Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 11(2): 4759–4786.
DOI:10.3934/math.2026194
Received: 11 November 2025
Revised: 06 February 2026
Accepted: 13 February 2026
Published: 26 February 2026

Research article

A new mini-batch negative momentum proximal stochastic variance reduction method for nonconvex optimization

Weihao Cui[1], **Chongyang He**[2], **Mingyuan Cao**[1] **and Yueting Yang**[1,*]

[1] School of Mathematics and Statistics, Beihua University, Jilin 132013, China
[2] School of Engineering, RMIT University, Melbourne 3000, Australia

* **Correspondence:** Email: yyt2858@163.com.

Abstract: First-order stochastic optimization algorithms have been widely applied to large-scale machine learning tasks. We have proposed a new stochastic optimization algorithm, which is inspired by an accelerated stochastic variance reduced method originally developed for convex optimization. The proposed algorithm addresses nonconvex and nonsmooth problems via the proximal operator and mitigates overfitting through a mini-batch strategy that exploits richer gradient information. We established sublinear convergence under standard assumptions. Extensive experiments on synthetic classification tasks, real-world datasets, nonconvex matrix factorization problems, and time series prediction tasks showed that the proposed algorithm consistently achieves faster convergence and competitive test performance compared to state-of-the-art stochastic optimization algorithms, underscoring its potential in practical applications for large-scale, nonsmooth learning problems.

Keywords: stochastic gradient method; machine learning; mini-batch strategy; proximal operator; nonconvex; nonsmooth optimization
Mathematics Subject Classification: 65K05, 90C30

1. Introduction

The empirical risk minimization problem (ERM) with regularization is widely encountered in fields such as word processing [1], deep learning [2, 3], and matrix factorization and completion [4]. Its structure can be expressed in the following form:

$$\min_{x\in\mathbb{R}^d} P(x) = f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x), \tag{1.1}$$

where n is the number of samples, $f_i(x) : \mathbb{R}^d \to \mathbb{R}$ $(i = 1, \ldots, n)$ represents the loss function, which is smooth but potentially nonconvex, and $h(x) : \mathbb{R}^d \to \mathbb{R}$ serves as the regularization term, which is convex but potentially nonsmooth. It is worth noting that if the regularization function $h(x)$ is smooth, it can be implicitly absorbed into the objective function $f(x)$. Different combinations of loss functions and regularization terms give rise to distinct optimization models for various machine learning tasks. For instance, combining the quadratic loss function with l_2-norm regularization leads to the ridge regression problem [5]; pairing quadratic loss with the l_1-norm yields the Lasso problem [6]; combining the hyperbolic tangent-based loss function with the l_1-norm constitutes a class of nonconvex and nonsmooth binary classification optimization problems [7]; and combining nonconvex logistic loss with the graph-guided fused l_1-norm gives rise to a class of structured nonconvex and nonsmooth optimization problems with variable dependencies [8].

The most fundamental method to solve (1.1) is the proximal gradient descent (ProxGD) method [9], where its iterative format is given by

$$x_{t+1} = \operatorname{prox}_{\eta h}\left(x_t - \eta \nabla f(x_t)\right),$$

where $\operatorname{prox}_{\eta h}(x) = \underset{y\in\mathbb{R}^d}{\operatorname{argmin}}\{h(y) - \frac{1}{2\eta}\|y - x\|^2\}$ is the proximal operator, and $\eta > 0$ is the step size (also known as the learning rate). The complexity and scale of (1.1) are closely related to n. If n becomes exceptionally large, then the cost of computing the full gradient increases significantly. Ghadimi et al. [10], inspired by the stochastic gradient descent (SGD) algorithm proposed by Robbins and Monro [11], proposed the proximal stochastic gradient descent (ProxSGD) method to solve problem (1.1). The algorithm adopted the idea of a mini-batch and combined proximal operators with the stochastic gradient descent algorithm. The key iterative step of this algorithm is as follows:

$$x_{t+1} = \operatorname{prox}_{\eta_t h}\left(x_t - \frac{\eta_t}{|I_t|}\sum_{i_t\in I_t}\nabla f_{i_t}(x_t)\right),$$

where I_t is a mini-batch randomly selected from $\{1,\ldots,n\}$, the size of I_t is typically denoted as $|I_t|$, and i_t is uniformly sampled from $\{1,\ldots,n\}$. The use of mini-batch sampling in the algorithm has been verified to be beneficial. Compared to the approach of randomly selecting a single sample at each iteration, it randomly uses $|I_t|$ samples to approximate the full gradient. It allows more useful gradient information to be utilized, thereby improving its efficiency. The performance of ProxSGD is highly sensitive to the choice of step size, and the algorithm typically requires a diminishing step size to ensure convergence. Due to the stochastic nature of the sampling process, the convergence rate of ProxSGD is limited to sublinear convergence.

To improve the convergence rate of SGD for ERM problems, a variety of new algorithms have been developed. Johnson and Zhang [12] introduced the stochastic variance reduced gradient (SVRG) algorithm. This method constructs a specific gradient estimator that ensures the variance of stochastic gradients is bounded above by a decreasing sequence, thereby accelerating convergence. Defazio and Bach [13] proposed the stochastic average gradient (SAGA) algorithm, inspired by the SVRG. The SAGA maintains an individual gradient estimate for each sample. During the iterative process, one sample is selected at a time, and its stored gradient is updated with the newly computed one. Compared to the SVRG, the SAGA requires n times more storage, which makes the SVRG more advantageous in scenarios where both the feature dimension and dataset size are large. Nguyen et al. [14] introduced the stochastic recursive gradient (SARAH) algorithm. Unlike the SAGA, the SARAH estimates gradients recursively within the inner loop using accumulated stochastic information, without the need to store all past gradients. Moreover, its inner loop achieves sublinear convergence.

To handle the nonsmooth regularization term in (1.1), many scholars combined the proximity operator with the above three algorithms, resulting in the proximal variants of these algorithms: ProxSVRG [15], ProxSAGA [16], and ProxSARAH [17]. Among them, the ProxSVRG method proposed by Xiao and Zhang [15] was the most widely applied. Its iterative update is as follows:

$$x_{t+1}^{s+1} = \operatorname{prox}_{\eta h}(x_t^{s+1} - \eta v_t^{s+1}),$$

where $v_t^{s+1} = \frac{1}{b}\sum_{i_t \in I_t}\left(\nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s)$, with $\tilde{x}^s$ being the snapshot point.

Another line of work on accelerating stochastic algorithms builds on Nesterov's pioneering method [18], which has been shown to outperform standard gradient descent. Given the limitations of second-order methods in large-scale optimization, first-order acceleration techniques have attracted growing attention in machine learning. However, in many tasks driven by the pursuit of low-rank structures and sparsity, the regularization term in problem (1.1) becomes non-differentiable, rendering the overall objective nonsmooth. Beck and Teboulle [19] proposed the fast iterative shrinkage-thresholding algorithm (FISTA) to address such nonsmooth problems. Other extrapolation-based acceleration methods have also been applied to optimization tasks across various domains [20–22]. Despite their empirical success, the theoretical foundations of Nesterov's acceleration remain not fully understood, leading researchers to explore explanations from multiple perspectives. From a primal-dual viewpoint, Allen-Zhu and Orecchia [23] interpreted the accelerated gradient method as a linear combination of gradient descent and mirror descent. Drawing on the ellipsoid method, Bubeck et al. [24] introduced a geometric acceleration scheme. Motivated by variational principles, Su et al. [25] examined Nesterov's method through the lens of differential equations.

