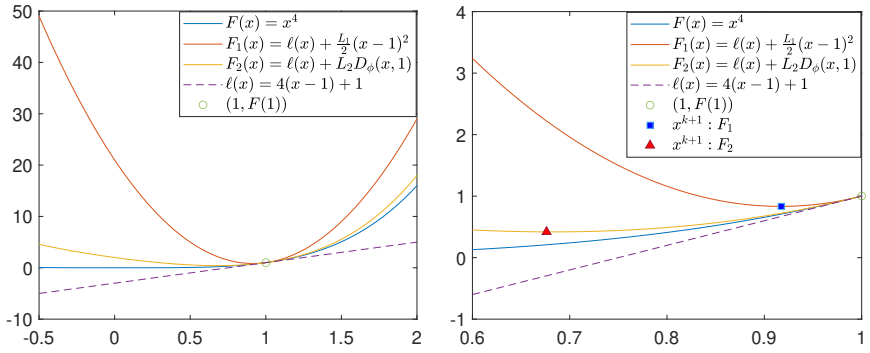
of stochastic gradient methods. In practice, the stepsize plays a decisive role—arguably the most important—in determining the convergence behavior of the algorithm. Finding the optimal stepsize can be time-consuming, as engineers often need to conduct numerous experiments to fine-tune it. From a theoretical perspective, choosing a cautious stepsize is necessary to ensure the descent property at each iteration, which is typically proportional to the inverse of the Lipschitz smoothness parameter. As a result, the absence of Lipschitz smoothness can lead to instability in classical stochastic first-order methods, complicating both their theoretical analysis and practical implementation.

To address these issues, classical approaches often resort to either line search or more complicated inner loops, but these methods can negatively impact the efficiency of the algorithm or even become intractable in a stochastic setting. For example, stochastic proximal point algorithm (PPA) models the approximation of $F(\boldsymbol{x})$ in (2) as $F_{\boldsymbol{x}^k}(\boldsymbol{x}) = F(\boldsymbol{x}, \boldsymbol{\xi}_k) + \frac{1}{2\alpha_k}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2$ (Bertsekas, 2011; Bianchi, 2016; Patrascu and Necoara, 2017; Rockafellar, 1976), making the selection of stepsize α_k more robust than the original model (2). However, the application of stochastic PPA is limited due to the difficulty of solving the subproblems, especially for complicated objectives, such as training deep neural networks. In such cases, solving the subproblem is almost as difficult as solving the original problem, rendering the method impractical. Recently, Bauschke et al. (2017); Lu et al. (2018) have proposed using Bregman proximity measures to relax the assumption of gradient Lipschitz continuity to smooth adaptivity. The Bregman gradient method was first introduced as the mirror descent scheme by Nemirovskij and Yudin (1983) for minimizing convex nonsmooth functions. From the modeling perspective, Bregman methods consider the following subproblem at each iteration:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}\in\overline{C}}{\operatorname{argmin}} \left\{ \underbrace{F(\boldsymbol{x}^k, \boldsymbol{\Xi}_k) + \langle \widetilde{\nabla}_k,\, \boldsymbol{x} - \boldsymbol{x}^k\rangle + \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{x}^k)}_{F_{\boldsymbol{x}^k}(\boldsymbol{x})\text{: model of } F \text{ at } \boldsymbol{x}^k} + R(\boldsymbol{x}) \right\}, \tag{3}$$

where $\mathcal{D}_\phi$ is the Bregman distance induced by the kernel function ϕ. To illustrate the advantage of the Bregman proximity model, we present a toy example. Consider the objective function $F(x) = x^4$, which does not admit a globally Lipschitz continuous gradient. We compare the performance of the upper quadratic approximation model (2) and Bregman proximity model (3). As shown in Figure (1)(a), the Bregman proximity model (3) ($F_2(x)$) with the kernel function $\phi(x) = \frac{1}{2}x^2 + \frac{1}{4}x^4$ can provide a closer approximation to $F(x)$ than the upper quadratic approximation model 2 ($F_1(x)$), as the yellow curve stays closer to the curve of the objective function $F(x) = x^4$. This improved approximation enables the Bregman gradient method to generate x^{k+1} that makes more substantial progress toward the optimal solution ($x^* = 0$) compared to the gradient descent method, as shown in Figure 1(b).

While several stochastic extensions of Bregman methods based on smooth adaptivity assumptions have been developed recently, the existing literature primarily focuses on stochastic convex problems (Dragomir et al., 2021b; Hanzely and Richtárik, 2021; Lu, 2019). Convergence analyses for Bregman methods in nonconvex settings (Latafat et al., 2022; Wang and Han, 2023) rely heavily on variance reduction techniques and finite sum structure.



(a) Two models' approximations of $F(x)$ (b) Zoomed-in of version plot (a)

Figure 1: For function $F(x) = x^4$, which does not admit a globally Lipschitz continuous gradient. We restrict the feasible set to $[-0.5, 2]$. Consider the models (2) and (3) of F at $x^k = 1$. The Lipschitz smooth constant of F with respect to the kernel $\phi(x) = \frac{1}{2}x^2$ is 48. The smooth adaptivity constant of F with respect to the kernel $\phi(x) = \frac{1}{2}x^2 + \frac{1}{4}x^4$ is 4. The figure in (b) is a zoomed-in version of the plot in (a) for the range $[0.6, 1]$. The unique minimum of $F(x)$ is at $x = 0$.

However, these approaches are either memory-intensive or require periodic computation of the full gradient, making them impractical for large-scale problems such as deep neural networks as demonstrated in (Defazio and Bottou, 2019).

Beyond the optimization literature, recent works in learning theory (Azizan et al., 2019; Li et al., 2021; Sun et al., 2022, 2023) have focused on specific scenarios, such as highly overparameterized models or linear models for classification and regression tasks, examining the implicit bias of Bregman-type methods. For instance, (Azizan et al., 2019) demonstrates that in overparameterized models, mirror descent iterates converge to the global minimum closest to the initial point in terms of Bregman divergence, provided the initial point is near the minimum set. Other works (Li et al., 2021; Sun et al., 2022, 2023) explore convergence towards the direction that maximizes the margin in classification problems. Our work, however, focuses on the optimization properties of stochastic Bregman proximal gradient (SBPG) in general nonconvex optimization problems, which presents a different perspective from the implicit bias literature. As a result, we will not review the broader literature on implicit bias in this context.

As we can see, stochastic Bregman methods have not been fully explored in the general large-scale nonconvex problems such as training neural networks, and rigorous numerical evaluations of their performance are limited. Additionally, the current literature gives insufficient attention to the robustness of stochastic Bregman methods, particularly regarding the selection of stepsizes and initial points—factors that can have a substantial impact on their effectiveness in large-scale applications.

In this paper, we consider stochastic Bregman proximal gradient methods (SBPG) for nonconvex problems with application to the training of deep neural networks. We establish the convergence result of a vanilla SBPG without Lipschitz smoothness assumption for

nonconvex problems. Moreover, we propose a momentum-based variant, denoted as MSBPG, and prove that it offers improved convergence properties compared to vanilla SBPG, particularly through a relaxed requirement on the mini-batch size. Both methods exhibit sample complexity of $\mathcal{O}(\epsilon^{-4})$, which matches the optimal bound for stochastic first order methods (Arjevani et al., 2023). We apply MSBPG to the training of deep neural networks with a polynomial kernel function, which ensures the smooth adaptivity of the loss function. Through an implicit reformulation, we observe that MSBPG enhances the robustness of neural network training by mitigating the gradient explosion phenomenon. For numerical illustrations, we conduct experiments on quadratic inverse problems (QIP) and testify vanilla SBPG's robustness to stepsize selection and initial point scaling. We also conduct extensive experiments on training deep neural networks for image classification and LSTMs for language modeling by employing MSBPG, which is well-suited for solving large-scale problems. Experimental results on representative benchmarks show that MSBPG converges to stationary points and, in some cases, nearly achieves the global minimum (i.e., training loss approaches zero). Moreover, MSBPG achieves comparable generalization performance to widely-used optimization algorithms, such as SGD (Robbins and Monro, 1951), Adam (Kingma and Ba, 2014), and AdamW (Loshchilov and Hutter, 2017), and in some instances, even outperforms these methods. The polynomial kernel function employed contributes to improved algorithmic stability compared to standard SGD, which may partly explain the good generalization performance of MSBPG observed in our experiments (see Appendix B for details). As the primary focus of this paper is on the optimization properties of Bregman-type methods, we only provide a preliminary analysis of algorithmic stability in the appendix. Furthermore, compared with standard SGD, SBPG/MSBPG is more robust to large stepsize and initial point scaling, which are the common reasons behind gradient explosion.

To summarize, our contributions are as follows:

1. **Development of SBPG for General Nonconvex Composite Problems:** We investigate Stochastic Bregman Proximal Gradient (SBPG) method to solve nonconvex problems without finite-sum structure, which employs Bregman distance to handle the non-Lipschitz gradient continuity. we establish its convergence properties along with optimal sample complexity of $\mathcal{O}(\epsilon^{-4})$. Furthermore, we propose a momentum-based variant, MSBPG, which improves the convergence property by relaxing the mini-batch size requirements, which is more suitable for large-scale problems. To our knowledge, this is the first integration of momentum techniques into a stochastic Bregman proximal gradient framework for nonconvex problems.

2. **Tailored MSBPG for Deep Neural Networks with Polynomial Kernel:** We apply MSBPG to training deep neural networks (DNN), which leverages on a suitable polynomial kernel function to ensure that the DNN's loss function is smooth adaptable with respect to the designed kernel function. Through an implicit reformulation, we observe that MSBPG is more robust than the traditional SGD in terms of stepsize selection and initialization. We highlight that MSBPG is a theoretically derived method that is able to ease the difficulty of selecting stepsize, mitigate gradient explosion, and maintain good generalization performance simultaneously. This distinguishes MSBPG from many existing techniques that rely on intuition and empirical observations.

3. **Empirical Evaluation of SBPG/MSBPG:** We demonstrate the efficiency and robustness of SBPG/MSBPG across various applications, including sparse quadratic inverse problems and large-scale deep neural networks. In the quadratic inverse problem, SBPG shows greater robustness to both stepsize and initial point selection. When training deep neural networks, MSBPG achieves comparable generalization performance to commonly used optimizers such as SGD, Adam, and AdamW, and in many cases, even outperforms them. Additionally, our method demonstrates robustness to stepsize selection and initialization. These results highlight the potential of MSBPG as a powerful tool for optimizing complex and large-scale deep neural networks, thus offering a promising direction for future research in this area.

The remainder of this paper is organized as follows. In Section 2, we present notation, some related preliminaries, and our problem setting. In Section 3, we first describe SBPG and establish its convergence results. Then, we propose a momentum-based SBPG (MSBPG) and prove its improved convergence properties. In Section 4, we adapt SBPG/MSBPG to the training of deep neural networks and analyze its capacity in mitigating gradient explosion. In Section 5, we present numerical experiments that demonstrate the efficiency and robustness of vanilla SBPG on quadratic inverse problems and MSBPG on training deep neural networks. Finally, we give some concluding remarks in Section 6. Additional supplementary materials are provided in the Appendix.

2. Preliminaries and Problem setting

In this paper, vectors are represented using boldface letters like $\boldsymbol{v}$, while scalars are represented using normal font. Given a proper, lower-semicontinuous function $F : \mathbb{R}^d \to \bar{\mathbb{R}} := [-\infty, \infty]$, $dom\, F = \{\boldsymbol{x} : F(\boldsymbol{x}) < \infty\}$. The Fenchel conjugate function of F is defined as $F^*(\boldsymbol{y}) = \sup\{\langle \boldsymbol{x}, \boldsymbol{y}\rangle - F(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^d\}$. Given a set $\mathcal{S} \subset \mathbb{R}^d$, $\bar{\mathcal{S}}$ denotes its closure, $int\, \mathcal{S}$ denotes the set of interior points. A function is of class $\mathcal{C}^k(\mathcal{S})$ if it is k times differentiable and the k-th derivative is continuous on $\mathcal{S}$. We say that F is level bounded if the set $\{x : F(\boldsymbol{x}) < \alpha\}$ is bounded for any real number α. Given a matrix A, $\mathrm{Vec}(A)$ denotes the vectorization of A by column order. $\mathrm{Mat}(\cdot)$ is the inverse operation of $\mathrm{Vec}(\cdot)$, which reshapes a vector back into its original matrix form. Define the operator $\mathrm{Diag}(\cdot)$ to map a vector into a diagonal matrix with diagonal elements equal to the corresponding entries of the vector. The Hadamard product is represented by the symbol $\circ$. If we use the notation $\|\cdot\|$ without any additional explanation, we assume that it refers to the Euclidean vector norm for vectors and the Frobenius matrix norm for matrices.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. Given a random variable $\boldsymbol{\xi}$ and a σ-algebra $\mathcal{F}$, we write $\boldsymbol{\xi} \lhd \mathcal{F}$ if $\boldsymbol{\xi}$ is measurable over $\mathcal{F}$. Let $\{\boldsymbol{\xi}_k\}_{k\geq 0}$ be a stochastic process, and $\{\mathcal{F}_k\}_{k\geq 0}$ be a filtration, where $\mathcal{F}_k$ defined by a σ-algebra $\mathcal{F}_k := \sigma(\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{k-1})$ on Ω. The conditional expectation is denoted by $\mathbb{E}[\cdot|\mathcal{F}_k]$. For simplicity, we use the notation $\mathbb{E}[\cdot]$ to denote $\mathbb{E}[\cdot|\mathcal{F}_\infty]$. The sequence $\{\boldsymbol{x}^k\}_{k\geq 0}$ generated by our proposed method is adapted to the filtration $\{\mathcal{F}_k\}_{k\geq 0}$, i.e. $\boldsymbol{x}^k \lhd \mathcal{F}_k$, for all $k \geq 0$. The notation $\widetilde{\nabla}_k$ represents an estimator of the exact gradient $\nabla F(\boldsymbol{x}^k)$, which satisties $\widetilde{\nabla}_k \lhd \mathcal{F}_{k+1}$. This estimator is applicable to both the vanilla and momentum cases. The stochastic error is denoted by $\boldsymbol{\varepsilon}_k = \nabla F(\boldsymbol{x}^k) - \widetilde{\nabla}_k$. The unbiasedness of the stochastic error $\boldsymbol{\varepsilon}_k$ is assumed throughout this paper, i.e., $\mathbb{E}[\boldsymbol{\varepsilon}_k|\mathcal{F}_k] = 0$.

The following supermartingale convergence lemma is a fundamental tool in the analysis of stochastic algorithms.

Lemma 1 *(Robbins and Monro, 1951) Let $\{y_k\}, \{u_k\}, \{a_k\}$ and $\{b_k\}$ be non-negative adapted processes with respect to the filtration $\{\mathcal{F}_k\}$ such that $\sum_{k=0}^{\infty} a_k < \infty, \sum_{k=0}^{\infty} b_k < \infty$, and for all k, $\mathbb{E}[y_{k+1} \mid \mathcal{F}_k] \leq (1 + a_k) y_k - u_k + b_k$ almost surely. Then, $\{y_k\}$ converges almost surely to a non-negative finite random variable and $\sum_{k=0}^{\infty} u_k < \infty$ almost surely.*

2.1 Smooth adaptable functions

In this subsection, we introduce the concept of smooth adaptivity, initially proposed by Bolte et al. (2018). This concept extends the idea of relative smoothness for convex functions, first introduced in Bauschke et al. (2017); Lu et al. (2018), and has since been applied in various contexts (Hanzely et al., 2021; Yang and Toh, 2021, 2022; Yang et al., 2024). We first give the definitions of kernel function and Bregman distance.

Definition 2 *(Kernel function and Bregman distance). Let $\mathcal{S}$ be a nonempty, convex and open subset of $\mathbb{R}^d$. A function $\phi : \mathbb{R}^d \to (-\infty, +\infty]$ is called a kernel function associated with $\mathcal{S}$ if it satisfies the following two conditions:*

1. *ϕ is proper, lower-semicontinuous and convex, with $dom\, \phi \subset \bar{\mathcal{S}}$, $dom\, \partial\phi = \mathcal{S}$.*
2. *$\phi \in \mathcal{C}^1(\mathcal{S})$ and $int\, dom\, \phi = \mathcal{S}$.*

Denote the class of kernel function associated with $\mathcal{S}$ by $\mathcal{M}(\mathcal{S})$. Given $\phi \in \mathcal{M}(\mathcal{S})$, the Bregman distance (Bregman, 1967) generated by ϕ is defined as $\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{y}) : dom\, \phi \times int\, dom\, \phi \to [0, +\infty)$, where

$$\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - \langle \nabla\phi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle.$$

Bregman distance measures the difference between the value of ϕ at $\boldsymbol{x}$ and its linear approximation at $\boldsymbol{y}$ based on the gradient of ϕ at $\boldsymbol{y}$. Some basic properties of Bregman distance can be found in (Chen and Teboulle, 1993; Teboulle, 2018). Some kernel functions commonly used in optimization are $\frac{1}{2}\|\boldsymbol{x}\|^2$, $\frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{\alpha}{4}\|\boldsymbol{x}\|^4$, $-\sum_{i=1}^{d} \log \boldsymbol{x}_i$ and $\sum_{i=1}^{d} \boldsymbol{x}_i \log \boldsymbol{x}_i$, where $\frac{1}{2}\|\boldsymbol{x}\|^2 \in \mathcal{M}(\mathbb{R}^d)$ recovers the classical half squared Euclidean distance. The kernel function $\frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{\alpha}{4}\|\boldsymbol{x}\|^4 \in \mathcal{M}(\mathbb{R}^d)$ has found applications in various problems, such as quadratic inverse problems, non-negative matrix factorization, and low-rank minimization (Bolte et al., 2018; Dragomir et al., 2021a). The entropy function $\sum_{i=1}^{d} \boldsymbol{x}_i \log \boldsymbol{x}_i \in \mathcal{M}(\mathbb{R}^d_{++})$ is commonly used in applications that involve probability constraints, where the resulting Bregman distance is known as the Kullback–Leibler (KL) divergence. Throughout the paper we will focus on the following pair of functions (f, ϕ) satisfying smooth adaptivity condition. We introduce this concept in the following definition:

Definition 3 *(Smooth adaptivity). Given a kernel function $\phi \in \mathcal{M}(\mathcal{S})$, a proper lower-semicontinuous function $f : \mathbb{R}^d \to (-\infty, +\infty]$ with $dom\, f \supset dom\, \phi$ that is $\mathcal{C}^1$ on $\mathcal{S}$. f is L-smooth adaptable with respect to ϕ if there exists $L > 0$, such that $L\phi + f$ and $L\phi - f$ are convex on $\mathcal{S}$.*

Alternative definition of smooth adaptivity is the two-side descent lemma (Bolte et al., 2018, Lemma 2.1). When both f and ϕ belong to $\mathcal{C}^2(\mathcal{S})$, we can verify their smooth adaptivity by comparing the Hessians of f and ϕ.