In recent years, researchers have increasingly integrated acceleration techniques with stochastic first-order methods and applied them to a wide range of learning tasks. To eliminate direct dependence on n, attention has shifted toward solving problem (1.1) in the dual space. Nitanda [26] introduced the Acc-Prox-SVRG method, incorporating Nesterov's acceleration into ProxSVRG to address the nonsmooth case of problem (1.1). More recently, Yang [27] proposed the SARAH-M method by integrating momentum techniques into SARAH, achieving linear convergence for strongly convex functions. Allen-Zhu [28] developed the well-known Katyusha algorithm, which introduced a negative momentum term to realize true acceleration in stochastic variance reduced gradient methods. He et al. [29] proposed the negative momentum SVRG (NMSVRG) algorithm, combining Katyusha's negative momentum mechanism with the classical SVRG framework. Their method requires fewer hyperparameter adjustments and achieves linear convergence under strong convexity. Table 1 summarizes the advantages and limitations of the algorithms discussed above. In addition, several other representative works can be found in [30–32].

Motivated by the NMSVRG and ProxSVRG algorithms, this paper introduces a new first-order mini-batch accelerated stochastic method for solving problem (1.1), termed the mini-batch negative momentum proximal stochastic variance reduced gradient (MNMPSVRG) method. The main contributions of this work are summarized as follows:

Table 1. Summary of the advantages and defects of the algorithms mentioned.

Methods	Advantages	Defects
SVRG [12]	Full-gradient correction reduces variance without the need to store historical gradients, and achieves linear convergence for strongly convex problems.	Periodic full-gradient computation raises per-iteration cost and demands careful hyperparameter tuning; in nonconvex optimization, the method heavily relies on good initialization to avoid poor local minima.
SAGA [13]	Leveraging unbiased gradient estimators, the algorithm attains a superior convergence rate compared to the SVRG in the strongly convex setting.	The necessity to store historical gradients leads to increased computational and memory costs.
SARAH [14]	The method achieves linear convergence in the inner loop through recursive gradient estimation, surpassing the performance of the SVRG, while eliminating the need for storing historical gradients.	The inner loop size is sensitive and requires careful tuning, and the convergence rate for strongly convex problems does not surpass that of the SVRG framework.
Nesterov [18]	The incorporation of a momentum term enables the method to attain the theoretically optimal convergence rate in deterministic convex optimization settings.	Its acceleration effect may deteriorate in stochastic or nonconvex problems, where oscillations and instability can occur without careful tuning.
Katyusha [28]	The incorporation of a "negative momentum" term enhances the convergence rate of the algorithm.	The algorithm's performance is sensitive to hyperparameter selection, necessitating meticulous parameter tuning for optimal results.
SARAH-M [27]	For strongly convex functions, the integration of momentum and adaptive step-size strategies enables efficient convergence while maintaining low variance.	The approach exhibits sensitivity to hyperparameter selection, is theoretically restricted to strongly convex settings, and involves relatively high computational overhead per iteration.

(1). We integrate the negative momentum framework into the ProxSVRG algorithm. Compared with the classical Katyusha acceleration scheme, our method requires only a single negative momentum parameter, which simplifies implementation and effectively reduces the cumulative error introduced by stochasticity.

(2). We incorporate the mini-batch strategy into the negative momentum-based ProxSVRG algorithm. By leveraging multiple samples in each iteration, the algorithm mitigates overfitting caused by single-sample updates and improves overall performance.

(3). We provide a rigorous theoretical convergence analysis of the proposed algorithm under nonconvex and nonsmooth settings, and prove that the MNMPSVRG algorithm achieves a sublinear convergence rate. Extensive numerical experiments are conducted on nonconvex and nonsmooth optimization tasks, including synthetic classification problems, real-world datasets, nonconvex matrix factorization, and time series prediction, which collectively demonstrate the superior performance of the proposed method.

In Section 2, we briefly review relevant definitions and their properties. Section 3 introduces the MNMPSVRG algorithm. A detailed analysis of its convergence properties is provided in Section 4. Section 5 presents empirical results to validate the effectiveness of our method. Finally, Section 6 concludes the paper.

2. Notations and preliminaries

In this paper, we use $\mathbb{R}^d$ to denote the d-dimensional Euclidean space, with its standard inner product denoted by $\langle \cdot, \cdot \rangle$. The Euclidean norm is denoted by $|| \cdot ||$, without specific mention. We use $|| \cdot ||_1$ to represent the l_1 norm. For a stochastic algorithm, we use $\mathbb{E}[\cdot]$ to denote the total expectation in terms of its whole iteration process.

Below we show several basic assumptions and definitions associated with their properties.

Assumption 2.1. *The gradient of each component function $f_i(x)$ is Lipschitz continuous, that is, for each i, there exists a constant $L_i > 0$ such that*

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

Then, it is easy to obtain $||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$, where $L = \sum_{i=1}^n L_i$.

Assumption 2.2. *The regularization $h(x)$: $\mathbb{R}^d \to \mathbb{R}$ in (1.1) is a lower semi-continuous and convex function, and its effective domain, $dom(h) = \{x \in \mathbb{R}^d | h(x) < +\infty\}$, is closed.*

Definition 2.1. *[33] For a convex function h, its proximity operator is defined as*

$$prox_h(x) = \underset{u \in \boldsymbol{dom}(h)}{\operatorname{argmin}} \{h(u) - \frac{1}{2}||u - x||^2\}.$$

The convergence criterion is an important aspect in the analysis of nonsmooth and nonconvex problems. For convex problems, the criterion we usually use is the optimality gap $P(x) - P(x^*)$. Due to the intractability of general nonconvex problems, using this criterion is unreasonable. For smooth nonconvex problems (i.e., $h \equiv 0$), a typical method is to measure stationarity, such as using $||\nabla P(x)||$, but this cannot be used for nonsmooth problems. A fitting alternative method is to use gradient mapping.

Definition 2.2. *[34] For nonsmooth problems, the gradient mapping of function $P(x)$ is defined as follows:*

$$\mathcal{G}_\eta(x) = \frac{1}{\eta}\left[x - \operatorname{prox}_{\eta h}(x - \eta\nabla f(x))\right]. \tag{2.1}$$

If $h \equiv 0$, this mapping reduces to $\mathcal{G}_\eta(x) = \nabla f(x) = \nabla P(x)$, that is, the gradient of function $P(x)$. We analyze our algorithms using the gradient mapping (2.1) as described more precisely above. The purpose of using the definition of gradient mapping is that, in the case of nonconvex functions, first-order algorithms generally can only guarantee convergence to a stable point and cannot guarantee convergence to the optimal solution. Similar to nonconvex smooth functions using the gradient norm as the stopping criterion, for nonconvex composite functions, we use the gradient mapping norm to characterize the convergence rate of an algorithm.

Definition 2.3. *[34] A point x output by a stochastic iterative algorithm for solving (1.1) is called an ε-accurate solution, if $\mathbb{E}[\|\mathcal{G}_\eta(x)\|^2] \le \varepsilon$ for some $\eta > 0$.*

3. The description of the MNMPSVRG algorithm

In this section, we introduce an accelerated mini-batch stochastic algorithm, termed the MNMPSVRG, for addressing problem (1.1). This method is built upon the concepts of a mini-batch and negative momentum, and employs a two-level nested loop structure, consisting of an outer and an inner loop. The acceleration effect is primarily realized within the inner loop, where the momentum mechanism significantly improves the convergence behavior.

In the classical Nesterov acceleration framework [18], an extrapolation step is introduced into certain stochastic algorithms, leading to the following iteration:

$$y_t^{s+1} = x_t^{s+1} + \beta_t(x_t^{s+1} - x_{t-1}^{s+1}),$$

where s and t denote the indices for the outer and inner iterations, respectively, and $\beta_t \in (0, 1)$ is the momentum parameter. The sequence $\{x_t^{s+1}\}$ is typically generated by various variance reduced stochastic methods. This design aims to accelerate convergence. However, in the presence of noisy data, the stochastic gradients may become unreliable, and incorporating momentum may exacerbate the bias, potentially impairing convergence.