Lemma 4 *f is L-smooth adaptable with respect to $\phi \in \mathcal{M}(\mathcal{S})$, if and only if*

$$|f(\boldsymbol{x}) - f(\boldsymbol{y}) - \langle \nabla f(\boldsymbol{y}),\, \boldsymbol{x} - \boldsymbol{y}\rangle| \le L\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{y}),\ \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathit{int}\,\mathit{dom}\,\phi.$$

Moreover, when both f and ϕ belong to $\mathcal{C}^2(\mathit{int}\,\mathit{dom}\,\phi)$, then the above is equivalent to

$$\exists L > 0,\ L\nabla^2\phi(\boldsymbol{x}) \pm \nabla^2 f(\boldsymbol{x}) \succeq 0,\ \textit{for all } \boldsymbol{x} \in \mathit{int}\,\mathit{dom}\phi.$$

The following four-point identity is frequently employed in our proofs, and can be easily verified.

Lemma 5 *(Four points identity) Given points $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ and any convex function ϕ which is differentiable at $\boldsymbol{a}$ and $\boldsymbol{b}$, then*

$$\langle \nabla\phi(\boldsymbol{a}) - \nabla\phi(\boldsymbol{b}),\, \boldsymbol{c} - \boldsymbol{d}\rangle = \mathcal{D}_\phi(\boldsymbol{c}, \boldsymbol{b}) + \mathcal{D}_\phi(\boldsymbol{d}, \boldsymbol{a}) - \mathcal{D}_\phi(\boldsymbol{c}, \boldsymbol{a}) - \mathcal{D}_\phi(\boldsymbol{d}, \boldsymbol{b}).$$

2.2 Bregman Proximal Mapping

Throughout this paper, we make the following basic assumptions.

Assumption 1 *(Basic requirements). In problem* (1)*:*

A1. F is a proper lower-semicontinuous function with $\mathit{dom}\,\phi \subset \mathit{dom}\,F$, and it is $\mathcal{C}^1$ on $\mathit{int}\,C$.

A2. The Legendre kernel (Definition 6) $\phi \in \mathcal{M}(C)$ is μ-strongly convex for some $\mu > 0$. $F(\cdot)$ is L_F-smooth adaptable with respect to ϕ.

A3. R is is a proper, lower-semicontinuous and convex function with $\mathit{dom}\,R \cap \mathit{int}\,C \neq \emptyset$.

A4. $\inf_{\boldsymbol{x}\in\overline{C}}\{\Phi(\boldsymbol{x})\} > -\infty$.

Assumption 1 is a standard requirement for Bregman-type methods and is usually satisfied in practice. It ensures the well-definedness of Bregman-type methods, as shown in (Bolte et al., 2018; Latafat et al., 2022). We also recall the definition of the Legendre function in (Latafat et al., 2022), which makes additional supercoercive conditions on the concept in (Rockafellar, 1997).

Definition 6 *(Legendre kernel). Let $\phi : \overline{C} \to (-\infty, \infty]$ be a proper lower-semicontinuous convex function. It is called essentially smooth if $\mathit{int}\,\mathit{dom}\,\phi$ is nonempty and ϕ is differentiable on $\mathit{int}\,\mathit{dom}\,\phi$, moreover $\lim_{k\to\infty} \|\nabla\phi(\boldsymbol{x}^k)\| = \infty$ whenever $\{\boldsymbol{x}^k\}_{k\in\mathbb{N}}$ converges to a boundary point of $\mathit{dom}\,\phi$. The function ϕ is called Legendre function if it is essentially smooth, strictly convex on $\mathit{int}\,\mathit{dom}\,\phi$ and supercoercive, i.e. $\lim_{\|\boldsymbol{x}\|\to\infty} \frac{\phi(\boldsymbol{x})}{\|\boldsymbol{x}\|} = \infty$.*

Definition 7 *Given a nonempty convex open set C, a proper lower-semicontinuous convex function R and a Legendre kernel function $\phi \in \mathcal{M}(C)$, $\boldsymbol{x} \in \mathit{int}\,\mathit{dom}\,\phi$, we denote the Bregman proximal mapping by $\mathrm{Prox}_R^\phi := (\nabla\phi + \partial R)^{-1}\nabla\phi$, which is equivalent to*

$$\mathrm{Prox}_R^\phi(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{u}\in\overline{C}} \{R(\boldsymbol{u}) + \mathcal{D}_\phi(\boldsymbol{u}, \boldsymbol{x})\}. \tag{4}$$

Note that the objective function of (4) is strictly convex on $\operatorname{dom}\phi \cap \operatorname{dom} R$, therefore (4) has at most one solution. To ensure that (4) is well-defined, the following result claims that $\operatorname{Prox}^{\phi}_{\alpha R}(\boldsymbol{x})$ is well-defined for any $\alpha > 0$, and moreover $\operatorname{Prox}^{\phi}_{\alpha R}(\boldsymbol{x}) \in \operatorname{int}\operatorname{dom}\phi$ under standard assumptions. The proof can be found in Appendix A.

Lemma 8 *Suppose Assumption 1 holds. Then* (4) *has a unique solution. Moreover, the solution* $\operatorname{Prox}^{\phi}_{\alpha R}(\boldsymbol{x}) \in C$.

The following proposition for Bregman proximal mapping generalizes the nonexpansive property of the classical proximal mapping (in the case $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2$). This property is commonly used in convergence proofs. The proof of the following proposition can be found in Appendix A.

Proposition 9 *Suppose Assumption 1 holds. Let* $\boldsymbol{x}_i^+ := \operatorname{Prox}^{\phi}_{R}(\nabla\phi^*(\boldsymbol{x}_i))$, $i = 1, 2$. *Then* $\|\boldsymbol{x}_1^+ - \boldsymbol{x}_2^+\| \leq \frac{1}{\mu}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$.

In this paper, we make the assumption that R and ϕ are simple enough so that (4) either has a closed-form solution or admits an efficient subroutine to solve it. Using the definition of the Bregman proximal mapping, we can then define the Bregman gradient mapping associated with (1). This mapping measures the solution accuracy of the methods we propose. Note that ϕ is a Legendre kernel, which implies that $\phi^* \in \mathcal{C}^1(\mathbb{R}^d)$ is strictly convex and $(\nabla\phi)^{-1} = \nabla\phi^*$ (Rockafellar, 1997, Corollary 13.3.1, Theorem 26.5). Therefore, the following concept is well-defined.

Definition 10 (Bregman Gradient Mapping) *Given* $\alpha > 0$, *a nonempty convex open set* C *and a Legendre kernel function* $\phi \in \mathcal{M}(C)$, *the Bregman gradient mapping associated with* (1) *is defined as follows*

$$\mathcal{G}_\alpha(\boldsymbol{x}) = \frac{\boldsymbol{x} - \operatorname{Prox}^{\phi}_{\alpha R}\left(\nabla\phi^*(\nabla\phi(\boldsymbol{x}) - \alpha\nabla F(\boldsymbol{x}))\right)}{\alpha}.$$

To simplify notation, we use $\mathcal{G}(\boldsymbol{x})$ *to denote* $\mathcal{G}_1(\boldsymbol{x})$ *when* $\alpha = 1$.

When the kernel function $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2$, the resulting Bregman Gradient Mapping becomes equivalent to the classical Gradient Mapping (Nesterov, 2003, 2005), which measures the solution's accuracy for proximal gradient methods.

Definition 11 *(Limiting subdifferential (Rockafellar and Wets, 1998, Definition 8.3)) Consider a function* $f : \mathbb{R}^d \to \bar{\mathbb{R}}$ *and a point* $\boldsymbol{x}$, *the regular subdifferential is defines as*

$$\hat{\partial} f(\boldsymbol{x}) = \{\boldsymbol{v} : f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{v}, \boldsymbol{y} - \boldsymbol{x}\rangle + o(\|\boldsymbol{y} - \boldsymbol{x}\|)\}.$$

The limiting subdifferential is defined as

$$\partial f(\boldsymbol{x}) = \{\boldsymbol{v} : \boldsymbol{x}_n \to \boldsymbol{x}, f(\boldsymbol{x}_n) \to f(\boldsymbol{x}), \boldsymbol{v}_n \in \hat{\partial} f(\boldsymbol{x}_n), \text{ and } \boldsymbol{v}_n \to \boldsymbol{v}\}.$$

Now, we restrict our attention on the case $C = \mathbb{R}^d$. By Fermat's rule (Rockafellar and Wets, 1998, Theorem 10.1), the set of critical point of Φ is given by

$$\operatorname{crit}\Phi = \left\{\boldsymbol{x} \in \mathbb{R}^d : \ 0 \in \partial\Phi(\boldsymbol{x}) \equiv \nabla F(\boldsymbol{x}) + \partial R(\boldsymbol{x})\right\}.$$

The Bregman Gradient Mapping can also be used to evaluate the solution accuracy for Bregman methods. Let $\boldsymbol{x}^+ = \mathrm{Prox}^{\phi}_{\alpha R}(\nabla\phi^*(\nabla\phi(\boldsymbol{x}) - \alpha\nabla F(\boldsymbol{x})))$. From Definition 10 and equation (4), it can be easily verified by definition that $0 \in \partial\Phi(\boldsymbol{x}) \Leftrightarrow 0 = \mathcal{G}_\alpha(\boldsymbol{x})$. Hence, $0 \in \partial\Phi(\boldsymbol{x}^+)$ for any $\alpha > 0$. The proof of this result is omitted for brevity. Furthermore, if $\nabla\phi$ is L_ϕ-Lipschitz continuous, then the following proposition holds, implying that $\|\mathcal{G}_\alpha(\boldsymbol{x})\|$ can be used as a reasonable criterion to measure the accuracy of $\boldsymbol{x}$.

Proposition 12 *Suppose Assumption 1 holds and that $\nabla\phi$ is L_ϕ Lipschitz continuous. Then, we have the following inequality:*

$$\mathrm{dist}\left(0, \partial\Phi(\boldsymbol{x}^+)\right) \leq (1 + \alpha L_F) L_\phi \|\mathcal{G}_\alpha(\boldsymbol{x})\|.$$

We also define the stochastic counterpart of Definition 10, which is commonly utilized to evaluate the accuracy of solutions for nonconvex stochastic proximal gradient methods, as discussed in (Ghadimi et al., 2016).

Definition 13 *(Stochastic Bregman Gradient Mapping). Given $\alpha > 0$ a nonempty convex open set C and a Legendre kernel function $\phi \in \mathcal{M}(C)$, the stochastic Bregman gradient mapping associated with* (1) *is defined as follows*

$$\widetilde{\mathcal{G}}_\alpha(\boldsymbol{x}) := \frac{\boldsymbol{x} - \mathrm{Prox}^{\phi}_{\alpha R}\left(\nabla\phi^*\left(\nabla\phi(\boldsymbol{x}) - \alpha\widetilde{\nabla}\right)\right)}{\alpha}, \text{ where } \widetilde{\nabla} \text{ is an estimator of } \nabla F(\boldsymbol{x}).$$

3. Stochastic Bregman Proximal Gradient Method

In this section, we will study the Stochastic Bregman Proximal Gradient method (SBPG) with the following update scheme:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}\in\overline{C}}{\mathrm{argmin}}\ R(\boldsymbol{x}) + \langle\widetilde{\nabla}_k, \boldsymbol{x} - \boldsymbol{x}^k\rangle + \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{x}^k). \tag{SBPG}$$

We call the above method as "vanilla" SBPG in this section, meaning that the method we study is a basic version without any additional techniques such as variance reduction, momentum, etc., except for the use of mini-batches. In this case, we suppose the following assumptions.

Assumption 2 *(Noise requirement). The estimator satisfies the following two conditions:*

$$\mathbb{E}[\widetilde{\nabla}_k|\mathcal{F}_k] = \nabla F(\boldsymbol{x}_k) \quad \textit{and} \quad \mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] \leq \frac{\sigma^2}{m_k},$$

where m_k is the size of the mini-batch in the k-th iteration.

Note that we do not assume a finite-sum structure for $F(\boldsymbol{x})$ in this section. The solution of (SBPG) can be written in the form of the Bregman proximal mapping. This is stated in the following proposition.

Proposition 14 *Suppose Assumption 1 holds. Then the solution of* (SBPG) *can be written as the following Bregman proximal mapping:*

$$\boldsymbol{x}^{k+1} = \mathrm{Prox}^{\phi}_{\alpha_k R}\left(\nabla\phi^*\left(\nabla\phi(\boldsymbol{x}^k) - \alpha_k\widetilde{\nabla}_k\right)\right).$$

Proof From the optimality condition of the main subproblem (SBPG), we have

$$0 \in \partial R(\boldsymbol{x}^{k+1}) + \widetilde{\nabla}_k + \frac{1}{\alpha_k}\left(\nabla\phi(\boldsymbol{x}^{k+1}) - \nabla\phi(\boldsymbol{x}^k)\right).$$

Let $\boldsymbol{u}^{k+1} := \mathrm{Prox}^{\phi}_{\alpha_k R}\left(\nabla\phi^*\left(\nabla\phi(\boldsymbol{x}^k) - \alpha_k\widetilde{\nabla}_k\right)\right)$. From the definition of Bregman proximal mapping, we have

$$\boldsymbol{u}^{k+1} = \arg\min_{\boldsymbol{u}}\ \left\{\alpha_k R(\boldsymbol{u}) + \mathcal{D}_\phi\left(\boldsymbol{u}, \nabla\phi^*\left(\nabla\phi(\boldsymbol{x}^k) - \alpha_k\widetilde{\nabla}_k\right)\right)\right\},$$

which is equivalent to

$$0 \in \alpha_k\partial R(\boldsymbol{u}^{k+1}) + \nabla\phi(\boldsymbol{u}^{k+1}) - \nabla\phi\big(\nabla\phi^*(\nabla\phi(\boldsymbol{x}^k) - \alpha_k\widetilde{\nabla}_k)\big).$$

Note that the function ϕ^* is the Fenchel conjugate of the Legendre kernel ϕ, which implies that $\nabla\phi(\nabla\phi^*(\boldsymbol{w})) = \boldsymbol{w}$ for all $\boldsymbol{w} \in \mathbb{R}^d$, as stated in (Rockafellar, 1997, Corollary 13.3.1, Theorem 26.5). Furthermore, since the objective function in (SBPG) is strictly convex, there exists a unique solution to the inclusion above. By comparing the two inclusions, we can conclude that $\boldsymbol{u}^{k+1} = \boldsymbol{x}^{k+1}$. ■

Based on Proposition 14 and definition of the definition of $\widetilde{\mathcal{G}}_\alpha(\boldsymbol{x})$, we can easily observe that $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)$. We can derive the following proposition, which bounds the difference between $\mathcal{G}_\alpha(\boldsymbol{x})$ and $\widetilde{\mathcal{G}}_\alpha(\boldsymbol{x})$ directly from Proposition 9. The proof is omitted for brevity.

Proposition 15 *Suppose Assumption 1 holds. At the k-th step, we have the estimation:*

$$\|\mathcal{G}_{\alpha_k}(\boldsymbol{x}^k) - \widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\| \le \frac{1}{\mu}\|\nabla F(\boldsymbol{x}^k) - \widetilde{\nabla}_k\| = \frac{\|\boldsymbol{\varepsilon}_k\|}{\mu},$$

where $\boldsymbol{\varepsilon}_k = \nabla F(\boldsymbol{x}^k) - \widetilde{\nabla}_k$.

Before presenting the main convergence result, we state the following one-step descent lemma below.

Lemma 16 *Suppose Assumption 1 holds. The sequence generated by SBPG satisfies the following condition:*

$$\Phi(\boldsymbol{x}^{k+1}) \le \Phi(\boldsymbol{x}^k) - \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) - \left(\frac{1}{\alpha_k} - L_F\right)\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k) + \langle\boldsymbol{\varepsilon}_k,\, \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\rangle.$$

Proof By the optimality condition of (SBPG), we obtain that

$$0 \in \partial R(\boldsymbol{x}^{k+1}) + \widetilde{\nabla}_k + \frac{1}{\alpha_k}\left(\nabla\phi(\boldsymbol{x}^{k+1}) - \nabla\phi(\boldsymbol{x}^k)\right).$$

Appealing to the convexity of R, we have

$$R(\boldsymbol{x}) - R(\boldsymbol{x}^{k+1}) \ge \left\langle -\widetilde{\nabla}_k - \frac{1}{\alpha_k}\left(\nabla\phi(\boldsymbol{x}^{k+1}) - \nabla\phi(\boldsymbol{x}^k)\right),\, \boldsymbol{x} - \boldsymbol{x}^{k+1}\right\rangle.$$

By the four points identity and the definition of $\boldsymbol{\varepsilon}_k$, we get

$$R(\boldsymbol{x})-R(\boldsymbol{x}^{k+1}) \geq \frac{1}{\alpha_k}\left[\mathcal{D}_\phi(\boldsymbol{x}^{k+1},\boldsymbol{x}^k)+\mathcal{D}_\phi(\boldsymbol{x},\boldsymbol{x}^{k+1})-\mathcal{D}_\phi(\boldsymbol{x},\boldsymbol{x}^k)\right]-\langle\nabla F(\boldsymbol{x}^k),\,\boldsymbol{x}-\boldsymbol{x}^{k+1}\rangle+\langle\boldsymbol{\varepsilon}_k,\,\boldsymbol{x}-\boldsymbol{x}^{k+1}\rangle.$$

Set $\boldsymbol{x}=\boldsymbol{x}^k$ in the above inequality, we have the following inequality:

$$R(\boldsymbol{x}^k)-R(\boldsymbol{x}^{k+1}) \geq \frac{1}{\alpha_k}\left[\mathcal{D}_\phi(\boldsymbol{x}^{k+1},\boldsymbol{x}^k)+\mathcal{D}_\phi(\boldsymbol{x}^k,\boldsymbol{x}^{k+1})\right]-\langle\nabla F(\boldsymbol{x}^k),\,\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\rangle+\langle\boldsymbol{\varepsilon}_k,\,\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\rangle.$$

By the smooth adaptivity of F, we have

$$F(\boldsymbol{x}^{k+1}) \leq F(\boldsymbol{x}^k)+\langle\nabla F(\boldsymbol{x}^k),\,\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\rangle+L_F\mathcal{D}_\phi(\boldsymbol{x}^{k+1},\boldsymbol{x}^k).$$

Combining the above two inequalities above, we complete the proof. ■

3.1 Convergence analysis of SBPG

In this subsection, we establish the convergence results for SBPG, which is an extension of the convergence result in (Ghadimi et al., 2016), in which the lassical Lipschitz gradient assumption is required. In many literature, the bounded sequence assumption is often required in the convergence analysis of stochastic algorithms. However, in this section, we relax this assumption and prove that under a certain condition, the sequence generated by (SBPG) is bounded almost surely. We need the following result to bound the stochastic error term $\langle\boldsymbol{\varepsilon}_k,\,\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\rangle$ in Lemma 16.