To mitigate such adverse effects, we adopt a negative momentum framework, incorporating a Katyusha-style interpolation step:

$$y_t^{s+1} = \theta x_t^{s+1} + (1-\theta)\tilde{x}^s,$$

where $\theta \in (0, 1)$, and $\tilde{x}^s$ denotes a snapshot point. This update performs a convex combination of the current iterate and the snapshot, which acts as a form of interpolation, in contrast to Nesterov's extrapolation. Unlike the original Katyusha method, which utilizes two auxiliary sequences, our algorithm introduces only a single auxiliary variable, thereby simplifying implementation and hyperparameter tuning.

Next, we compute a variance reduced stochastic gradient estimate at the interpolation point

$$v_t^{s+1} = \frac{1}{b}\sum_{i_t \in I_t}\left(\nabla f_{i_t}(y_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s).$$

Since the mini-batch I_t is drawn randomly, the gradient estimate remains unbiased. Specifically, we have

$$\mathbb{E}[v_t^{s+1}] = \mathbb{E}\left[\frac{1}{b}\sum_{i_t \in I_t}\left(\nabla f_{i_t}(y_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s)\right] = \nabla f(y_t^{s+1}) - \nabla f(\tilde{x}^s) + \nabla f(\tilde{x}^s) = \nabla f(y_t^{s+1}).$$

Finally, we update the iterate using the proximal operator

$$x_{t+1}^{s+1} = \mathrm{prox}_{\eta h}\left(x_t^{s+1} - \eta v_t^{s+1}\right).$$

With this design, the proposed MNMPSVRG algorithm achieves improved convergence behavior and enhanced optimization efficiency.

Algorithm 1 describes the detailed process of the MNMPSVRG algorithm for solving nonconvex and nonsmooth optimization.

Algorithm 1 MNMPSVRG.

```
Require: Negative momentum coefficient $\theta \in (0, 1)$, stepsize $0 < \eta < \frac{1}{2L}$, initial point $\tilde{x}^0 = x_m^0 = x^0 \in \mathbb{R}^d$, epoch length $m$, $S = \lceil T/m \rceil$, and mini-batch size $b$.
 1: for $s = 0$ to $S - 1$ do
 2:     $x_0^{s+1} \leftarrow \tilde{x}^s$
 3:     $\nabla f(\tilde{x}^s) \leftarrow \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x}^s)$
 4:     for $t = 0$ to $m - 1$ do
 5:         Uniformly select subset $I_t \subset \{1, \ldots, n\}$ with $|I_t| = b$
 6:         $y_t^{s+1} \leftarrow \theta x_t^{s+1} + (1 - \theta)\tilde{x}^s$
 7:         $v_t^{s+1} \leftarrow \frac{1}{b}\sum_{i \in I_t}\left(\nabla f_i(y_t^{s+1}) - \nabla f_i(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s)$
 8:         $x_{t+1}^{s+1} \leftarrow \mathrm{prox}_{\eta h}\left(x_t^{s+1} - \eta v_t^{s+1}\right)$
 9:     end for
10:     $\tilde{x}^{s+1} \leftarrow x_m^{s+1}$
11: end for
Ensure: Iterate $x_a$ chosen uniformly random from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$.
```

4. Convergence analysis

In this section, we establish the convergence results of the proposed algorithm. Before providing the main theoretical results, we need to give the following assumptions and lemmas.

Lemma 4.1. *Let us define*

$$y = \mathrm{prox}_{\eta h}(x - \eta d')$$

for some $d' \in \mathbb{R}^d$. Then for y, the following inequality holds:

$$P(y) \le P(z) + \langle y - z, \nabla f(x) - d' \rangle + \left(\frac{L}{2} - \frac{1}{2\eta}\right)\|y - x\|^2 + \left(\frac{L}{2} + \frac{1}{2\eta}\right)\|z - x\|^2 - \frac{1}{2\eta}\|y - z\|^2$$

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

We note that the proof of Lemma 4.1 is similar to what is described in [34].

Lemma 4.2. *For the iterates x_t^{s+1}, v_t^{s+1}, and $\tilde{x}^s$ in Algorithm 1, the following inequality holds:*

$$\mathbb{E}\|\nabla f(x_t^{s+1}) - v_t^{s+1}\|^2 \leq \frac{(1+\tau)L^2}{b}\mathbb{E}\|y_t^{s+1} - \tilde{x}^s\|^2 + \frac{(1+\tau)L^2}{\tau}\mathbb{E}\|x_t^{s+1} - \tilde{x}^s\|^2, \quad \tau > 0.$$

Proof. According to the definition of v_t^{s+1}, we have

$$\begin{aligned}
&\mathbb{E}\|\nabla f(x_t^{s+1}) - v_t^{s+1}\|^2 \\
&= \mathbb{E}\|\nabla f_{I_t}(y_t^{s+1}) - \nabla f_{I_t}(\tilde{x}^s) - (\nabla f(x_t^{s+1}) - \nabla f(\tilde{x}^s))\|^2 \\
&\leq (1+\tau)\mathbb{E}\|\nabla f_{I_t}(y_t^{s+1}) - \nabla f_{I_t}(\tilde{x}^s)\|^2 + \frac{1+\tau}{\tau}\mathbb{E}\|\nabla f(x_t^{s+1}) - \nabla f(\tilde{x}^s)\|^2 \\
&\leq \frac{(1+\tau)L^2}{b}\mathbb{E}\|y_t^{s+1} - \tilde{x}^s\|^2 + \frac{(1+\tau)L^2}{\tau}\mathbb{E}\|x_t^{s+1} - \tilde{x}^s\|^2,
\end{aligned}$$

where the first inequality follows from the Cauchy-Schwarz and Young inequalities, and the second inequality is derived from Assumption 2.1. □

Before studying the convergence of the algorithm in the paper, we set up a recursion for which we use the following Lyapunov function:

$$R_t^{s+1} = \mathbb{E}\left[P(x_t^{s+1}) + c_t\|x_t^{s+1} - \tilde{x}^s\|^2\right], \quad t = 0, \ldots, m-1, \tag{4.1}$$

where $c_t = c_{t+1}(1 + \frac{1}{m}) + T_1 + T_2$, $T_1 = \frac{(1+\tau)\theta^2 L}{4b}$, $T_2 = \frac{(1+\tau)L}{4\tau}$.

Lemma 4.3. *For the sequence $\{c_t\}$ satisfying the boundary condition $c_m = 0$ with $c_t, c_{t+1} > 0$, if*

$$4m(T_1 + T_2)(1+m) + L \leq \frac{1}{\eta},$$

then the following inequality holds:

$$c_{t+1}(1+m) + \frac{L}{2} < \frac{1}{2\eta}.$$

Proof. From the definition of c_t and $c_m = 0$, we have

$$\begin{aligned}
c_t &= c_{t+1}(1 + \frac{1}{m}) + T_1 + T_2 \\
&= c_m(1 + \frac{1}{m})^{m-t} + \left[(1 + \frac{1}{m})^{m-t-1} + \cdots + (1 + \frac{1}{m}) + 1\right](T_1 + T_2) \\
&= m\left[\left(1 + \frac{1}{m}\right)^{m-t} - 1\right](T_1 + T_2) \\
&\leq m\left[\left(1 + \frac{1}{m}\right)^{m} - 1\right](T_1 + T_2) \\
&\leq 2m \cdot \frac{e-1}{2}(T_1 + T_2) \\
&\leq 2m(T_1 + T_2),
\end{aligned} \tag{4.2}$$

where the second inequality follows upon noting that (i) $\lim_{m\to+\infty}(1+\frac{1}{m})^m = e$ and (ii) $(1+\frac{1}{m})^m$ is an increasing function for $m > 0$ (here, e is Euler's number). By combining (4.2) with the inequality $4m(T_1+T_2)(1+m)+L \le \frac{1}{\eta}$, we can draw the conclusion. □

Theorem 4.1. *Let the sequence $\{x_a\}$ be generated by Algorithm 1. Denote $D = \max\{4m(T_1+T_2)(1+m)+L, 2L\}$. Suppose the total number of iterations T is a multiple of m, and the step size η satisfies $\eta \le \frac{1}{D}$. Then the following bound holds:*

$$\mathbb{E}\left[\|\mathcal{G}_\eta(x_a)\|^2\right] \le \frac{2\left(P(\tilde{x}^0)-P(x^*)\right)}{\eta T - 2\eta^2 TL},$$

where x^* is an optimal solution to problem (1.1).

Proof. We begin by introducing the full gradient-based reference point, defined as

$$\bar{x}_{t+1}^{s+1} = \mathrm{prox}_{\eta h}\left(x_t^{s+1} - \eta\nabla f(x_t^{s+1})\right), \tag{4.3}$$

which serves solely for theoretical analysis and is not computed during actual execution. Applying Lemma 4.1 to this point (with $y = \bar{x}_{t+1}^{s+1}$, $z = x = x_t^{s+1}$, and $d' = \nabla f(x_t^{s+1})$), and taking expectations, we obtain

$$\begin{aligned}\mathbb{E}[P(\bar{x}_{t+1}^{s+1})] \le\ & \mathbb{E}\Bigg[P(x_t^{s+1}) + \langle \bar{x}_{t+1}^{s+1} - x_t^{s+1}, \nabla f(x_t^{s+1}) - \nabla f(x_t^{s+1})\rangle + \left(\frac{L}{2}-\frac{1}{2\eta}\right)\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2 \\ & + \left(\frac{L}{2}+\frac{1}{2\eta}\right)\|x_t^{s+1} - x_t^{s+1}\|^2 - \frac{1}{2\eta}\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2\Bigg] \\ \le\ & \mathbb{E}\left[P(x_t^{s+1}) + \left(\frac{L}{2}-\frac{1}{\eta}\right)\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2\right].\end{aligned} \tag{4.4}$$

In contrast, the actual update in Algorithm 1 is given by

$$x_{t+1}^{s+1} = prox_{\eta h}(x_t^{s+1} - \eta v_t^{s+1}), \tag{4.5}$$

where v_t^{s+1} is defined as a variance reduced stochastic gradient estimator:

$$v_t^{s+1} = \frac{1}{b}\sum_{i_t\in I_t}\left(\nabla f_{i_t}(y_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s).$$

Applying Lemma 4.1 on update (4.5) (with $y = x_{t+1}^{s+1}$, $z = \bar{x}_{t+1}^{s+1}$, $x = x_t^{s+1}$, and $d' = v_t^{s+1}$) and taking expectations, we obtain

$$\begin{aligned}\mathbb{E}[P(x_{t+1}^{s+1})] \le\ & \mathbb{E}\Bigg[P(\bar{x}_{t+1}^{s+1}) + \langle x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}, \nabla f(x_t^{s+1}) - v_t^{s+1}\rangle + \left(\frac{L}{2}-\frac{1}{2\eta}\right)\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 \\ & + \left(\frac{L}{2}+\frac{1}{2\eta}\right)\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2 - \frac{1}{2\eta}\|x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}\|^2\Bigg].\end{aligned} \tag{4.6}$$

By summing inequalities (4.4) and (4.7), we consolidate the bounds on $\mathbb{E}[P(\bar{x}_{t+1}^{s+1})]$ and $\mathbb{E}[P(x_{t+1}^{s+1})]$, yielding

$$\begin{aligned}\mathbb{E}[P(x_{t+1}^{s+1})] \leq \mathbb{E}\bigg[&P(x_t^{s+1}) + \left(L - \frac{1}{2\eta}\right)\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2 + \left(\frac{L}{2} - \frac{1}{2\eta}\right)\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 \\ &- \frac{1}{2\eta}\|x_{t+1}^{s+1} - \bar{x}_t^{s+1}\|^2 + \langle x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}, \nabla f(x_t^{s+1}) - v_t^{s+1}\rangle\bigg]. \qquad (4.7)\end{aligned}$$

In inequality (4.7) we take $Q_1 = \langle x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}, \nabla f(x_t^{s+1}) - v_t^{s+1}\rangle$, and we can bound the term Q_1 as follows:

$$\begin{aligned}\mathbb{E}[Q_1] &\leq \frac{1}{2\eta}\mathbb{E}\left[\|x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}\|^2\right] + \frac{\eta}{2}\mathbb{E}\left[\|\nabla f(x_t^{s+1}) - v_t^{s+1}\|^2\right] \\ &\leq \frac{1}{2\eta}\mathbb{E}\left[\|x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}\|^2\right] + \frac{(1+\tau)\eta L^2}{2b}\mathbb{E}\left[\|y_t^{s+1} - \tilde{x}^s\|^2\right] + \frac{(1+\tau)\eta L^2}{2\tau}\mathbb{E}\left[\|x_t^{s+1} - \tilde{x}^s\|^2\right] \\ &\leq \frac{1}{2\eta}\mathbb{E}\left[\|x_{t+1}^{s+1} - \bar{x}_{t+1}^{s+1}\|^2\right] + \frac{(1+\tau)L}{4b}\mathbb{E}\left[\|y_t^{s+1} - \tilde{x}^s\|^2\right] + \frac{(1+\tau)L}{4\tau}\mathbb{E}\left[\|x_t^{s+1} - \tilde{x}^s\|^2\right], \qquad (4.8)\end{aligned}$$

where the above three inequalities follow from the Cauchy inequality, Lemma 4.2, and Algorithm 1, respectively. From the definition of y_t^{s+1}, we know

$$y_t^{s+1} - \tilde{x}^s = \theta x_t^{s+1} + (1-\theta)\tilde{x}^s - \tilde{x}^s = \theta(x_t^{s+1} - \tilde{x}^s).$$

Taking the expectation of the above equation, we have

$$\mathbb{E}\|y_t^{s+1} - \tilde{x}^s\|^2 = \theta^2\mathbb{E}\|x_t^{s+1} - \tilde{x}^s\|^2. \qquad (4.9)$$

Combine formulas (4.7)–(4.9), we see that

$$\begin{aligned}\mathbb{E}[P(x_{t+1}^{s+1})] \leq \mathbb{E}\bigg[&P(x_t^{s+1}) + \left(L - \frac{1}{2\eta}\right)\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2 + \left(\frac{L}{2} - \frac{1}{2\eta}\right)\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 \\ &+ \left(\frac{(1+\tau)\theta^2 L}{4b} + \frac{(1+\tau)L}{4\tau}\right)\|x_t^{s+1} - \tilde{x}^s\|^2\bigg]. \qquad (4.10)\end{aligned}$$

To further analyze (4.10), we establish a recursion using the following Lyapunov function. Recall that the function R_t^{s+1} is defined by (4.1):

$$R_t^{s+1} = \mathbb{E}\left[P(x_t^{s+1}) + c_t\|x_t^{s+1} - \tilde{x}^s\|^2\right].$$