Lemma 17 *Suppose Assumption 1, 2 hold. We have the following estimation of the error term:*

$$\mathbb{E}\left[\langle\boldsymbol{\varepsilon}_k,\,\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\rangle\right] \leq \frac{\alpha_k}{\mu}\mathbb{E}[\|\boldsymbol{\varepsilon}_k\|^2] \leq \frac{\alpha_k\sigma^2}{\mu m_k}.$$

Proof Define $\bar{\boldsymbol{x}}^{k+1} := \mathrm{Prox}^{\phi}_{\alpha_k R}(\nabla\phi^*(\nabla\phi(\boldsymbol{x}^k)-\alpha_k\nabla F(\boldsymbol{x}^k)))$. By Proposition 14 and the optimality condition for $\bar{\boldsymbol{x}}^{k+1}$, we have

$$0 \in \partial R(\bar{\boldsymbol{x}}^{k+1})+\nabla F(\boldsymbol{x}^k)+\frac{1}{\alpha_k}(\nabla\phi(\bar{\boldsymbol{x}}^{k+1})-\nabla\phi(\boldsymbol{x}^k)).$$

Similarly,

$$0 \in \partial R(\boldsymbol{x}^{k+1})+\widetilde{\nabla}_k+\frac{1}{\alpha_k}(\nabla\phi(\boldsymbol{x}^{k+1})-\nabla\phi(\boldsymbol{x}^k)).$$

By the monotonicity of ∂R and Lemma 5, we have

$$\left\langle\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^{k+1},\,-\boldsymbol{\varepsilon}_k-\frac{1}{\alpha_k}(\nabla\phi(\bar{\boldsymbol{x}}^{k+1})-\nabla\phi(\boldsymbol{x}^{k+1}))\right\rangle \geq 0.$$

Therefore,

$$\langle\boldsymbol{x}^{k+1}-\bar{\boldsymbol{x}}^{k+1},\,\boldsymbol{\varepsilon}_k\rangle \geq \langle\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^{k+1},\,\frac{1}{\alpha_k}(\nabla\phi(\bar{\boldsymbol{x}}^{k+1})-\nabla\phi(\boldsymbol{x}^{k+1}))\rangle \geq \frac{\mu}{\alpha_k}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^{k+1}\|^2.$$

By Cauchy-Schwarz inequality, we get $\|\bar{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^{k+1}\| \leq \frac{\alpha_k}{\mu}\|\boldsymbol{\varepsilon}_k\|$.

Now, we are ready to prove Lemma 17. From the definition, we know that $\bar{\boldsymbol{x}}^{k+1} \lhd \mathcal{F}_k$. Therefore, $\mathbb{E}[\langle \boldsymbol{\varepsilon}_k, \boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k+1}\rangle] = \mathbb{E}[\mathbb{E}[\langle \boldsymbol{\varepsilon}_k, \boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k+1}\rangle|\mathcal{F}_k]] = \mathbb{E}[\langle \mathbb{E}[\boldsymbol{\varepsilon}_k|\mathcal{F}_k], \boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k+1}\rangle] = 0$, where the first equality is from the tower rule of conditional expectation, the second comes from the fact that $\boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k+1} \lhd \mathcal{F}_k$. Hence,

$$\mathbb{E}\left[\langle \boldsymbol{\varepsilon}_k, \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\rangle\right] = \mathbb{E}\left[\langle \boldsymbol{\varepsilon}_k, \boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^{k+1}\rangle\right] - \mathbb{E}\left[\langle \boldsymbol{\varepsilon}_k, \boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k+1}\rangle\right] \leq \frac{\alpha_k}{\mu}\mathbb{E}[\|\boldsymbol{\varepsilon}_k\|^2] \leq \frac{\alpha_k\sigma^2}{\mu m_k},$$

which completes the proof. ■

Lemma 18 (Bounded sequence) *Suppose Assumption 1, 2 hold. If* $\sum_k \frac{\alpha_k}{m_k} < \infty$, $\sup_k \alpha_k \leq \bar{\alpha} < \frac{1}{L_F}$, *then,*

1. $\sum_{k=0}^{\infty} \mathbb{E}[\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k)] < \infty$.
2. *If* Φ *is level bounded, then* $\{\boldsymbol{x}^k\}_{k\geq 0}$ *is bounded almost surely.*

Proof By Cauchy-Young inequality, we have

$$|\langle \boldsymbol{\varepsilon}_k, \boldsymbol{x}^k - \boldsymbol{x}^{k+1}\rangle| \leq \frac{\mu}{2\alpha_k}\|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\|^2 + \frac{\alpha_k}{2\mu}\|\boldsymbol{\varepsilon}_k\|^2 \leq \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) + \frac{\alpha_k}{2\mu}\|\boldsymbol{\varepsilon}_k\|^2.$$

By Lemma 16, we have

$$\left(\frac{1}{\alpha_k} - L_F\right)\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k) \leq \Phi(\boldsymbol{x}^k) - \Phi(\boldsymbol{x}^{k+1}) + \frac{\alpha_k}{2\mu}\|\boldsymbol{\varepsilon}_k\|^2. \tag{5}$$

Taking conditional expectation for both sides of (5), we get

$$\mathbb{E}\left[\left(\frac{1}{\alpha_k} - L_F\right)\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k)|\mathcal{F}_k\right] \leq \Phi(\boldsymbol{x}^k) - \mathbb{E}[\Phi(\boldsymbol{x}^{k+1})|\mathcal{F}_k] + \frac{\alpha_k}{2\mu}\mathbb{E}[\|\boldsymbol{\varepsilon}_k\|^2|\mathcal{F}_k].$$

Since $\sum_{k\geq 0}\frac{\alpha_k}{2\mu}\mathbb{E}[\|\boldsymbol{\varepsilon}_k\|^2|\mathcal{F}_k] \leq \sum_{k\geq 0}\frac{\alpha_k\sigma^2}{2\mu m_k} < \infty$, applying Theorem 1, we have that $\Phi(\boldsymbol{x}^k)$ converges and $\sum_{k\geq 0}\mathbb{E}\left[\left(\frac{1}{\alpha_k} - L_F\right)\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k)|\mathcal{F}_k\right] < \infty$ almost surely. By the tower rule of conditional expectation, we have $\sum_{k=0}^{\infty}\mathbb{E}[\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k)] < \infty$. Since $\Phi(\boldsymbol{x}^k)$ converges almost surely, thus $\{\Phi(\boldsymbol{x}^k)\}_{k\geq 0}$ is bounded almost surely. By the level boundness of Φ, we deduce that $\{\boldsymbol{x}^k\}_{k\geq 0}$ is bounded almost surely. ■

Now, we present our main convergence result for the vanilla SBPG, which is in the sense of expectation.

Theorem 19 (Convergence result in expectation) *Suppose Assumption 1, 2 hold,* $\alpha_k < \frac{1}{L_F}\min\{1, \frac{1}{\mu}\}$. *Define a random variable* r *with the distribution* $\mathbb{P}\{r = k\} = \frac{\alpha_k}{\sum_{k=0}^{N-1}\alpha_k}$ *for* $k = 0, ..., N-1$. *Then,*

$$\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_r}(\boldsymbol{x}^r)\|^2] \leq \frac{2\Delta_0 + 2\sum_{k=0}^{N-1}\frac{\alpha_k\sigma^2}{\mu m_k}}{\mu\sum_{k=0}^{N-1}\alpha_k}, \tag{6}$$

where $\Delta_0 := \Phi(\boldsymbol{x}^0) - \Phi^*$. *If* $\sum_k \frac{\alpha_k}{m_k} < +\infty$ *and* $\sum_k \alpha_k = +\infty$, *then the right hand side of* (6) *converges to zero. Moreover, if* Φ *is level bounded, then the sequence* $\{\boldsymbol{x}^k\}_{k\geq 0}$ *is bounded almost surely.*

Proof Note that $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k \widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)$ and by the strongly convexity of ϕ, Lemma 16 yields

$$\begin{aligned}\mu(\alpha_k - \frac{L_F\alpha_k^2}{2})\|\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\|^2 &\leq \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) + \left(\frac{1}{\alpha_k} - L_F\right)\mathcal{D}_\phi(\boldsymbol{x}^{k+1}, \boldsymbol{x}^k)\\ &\leq \Phi(\boldsymbol{x}^k) - \Phi(\boldsymbol{x}^{k+1}) + \langle \boldsymbol{\varepsilon}_k,\, \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\rangle.\end{aligned}$$

Taking expectations, telescoping from $k = 0...N-1$, and using Lemma 17, we obtain

$$\sum_{k=0}^{N-1} \mu(\alpha_k - \frac{\mu L_F\alpha_k^2}{2})\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\|^2] \leq \Phi(\boldsymbol{x}^0) - \Phi(\boldsymbol{x}^N) + \sum_{k=0}^{N-1}\frac{\alpha_k\sigma^2}{\mu m_k}. \tag{7}$$

By utilizing the inequality $\alpha_k - \frac{\mu L_F\alpha_k^2}{2} \geq \frac{\alpha_k}{2}$, the condition $\Phi(\boldsymbol{x}^N) \geq \Phi^*$, and considering the definition of the random variable r, we can derive (6) from (7). ■

Remark 20 *We give some remarks for Theorem 19.*

1. *The mini-batch size plays a crucial role in ensuring convergence, as it allows us to control the stochastic error term in Lemma 16 and provide a bound for* $\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\|^2]$ *that converges to* 0 *as* k *tends to infinity. If* $m_k = 1$ *for all* k, *then the upper bound for* $\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\|^2]$ *will not converge to* 0, *no matter how* $\{\alpha_k\}$ *is selected.*

2. *In (Ghadimi et al., 2016), a similar convergence result is established for mini-batch stochastic proximal gradient methods, but our analysis differs in a crucial aspect in that we do not assume the Lipschitz continuity of* $F(\boldsymbol{x})$. *Instead, we rely on the smooth adaptivity of* $F(\boldsymbol{x})$, *which is a more relaxed assumption. Additionally, we provide specific conditions on the stepsizes* $\{\alpha_k\}$ *and mini-batch sizes* $\{m_k\}$ *that guarantee the convergence of* $\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\|^2]$ *to* 0, *as well as the almost sure boundedness of the sequence* $\{\boldsymbol{x}^k\}$.

3. *We now provide a specific choice of* $\{\alpha_k\}$ *and* $\{m_k\}$ *to establish a convergence rate in terms of expected stationarity. For given positive constants* $c_1, c_2, \gamma \in (0,\infty)$ *and* $\delta \in [0,1)$, *we choose*

$$\alpha_k = \frac{c_1}{(k+1)^\delta},\ m_k = \lceil c_2(k+1)^\gamma \rceil.$$

Under these choices, we have $\sum_{i=0}^k \alpha_i = \mathcal{O}(k^{1-\delta})$ *and* $\sum_{i=0}^k \frac{\alpha_k}{m_k} = \mathcal{O}(k^{1-(\delta+\gamma)})$. *As a result, the RHS of* (6) *has the bound of* $\mathcal{O}(k^{-(1-\delta)} + k^{-\gamma})$. *When* $\delta = 0$ *and* $\gamma \geq 1$, *the order is* $\mathcal{O}(k^{-1})$ *which recovers the optimal rate of deterministic first order method. In this case, the mini-batch size* m_k *increases rapidly and the variance in the stochastic gradient reduces to zero quickly, so the algorithm behaves like a deterministic algorithm. However, such a favorable convergence rate comes at the expense of a heavy computational burden and high memory cost in each iterative step.*

Next, we present the sample complexity of the SBPG method. We define each computation of $\nabla f(\boldsymbol{x}, \boldsymbol{\xi})$ for a fix $\boldsymbol{\xi}$ as an oracle evaluation.

Corollary 21 *Given an accuracy level $\epsilon > 0$ and a positive constant $\gamma > 0$, choose $\alpha_k = \alpha < \frac{1}{L_F}\min\{1, \frac{1}{\mu}\}$. Then to achieve an ϵ-stationary point in expectation, at most $\bar{N} := \left\lceil \max\left\{\gamma^2, \frac{4\sigma^4}{\mu^4\epsilon^4}\left(\frac{2\Delta_0\mu\gamma}{\alpha\sigma^2} + \frac{1}{\gamma}\right)^2\right\}\right\rceil$ oracle evaluations are required. In this case, we need to set $m_k = m := \lceil \min\{\gamma\sqrt{\bar{N}}, \bar{N}\}\rceil$ for all k.*

Proof Let the total number of oracle evaluations be $\bar{N}$. Then $N = \left\lfloor \frac{\bar{N}}{m} \right\rfloor$ and $N \geq \frac{\bar{N}}{2m}$. We have that

$$\begin{aligned}\text{RHS of (6)} \leq & \frac{4\Delta_0 m}{\alpha\mu\bar{N}} + \frac{2\sigma^2}{\mu^2 m} \leq \frac{4\Delta_0}{\alpha\mu\bar{N}}\left(\gamma\sqrt{\bar{N}}\right) + \frac{2\sigma^2}{\mu^2}\max\left\{\frac{1}{\gamma\sqrt{\bar{N}}}, \frac{1}{\bar{N}}\right\} \\ \leq & \frac{2\sigma^2}{\mu^2\sqrt{\bar{N}}}\left(\frac{2\Delta_0\mu\gamma}{\alpha\sigma^2} + \frac{1}{\gamma}\right),\end{aligned}$$

where $\Delta_0 := \Phi(\boldsymbol{x}^0) - \Phi(\boldsymbol{x}^*)$. By the choice of $\bar{N}$, we have RHS of (6) $\leq \epsilon^2$. This completes the proof. ■

This oracle complexity matches the lower bound provided by (Arjevani et al., 2023), thus it cannot be improved in general.

3.2 Momentum based Stochastic Bregman Gradient Descent Method

Remark 20 and Corollary 21 suggest that a large mini-batch size m_k is necessary to achieve a small stationarity error. However, employing a large mini-batch size in each iteration can be computationally expensive and memory inefficient, especially in modern large-scale problems such as deep neural network training. In this part, we resort to the momentum technique to address this issue. Specifically, we consider using a stochastic moving average estimator (SMAE) for the true gradient given by:

$$\boldsymbol{v}^k = (1-\beta_k)\boldsymbol{v}^{k-1} + \beta_k\widetilde{\nabla}_k, \quad \text{where} \quad \mathbb{E}[\widetilde{\nabla}_k|\mathcal{F}_k] = \nabla F(\boldsymbol{x}^k), \tag{8}$$

where $\boldsymbol{v}^{k-1}$ can be viewed as the momentum which contains the information of all historical stochastic gradients, and $\mathbb{E}[\|\widetilde{\nabla}_k\|^2|\mathcal{F}_k] \leq \frac{\sigma^2}{m_k}$. We expect that incorporating the SMAE technique can achieve a certain level of variance reduction without requiring an increase in the mini-batch size or the computation of the full gradient. In our approach, we utilize the gradient estimator $\boldsymbol{v}^k$ within SBPG, and we refer to the resulting method as MSBPG. Specifically, we consider the following update scheme:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}\in\overline{C}}{\operatorname{argmin}}\ R(\boldsymbol{x}) + \langle \boldsymbol{v}^k, \boldsymbol{x} - \boldsymbol{x}^k\rangle + \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{x}^k). \tag{9}$$

We need the following assumption that the difference of gradients of F can be bounded by the Bregman distance.

Assumption 3 *There exists $\kappa > 0$, such that $\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|^2 \leq \kappa\mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{y})$ for all $\boldsymbol{x} \in dom\,\phi$, $\boldsymbol{y} \in int\,dom\,\phi$.*

Remark 22 *This assumption generalizes the case of Lipschitz kernel function. If F is L_F-smooth adaptable to ϕ, it can be easily shown that if ϕ has L_ϕ-Lipschitz gradient, this assumption holds for $\kappa \geq \frac{2L_F^2 L_\phi^2}{\mu}$. In this paper, we are particularly interested in polynomial kernel functions. For functions with polynomially bounded growth rates, this assumption is not restrictive. For example, consider the one-dimensional objective function $F(x) = \frac{1}{4}x^4$ and the kernel function $\phi(x) = \frac{1}{2}x^2 + \frac{1}{8}x^8$. Then, by (Lu et al., 2018, Proposition 2.1), we know that F is smooth adaptable with respect to ϕ. Simple algebra shows that $\mathcal{D}_\phi(x,y) = \frac{1}{8}(x-y)^2(x^6+2x^5y+3x^4y^2+4x^3y^3+5x^2y^4+6xy^5+7y^6+4)$ and $(F'(x)-F'(y))^2 = (x-y)^2(x^2+xy+y^2)^2$. Numerical computation shows that $(x^6+2x^5y+3x^4y^2+4x^3y^3+5x^2y^4+6xy^5+7y^6+4)-(x^2+xy+y^2)^2 \geq 3.71$. Therefore, $(F'(x)-F'(y))^2 \leq 8\mathcal{D}_\phi(x,y)$, which holds globally for any x and y in $\mathbb{R}^d$.*

Next, we present a recursion lemma that allows us to estimate the accuracy of the SMAE. While similar lemmas have been proposed in the literature, such as in (Wang et al., 2017), their bounds are not directly applicable in the Bregman setting. As a result, we have developed a version of the recursion lemma that is tailored to our specific context.

Lemma 23 *The following recursion holds*

$$\mathbb{E}[\|\boldsymbol{v}^k-\nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] \leq (1-\beta_k)\|\boldsymbol{v}^{k-1}-\nabla F(\boldsymbol{x}^{k-1})\|^2+\beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k-\nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k]+\frac{\|\nabla F(\boldsymbol{x}^{k-1})-\nabla F(\boldsymbol{x}^k)\|^2}{\beta_k}.$$

Proof Note that $\boldsymbol{v}^k - \nabla F(\boldsymbol{x}^k) = (1-\beta_k)(\boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1})) + (1-\beta_k)(\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)) + \beta_k(\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k))$, and $\mathbb{E}[\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)|\mathcal{F}_k] = 0$. Then we have

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{v}^k - \nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] \\
&= (1-\beta_k)^2\|\boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1})\|^2 + (1-\beta_k)^2\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2 + \\
&\quad \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] + 2(1-\beta_k)^2\langle \boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1}),\, \nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\rangle \\
&\leq (1-\beta_k)^2\|\boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1})\|^2 + (1-\beta_k)^2\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2 + \\
&\quad \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] + \beta_k(1-\beta_k)\|\boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1})\|^2 + \frac{(1-\beta_k)^3}{\beta_k}\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2 \\
&= (1-\beta_k)\|\boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1})\|^2 + \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] + \frac{(1-\beta_k)^2\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2}{\beta_k} \\
&\leq (1-\beta_k)\|\boldsymbol{v}^{k-1} - \nabla F(\boldsymbol{x}^{k-1})\|^2 + \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2|\mathcal{F}_k] + \frac{\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2}{\beta_k}.
\end{aligned}$$

This completes the proof. ■

Now we are ready to provide the convergence result for our momentum based SBPG.

Theorem 24 *Suppose Assumption 1, 2 and 3 hold. Let $\alpha_k = c\mu\beta_{k+1}$ for any $c \in (0, \frac{1}{2\sqrt{\mu\kappa}}]$. Then, it holds that*

$$\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_r}(\boldsymbol{x}^r)\|^2] \leq \frac{\Delta_0 + c\|\boldsymbol{v}_0 - \nabla F(\boldsymbol{x}^0)\|^2 + \sum_{k=0}^{N-1}\frac{\alpha_k^2\sigma^2}{c\mu^2 m_k}}{\sum_{k=0}^{N-1}\frac{\mu\alpha_k}{8}}, \tag{10}$$

where r is a random variable with distribution $\mathbb{P}\{r=k\} = \frac{\alpha_k}{\sum_{k=0}^{N-1}\alpha_k}$, for $k = 0, ..., N-1$.