We can then derive an upper bound for R_{t+1}^{s+1} as follows:

$$\begin{aligned}R_{t+1}^{s+1} &= \mathbb{E}\left[P(x_{t+1}^{s+1}) + c_{t+1}\|x_{t+1}^{s+1} - \tilde{x}^s\|^2\right] \\ &= \mathbb{E}\left[P(x_{t+1}^{s+1}) + c_{t+1}\|x_{t+1}^{s+1} - x_t^{s+1} + x_t^{s+1} - \tilde{x}^s\|^2\right] \\ &\leq \mathbb{E}\left[P(x_{t+1}^{s+1}) + c_{t+1}(1+m)\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 + c_{t+1}(1+\frac{1}{m})\|x_t^{s+1} - \tilde{x}^s\|^2\right] \\ &\leq \mathbb{E}\bigg[P(x_t^{s+1}) + \left(L - \frac{1}{2\eta}\right)\|\bar{x}_{t+1}^{s+1} - x_t^{s+1}\|^2 + \left[c_{t+1}(1+m) + \frac{L}{2} - \frac{1}{2\eta}\right]\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 \qquad (4.11)\end{aligned}$$

$$+\left[c_{t+1}(1+\frac{1}{m})+\frac{(1+\tau)\theta^2 L}{4b}+\frac{(1+\tau)L}{4\tau}\right]\|x_t^{s+1}-\tilde{x}^s\|^2\Bigg]$$
$$\leq R_t^{s+1}+\left(L-\frac{1}{2\eta}\right)\|\bar{x}_{t+1}^{s+1}-x_t^{s+1}\|^2,$$

where the first inequality follows from the Cauchy-Schwarz and Young inequalities, the second inequality is due to the bound (4.10), while the final equality is due to the Lyapunov function R_t^{s+1} and Lemma 4.3. The sequence of values c_t satisfies the following bound:

$$c_{t+1}(1+m)+\frac{L}{2}\leq\frac{1}{2\eta}.$$

Now, adding (4.11) across all the iterations in epoch $s+1$ and the telescoping sums, we get

$$R_m^{s+1}\leq R_0^{s+1}+\sum_{t=0}^{m-1}\left(L-\frac{1}{2\eta}\right)\mathbb{E}\left[\|\bar{x}_{t+1}^{s+1}-x_t^{s+1}\|^2\right], \tag{4.12}$$

where $c_m = 0$ and from the definition of $\tilde{x}^{s+1}$, it follows that $R_m^{s+1} = \mathbb{E}[P(x_m^{s+1})] = \mathbb{E}[P(\tilde{x}^{s+1})]$. Furthermore, $R_0^{s+1} = \mathbb{E}[P(x_0^{s+1})] = \mathbb{E}[P(\tilde{x}^s)]$. This is due to the fact that $x_0^{s+1} = \tilde{x}^s$. Therefore, using the above reasoning in inequality (4.12), we have

$$E\left[P(\tilde{x}^{s+1})\right]\leq E\left[P(\tilde{x}^s)\right]+\sum_{t=0}^{m-1}\left(L-\frac{1}{2\eta}\right)\mathbb{E}\left[\|\bar{x}_{t+1}^{s+1}-x_t^{s+1}\|^2\right],\quad t=0,\ldots,m-1,\ s=0,\ldots,S-1. \tag{4.13}$$

Adding (4.13) across all the epochs and rearranging the terms, we obtain the bound

$$\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\left(\frac{1}{2\eta}-L\right)\mathbb{E}\left[\|\bar{x}_{t+1}^{s+1}-x_t^{s+1}\|^2\right]\leq P(\tilde{x}^0)-\mathbb{E}[P\left(\tilde{x}^S\right)]\leq P(\tilde{x}^0)-P(x^*), \tag{4.14}$$

where the second inequality follows from the optimality of x^*. Recall that the gradient mapping $\mathcal{G}_\eta(x)$ is defined by (2.1):

$$\mathcal{G}_\eta(x_t^{s+1})=\frac{1}{\eta}\left[x_t^{s+1}-prox_{\eta h}\left(x_t^{s+1}-\eta\nabla f(x_t^{s+1})\right)\right].$$

By using this relationship in (4.14), we see that

$$\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\left(\frac{1}{2\eta}-L\right)\eta^2\mathbb{E}\left[\|\mathcal{G}_\eta(x_t^{s+1})\|^2\right]\leq P(\tilde{x}^0)-P(x^*).$$

Now using the definition of x_a from Algorithm 1, we know

$$T\left(\frac{1}{2\eta}-L\right)\eta^2\mathbb{E}\left[\|\mathcal{G}_\eta(x_a)\|^2\right]=\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\left(\frac{1}{2\eta}-L\right)\eta^2\mathbb{E}\left[\|\mathcal{G}_\eta(x_t^{s+1})\|^2\right],$$

and then, we have

$$\mathbb{E}\left[\|\mathcal{G}_\eta(x_a)\|^2\right]\leq\frac{C}{\eta T-2\eta^2 TL}=O\left(\frac{1}{\eta T-2\eta^2 TL}\right),$$

where $C = 2(P(\tilde{x}^0) - P(x^*))$. □

We compare the convergence rate of the proposed algorithm with several commonly used first-order stochastic optimization methods and present their corresponding choices of learning rate and mini-batch size. The detailed comparison is summarized in Table 2. For ProxSTORM [35], σ^2 denotes an upper bound on the variance of the stochastic gradient estimator.

Table 2. Summary of the convergence rates, learning rates, and mini-batch settings.

Algorithm	**Learning Rate**	**Mini-Batch Size**	**Convergence Rate**
ProxSVRG [15]	$\eta = \frac{1}{3L}$	$b = n^{2/3}$	$O\left(\frac{L}{T}\right)$
ProxSAGA [16]	$\eta = \frac{1}{5L}$	$b = n^{2/3}$	$O\left(\frac{L}{T}\right)$
ProxSTORM [35]	$\eta < \frac{1}{L}$	$b \le n$	$O\left(\frac{\sigma^2}{\eta(T+1)}\right)$
MNMPSVRG	$\eta \le \frac{1}{D}$	$b \le n$	$O\left(\frac{1}{\eta T - 2\eta^2 TL}\right)$

For ProxSVRG and ProxSAGA, existing theoretical analyzes restrict step size to $\eta = 1/(3L)$ and $\eta = 1/(5L)$, respectively, leading to constants $1/\eta = 3L$ and $5L$. For the convergence rate $O\left(\frac{1}{\eta T - 2\eta^2 TL}\right)$ of the MNMPSVRG, $\eta T - 2\eta^2 TL$ is a quadratic function on η. It remains positive for $0 < \eta < \frac{1}{2L}$, and achieves the maximum $\frac{T}{8L}$ at $\eta = \frac{1}{4L}$. At this time, the convergence rate of the MNMPSVRG should be $O\left(\frac{L}{T}\right)$.

5. Numerical experiments

All experiments in this paper were conducted on a server equipped with a 12th Gen Intel(R) Core(TM) i5-1240P processor, along with Python 3.10.14 and PyTorch 2.4.0. We evaluate the convergence behavior and characteristics of the proposed MNMPSVRG algorithm on nonconvex and nonsmooth models, on nonconvex matrix factorization problems and time series prediction tasks. Table 3 lists four properties of the different models.

Table 3. Comparison of model properties.

Model / Loss	**Convexity**	**Robustness**	**Sparsity (ℓ_1)**	**Differentiability**
Tanh Loss + ℓ_1	✗	✓	✓	✓
Ramp Loss + ℓ_1	✗	✓	✓	Partial ✗
Matrix Factorization	✗	✗	✓	✗

To ensure experimental consistency and reproducibility, we employed two types of synthetic datasets under controlled settings. The synthetic Gaussian dataset is constructed with high-dimensional features sampled from a standard normal distribution. Binary labels are generated using a linear model with additive Gaussian noise. This dataset is linearly separable with a margin, making it suitable for evaluating optimization algorithms in both convex and nonconvex regimes on a large scale. The Moons dataset, generated using the make-moons function in scikit-learn, is a classic toy dataset for binary classification with nonlinearly separable classes. It consists of two interleaving half circles with added Gaussian noise and is commonly employed to evaluate algorithmic performance under nonconvex decision boundaries.