Proof From Lemma 16 and Cauchy-Young's inequality, we have

$$\begin{aligned}\Phi(\boldsymbol{x}^{k+1}) &\leq \Phi(\boldsymbol{x}^k) - \frac{1}{\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) + \frac{\mu}{4\alpha_k}\|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\|^2 + \frac{\alpha_k}{\mu}\|\boldsymbol{\varepsilon}_k\|^2 \\ &\leq \Phi(\boldsymbol{x}^k) - \frac{1}{2\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) + \frac{\alpha_k}{\mu}\|\boldsymbol{\varepsilon}_k\|^2.\end{aligned}$$

where we have defined $\boldsymbol{\varepsilon}_k := \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k$. Summing the above inequality over $k = 0, \ldots, N-1$ and rearranging the terms, we get

$$\sum_{k=0}^{N-1} \frac{1}{2\alpha_k}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) \leq \Phi(\boldsymbol{x}^0) - \Phi^* + \sum_{k=0}^{N-1} \frac{\alpha_k}{\mu}\|\boldsymbol{\varepsilon}_k\|^2.$$

By applying Lemma 23, we can obtain the following inequality:

$$\beta_k\mathbb{E}[\|\boldsymbol{\varepsilon}_{k-1}\|^2] \;\leq\; \mathbb{E}[\|\boldsymbol{\varepsilon}_{k-1}\|^2] - \mathbb{E}[\|\boldsymbol{\varepsilon}_k\|^2] + \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2] + \mathbb{E}\left[\frac{\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2}{\beta_k}\right].$$

Hence

$$\sum_{k=0}^{N-1} \beta_{k+1}\mathbb{E}[\|\boldsymbol{\varepsilon}_k\|^2] = \sum_{k=1}^{N} \beta_k\mathbb{E}[\|\boldsymbol{\varepsilon}_{k-1}\|^2] \leq \|\boldsymbol{\varepsilon}_0\|^2 + \sum_{k=1}^{N} \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2] + \sum_{k=1}^{N} \mathbb{E}\left[\frac{\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2}{\beta_k}\right].$$

Since $\frac{\alpha_k}{\mu} = c\beta_{k+1}$ for some constant c, we get the following inequality:

$$\sum_{k=0}^{N-1} \frac{1}{2\alpha_k}\mathbb{E}[\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1})] \leq \Phi(\boldsymbol{x}^0) - \Phi^* + c\left(\|\boldsymbol{\varepsilon}_0\|^2 + \sum_{k=1}^{N} \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2] + \sum_{k=1}^{N} \mathbb{E}\left[\frac{\|\nabla F(\boldsymbol{x}^{k-1}) - \nabla F(\boldsymbol{x}^k)\|^2}{\beta_k}\right]\right).$$

By using Assumption 3, we obtain that

$$\frac{\|\nabla F(\boldsymbol{x}^k) - \nabla F(\boldsymbol{x}^{k+1})\|^2}{\beta_{k+1}} \leq \frac{\kappa}{\beta_{k+1}}\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}).$$

Combining above two inequalities, we get

$$\sum_{k=0}^{N-1} \frac{1}{2\alpha_k}\mathbb{E}[\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1})] \leq \Phi(\boldsymbol{x}^0) - \Phi^* + c\left(\|\boldsymbol{\varepsilon}_0\|^2 + \sum_{k=1}^{N} \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2] + \sum_{k=0}^{N-1} \frac{\kappa}{\beta_{k+1}}\mathbb{E}[\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1})]\right).$$

Since $c \leq \frac{1}{2\sqrt{\mu\kappa}}$ and $\frac{\alpha_k}{\mu} = c\beta_{k+1}$, we can deduce that $\frac{c\kappa}{\beta_{k+1}} \leq \frac{1}{4\alpha_k}$. Using this condition, we obtain the inequality:

$$\sum_{k=0}^{N-1} \frac{1}{4\alpha_k}\mathbb{E}[\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1})] \leq \Phi(\boldsymbol{x}^0) - \Phi^* + c\left(\|\boldsymbol{\varepsilon}_0\|^2 + \sum_{k=1}^{N} \beta_k^2\mathbb{E}[\|\widetilde{\nabla}_k - \nabla F(\boldsymbol{x}^k)\|^2]\right).$$

Note that $\mathcal{D}_\phi(\boldsymbol{x}^k, \boldsymbol{x}^{k+1}) \geq \frac{\mu}{2}\|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\|^2 = \frac{\mu\alpha_k^2}{2}\|\widetilde{\mathcal{G}}_{\alpha_k}(\boldsymbol{x}^k)\|^2$ and by the definition of the random variable a, we get

$$\mathbb{E}[\|\widetilde{\mathcal{G}}_{\alpha_a}(\boldsymbol{x}^a)\|^2] \leq \frac{\Phi^0 - \Phi^* + c\|\boldsymbol{\varepsilon}_0\|^2 + c\sum_{k=1}^{N} \frac{\beta_k^2\sigma^2}{m_k}}{\sum_{k=0}^{N-1} \frac{\mu\alpha_k}{8}},$$

which completes the proof. ■

Remark 25 *Now we give some remarks for Theorem 24.*

1. *When the sequence $\{\boldsymbol{x}_k\}$ is bounded, an alternative to Assumption 3 is to assume that $C = \mathbb{R}^d$ and that ϕ has a locally Lipschitz gradient, as made in (Bolte et al., 2018, Theorem 4.1) and (Latafat et al., 2022, Theorem 4.7). Under these conditions, we can conclude that there exists a compact set $\mathcal{U}$ containing $\{\boldsymbol{x}_k\}$. Therefore, there exists a constant $L_{\phi,\mathcal{U}} > 0$ such that $\nabla\phi$ is Lipschitz continuous over $\mathcal{U}$, and we can derive that $\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|^2 \le L_F^2 L_{\phi,\mathcal{U}}^2 \|\boldsymbol{x} - \boldsymbol{y}\|^2 \le \frac{2L_F^2 L_{\phi,\mathcal{U}}^2}{\mu} \mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{y})$ holds.*

2. *The stationarity error for SBPG in Theorem 19 is given by $\mathcal{O}\left(\frac{1}{\sum_{i=0}^k \alpha_i} + \frac{\sum_{i=0}^k \frac{\alpha_i}{m_i}}{\sum_{i=0}^k \alpha_i}\right)$, while for MSBPG in Theorem 24, the error is $\mathcal{O}\left(\frac{1}{\sum_{i=0}^k \alpha_i} + \frac{\sum_{i=0}^k \frac{\alpha_i^2}{m_i}}{\sum_{i=0}^k \alpha_i}\right)$. Notably, compared to SBPG, even with a small constant mini-batch size m_k, MSBPG can still achieve convergence to a zero error bound by carefully selecting the stepsize sequence $\{\alpha_k\}$. A typical stepsize condition is $\sum_{k=0}^\infty \alpha_k = \infty$, $\sum_{k=0}^\infty \alpha_k^2 < \infty$, which coincides with the classical stepsize condition ensuring a moderate decrease of the stepsize, as discussed in (Bertsekas and Tsitsiklis, 2000). Thus, by incorporating the momentum technique, we can achieve improved convergence properties, particularly in terms of relaxed mini-batch size requirements, with minimal additional computational cost. This favorable convergence property theoretically supports the application of MSBPG for large-scale problems, such as deep neural networks training, without the use of very large mini-batch size. To illustrate, we provide a specific choice of $\{\alpha_k\}$ and $\{m_k\}$ that yields a convergence rate in terms of expected stationarity: Set $m_k = 1$, $\alpha_k = \frac{c}{\sqrt{k+1}}$, the convergence rate is $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{k}}\right)$ with logarithmic terms hidden.*

Next, we present the sample complexity of MSBPG.

Corollary 26 *Given an accuracy level $\epsilon > 0$, a constant $\alpha_0 > 0$, and any integer $m > 0$, set $m_k = m$. Then to achieve an ϵ-stationary point in expectation, at most $\bar{N} := \left\lceil \max\left\{(1+\mu^2)\alpha_0^2 L_F^2, \frac{1}{\epsilon^4}\left(\frac{16(\Delta_0 + c\|\varepsilon_0\|^2)}{\mu\alpha_0} + \frac{8\alpha_0\sigma^2}{c\mu m}\right)\right\}\right\rceil$ oracle evaluations are required. To achieve this complexity, we can choose $\alpha_k = \alpha = \frac{\alpha_0}{\sqrt{\bar{N}}}$.*

Proof Let the total number of oracle evaluations be $\bar{N}$. Then $N = \left\lfloor \frac{\bar{N}}{m} \right\rfloor$ and $N \ge \frac{\bar{N}}{2m}$. By choosing $\alpha_k = \alpha = \frac{\alpha_0}{\sqrt{\bar{N}}}$ and using the definition of $\bar{N}$, we have that $\alpha_k < \frac{1}{L_F}\min\left\{1, \frac{1}{\mu}\right\}$, satisfying the conditions for Theorem 24. Then we have

$$\text{RHS of (10)} \le \frac{8(\Delta_0 + c\,\|\varepsilon_0\|^2)m}{\mu\alpha\bar{N}} + \frac{8\sigma^2\alpha}{c\mu m}.$$

By the choices of $\bar{N}$ and α, we have RHS of (24) $\le \epsilon^2$. This completes the proof. ■

We observe that MSBPG has the same complexity order as SBPG, with both achieving $\mathcal{O}(\epsilon^{-4})$, which, as shown in (Arjevani et al., 2023), is the optimal bound and cannot be improved. However, to reach this complexity bound, SBPG requires a large mini-batch

size, whereas MSBPG allows for an arbitrary mini-batch size. This relaxed requirement for the mini-batch size makes MSBPG more practical for training deep neural networks, as it is more memory-efficient. Therefore, the convergence properties of MSBPG are improved in terms of reduced mini-batch size requirements.

As a final remark for this section, in this work, we have only established convergence to a stationary point. However, as demonstrated by (Lee et al., 2019) and (Panageas et al., 2019), the Bregman gradient method almost always converges to a local minimum when the loss function has the strict saddle property (i.e. all the saddle points of f are strict saddle points). In general, finding the global minimum of a nonconvex function is NP-hard. However, in many applications, the objective function is "benign" nonconvex, meaning that all local minima are global minima. Examples include matrix completion (Ge et al., 2016), matrix factorization (Chi et al., 2019), and phase retrieval (Sun et al., 2018), where convergence to a local minimum implies convergence to the global minimum. Additionally, when the objective function satisfies the Polyak–Łojasiewicz (PL) condition (Polyak, 1963), convergence to a stationary point also guarantees convergence to a global minimum. In the experimental section, we observe that in some instances of training neural networks, the training loss can even approach zero, indicating near convergence to a global minimum.

4. Application in training deep neural networks

In this section, we present a detailed description of MSBPG applied to training deep neural networks. Throughout this section, we assume that the optimization domain $\overline{C}$ is the entire space $\mathbb{R}^d$, so that $\phi \in \mathcal{M}(\mathbb{R}^d)$ and $F \in \mathcal{C}^1(\mathbb{R}^d)$. For simplicity, we omit the explicit mention of the feasible set $\mathbb{R}^d$ in this section. In this context, we utilize a polynomial kernel function.

The optimization problem we consider here is given by:

$$\min_{\boldsymbol{W}} \underbrace{\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathcal{DNN}(\boldsymbol{W}, \boldsymbol{x}_i), y_i) + \lambda_2 \|\boldsymbol{W}\|_2^2}_{F(\boldsymbol{W})} + \lambda_1 \|\boldsymbol{W}\|_1, \tag{11}$$

where $\mathcal{DNN}(\boldsymbol{W}, \boldsymbol{x})$ is the neural network function with training parameters $\boldsymbol{W}$ and input data $\boldsymbol{x}$, $\mathcal{L}$ is the loss function that measures the difference between the output of the neural network $\mathcal{DNN}(\boldsymbol{W}, \boldsymbol{x}_i)$ and the label y_i, $\lambda_2 \|\boldsymbol{W}\|_2^2$ is the weight decay term, and $\lambda_1 \|\boldsymbol{W}\|_1$ is the L_1 regularization term that is the sparsity induced operator and often used to avoid overfitting in training deep neural networks (Ng, 2004). To illustrate the neural network function $\mathcal{DNN}(\boldsymbol{W}, \boldsymbol{x})$, in the L-layer fully connected neural network, we have $\boldsymbol{W} = [\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_L]$ and

$$\mathcal{DNN}(\boldsymbol{W}, \boldsymbol{x}) = \sigma_L(\boldsymbol{W}_L(\sigma_{L-1}(\boldsymbol{W}_{L-1}(...(\sigma_1(\boldsymbol{W}_1\boldsymbol{x}))...)))), \tag{12}$$

where σ_i is the nonlinear activation function. In this paper, we focus on smooth activation functions.

At the k-th iteration, MSBPG has the following update scheme:

$$\boldsymbol{v}^k = (1-\beta_k)\boldsymbol{v}^{k-1} + \beta_k \widetilde{\nabla}_k \tag{13}$$

$$\boldsymbol{W}^{k+1} = \operatorname*{argmin}_{\boldsymbol{W}} \langle \boldsymbol{v}^k, \boldsymbol{W} - \boldsymbol{W}^k \rangle + \frac{1}{\alpha_k} \mathcal{D}_\phi(\boldsymbol{W}, \boldsymbol{W}^k) + \lambda_1 \|\boldsymbol{W}\|_1, \tag{14}$$

where $\widetilde{\nabla}_k$ is mini-batch gradient of $F(\boldsymbol{W})$. Omitting all the constants, the subproblem takes the form of:

$$\boldsymbol{W}^{k+1} = \underset{\boldsymbol{W}}{\operatorname{argmin}}\ \phi(\boldsymbol{W}) + \langle \boldsymbol{p}^k,\, \boldsymbol{W}\rangle + \alpha_k\lambda_1\|\boldsymbol{W}\|_1, \tag{15}$$

where $\boldsymbol{p}^k = \alpha_k\boldsymbol{v}^k - \nabla\phi(\boldsymbol{W}^k)$. Here we adopt the kernel function $\phi(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{\delta}{r}\|\boldsymbol{W}\|^r$ $(r \geq 2)$ for training neural networks, and then we have an explicit solution for (15) in Proposition 27.

Proposition 27 *Given $\boldsymbol{p}^k \in \mathbb{R}^d$, positive constant α_k, λ, and the kernel function $\phi(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{\delta}{r}\|\boldsymbol{W}\|^r$ $(r \geq 2,\ \delta > 0)$. The solution of the subproblem (15) is given by*

$$\boldsymbol{W}^{k+1} = -t^*\boldsymbol{p}^+,$$

where t^* is the unique positive real root of the equation

$$(\delta\|\boldsymbol{p}^+\|^{r-2})t^{r-1} + t - 1 = 0, \tag{16}$$

and $\boldsymbol{p}^+$ is given by

$$\boldsymbol{p}^+ = \underset{\boldsymbol{p}}{\operatorname{argmin}}\left\{\frac{1}{2}\|\boldsymbol{p} - \boldsymbol{p}^k\|^2 + \alpha_k\lambda\|\boldsymbol{p}\|_1\right\}$$

which has an explicit expression given by $\boldsymbol{p}_j^+ = \operatorname{sign}(\boldsymbol{p}_j^k)\max(|\boldsymbol{p}_j^k| - \alpha_k\lambda, 0)$ for the j-th coordinate.

Proof The optimality condition of (15) is given by

$$0 = \boldsymbol{W}^{k+1}(1 + \delta\|\boldsymbol{W}^{k+1}\|^{r-2}) + \boldsymbol{p}^k + \alpha_k\lambda\boldsymbol{\Gamma}^k,\ \text{where } \boldsymbol{\Gamma}^k \in \partial\|\cdot\|_1(\boldsymbol{W}^{k+1}).$$

Let $\boldsymbol{p}^+ = \boldsymbol{p}^k + \alpha_k\lambda\boldsymbol{\Gamma}^k$. By the optimality condition, we have $\boldsymbol{W}^{k+1} = -t\boldsymbol{p}^+$ for some positive scalar t, and

$$(-t - \delta\|\boldsymbol{p}^+\|^{r-2}t^{r-1} + 1)\boldsymbol{p}^+ = 0.$$

If $\boldsymbol{p}^+ \neq 0$, then $\delta\|\boldsymbol{p}^+\|^{r-2}t^{r-1} + t - 1 = 0$. If $\boldsymbol{p}^+ = 0$, then $\boldsymbol{W}^{k+1} = -t\boldsymbol{p}^+ = 0$. Since $t > 0$, then we have $\partial\|\cdot\|_1(\boldsymbol{W}^{k+1}) = \partial\|\cdot\|_1(-t\boldsymbol{p}^+) = -\partial\|\cdot\|_1(\boldsymbol{p}^+)$. Recall the definition of $\boldsymbol{p}^+$, we have

$$\boldsymbol{p}^+ = \boldsymbol{p}^k + \alpha_k\lambda\boldsymbol{\Gamma}^k \in \boldsymbol{p}^k - \alpha_k\lambda\partial\|\cdot\|_1(\boldsymbol{p}^+),$$

which is sufficient and necessary optimality condition of the convex optimization problem:

$$\boldsymbol{p}^+ = \underset{\boldsymbol{p}}{\operatorname{argmin}}\left\{\frac{1}{2}\|\boldsymbol{p} - \boldsymbol{p}^k\|^2 + \alpha_k\lambda\|\boldsymbol{p}\|_1\right\}.$$

This completes the proof by noting the the above minimization problem is the well-known soft threshold operator, see for example (Friedman et al., 2010). ■

In the absence of L_1-regularization, that is, when $\lambda_1 = 0$, then $\boldsymbol{p}^+ = \boldsymbol{p}^k$ and the update formula for MSBPG at the k-th iteration simplifies to $\boldsymbol{W}^{k+1} = -t^*\boldsymbol{p}^k$, where t^* is the positive root of the equation (16). In this case, $\boldsymbol{W}^{k+1} = t^*(\nabla\phi(\boldsymbol{W}^k) - \alpha_k\boldsymbol{v}^k)$.

Furthermore, if we choose the kernel function simply as the square of Euclidean distance, i.e. $\delta = 0$, then SBPG reduces to SGD with momentum. Specifically, we have $t^* = 1$ and the update $\boldsymbol{W}^{k+1} = \boldsymbol{W}^k - \alpha_k \boldsymbol{v}^k$.

Determining degree of kernel function We now turn our attention to selecting an appropriate parameter r for the kernel function. Intuitively, in order to bound the Hessian of the loss function in (11), particularly when the number of layers L in (12) is large, r should also be chosen larger, so that $\nabla^2 F \preceq \frac{1}{\alpha}\nabla^2\phi$ holds globally for some $\alpha > 0$. However, this can lead to numerical issues when computing when computing $\|\boldsymbol{W}\|^{r-2}$. This problem can be avoided if the deep neural network exhibits some special structure such that a moderate r can make $F(\boldsymbol{W})$ smooth adaptable with respect to $\phi(\boldsymbol{W})$. For simplicity of analysis, we assume all the given labels y_i as zero and consider a sum of squares loss function. Then, we have a two-layer model defined as follows:

$$\min_{\boldsymbol{W}=(\boldsymbol{u},\boldsymbol{v})} F(\boldsymbol{W}) = \frac{1}{2}\sum_{i=1}^{N}\left(\|\sigma\left(\mathrm{Mat}(\boldsymbol{u})(g_i(\boldsymbol{v}))\right)\|^2\right) + \frac{\lambda_2}{2}(\|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2), \tag{17}$$

where $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^{km}$, $g_i : \mathbb{R}^n \to \mathbb{R}^m$, $\sigma : \mathbb{R} \to \mathbb{R}$, $\mathrm{Mat}(\boldsymbol{u}) \in \mathbb{R}^{k\times m}$ and $\sigma(\cdot)$ is a coordinate-wise operator. Notably, any deep neural network can be reformulated as the two-layer model in (17). For instance, if we define $\boldsymbol{v} = (\boldsymbol{W}_1, ..., \boldsymbol{W}_{L-1})$, $\boldsymbol{u} = \mathrm{Vec}(\boldsymbol{W}_L)$, $g_i(\boldsymbol{W}_1, ..., \boldsymbol{W}_{L-1}) = \sigma_{L-1}(\boldsymbol{W}_{L-1}(...(\sigma_1(\boldsymbol{W}_1\boldsymbol{x}_i))...))$, then model (12) can be transformed into (17). We make the following assumptions in this section, which guarantees that we can find a polynomial kernel function ϕ with a moderate degree, such that F in (17) is smooth adaptable to ϕ.

Assumption 4 *σ is twice differentiable and σ' and $\sigma \cdot \sigma''$ are globally bounded.*

Assumption 5 *Each g_i is twice differentiable. All partial derivatives of order zero, one, and two of g_i are globally bounded.*

Remark 28 *Now we give some remarks on the above assumptions.*

1. *Assumption 4 is typically valid for various commonly used smooth activation functions. For example, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ satisfies global boundedness for both σ and σ''. Certain activation function may not have bounded function value, such as GELU (Hendrycks and Gimpel, 2016), which takes the formulation of $\sigma(x) = x\Phi(x)$ where Φ is the standard Gaussian cumulative distribution function. Nonetheless, the product $\sigma \cdot \sigma''$ is globally bounded. Another type of activation function satisfying Assumption 4 is the smoothed ReLU function, for example, the following smoothed ReLU function, which will be considered in our numerical experiments:*

$$\sigma_\epsilon(x) = \begin{cases} 0 & x \le 0 \\ x^3\left(\frac{1}{\epsilon^2} - \frac{x}{2\epsilon^3}\right) & 0 < x \le \epsilon \\ x - \frac{\epsilon}{2} & x > \epsilon. \end{cases}$$

We observe that as ϵ tends to zero, σ_ϵ converges to the ReLU function. It is straightforward to verify that $\sigma_\epsilon \cdot \sigma_\epsilon''$ is globally bounded. Specifically, $\frac{3}{4}$ is a uniform bound on $\sigma_\epsilon \cdot \sigma_\epsilon''$ for $\epsilon \in (0, \frac{1}{2})$.

2. *In many popular neural network frameworks, batch normalization (BN) layers (Ioffe and Szegedy, 2015) are often used before the fully connected layers. For example, in the VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016), BN layers are usually used before the last linear layer. In this case, we can treat all layers except the last one as one layer, which can be modeled as* (17)*. It is expected that the BN layer can make the function g_i sufficiently smooth, thereby satisfying Assumption 5.*

By applying the chain rule, we can compute the Hessian of F and determine a suitable degree parameter r in the kernel function, which will ensure that $\nabla^2 F$ is bounded by $\nabla^2\phi$ globally. Consequently, F is smooth adaptable with respect to ϕ. In order to compute the Hessian of F, two formulas are required, which can be verified directly.

Lemma 29 *Let $\boldsymbol{u} \in \mathbb{R}^{km}$, $\boldsymbol{g} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{b} \in \mathbb{R}^k$. Consider two linear maps: $\boldsymbol{u} \mapsto \mathrm{Mat}(\boldsymbol{u})\boldsymbol{g}$ and $\boldsymbol{u} \mapsto \boldsymbol{A}(\mathrm{Mat}(\boldsymbol{u}))^T\boldsymbol{b}$, then, the Jacobian of the two maps are given by*

$$J_{\boldsymbol{u}}\left[\mathrm{Mat}(\boldsymbol{u})\boldsymbol{g}\right] = \boldsymbol{g}^T \otimes \mathbb{I}_k,$$

$$J_{\boldsymbol{u}}[\boldsymbol{A}(\mathrm{Mat}(\boldsymbol{u}))^T\boldsymbol{b}] = \boldsymbol{A} \otimes \boldsymbol{b}^T.$$

Proposition 30 *Suppose Assumptions 4 and 5 hold. Then, for any given $\delta > 0$ and any $r \geq 4$, the function F defined in* (17) *is smooth adaptable with respect to $\phi(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{\delta}{r}\|\boldsymbol{W}\|^r$.*

Proof We denote $\mathrm{Mat}(\boldsymbol{u})$ by $\boldsymbol{M}$. The Jacobian of g is denoted by Jg, while its transpose is denoted by $J^T g$. $\mathbb{I}_k$ is $k \times k$ identity matrix. We only need to prove the single sample case, i.e. $N = 1$. Using Lemma 29, we can compute the Jacobian and Hessian of F as follows:

Jacobian of F :

$$\begin{aligned}\frac{\partial F}{\partial \boldsymbol{u}} &= (g(\boldsymbol{v}) \otimes \mathbb{I}_k)\left[\sigma'(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma(\boldsymbol{M}g(\boldsymbol{v}))\right] + \lambda_2\boldsymbol{u},\\ \frac{\partial F}{\partial \boldsymbol{v}} &= J^T g(\boldsymbol{v})\boldsymbol{M}^T\left[\sigma'(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma(\boldsymbol{M}g(\boldsymbol{v}))\right] + \lambda_2\boldsymbol{v}.\end{aligned} \tag{18}$$

Hessian of F :

$$\begin{aligned}\frac{\partial^2 F}{\partial \boldsymbol{u}^2} &= (1) + (2) + \lambda_2\mathbb{I}_{km},\\ \text{where } (1) &= (g(\boldsymbol{v}) \otimes \mathbb{I}_k)\,\mathrm{Diag}\big(\sigma(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma''(\boldsymbol{M}g(\boldsymbol{v}))\big)(\boldsymbol{g}^T(v) \otimes \mathbb{I}_k)\\ (2) &= (g(\boldsymbol{v}) \otimes \mathbb{I}_k)\,\mathrm{Diag}\big(\sigma'(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma'(\boldsymbol{M}g(\boldsymbol{v}))\big)(\boldsymbol{g}^T(v) \otimes \mathbb{I}_k).\end{aligned} \tag{19}$$

$$\begin{aligned}\frac{\partial^2 F}{\partial \boldsymbol{u}\partial \boldsymbol{v}} &= (1) + (2) + (3),\\ \text{where } (1) &= \big(J^T g(\boldsymbol{v})\big) \otimes \left[\sigma'(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma'(\boldsymbol{M}g(\boldsymbol{v}))\right]^T\\ (2) &= J^T g(\boldsymbol{v})\boldsymbol{M}^T\mathrm{Diag}[\sigma(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma''(\boldsymbol{M}g(\boldsymbol{v}))]\left(g^T(\boldsymbol{v}) \otimes \mathbb{I}_k\right)\\ (3) &= J^T g(\boldsymbol{v})\boldsymbol{M}^T\mathrm{Diag}[\sigma'(\boldsymbol{M}g(\boldsymbol{v})) \circ \sigma'(\boldsymbol{M}g(\boldsymbol{v}))]\left(g^T(\boldsymbol{v}) \otimes \mathbb{I}_k\right).\end{aligned} \tag{20}$$

$$\begin{aligned}
\frac{\partial^2 F}{\partial \boldsymbol{v}^2} &= (1)+(2)+(3)+\lambda_2 \mathbb{I}_n, \\
\text{where } (1) &= D^2 g(\boldsymbol{v})\left[\boldsymbol{M}^T[\sigma'(\boldsymbol{M}g(\boldsymbol{v}))\circ\sigma(\boldsymbol{M}g(\boldsymbol{v}))]\right] = \sum d_i \nabla^2 g_i(\boldsymbol{v}) \\
(2) &= J^T g(\boldsymbol{v})\boldsymbol{M}^T \mathrm{Diag}[\sigma(\boldsymbol{M}g(\boldsymbol{v}))\circ\sigma''(\boldsymbol{M}g(\boldsymbol{v}))]\boldsymbol{M}Jg(\boldsymbol{v}) \\
(3) &= J^T g(\boldsymbol{v})\boldsymbol{M}^T \mathrm{Diag}[\sigma'(\boldsymbol{M}g(\boldsymbol{v}))\circ\sigma'(\boldsymbol{M}g(\boldsymbol{v}))]\boldsymbol{M}Jg(\boldsymbol{v}),
\end{aligned} \tag{21}$$

where $\boldsymbol{d} = \boldsymbol{M}^T[\sigma'(\boldsymbol{M}g(\boldsymbol{v}))\circ\sigma(\boldsymbol{M}g(\boldsymbol{v}))]$. Now, we are ready to prove this proposition. For any $\boldsymbol{w}\in\mathbb{R}^{km+n}$ and $\boldsymbol{h}=[\boldsymbol{h}^u;\boldsymbol{h}^v]\in\mathbb{R}^{km+n}$, it suffices to prove that $\langle\nabla^2 F(\boldsymbol{w})\boldsymbol{h},\,\boldsymbol{h}\rangle = \mathcal{O}(\langle\nabla^2\phi(\boldsymbol{w})\boldsymbol{h},\,\boldsymbol{h}\rangle)$. From (19)(20)(21) and Assumption 4, 5, we can easily get $\langle\nabla^2 F(\boldsymbol{w})\boldsymbol{h},\,\boldsymbol{h}\rangle = \mathcal{O}((1+\|\boldsymbol{w}\|^2)\|\boldsymbol{h}\|^2)$. On the other hand, $\nabla^2\phi(\boldsymbol{w}) = I(1+\|\boldsymbol{w}\|^{r-2}) + (r-2)\|\boldsymbol{w}\|^{r-4}\boldsymbol{w}\boldsymbol{w}^T$. Hence $\langle\nabla^2\phi(\boldsymbol{w})\boldsymbol{h},\,\boldsymbol{h}\rangle \geq (1+\|\boldsymbol{w}\|^{r-2})\|\boldsymbol{h}\|^2$. So, we only require $r-2\geq 2$. This completes the proof. ■

Layerwise kernel function In Proposition 27, we use the kernel function $\phi(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|^2 + \frac{\delta}{r}\|\boldsymbol{W}\|^r$, which applies the same Bregman distance across all layers of the deep neural network. However, different layers exhibit distinct geometric properties (You et al., 2019), and computing $\|\boldsymbol{W}\|^r$ with $r>2$ may result in numerical instability for neural networks with millions of parameters, such as in VGG (Simonyan and Zisserman, 2014). To take advantage of the layerwise structure of neural networks, we design a layerwise kernel function for a L-layer neural network as follows:

$$\phi(\boldsymbol{W}) = \sum_{i=1}^{L}\phi_i(\boldsymbol{W}_i), \quad \phi_i(\boldsymbol{W}_i) = \frac{1}{2}\|\boldsymbol{W}_i\|^2 + \frac{\delta}{r}\|\boldsymbol{W}_i\|^r. \tag{22}$$

Note that δ and r can vary from layer to layer, here we take the same δ and r for all layers for simplicity. With this structure, the Bregman distance takes the form $\mathcal{D}_\phi = \sum_{i=1}^{L}\mathcal{D}_{\phi_i}$. By incorporating this layerwise Bregman distance into the subproblem (14), our MSBPG algorithm can be implemented in a layerwise manner. Details of the implementation are provided in Algorithm 1.

Mitigating gradient explosion In the training of deep neural networks, gradient explosion is a common undesired phenomenon, where the gradients of the loss function grow exponentially from layer to layer, leading to numerical instability or even collapse of the training process (Hochreiter, 1991; Manchev and Spratling, 2020). The reasons for gradient explosion include selecting a large stepsize and choosing an improper initialization for the model's parameters (Pascanu et al., 2013). In the following, we will show that MSBPG provides a novel approach to mitigate gradient explosion. Considering MSBPG without L_1-regularization, the update rule is given by:

$$\boldsymbol{W}_i^{k+1} = -t_i^k \boldsymbol{p}_i^k = t_i^k\left((1+\delta\|\boldsymbol{W}_i{}^k\|^{r-2})\boldsymbol{W}_i{}^k - \alpha_k \boldsymbol{v}_i^k\right), \tag{23}$$

where $t_i^k\in(0,1)$ is the unique positive root of

$$\left(\delta\left\|(1+\delta\|\boldsymbol{W}_i^k\|^{r-2})\boldsymbol{W}_i^k - \alpha_k\boldsymbol{v}_i^k\right\|^{r-2}\right)t^{r-1} + t - 1 = 0. \tag{24}$$

Algorithm 1 Momentum based Stochastic Bregman Proximal Gradient (MSBPG) for training neural networks

Input: Total number of iterations K, stepsize $\alpha_k > 0$, momentum parameter $\beta_k \in (0,1)$, $\delta > 0$ and integer $r \geq 4$ to determine the kernel function ϕ, and λ_1.
Initialize: Set $\boldsymbol{W} = \boldsymbol{W}^0$, $\boldsymbol{v}^0 = 0$.
for $k = 0, \cdots, K-1$ **do**
  Compute mini-batch gradient $\widetilde{\nabla}_k$ of F;
  Compute SMAE: $\boldsymbol{v}^k = (1-\beta_k)\boldsymbol{v}^{k-1} + \beta_k \widetilde{\nabla}_k$;
  for $i = 1, \ldots, L$ **do**
    $\boldsymbol{p}_i^k = \alpha_k \boldsymbol{v}_i^k - \nabla\phi(\boldsymbol{W}_i^k)$;
    $\boldsymbol{p}_i^+ = \mathrm{argmin}_{\boldsymbol{p}_i}\{\frac{1}{2}\|\boldsymbol{p}_i - \boldsymbol{p}_i^k\|^2 + \alpha_k\lambda_1\|\boldsymbol{p}_i\|_1\}$;
    Solve $(\delta\|\boldsymbol{p}_i^+\|^{r-2})t_i^{r-1} + t_i - 1 = 0$ to get t_i^k;
    $\boldsymbol{W}_i^{k+1} = -t_i^k \boldsymbol{p}_i^+$;
  end for
end for
Output: $\boldsymbol{W}^1, \cdots, \boldsymbol{W}^K$.

Combining (23) and (24), we have the following equivalent implicit update scheme for the i-th layer:

$$\boldsymbol{W}_i^{k+1} = \frac{1+\delta\|\boldsymbol{W}_i^k\|^{r-2}}{1+\delta\|\boldsymbol{W}_i^{k+1}\|^{r-2}}\boldsymbol{W}_i^k - \frac{\alpha_k}{1+\delta\|\boldsymbol{W}_i^{k+1}\|^{r-2}}\boldsymbol{v}_i^k. \tag{25}$$

It is observed in practice that with large stepsize or large initial point, the gradient $\boldsymbol{v}_i^k$ tends to explode if no scaling or clipping is applied, while the norm of the weight $\|\boldsymbol{W}_i^{k+1}\|$ also tends to be large. In (25), scaling the gradient by $\frac{1}{1+\delta\|\boldsymbol{W}_i^{k+1}\|^{r-2}}$ prevents the weight $\boldsymbol{W}_i^{k+1}$ from moving too drastically in the direction of the gradient, thereby controlling the rapid growth of its norm. At the same time, if the norm $\|\boldsymbol{W}_i^{k+1}\|$ does not change significantly, the coefficient of $\boldsymbol{W}_i^k$ in (25) will remain approximately 1. Thus, the implicit update (25) provides automatic scaling of the gradient, effectively mitigating the rapid growth of the weight and preventing subsequent gradient explosion.

Experimental results in Section 5.2 indeed verify MSBPG 's ability to mitigate gradient explosion for training deep neural networks. An intuitive illustration of SBPG 's "pull-back" ability is given in Figure 12 in Appendix D, and this "pull-back" ability originates from the Bregman proximity model and the polynomial kernel function we adopt.

5. Numerical experiments

In this section, we conduct numerical experiments to demonstrate the effectiveness and robustness of MSBPG in comparison to some commonly used optimizers in deep learning. We assess the impact of stepsize and initial point selection on the performance of our method. Our experiments consist of two parts. In the first part, we use a quadratic inverse problem as a toy example to illustrate the capabilities of vanilla SBPG. The second part is the main focus of this section, where we evaluate the performance of MSBPG in training deep neural networks. The experiments for the quadratic inverse problem are conducted

using MATLAB R2021b on a Windows workstation equipped with a 12-core Intel Xeon E5-2680 @ 2.50GHz processor and 128GB of RAM. For the deep learning experiments, we conducted the experiments using PyTorch running on a single RTX3090 GPU.

5.1 Quadratic inverse problem

The quadratic inverse problem, as formulated in (Bolte et al., 2018), is given by:

$$\min\left\{\Phi(\boldsymbol{x}) := \underbrace{\frac{1}{4}\sum_{i=1}^{n}(\langle A_i\boldsymbol{x},\,\boldsymbol{x}\rangle - b_i)^2}_{F(\boldsymbol{x})} + \lambda R(\boldsymbol{x}) : \boldsymbol{x}\in\mathbb{R}^d\right\},$$

which has practical applications (Beck and Eldar, 2013), including the phase retrieval problem as a special case (Luke, 2017). In this experiment, we consider the L_1 regularization $R(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ with $\lambda = 1\times 10^{-3}$, and solve the quadratic inverse problem using SBPG and stochastic (Euclidean) proximal gradient (SPG) method (Bertsekas, 2011). Notably, SPG is a special case of SBPG, where $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2$. Since the smooth term in the objective function $F(\boldsymbol{x})$ does not admit a globally Lipschitz continuous gradient, we employ the kernel function $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{1}{r}\|\boldsymbol{x}\|^r$ with $r = 4$. It has been shown in (Lu et al., 2018) that any $r \geq 4$ guarantees that F is ϕ-smooth adaptable globally. Moreover, according to (Bolte et al., 2018), the smooth adaptable constant L_F can be chosen such that $L_F \geq \sum_{i=1}^{n}(3\|A_i\|^2 + \|A_i\|\|b_i|)$ for $r = 4$. In this experiment, we randomly generate the data by the following `MATLAB` commands:

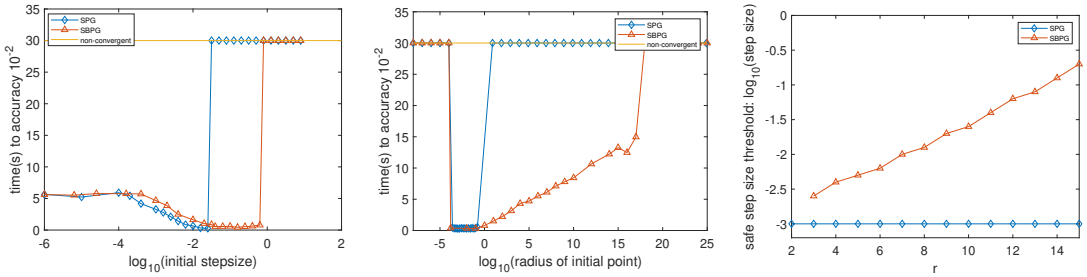
$$\texttt{ai} = \texttt{randn}(\texttt{d},1);\ \texttt{Ai} = \texttt{ai}*\texttt{ai}';$$
$$\texttt{x\_true} = \texttt{sprandn}(\texttt{d},1,\texttt{density\_x});\ \texttt{b\_i} = \texttt{x\_true}'*(\texttt{Ai}*\texttt{x\_true});$$

The true solution for the quadratic inverse problem is chosen as a sparse vector $\boldsymbol{x}^*$ that satisfies $\langle A_i\boldsymbol{x}^*,\,\boldsymbol{x}^*\rangle = b_i$ for $i = 1,\ldots,n$. We set the mini-batch size for all algorithms to be $m = 1$. To evaluate the effectiveness of each algorithm, we use the following criterion that takes into account the possibility of stationary points being local minimum or saddle points:

$$\epsilon_k = \max\left\{\|\mathcal{G}(\boldsymbol{x}^k)\|,\ \epsilon_{\texttt{obj}} := \frac{\texttt{obj}_k - \texttt{obj}_*}{1+\texttt{obj}_*}\right\}, \tag{26}$$

where $\texttt{obj}_k = \Phi(\boldsymbol{x}^k)$ and $\texttt{obj}_* = \Phi(\boldsymbol{x}^*)$. The term $\|\mathcal{G}(\boldsymbol{x}^k)\|$ measures the stationarity of the solution, while a small $\epsilon_{\texttt{obj}}$ indicates that the solution is a "nearly" global minimum.