We also incorporate two real-world binary classification datasets from the UCI repository*. The Adult Income dataset is used to predict whether an individual earns more than \$50K annually. It contains both numerical and categorical features, but only numerical features are used in our experiments. This dataset represents a medium-dimensional classification task.

The Bank Marketing dataset is another real-world dataset related to the direct marketing campaigns of a Portuguese banking institution. It contains mixed-type attributes, with all categorical features one-hot encoded, resulting in a high-dimensional sparse feature space. The objective is to predict whether a client will subscribe to a term deposit.

For matrix factorization tasks, we use the widely adopted MovieLens-100K dataset, which contains 100,000 real user ratings collected from the MovieLens movie recommendation platform. This dataset is commonly used to evaluate collaborative filtering and low-rank matrix approximation techniques.

Furthermore, all datasets were partitioned into 70% for training and 30% for testing, using a fixed random seed of 42 to ensure reproducibility. A comprehensive summary of the dataset characteristics is provided in Table 4. For each set of machine learning experiments, we conducted five independent runs and report the averaged results.

Table 4. Dataset statistics used in experiments.

Dataset	Train Samples	Test Samples	Feature Dimension
Synthetic Gaussian	70,000	30,000	100
Moons	70,000	30,000	2
Adult Income	32,561	13,955	6
Bank Marketing	30,960	13,270	62
MovieLens-100K	70,000	30,000	✗

We compare the proposed algorithm with several common first-order stochastic optimization methods, such as SGD [10], SVRG [12], SGDM [36], Adagrad [30], RMSProp [31], and Adam [32]. The mini-batch size b was chosen to be consistent with the literature used for the comparative algorithms. Specifically, we set $b = \lceil \sqrt{n} \rceil$ in Sections 5.1 and 5.2, $b = 64$ in Section 5.3, and $b = 128$ in Section 5.4.

5.1. *Tanh loss with ℓ_1 regularization*

We consider the nonconvex binary classification problem (referred to as Model 1) with a tanh loss term and a nonsmooth ℓ_1 regularization term. The objective function is formulated as follows:

$$\min_{x\in\mathbb{R}^d} P(x) = \frac{1}{n}\sum_{i=1}^{n}[1 - \tanh(b_i\langle a_i, x\rangle)] + \lambda\|x\|_1,$$

where $a_i \in \mathbb{R}^d$ denotes the feature vector of the i-th sample, and $b_i \in \{-1, +1\}$ is the corresponding binary class label. The model parameter vector is $w \in \mathbb{R}^d$. The term $\ell(z) = 1 - \tanh(z)$ is a nonconvex

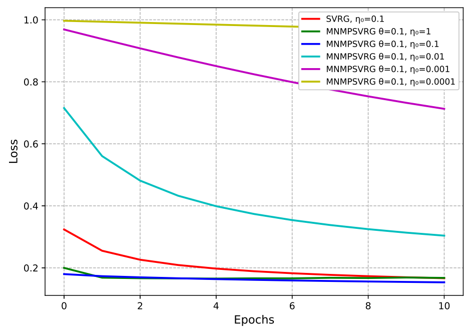
*UCI Machine Learning Repository: `https://archive.ics.uci.edu/`.

yet smooth classification loss function. The regularization parameter $\lambda > 0$ controls the sparsity level through the ℓ_1-norm penalty. The regularization parameter λ is set to 1×10^{-3}, the mini-batch size is $b = \lceil \sqrt{n} \rceil$, and the epoch length is $m = 50$ throughout this paper.

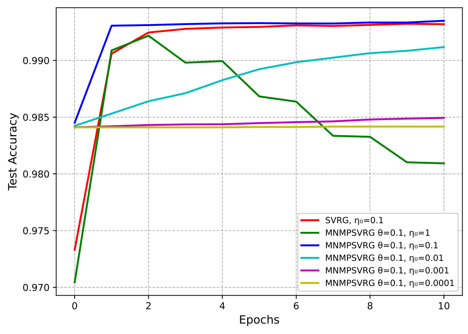
Figure 1 provides a comprehensive evaluation of the performance of the proposed MNMPSVRG method with different initial step sizes η_0 on two synthetic datasets, in comparison with the baseline SVRG algorithm.

Figures 1(a) and 1(b) respectively illustrate the training loss and test accuracy on the Synthetic Gaussian dataset. If $\eta_0 = 0.1$, the MNMPSVRG exhibits the fastest convergence and achieves the lowest final training loss. It also reaches a peak test accuracy close to 0.992 at the early stages of training and maintains high stability throughout. In contrast, although the SVRG with $\eta_0 = 0.1$ also performs reasonably well, its convergence is slower and the final loss is higher. Furthermore, as shown in Figure 1(b), an excessively large step size ($\eta_0 = 1$) or an overly small one ($\eta_0 = 0.0001$) can lead to instability or significantly slower convergence in the MNMPSVRG.

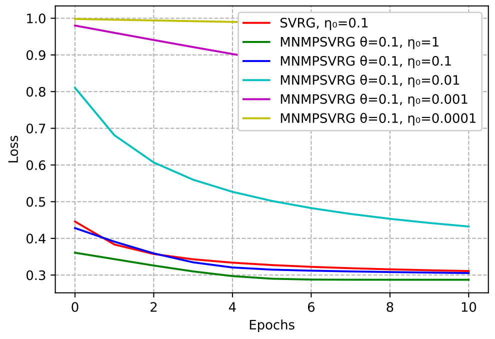
Figures 1(c) and 1(d) present the training loss and test accuracy on the Moons dataset. In Figure 1(c), the configuration with $\eta_0 = 1$ yields the lowest final loss. Notably, if $\eta_0 = 0.1$, the performance of the MNMPSVRG is comparable to that of the SVRG, while extremely small step sizes (e.g., $\eta_0 = 0.0001$) cause training stagnation and consistently high loss. Figure 1(d) further shows that the MNMPSVRG with $\eta_0 = 1$ achieves the highest accuracy (exceeding 0.86). Although its performance is initially inferior to the SVRG, it surpasses the SVRG after approximately four epochs and remains superior thereafter.



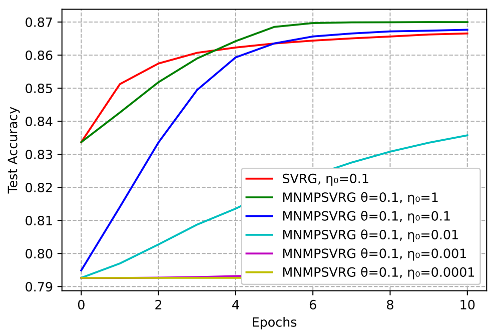
(a) Loss on Synthetic Gaussian



(b) Accuracy on Synthetic Gaussian



(c) Loss on Moons



(d) Accuracy on Moons

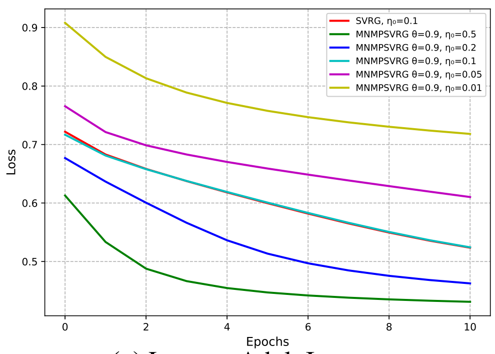
Figure 1. Effect of initial step size η_0 with the MNMPSVRG method on Synthetic Gaussian and Moons.

These results demonstrate that the MNMPSVRG is more robust to step size selection than the SVRG. A moderately large step size, such as $\eta_0 = 0.01$, offers a favorable trade-off between convergence speed and accuracy, contributing to both efficient and stable optimization.

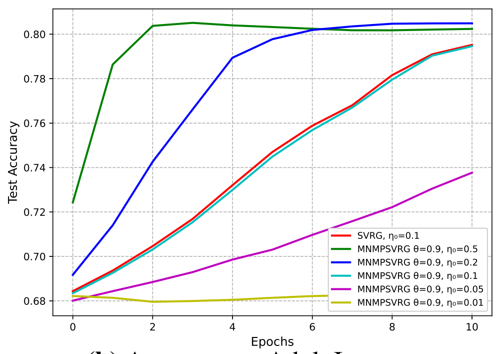
Figure 2 provides a detailed analysis of the performance of the MNMPSVRG method with different initial step sizes η_0 on two real-world classification datasets, in comparison with the baseline SVRG algorithm.

Figures 2(a) and 2(b) present the training loss and test accuracy on the Adult Income dataset. If $\eta_0 = 0.5$, the MNMPSVRG achieves the lowest training loss and the fastest convergence. For $\eta_0 = 0.2$, the performance of the MNMPSVRG is comparable to that of the SVRG. In terms of test accuracy, although the MNMPSVRG with $\eta_0 = 0.5$ initially reaches a high accuracy, it begins to decline around the third epoch. In contrast, the MNMPSVRG with $\eta_0 = 0.2$ achieves the best overall accuracy, reaching approximately 0.80, outperforming the SVRG with the same step size.

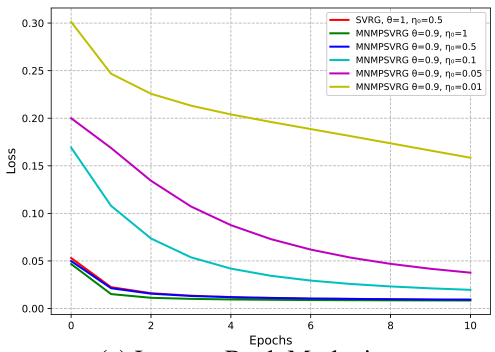
Figures 2(c) and 2(d) show the training loss and test accuracy on the Bank Marketing dataset. The MNMPSVRG with $\eta_0 = 1$ achieves both the lowest training loss and the highest test accuracy. With $\eta_0 = 0.5$, the MNMPSVRG is competitive with the SVRG: although its initial performance is slightly inferior, the accuracy curves of both methods closely align as the number of epochs increases.



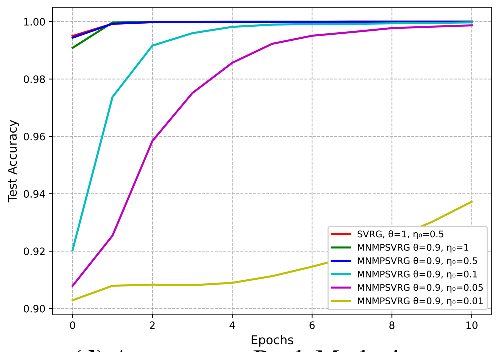
(a) Loss on Adult Income



(b) Accuracy on Adult Income



(c) Loss on Bank Marketing

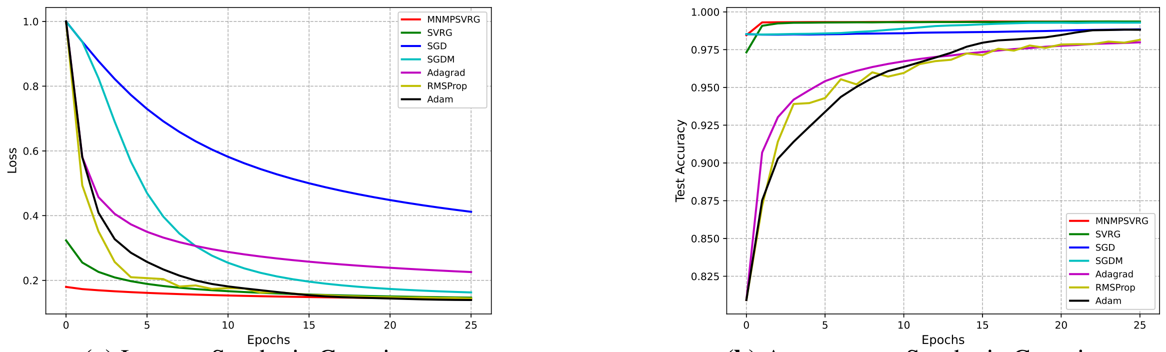


(d) Accuracy on Bank Marketing

Figure 2. Effect of initial step size η_0 with the MNMPSVRG method on Adult Income and Bank Marketing.

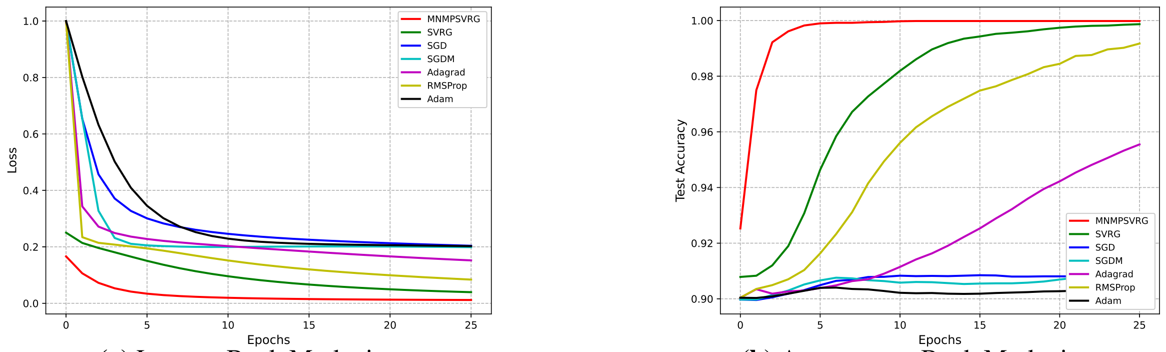
These results further confirm that the MNMPSVRG demonstrates strong robustness under different step size configurations. If the step size is appropriately selected, the MNMPSVRG generally outperforms the SVRG. Specifically, a step size of $\eta_0 = 0.2$ on the Adult Income dataset and $\eta_0 = 0.5$ on the Bank Marketing dataset achieves a favorable balance between convergence speed and generalization performance.

Figures 3 and 4 compare the performance of the MNMPSVRG with other modern optimization methods on Model 1 using two distinct datasets: Synthetic Gaussian (Figure 3) and Bank Marketing (Figure 4). For all compared methods, we adopted their default parameter settings. On the Synthetic Gaussian dataset, we set the parameter $\theta = 0.1$ for the MNMPSVRG and use a unified initial step size of $\eta_0 = 0.1$ for all methods. On the Bank Marketing dataset, we set $\theta = 0.9$ and $\eta_0 = 0.5$ for the MNMPSVRG. For other optimizers, we use a common initial step size of $\eta_0 = 0.1$ for SGD, SGDM, and Adagrad, and $\eta_0 = 0.02$ for RMSProp, Adam, and SVRG.



(a) Loss on Synthetic Gaussian **(b)** Accuracy on Synthetic Gaussian

Figure 3. Comparison of the MNMPSVRG and other modern methods on Model 1 over Synthetic Gaussian.



(a) Loss on Bank Marketing **(b)** Accuracy on Bank Marketing

Figure 4. Comparison of the MNMPSVRG and other modern methods on Model 1 over Bank Marketing.