We conduct experiments on a problem with data size $d = 100$ and $\texttt{density\_x} = 0.05$. All methods are run until they reach an accuracy of $\epsilon_k \leq 0.01$ within a time limit of 30 seconds. To ensure statistical significance, we run each algorithm 10 times and report the median value. The results are presented in Figure 2. For Figures 2(a), we randomly select initial points within a ball centered at the origin with radius 1×10^{-2}. We use the stepsize schedule of $\alpha_k = \max\left\{10^{-4}, \frac{\alpha_0}{\sqrt{1+k}}\right\}$, where α_0 is the initial stepsize. For Figure 2(b), we set constant stepsize schedule 1×10^{-3}. For Figures 2(c), we randomly select initial points within a ball centered at the origin with radius 1×10^{-2}. We use a constant stepsize



(a) Stepsize robustness (b) Initial point robustness (c) Safe stepsize threshold

Figure 2: Comparison of SBPG and SPG in terms of their robustness with respect to stepsize and initial point selction. A method is considered non-convergent if it fails to reach an accuracy of $\epsilon_k < 10^{-2}$ within 30 seconds or if it collapses. Generally, choosing large stepsize and large radius for the initial point can cause an algorithm to collapse. The safe stepsize threshold is the maximum stepsize (constant schedule) that a method does not collapse. We run 10 tests for each algorithm and report the median of the results.

schedule. To prevent excessively small stepsizes that can slow down all methods, we set a lower bound for the stepsize.

Figure 2(a) demonstrates that SBPG has a wider range of convergent stepsizes than SPG, indicating that SBPG is more robust in terms of stepsize selection. The effect of the initial stepsize on the performance of the algorithms is also depicted in this figure. Figure 2(b) highlights that SBPG is significantly more robust than SPG with respect to initial point selection, showing high resilience and preventing the training process from collapsing. Additionally, Figure 2(c) illustrates that increasing the degree r in the kernel function raises the threshold for safe stepsizes. These observations are explained in Section 4. When a large stepsize or a large initial point radius leads to a potential gradient explosion, the Bregman proximal mapping effectively pulls back the iterate, guiding it toward a more stable region and preventing gradient explosion.

5.2 Deep neural network

To evaluate MSBPG's performance in training deep neural networks, we consider a model with L_2 regularization to enhance generalization:

$$\min_{\boldsymbol{W}} \underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\mathcal{DNN}(\boldsymbol{W},\boldsymbol{x}_i),y_i)}_{F(\boldsymbol{W})} + \lambda_2\|\boldsymbol{W}\|_2^2. \tag{27}$$

We employ MSBPG to solve this large-scale problem. Following AdamW (Loshchilov and Hutter, 2017), we employ the strategy of decoupled weight decay. We use MSBPG to solve this large-scale optimization problem. Following the AdamW approach (Loshchilov and Hutter, 2017), we implement a decoupled weight decay strategy. Specifically, we compute

the stochastic gradient only for the loss function F, treating F and L_2 regularization separately. After performing the Bregman proximal mapping based on the stochastic gradient, we apply a weight decay step.

The detailed algorithm is summarized in Algorithm 2. At iteration k, MSBPG first uses automatic differentiation to compute the mini-batch gradient $\widetilde{\nabla}_k$ of F. Then, it maintains a bias-corrected gradient estimator $\bar{\boldsymbol{v}}^k$ (Kingma and Ba, 2014) and uses it to calculate the layerwise $\boldsymbol{p}_i^k$. With $\boldsymbol{p}_i^k$, MSBPG solves a univariate equation to get t_i^k and updates the weight of the i-th layer to $\boldsymbol{W}_i^k$. In the end, MSBPG conducts decoupled weight decay.

Algorithm 2 MSBPG with L_2 regularization

Input: Total number of training epochs K, momentum coefficient β, stepsize α_k, weight decay coefficient λ_2, δ and r to determine the kernel function ϕ.
Initialize: Set $\boldsymbol{W} = \boldsymbol{W}^0$, $\boldsymbol{v}^0 = \boldsymbol{0}$.
for $k = 1, \cdots, K$ **do**
  Compute mini-batch gradient $\widetilde{\nabla}_k$;
  $\boldsymbol{v}^k = \beta \boldsymbol{v}^{k-1} + (1-\beta)\widetilde{\nabla}_k$, $\bar{\boldsymbol{v}}^k = \boldsymbol{v}^k/(1-\beta^k)$;
  for $i = 1, \cdots, L$ **do**
    $\boldsymbol{p}_i^k = \alpha_k \bar{\boldsymbol{v}}_i^k - \nabla\phi(\boldsymbol{W}_i^{k-1})$;
    Solve $(\delta\|\boldsymbol{p}_i^k\|^{r-2})t_i^{r-1} + t_i - 1 = 0$ to get t_i^k;
    $\widetilde{\boldsymbol{W}}_i^k = -t_i^k \boldsymbol{p}_i^k$;
  end for
  $\boldsymbol{W}^k = \widetilde{\boldsymbol{W}}^k - \alpha_k \lambda_2 \boldsymbol{W}^{k-1}$;
end for
Output: $\boldsymbol{W}^1, \cdots, \boldsymbol{W}^K$

We conducted experiments on several representative benchmarks, including VGG16 (Simonyan and Zisserman, 2014), ResNet34 (He et al., 2016) on CIFAR10 dataset (Krizhevsky et al., 2009), ResNet34 (He et al., 2016), DenseNet121 (Huang et al., 2017) on CIFAR100 dataset (Krizhevsky et al., 2009), and LSTMs (Hochreiter and Schmidhuber, 1997) on the Penn Treebank dataset (Marcinkiewicz, 1994). We compare MSBPG with the most popular optimization algorithms used for training neural networks, including SGD (Sutskever et al., 2013), Adam (Kingma and Ba, 2014), and AdamW (Loshchilov and Hutter, 2017). Experimental results show that MSBPG has good convergence performance and generalization capacity for both the task that SGD dominates (image classification with CNNs) and the task that Adam dominates (language modeling with LSTMs). We also conducted experiments to compare MSBPG with SGD on different initial stepsizes and different scales of the initial point. Our experimental results demonstrate the robustness of MSBPG in training neural networks.

To further evaluate the performance of MSBPG on more recent neural network architectures, we conducted additional experiments, with the results presented in Appendix C.

Before getting into the details of our experiments, we first clarify the activation function. The commonly used ReLU activation function in VGG, ResNet, and DenseNet is defined as $\mathrm{ReLU}(x) = \max(0, x)$, which is not continuously differentiable. To address this, we design a smooth approximation of ReLU with a parameter ϵ, which is twice continuously

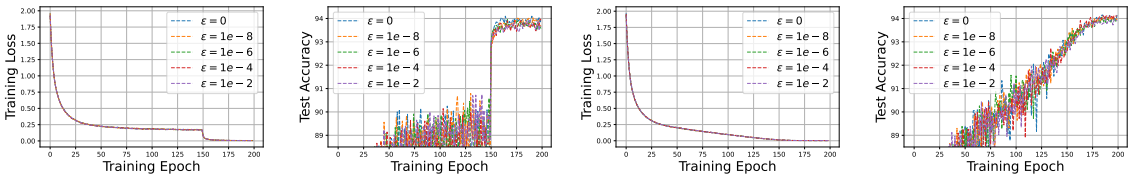
differentiable and satisfies our Assumption 4:

$$\sigma_\epsilon(x) = \begin{cases} 0 & x \le 0 \\ x^3\left(\frac{1}{\epsilon^2} - \frac{x}{2\epsilon^3}\right) & 0 < x \le \epsilon \\ x - \frac{\epsilon}{2} & x > \epsilon. \end{cases}$$

The gradient of this activation function is given by:

$$\sigma'_\epsilon(x) = \begin{cases} 0 & x \le 0 \\ x^2\left(\frac{3}{\epsilon^2} - \frac{2x}{\epsilon^3}\right) & 0 < x \le \epsilon \\ 1 & x > \epsilon. \end{cases}$$

As ϵ tends to 0, this smooth activation function converges to the standard ReLU function. We conducted experiments using VGG16 on the CIFAR-10 dataset, replacing all activation functions in VGG16 with σ_ϵ. As shown in Figure 4, our algorithm MSBPG 's performance does not degrade as ϵ tends to 0. Therefore, in the subsequent experiments, we use the original neural network architectures with the ReLU activation function to evaluate our method MSBPG.



(a) Step decay. (b) Cosine annealing.

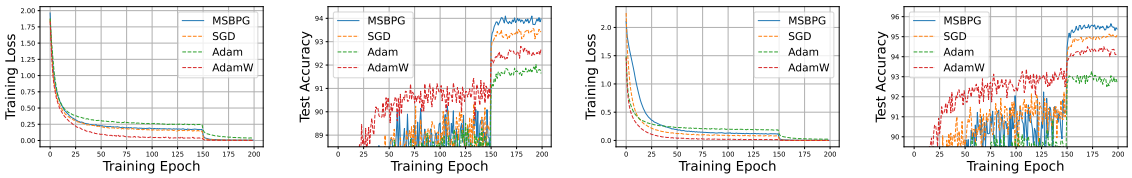
Figure 3: Training loss and test accuracy (%) of VGG16 on CIFAR10 dataset under two frequently used training settings. Here the activation function of VGG16 adopts smoothed ReLU activation function σ_ϵ with different choices of ϵ ($\epsilon = 0$ denotes adopting the original ReLU activation function).

CNNs on image classification We conducted experiments with VGG16 and ResNet34 on the CIFAR-10 dataset, and ResNet34 and DenseNet121 on the CIFAR-100 dataset. SGD usually has better generalization performance than adaptive gradient algorithms such as Adam and AdamW when training CNNs on image classification tasks. For our experiments, we utilized two common training strategies: reducing the stepsize to 10% of its original value near the end of training (Zhuang et al., 2020; Chen et al., 2021; Luo et al., 2019), and using a cosine annealing schedule for stepsizes (Loshchilov and Hutter, 2016, 2017). These strategies are designed to accelerate convergence and allow for a fair comparison of the generalization capacities of different optimizers. We use the default training hyperparameters of SGD, Adam, and AdamW in these settings (He et al., 2016; Zhuang et al., 2020; Chen et al., 2021), and set MSBPG 's learning rate (initial stepsize) as 0.1, momentum coefficient β as 0.9, weight decay coefficient λ_2 as 1×10^{-3}. For the layerwise kernel function $\phi_i(\boldsymbol{W}_i) = \frac{1}{2}\|\boldsymbol{W}_i\|^2 + \frac{\delta}{r}\|\boldsymbol{W}_i\|^r$, we set $r = 4$, $\delta = 1 \times 10^{-2}$ for VGG16 and $r = 6$, $\delta = 1 \times 10^{-3}$ for

ResNet34 on CIFAR10 dataset, and $r = 4,\ \delta = 1\times10^{-2}$ for ResNet34 and $r = 4,\ \delta = 1\times10^{-3}$ for DenseNet121 on CIFAR100 dataset.

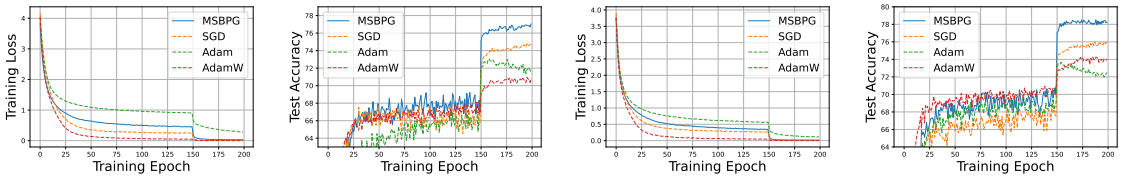
Our convergence analysis demonstrates that MSBPG converges to a stationary point. As shown by (Lee et al., 2019; Panageas et al., 2019), the Bregman gradient method almost always converges to a local minimum for loss functions with the strict saddle property. When the neural network is highly overparameterized and exhibits a benign nonconvexity in the search region, stochastic first-order methods tend to find the global minimum, where the loss function value is 0.

From the experimental results in Figure 4, 5, 6, 7, we can observe that MSBPG attains almost zero training loss in all the training settings. This implies that our method can find the global minimum in these instances. Furthermore, MSBPG consistently achieves the highest test accuracy for all experimental settings and attains at least 0.5% test accuracy improvement compared with the second-best optimization algorithm. This generalization advantage of MSBPG can be attributed to the Bregman proximity model we adopt. In Appendix B, we further discuss SBPG's performance from the perspective of algorithmic stability, which can influence the generalization gap. We demonstrate that the high-order polynomial kernel used helps reduce the generalization gap bound in high-dimensional scenarios, which partially explains the good generalization performance of our proposed methods.



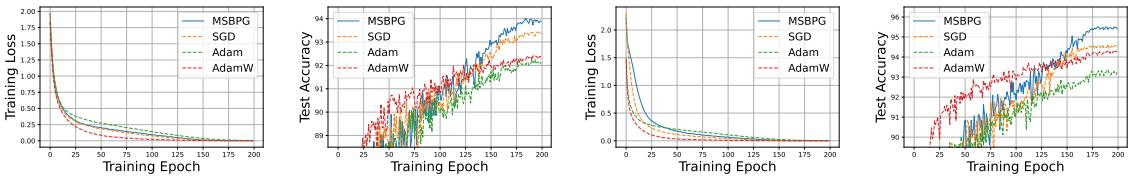
(a) VGG16 on CIFAR10 (b) ResNet34 on CIFAR10

Figure 4: Training loss and test accuracy (%) of CNNs on CIFAR10 dataset with learning rate reduced to 0.1 times of the original value at the 150th epoch.



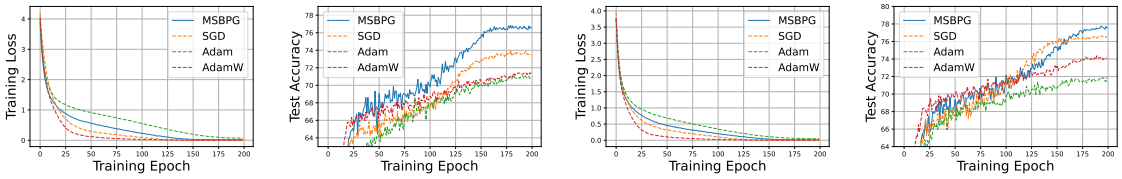
(a) ResNet34 on CIFAR100 (b) DenseNet121 on CIFAR100

Figure 5: Training loss and test accuracy (%) of CNNs on CIFAR100 dataset with learning rate reduced to 0.1 times of the original value at the 150th epoch.



(a) VGG16 on CIFAR10 (b) ResNet34 on CIFAR10

Figure 6: Training loss and test accuracy (%) of CNNs on CIFAR10 dataset with learning rate using the cosine annealing schedule.

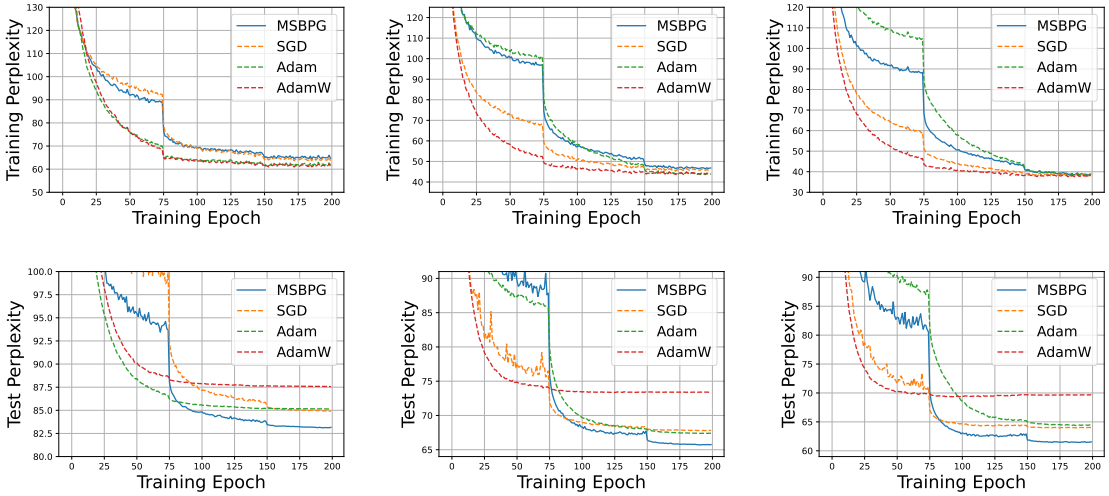


(a) ResNet34 on CIFAR100 (b) DenseNet121 on CIFAR100

Figure 7: Training loss and test accuracy (%) of CNNs on CIFAR100 dataset with learning rate using the cosine annealing schedule.

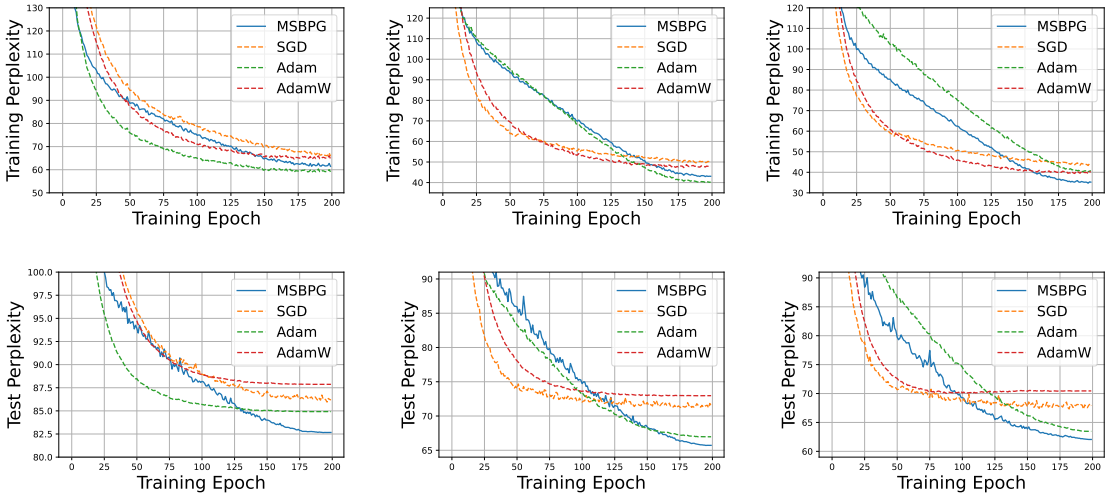
LSTMs on language modeling To further evaluate the performance of MSBPG, we conducted experiments on LSTMs using the Penn Treebank dataset, reporting both training and test perplexity (lower is better). Adam is generally favored over SGD for language modeling tasks due to its better generalization capacity (Fu et al., 2016; Siami-Namini et al., 2019), and thus it is the default optimization algorithm for training LSTMs. We followed the standard experimental setup for training LSTMs (Zhuang et al., 2020; Chen et al., 2021), where the learning rate is reduced to 10% of its original value twice (at the 75th and 150th epochs). Additionally, we experimented with the cosine annealing learning rate schedule (Loshchilov and Hutter, 2016), which is commonly used in practice. For the training hyperparameters, we used the default settings for SGD, Adam, and AdamW when training 1-, 2-, and 3-layer LSTMs (Zhuang et al., 2020; Chen et al., 2021). For MSBPG, we set the learning rate to 25, 80, and 80 for 1-, 2-, and 3-layer LSTMs, respectively, with a momentum parameter $\beta = 0.9$, weight decay coefficient $\lambda_2 = 2 \times 10^{-6}$. For the layerwise kernel function $\phi_i(\boldsymbol{W}_i) = \frac{1}{2}\|\boldsymbol{W}_i\|^2 + \frac{\delta}{r}\|\boldsymbol{W}_i\|^r$, we set $r = 4$ and $\delta = 1 \times 10^{-6}$. From Figure 8 and Figure 9, we observe that MSBPG converges well on the training dataset for 1-, 2-, and 3-layer LSTMs across both training strategies. In contrast, SGD with the cosine annealing learning rate schedule fails to fully converge on the training dataset, as shown in Figure 9. Additionally, MSBPG consistently achieves lower test perplexity in all experiments, outperforming other methods by at least 1 unit. This excellent generalization capacity can be attributed to the Bregman proximity model employed by MSBPG. As an additional evaluation, we further assess the performance of MSBPG on a recently popular transformer

model, Transformer-XL (Dai et al., 2019), which is designed for long-sequence tasks. The results are provided in Appendix C.



(a) 1-layer LSTM (b) 2-layer LSTM (c) 3-layer LSTM

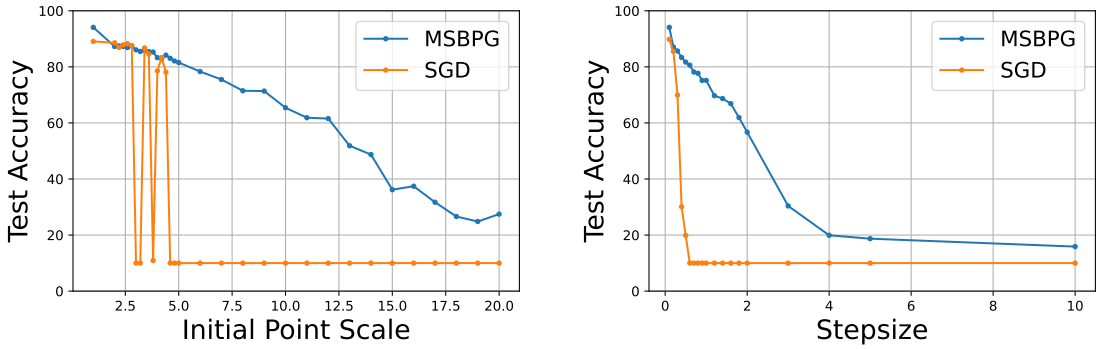
Figure 8: Training and test perplexity (lower is better) of LSTMs on Penn Treebank dataset with learning rate reduced to 0.1 times of the original value at the 75th epoch and 150th epoch.



(a) 1-layer LSTM (b) 2-layer LSTM (c) 3-layer LSTM

Figure 9: Training and test perplexity (lower is better) of LSTMs on Penn Treebank dataset with learning rate using the cosine annealing schedule.

Robustness to initial point scale and stepsize As demonstrated in Section 4, MSBPG can mitigate the issue of gradient explosion, which typically occurs when using large stepsizes or large initial point scales. To verify MSBPG's robustness in training neural networks, we conducted experiments using VGG16 on the CIFAR-10 dataset. Specifically, we compared the performance of MSBPG and SGD under different initial point scales and stepsizes, as both algorithms share the same default learning rate of (1×10^{-1}). Since adaptive gradient algorithms, such as Adam, use a different default learning rate scale (1×10^{-3}), they were not included in this comparison. For various initial point scales and stepsizes, we ran each optimization algorithm for 50 iterations and reported the best test accuracy. As shown in Figure 10, MSBPG is more robust than SGD to large initial points and stepsizes. Training deep neural networks, which have millions or billions of parameters, is highly sensitive to both the initial point scale and stepsize. From Figure 10, we can see that SGD fails to converge when the initial point scale is increased to 4.6 or when the stepsize is increased from 0.1 to 0.6. In contrast, MSBPG converges with an initial point scale as large as 20 and a stepsize as large as 5. This robustness of MSBPG can ease the tuning of hyperparameters for training neural networks, and can also make the training process more robust to noises and errors.



(a) Initial point robustness

(b) Stepsize robustness

Figure 10: Test accuracy (%) of VGG16 on CIFAR10 dataset with different initial point scale and stepsize choice.

6. Conclusion

In this paper, we introduce a family of nonconvex stochastic Bregman proximal gradient (SBPG) methods to solve optimization problems without the Lipschitz smoothness assumption. By leveraging Bregman proximity measures, SBPG offers a more flexible and robust framework than classical stochastic gradient methods. We establish convergence results for the vanilla SBPG method in the nonconvex setting and propose a momentum-based variant, MSBPG, which improves convergence property by relaxing the mini-batch size requirement. Both methods achieve optimal sample complexity $\mathcal{O}(\epsilon^{-4})$, making them well-suited for large-scale problems. MSBPG is applied to training deep neural networks, where it mit-

igates gradient explosion and enhances generalization performance. Numerical experiments on sparse quadratic inverse problems and deep neural networks demonstrate MSBPG's robustness and superior performance compared to commonly used optimizers such as SGD, Adam, and AdamW. In conclusion, MSBPG provides an effective and efficient optimization approach for large-scale nonconvex problems, combining theoretical robustness with practical advantages. Future work can explore further refinements and applications in more complex machine learning tasks.

Appendix A. Proofs in Preliminaries

Proof of Lemma 8 First, we prove the uniqueness of the solution. Problem (4) is equivalent to the following problem:

$$\arg\min_{\boldsymbol{u}\in\overline{C}} \Psi(\boldsymbol{u}) := \alpha R(\boldsymbol{u}) + \phi(\boldsymbol{u}) - \langle \nabla\phi(\boldsymbol{x}),\, \boldsymbol{u}\rangle.$$

We have that

$$\Psi(\boldsymbol{u}) \geq \alpha R(\boldsymbol{u}) + \phi(\boldsymbol{u}) - \|\nabla\phi(\boldsymbol{x})\|\,\|\boldsymbol{u}\| \geq \|\boldsymbol{u}\|\Big(\frac{\alpha R(\boldsymbol{u}) + \phi(\boldsymbol{u})}{\|\boldsymbol{u}\|} - \|\nabla\phi(\boldsymbol{x})\|\Big).$$

As $\|\boldsymbol{u}\| \to \infty$, we have $\Psi(\boldsymbol{u}) \geq \|\boldsymbol{u}\|\Big(\frac{\alpha R(\boldsymbol{u})+\phi(\boldsymbol{u})}{\|\boldsymbol{u}\|} - \|\nabla\phi(\boldsymbol{x})\|\Big) = \infty$, where we use the fact that ϕ is supercoercive and R is convex. Since Ψ is a proper lower-semicontinuous convex function, by the modern form of Weierstrass theorem (Rockafellar, 1997, Chapter 1), we know that the solution set of (4) is a nonempty compact set. Also note that Ψ is a strictly convex function, which implies the uniqueness of the solution. For any Legendre function ϕ, from (Rockafellar, 1997, Chapter 26), we have $dom\,\partial\phi = int\,dom\phi$ with $\partial\phi(\boldsymbol{x}) = \{\nabla\phi(\boldsymbol{x})\}$ for all $\boldsymbol{x} \in int\,dom\,\phi$. The optimality condition implies that $\partial\phi(\mathrm{Prox}^{\phi}_{\alpha R}(\boldsymbol{x}))$ is nonempty, which automatically forces $\mathrm{Prox}^{\phi}_{\alpha P}(\boldsymbol{x}) \in \mathrm{int\ dom}\phi$. This completes the proof. □

Proof of Proposition 12 Note that $\|\nabla\phi(\boldsymbol{x}^+) - \nabla\phi(\boldsymbol{x})\| \leq L_\phi\|\boldsymbol{x}^+ - \boldsymbol{x}\|$ and $\|\nabla F(\boldsymbol{x}^+) - \nabla F(\boldsymbol{x})\| \leq L_F L_\phi\|\boldsymbol{x}^+ - \boldsymbol{x}\|$. By the definition of $\boldsymbol{x}^+$, we have

$$\nabla F(\boldsymbol{x}^+) - \nabla F(\boldsymbol{x}) + \frac{\nabla\phi(\boldsymbol{x}) - \nabla\phi(\boldsymbol{x}^+)}{\alpha} \in \nabla F(\boldsymbol{x}^+) + \partial R(\boldsymbol{x}^+).$$

Thus, we obtain

$$\mathrm{dist}\left(0, \partial\Phi(\boldsymbol{x}^+)\right) \leq \left\|\nabla F(\boldsymbol{x}^+) - \nabla F(\boldsymbol{x}) + \frac{\nabla\phi(\boldsymbol{x}) - \nabla\phi(\boldsymbol{x}^+)}{\alpha}\right\| \leq \left(L_F L_\phi + \frac{L_\phi}{\alpha}\right)\|\boldsymbol{x}^+ - \boldsymbol{x}\|.$$

Note that $\|\boldsymbol{x}^+ - \boldsymbol{x}\| = \alpha\|\mathcal{G}_\alpha(\boldsymbol{x})\|$, which completes the proof. □

Proof of Proposition 9 By the definition of $\mathrm{Prox}^{\phi}_{R}(\cdot)$, $x_i \in \partial R(\boldsymbol{x}_i^+) + \nabla\phi(\boldsymbol{x}_i^+)$, $i = 1, 2$. Since $\partial R(\cdot)$ is monotone, then $\langle \boldsymbol{x}_1 - \boldsymbol{x}_2 - (\nabla\phi(\boldsymbol{x}_1^+) - \nabla\phi(\boldsymbol{x}_2^+)),\, \boldsymbol{x}_1^+ - \boldsymbol{x}_2^+\rangle \geq 0$. From the μ-strong convexity of ϕ, it follows that $\langle \boldsymbol{x}_1 - \boldsymbol{x}_2,\, \boldsymbol{x}_1^+ - \boldsymbol{x}_2^+\rangle \geq \langle \nabla\phi(\boldsymbol{x}_1^+) - \nabla\phi(\boldsymbol{x}_2^+),\, \boldsymbol{x}_1^+ - \boldsymbol{x}_2^+\rangle \geq \mu\|\boldsymbol{x}_1^+ - \boldsymbol{x}_2^+\|^2$. Therefore, $\|\boldsymbol{x}_1^+ - \boldsymbol{x}_2^+\| \leq \frac{1}{\mu}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$. □

Appendix B. Algorithmic stability analysis

Consider a dataset $\boldsymbol{\Xi} = (\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_n)$ and an algorithm $\mathcal{A}$. Let $\Phi_{\boldsymbol{\Xi}}(\boldsymbol{x}) = F(\boldsymbol{x}, \boldsymbol{\Xi}) + R(\boldsymbol{x})$, and let $\mathcal{A}(\boldsymbol{\Xi})$ be the output of the algorithm $\mathcal{A}$ based on the dataset $\boldsymbol{\Xi}$. Since the underlying distribution of $\boldsymbol{\xi}$ is unknown, the population risk can be decomposed into two parts:

$$\mathbb{E}_{\boldsymbol{\Xi},\mathcal{A}}[\Phi(\mathcal{A}(\boldsymbol{\Xi})) - \Phi(\boldsymbol{x}^*)] = \underbrace{\mathbb{E}_{\boldsymbol{\Xi},\mathcal{A}}\left[\Phi_{\boldsymbol{\Xi}}(\mathcal{A}(\boldsymbol{\Xi})) - \Phi(\mathcal{A}(\boldsymbol{\Xi}))\right]}_{\varepsilon_{\rm gen}} + \underbrace{\mathbb{E}_{\boldsymbol{\Xi},\mathcal{A}}\left[\Phi_{\boldsymbol{\Xi}}(\mathcal{A}(\boldsymbol{\Xi})) - \Phi(\boldsymbol{x}^*)\right]}_{\varepsilon_{\rm opt}},$$

where $\boldsymbol{x}^*$ is independent of $\boldsymbol{\Xi}$ and $\mathcal{A}$. The first term represents the expected generalization error, and the second term represents the optimization error. Our goal is to assess the expected generalization error $\varepsilon_{\rm gen}$. For clarity, we focus on the simple case where $R(x) \equiv 0$, in which case $\Phi(\boldsymbol{x}) = F(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\xi}}[f(\boldsymbol{x}, \boldsymbol{\xi})]$. Consider any two data sets $\boldsymbol{\Xi}, \boldsymbol{\Xi}'$ that differ by at most one example. As shown by (Bousquet and Elisseeff, 2002; Shalev-Shwartz et al., 2010; Hardt et al., 2016), the absolute expected generalization error $|\varepsilon_{\rm gen}|$ can be bounded by ε if the algorithm is ε-uniform stable that is defined as follows:

Definition 31 *A randomized algorithm $\mathcal{A}$ is ε-uniformly stable, if for any two datasets $\boldsymbol{\Xi}$ and $\boldsymbol{\Xi}'$ which differ by at most one example, it holds that*

$$\sup_{\boldsymbol{z}} \mathbb{E}_{\mathcal{A}}[f(\mathcal{A}(\boldsymbol{\Xi}), \boldsymbol{z}) - f(\mathcal{A}(\boldsymbol{\Xi}'), \boldsymbol{z})] \leq \varepsilon.$$

Lemma 32 *Let $\mathcal{A}$ be ε-uniformly stable, then $|\varepsilon_{\rm gen}| \leq \varepsilon$.*

In this section, we demonstrate that the polynomial kernel employed in our method (as discussed in Section 4) enhances algorithmic stability, particularly in high-dimensional scenarios such as training deep neural networks. Inspired by the ODE approach for Bregman gradient-type methods (Alvarez et al., 2004; Ding and Toh, 2024), the ODE corresponding to SBPG is given by:

$$\frac{d\nabla\phi(\boldsymbol{x}(t))}{dt} = -\nabla F(\boldsymbol{x}(t)),$$

which is equivalent to the following ODE:

$$\dot{\boldsymbol{x}}(t) = -[\nabla^2\phi(\boldsymbol{x}(t))]^{-1}\nabla F(\boldsymbol{x}(t)).$$

The discrete version of this ODE then leads to the Hessian preconditioned gradient method:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k[\nabla^2\phi(\boldsymbol{x}^k)]^{-1}\nabla F(\boldsymbol{x}^k).$$

This connection between the Bregman gradient method and the Hessian preconditioned gradient method inspires us to derive the following estimation on the expected generalization gap:

Theorem 33 *Given a dataset $\boldsymbol{\Xi}$ containing n samples. Let $\mathcal{A}$ be the method* (SBPG) *where the last iterate is the output, and let $\mathcal{E}$ be the event where the iterates generated by $\mathcal{A}$ is contained within a compact set $\mathcal{B}$. Suppose f is differentiable and the kernel function ϕ is twice differentiable. Under the event $\mathcal{E}$, for any sufficiently small $\{\alpha_k\}$, we have*

$$|\mathbb{E}_{\boldsymbol{\Xi},\mathcal{A}}[\Phi_{\boldsymbol{\Xi}}(\mathcal{A}(\boldsymbol{\Xi})) - \Phi(\mathcal{A}(\boldsymbol{\Xi}))]| \leq \sum_{t=1}^{k} \exp\left(\left(1 - \frac{1}{n}\right) L_{\mathcal{B}} \sum_{i=t+1}^{k} \alpha_i\right) \frac{3\ell_{f,\mathcal{B}}^2 \ell_{\mathcal{B}} \alpha_t}{n} =: \varepsilon_{\rm gen}^{\phi}, \quad (28)$$

where $L_{\mathcal{B}} = \sup_{\boldsymbol{x}\in\mathcal{B},\boldsymbol{y}\in\mathcal{B},\boldsymbol{z}} \frac{\left\|[\nabla^2\phi(\boldsymbol{x})]^{-1}\nabla f(\boldsymbol{x},\boldsymbol{z})-[\nabla^2\phi(\boldsymbol{y})]^{-1}\nabla f(\boldsymbol{y},\boldsymbol{z})\right\|}{\|\boldsymbol{x}-\boldsymbol{y}\|}$, $\ell_{f,\mathcal{B}} = \sup_{\boldsymbol{x}\in\mathcal{B},\boldsymbol{y}\in\mathcal{B},\boldsymbol{z}} \frac{\|f(\boldsymbol{x},\boldsymbol{z})-f(\boldsymbol{y},\boldsymbol{z})\|}{\|\boldsymbol{x}-\boldsymbol{y}\|}$, *and* $\ell_{\mathcal{B}} = \sup_{\boldsymbol{x}\in\mathcal{B}} \left\|[\nabla^2\phi(\boldsymbol{x})]^{-1}\right\|$.

Proof Given the dataset $\boldsymbol{\Xi}$, recall the update scheme of SBPG:

$$\boldsymbol{x}_{k+1} = \nabla\phi^*(\nabla\phi(\boldsymbol{x}_k) - \alpha_k\nabla f(\boldsymbol{x}_k,\boldsymbol{z}_k)).$$

Noting that $\nabla\phi^*(\nabla\phi(\boldsymbol{x})) = \boldsymbol{x}$ and $\nabla^2\phi^*(\nabla\phi(\boldsymbol{x})) = [\nabla^2\phi(\boldsymbol{x})]^{-1}$, we have

$$\begin{aligned}\boldsymbol{x}_{k+1} =& \nabla\phi^*(\nabla\phi(\boldsymbol{x}_k) - \alpha_k\nabla f(\boldsymbol{x}_k,\boldsymbol{z}_k))\\ =& \nabla\phi^*(\nabla\phi(\boldsymbol{x}_k)) - \alpha_k\nabla^2\phi^*(\nabla\phi(\boldsymbol{x}_k))\nabla f(\boldsymbol{x}_k,\boldsymbol{z}_k) + \mathcal{O}(\alpha_k^2)\\ =& \boldsymbol{x}_k - \alpha_k[\nabla^2\phi(\boldsymbol{x}_k)]^{-1}\nabla f(\boldsymbol{x}_k,\boldsymbol{z}_k) + \mathcal{O}(\alpha_k^2),\end{aligned}$$

where the second equality comes from Taylor's expansion. Let $\boldsymbol{\Xi}'$ be a dataset that differs from $\boldsymbol{\Xi}$ by one example. Define $\delta_{k+1} := \mathbb{E}\left[\left\|\boldsymbol{x}_{k+1} - \boldsymbol{x}'_{k+1}\right\|\right]$, where $\boldsymbol{x}'_{k+1}$ corresponds to the dataset $\boldsymbol{\Xi}'$. From the inequality above, we have

$$\begin{aligned}&\left\|\boldsymbol{x}_{k+1} - \boldsymbol{x}'_{k+1}\right\|\\ =&\left\|\left(\boldsymbol{x}_k - \alpha_k[\nabla^2\phi(\boldsymbol{x}_k)]^{-1}\nabla f(\boldsymbol{x}_k,\boldsymbol{z}_k)\right) - \left(\boldsymbol{x}'_k - \alpha_k[\nabla^2\phi(\boldsymbol{x}'_k)]^{-1}\nabla f(\boldsymbol{x}'_k,\boldsymbol{z}'_k)\right)\right\| + \mathcal{O}(\alpha_k^2)\\ \le&\left\|\boldsymbol{x}_k - \boldsymbol{x}'_k\right\| + \alpha_k\left\|[\nabla^2\phi(\boldsymbol{x}_k)]^{-1}\nabla f(\boldsymbol{x}_k,\boldsymbol{z}_k) - [\nabla^2\phi(\boldsymbol{x}'_k)]^{-1}\nabla f(\boldsymbol{x}'_k,\boldsymbol{z}'_k)\right\| + \mathcal{O}(\alpha_k^2).\end{aligned}$$

Since $\boldsymbol{x}_0 = \boldsymbol{x}'_0$, then we have

$$\begin{aligned}\delta_{k+1} &= \left(1-\frac{1}{n}\right)(1+\alpha_k L_{\mathcal{B}})\,\delta_k + \frac{1}{n}\left(\delta_k + 2\ell_{f,\mathcal{B}}\ell_{\mathcal{B}}\alpha_k\right) + C\alpha_k^2\\ &\le \left(1+\alpha_k\left(1-\frac{1}{n}\right)L_{\mathcal{B}}\right)\delta_k + \frac{3\ell_{f,\mathcal{B}}\ell_{\mathcal{B}}\alpha_k}{n}\\ &\le \exp\left(\alpha_k\left(1-\frac{1}{n}\right)L_{\mathcal{B}}\right)\delta_k + \frac{3\ell_{f,\mathcal{B}}\ell_{\mathcal{B}}\alpha_k}{n}.\end{aligned} \tag{29}$$

The first inequality comes from the fact that α_k is sufficiently small. Since $\delta_0 = 0$, by recursion (29), we have

$$\delta_k \le \sum_{t=1}^{k}\exp\left(\left(1-\frac{1}{n}\right)L_{\mathcal{B}}\sum_{i=t+1}^{k}\alpha_i\right)\frac{3\ell_{f,\mathcal{B}}\ell_{\mathcal{B}}\alpha_t}{n}.$$

Finally, using the bound $\sup_{\boldsymbol{z}}\mathbb{E}_{\mathcal{A}}[f(\boldsymbol{x}_k,\boldsymbol{z}) - f(\boldsymbol{x}'_k,\boldsymbol{z})] \le \ell_{f,\mathcal{B}}\delta_k$ and Lemma 32, we complete the proof. ■

Now, we make some remarks on Theorem 33.

Remark 34 *1. When* $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2$, *we use the notation* $\bar{L}_{\mathcal{B}} = \sup_{\boldsymbol{x}\in\mathcal{B},\boldsymbol{y}\in\mathcal{B},\boldsymbol{z}} \frac{\|\nabla f(\boldsymbol{x},\boldsymbol{z})-\nabla f(\boldsymbol{y},\boldsymbol{z})\|}{\|\boldsymbol{x}-\boldsymbol{y}\|}$, $\bar{\ell}_{\mathcal{B}} = 1$. *If* $\alpha_k \le \frac{c}{k}$ *for some constant c, the right-hand side of* (28) *recovers the bound in (Hardt et al., 2016, Theorem 3.12).*

2. *If we choose the polynomial kernel from Proposition 27 with* $\delta = 1$ *and* $r = 4$, *i.e.* $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{1}{4}\|\boldsymbol{x}\|^4$, *then* $\nabla^2\phi(\boldsymbol{x})^{-1} = \frac{1}{1+\|\boldsymbol{x}\|^2}I - \frac{2\boldsymbol{x}\boldsymbol{x}^T}{(1+3\|\boldsymbol{x}\|^2)(1+\|\boldsymbol{x}\|^2)}$. *Thus, we have* $\ell_{\mathcal{B}} = \sup_{\boldsymbol{x}\in\mathcal{B}} \left\| \frac{1}{1+\|\boldsymbol{x}\|^2}I - \frac{2\boldsymbol{x}\boldsymbol{x}^T}{(1+3\|\boldsymbol{x}\|^2)(1+\|\boldsymbol{x}\|^2)} \right\| \leq 1$. *When* $\mathcal{B}$ *is a convex compact set, by the intermediate theorem, we have*

$$L_{\mathcal{B}} = \ell_{f,\mathcal{B}} \sup_{\boldsymbol{x}\in\mathcal{B}} \left\{ \left\| D_{\nabla^2\phi(\boldsymbol{x})^{-1}} \right\| \right\} + \ell_{\mathcal{B}}\bar{L}_{\mathcal{B}},$$

where $D_{\nabla^2\phi(\boldsymbol{x})^{-1}}$ *is the first order differential operator of* $\nabla^2\phi(\boldsymbol{x})^{-1}$. *Furthermore, if* $\operatorname{dist}(0,\mathcal{B}) \geq M > 0$, *then* $\ell_{\mathcal{B}} \leq \frac{1}{1+M^2} < 1$. *Moreover by some basic algebraic calculations, we have* $D_{\mathcal{B}} := \sup_{\boldsymbol{x}\in\mathcal{B}} \left\{ \left\| D_{\nabla^2\phi(\boldsymbol{x})^{-1}} \right\| \right\} \leq \begin{cases} 6\|\boldsymbol{x}\|(1+4\|\boldsymbol{x}\|^2), & \|\boldsymbol{x}\| \leq 1, \\ \frac{12}{1+3\|\boldsymbol{x}\|^2}, & \|\boldsymbol{x}\| > 1. \end{cases}$
When M *is sufficiently large, which usually occurs in high-dimensional scenarios,* $D_{\mathcal{B}}$ *becomes very small. Thus,* $L_{\mathcal{B}} < \bar{L}_{\mathcal{B}}$. *In summary, when a high-order polynomial kernel is employed,* (SBPG) *tends to have a better stability bound compared to standard SGD.*

Appendix C. Additional Experimental Results

In this section, we provide further experimental results and assess the performance of MSBPG on more recent neural network architectures, including ConvNext (Liu et al., 2022) and the Vision Transformer (ViT) (Dosovitskiy, 2020). The networks are trained on the CIFAR-100 dataset for 200 epochs, utilizing a cosine annealing learning rate schedule (Loshchilov and Hutter, 2016), with a batch size of 128. The test accuracies for different optimizers, including MSBPG, SGD, Adam, and AdamW, are reported in Table 1. Notably, MSBPG achieved slightly higher test accuracy compared to the other methods. This advantage can be attributed to the Bregman proximity model used in our approach.

Method	MSBPG	SGD	Adam	AdamW
ViT Tiny	62.37	60.65	56.88	58.77
ViT Small	63.59	61.90	57.66	59.39
ConvNext Atto	75.98	74.46	73.76	75.29
ConvNext Femto	76.47	75.96	74.35	75.65

Table 1: Test accuracy (%) of different optimizers on CIFAR-100 dataset.

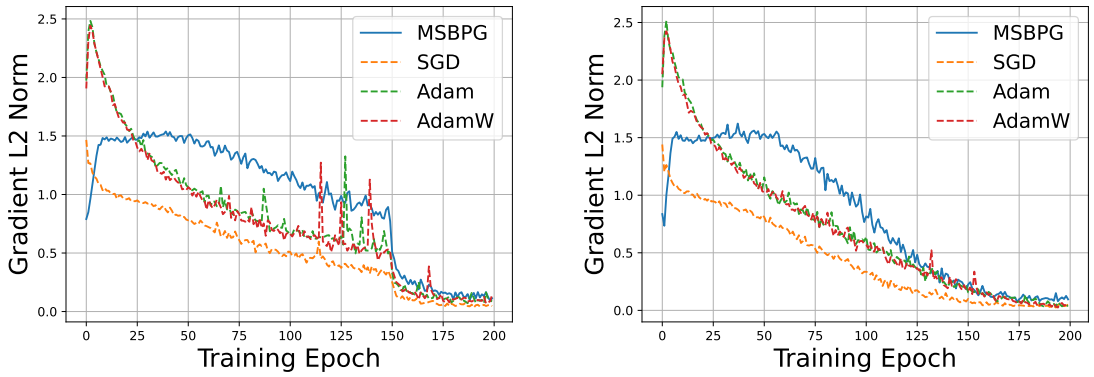
To further evaluate the performance of MSBPG, we applied it to Transformer-XL (Dai et al., 2019), a model designed for long-sequence tasks. We followed the official configuration to train the Transformer-XL-based model on the WikiText-103 dataset (Merity et al., 2016), a large-scale word-level language modeling benchmark that involves long-term dependencies. The performance of MSBPG, along with that of SGD, Adam, and AdamW, was measured by the test perplexity after 50,000 training steps. The results are presented in Table 2.

We also plot the gradient norm of the objective function, which indicates the stationarity of the minimization problem when the nonsmooth term is absent, as shown in Figure 11. From the figure, we observe that our method, MSBPG, successfully reaches a stationary point, similar to other optimization methods. It is important to highlight that our method

	MSBPG	SGD	Adam	AdamW
Transformer-XL	31.07	33.81	33.53	32.17

Table 2: Test perplexity (lower is better) for Transformer-XL-based model on the WikiText-103 dataset.

theoretically converges to the stationary point without requiring Lipschitz smoothness, while the convergence of traditional methods relies on this property. With careful tuning of the learning rate, methods such as SGD, Adam, and AdamW can also achieve stationary points, but they are more sensitive to learning rate choices compared to MSBPG. This sensitivity is further demonstrated experimentally in Figure 10, and is partially attributed to the absence of Lipschitz smoothness, which increases the sensitivity to the learning rate.



(a) Step decay

(b) Cosine annealing

Figure 11: Gradient norm of the objective function. When the nonsmooth term is absent, the gradient norm can imply stationarity of the minimization problem.

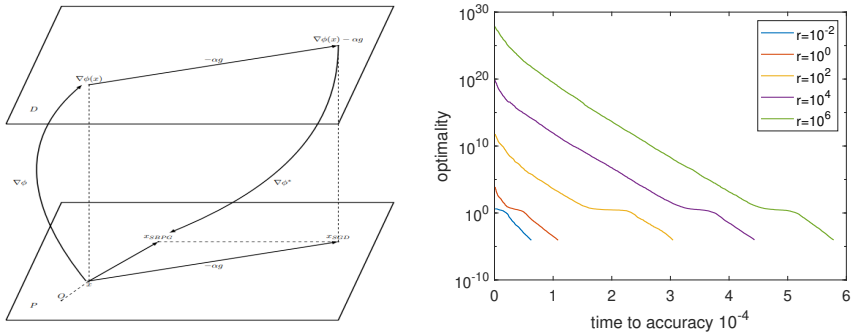
Appendix D. Additional Figures

References

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018.

Felipe Alvarez, Jérôme Bolte, and Olivier Brahic. Hessian riemannian gradient flows in convex programming. *SIAM journal on control and optimization*, 43(2):477–501, 2004.

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1):165–214, 2023.



(a) Illustration of SBPG and SGD updates

(b) Different initial points for SBPG

Figure 12: Figure (a) depicts SGD and SBPG updates. SBPG includes a "pull back" mechanism that prevents the point from moving excessively in any given direction. "P" and "D" refer to the primal and dual spaces, respectively, and these terms are commonly used in the mirror descent method literature (see, e.g., Bubeck et al. (2015); Nemirovskij and Yudin (1983)). Figure (b) illustrates the effect of choosing the initial point from a ball of radius r on the SBPG when r changes for the QIP example with $d = 100$ and $n = 5000$. All initial step sizes are set to 1×10^{-3}. As shown in Figure (b), even for an initial point that is far from the optimal point, SBPG can pull back the iterates to the optimal point.

Navid Azizan, Sahin Lale, and Babak Hassibi. Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. *arXiv preprint arXiv:1906.03830*, 2019.

Heinz H Bauschke, Jérôme Bolte, and Marc Teboulle. A descent lemma beyond Lipschitz gradient continuity: first-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2017.

Amir Beck and Yonina C Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal on Optimization*, 23(3):1480–1509, 2013.

Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2):163–195, 2011.

Dimitri P Bertsekas and John N Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

Pascal Bianchi. Ergodic convergence of a stochastic proximal point algorithm. *SIAM Journal on Optimization*, 26(4):2235–2260, 2016.

Jérôme Bolte, Shoham Sabach, Marc Teboulle, and Yakov Vaisbourd. First order methods beyond convexity and Lipschitz gradient continuity with applications to quadratic inverse problems. *SIAM Journal on Optimization*, 28(3):2131–2151, 2018.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

Gong Chen and Marc Teboulle. Convergence analysis of a proximal-like minimization algorithm using bregman functions. *SIAM Journal on Optimization*, 3(3):538–543, 1993.

Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3267–3275, 2021.

Yuejie Chi, Yue M Lu, and Yuxin Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *IEEE Transactions on Signal Processing*, 67(20):5239–5269, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Kuangyu Ding and Kim-Chuan Toh. Stochastic bregman subgradient methods for nonsmooth nonconvex optimization problems. *arXiv preprint arXiv:2404.17386*, 2024.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Radu-Alexandru Dragomir, Alexandre d'Aspremont, and Jérôme Bolte. Quartic first-order methods for low-rank minimization. *Journal of Optimization Theory and Applications*, 189:341–363, 2021a.

Radu Alexandru Dragomir, Mathieu Even, and Hadrien Hendrikx. Fast stochastic bregman gradient methods: Sharp analysis and variance reduction. In *International Conference on Machine Learning*, pages 2815–2825. PMLR, 2021b.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Rui Fu, Zuo Zhang, and Li Li. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 2016.

Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. *Advances in neural information processing systems*, 29, 2016.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

Bin Gu, Wenhan Xian, Zhouyuan Huo, Cheng Deng, and Heng Huang. A unified q-memorization framework for asynchronous stochastic optimization. *The Journal of Machine Learning Research*, 21(1):7761–7813, 2020.

Filip Hanzely and Peter Richtárik. Fastest rates for stochastic mirror descent methods. *Computational Optimization and Applications*, 79:717–766, 2021.

Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated Bregman proximal gradient methods for relatively smooth convex optimization. *Computational Optimization and Applications*, 79:405–440, 2021.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR, 2016.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*, volume 305. Springer science & business media, 1993.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. pmlr, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Puya Latafat, Andreas Themelis, Masoud Ahookhosh, and Panagiotis Patrinos. Bregman finito/miso for nonconvex regularized finite sum minimization without lipschitz gradient continuity. *SIAM Journal on Optimization*, 32(3):2230–2262, 2022.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Mathematical programming*, 176:311–337, 2019.

Yan Li, Caleb Ju, Ethan X Fang, and Tuo Zhao. Implicit regularization of bregman proximal point algorithm and mirror descent on separable data. *arXiv preprint arXiv:2108.06808*, 2021.

Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust region newton methods for large-scale logistic regression. In *Proceedings of the 24th international conference on Machine learning*, pages 561–568, 2007.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Haihao Lu. "Relative continuity" for non-Lipschitz nonsmooth convex optimization using stochastic (or deterministic) mirror descent. *INFORMS Journal on Optimization*, 1(4): 288–303, 2019.

Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

D Russell Luke. Phase retrieval, what's new. *SIAG/OPT Views and News*, 25(1):1–5, 2017.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

Nikolay Manchev and Michael Spratling. Target propagation in recurrent neural networks. *The Journal of Machine Learning Research*, 21(1):250–282, 2020.

Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273, 1994.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. *Wiley Interscience*, 1983.

Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

Andrew Y Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 78, 2004.

Ioannis Panageas, Georgios Piliouras, and Xiao Wang. First-order methods almost always avoid saddle points: The case of vanishing step-sizes. *Advances in Neural Information Processing Systems*, 32, 2019.

Alasdair Paren, Leonard Berrada, Rudra PK Poudel, and M Pawan Kumar. A stochastic bundle method for interpolating networks. *Journal of Machine Learning Research*, 23: 1–57, 2022.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. Pmlr, 2013.

Andrei Patrascu and Ion Necoara. Nonasymptotic convergence of stochastic proximal point methods for constrained convex optimization. *The Journal of Machine Learning Research*, 18(1):7204–7245, 2017.

Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki*, 3(4):643–653, 1963.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

R Tyrrell Rockafellar. *Convex analysis*, volume 11. Princeton university press, 1997.

R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational Analysis*. Springer Verlag, Heidelberg, Berlin, New York, 1998.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research*, 11:2635–2670, 2010.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of LSTM and BiLSTM in forecasting time series. In *2019 IEEE International Conference on Big Data*, pages 3285–3292. IEEE, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Haoyuan Sun, Kwangjun Ahn, Christos Thrampoulidis, and Navid Azizan. Mirror descent maximizes generalized margin and can be implemented efficiently. *Advances in Neural Information Processing Systems*, 35:31089–31101, 2022.

Haoyuan Sun, Khashayar Gatmiry, Kwangjun Ahn, and Navid Azizan. A unified approach to controlling implicit regularization via mirror descent. *Journal of Machine Learning Research*, 24(393):1–58, 2023.

Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18:1131–1198, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.

Marc Teboulle. A simplified view of first order methods for optimization. *Mathematical Programming*, 170(1):67–96, 2018.

Bokun Wang, Shiqian Ma, and Lingzhou Xue. Riemannian stochastic proximal gradient methods for nonsmooth optimization over the stiefel manifold. *The Journal of Machine Learning Research*, 23(1):4599–4631, 2022.

Mengdi Wang, Ethan X Fang, and Han Liu. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161:419–449, 2017.

Qingsong Wang and Deren Han. A Bregman stochastic method for nonconvex nonsmooth problem beyond global Lipschitz gradient continuity. *Optimization Methods and Software*, 38(5):914–946, 2023.

Lei Yang and Kim-Chuan Toh. Inexact Bregman Proximal Gradient Method and its Inertial Variant with Absolute and Relative Stopping Criteria. *arXiv preprint arXiv:2109.05690*, 2021.

Lei Yang and Kim-Chuan Toh. Bregman proximal point algorithm revisited: A new inexact version and its inertial variant. *SIAM Journal on Optimization*, 32(3):1523–1554, 2022.

Lei Yang, Jingjing Hu, and Kim-Chuan Toh. An inexact bregman proximal difference-of-convex algorithm with two types of relative stopping criteria. *arXiv preprint arXiv:2406.04646*, 2024.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 116, 2004.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in Neural Information Processing Systems*, 33:18795–18806, 2020.