Figure 3 presents a comparative study between the MNMPSVRG method and several commonly used optimization algorithms on the Synthetic Gaussian dataset using Model 1. As illustrated in Figure 3(a), the MNMPSVRG consistently achieves the fastest convergence and the lowest training loss across all epochs. While the SVRG performs relatively well among the baseline methods, its final loss remains higher than that of the MNMPSVRG. In contrast, methods such as SGD, SGDM, Adagrad, RMSprop, and Adam converge more slowly and settle at significantly higher loss values, with SGD exhibiting the poorest performance in terms of loss reduction. Figure 3(b) further validates the superior generalization capability of the MNMPSVRG. It rapidly reaches near-perfect test accuracy in the early stages of training and maintains this high level throughout the entire process. Although the SVRG eventually achieves a similar final accuracy, its convergence is noticeably slower. Adaptive methods like Adam and RMSprop, as well as classical approaches such as SGD, yield lower final accuracies and require more epochs to stabilize. The MNMPSVRG demonstrates clear advantages in both convergence efficiency and model generalization. These results highlight the effectiveness and robustness of the proposed algorithm in synthetic data optimization tasks.

Figure 4 presents a comprehensive comparison of the MNMPSVRG method with several commonly used optimization algorithms on the Bank Marketing dataset using Model 1. Similar to the results observed in Figure 3, the MNMPSVRG achieves the lowest training loss and the highest test accuracy across all methods.

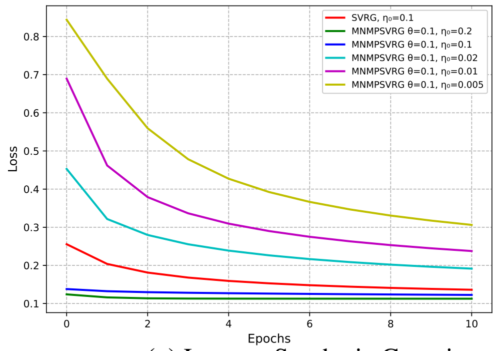
5.2. *Ramp loss with l_1 regularization*

We also consider a nonconvex binary classification problem with ramp loss and an ℓ_1 regularization term, referred to as Model 2. The objective function is defined as follow:

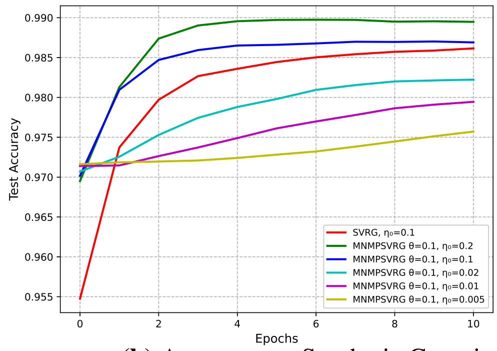
$$\min_{x\in\mathbb{R}^d} P(x) = \frac{1}{n}\sum_{i=1}^{n}\left[1 - \max\left(0, \min\left(1, b_i\langle a_i, x\rangle\right)\right)\right] + \lambda\|x\|_1.$$

This model captures the robustness of the ramp loss against outliers while promoting sparsity through the nonsmooth ℓ_1 regularizer.

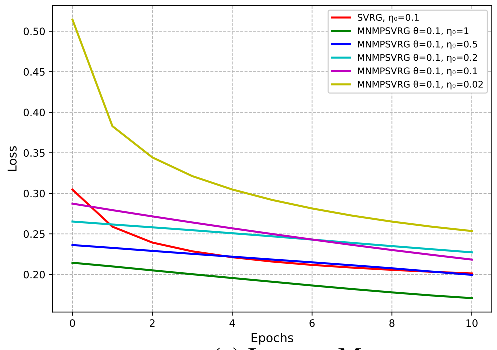
Figure 5 illustrates the impact of different initial step sizes η_0 on the performance of the proposed MNMPSVRG method across two synthetic datasets, with a comparative analysis against the baseline algorithm SVRG. From the figure, it can be observed that the MNMPSVRG demonstrates greater robustness to the choice of step size compared to the SVRG.



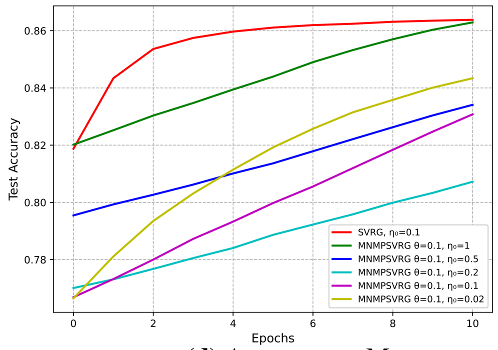
(a) Loss on Synthetic Gaussian



(b) Accuracy on Synthetic Gaussian



(c) Loss on Moons



(d) Accuracy on Moons

Figure 5. Effect of initial step size η_0 with the MNMPSVRG method on Synthetic Gaussian and Moons.

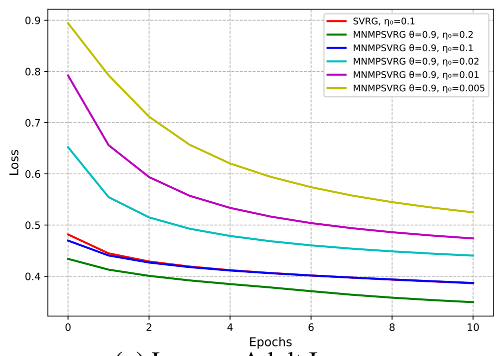
Figures 5(a) and 5(b) present the training loss and test accuracy on the Synthetic Gaussian dataset, respectively. The MNMPSVRG with $\eta_0 = 0.2$ achieves the fastest convergence rate and the lowest final loss. In contrast, although the SVRG with $\eta_0 = 0.1$ performs reasonably well, it converges more slowly and results in a higher final loss, exhibiting inferior performance compared to the MNMPSVRG under the same step size. It is also noteworthy that excessively small step sizes (e.g., $\eta_0 = 0.0001$) cause the MNMPSVRG to become unstable or converge very slowly during training.

Figures 5(c) and 5(d) show the corresponding results on the Moons dataset. The MNMPSVRG with $\eta_0 = 1$ achieves the lowest final loss and the best test accuracy. Notably, although the test accuracy of the MNMPSVRG is slightly lower than that of the SVRG during the initial few iterations, it consistently surpasses the SVRG after four epochs.

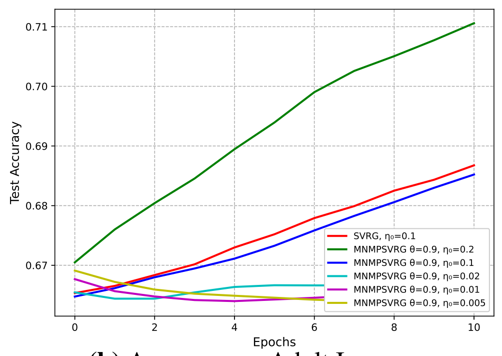
Figure 6 investigates the effect of the initial step size η_0 on the performance of the MNMPSVRG method across two real-world datasets, with a comparative analysis against the baseline SVRG algorithm. The results indicate that the MNMPSVRG exhibits greater robustness to step size selection in practical applications.

Figures 6(a) and 6(b) show the training loss and test accuracy on the Adult Income dataset, respectively. If $\eta_0 = 0.2$, the MNMPSVRG achieves the best convergence behavior, reaching the lowest final loss while maintaining stable accuracy throughout the training process.

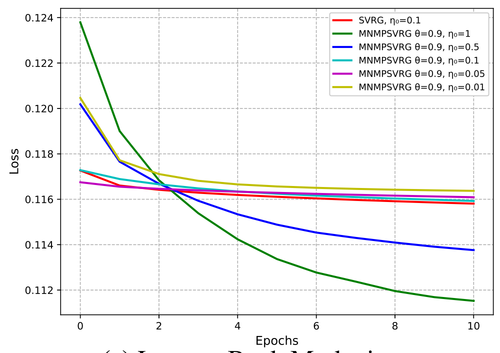
Figures 6(c) and 6(d) present the corresponding metrics on the Bank Marketing dataset. Under the configuration $\eta_0 = 0.5$, the MNMPSVRG demonstrates optimal performance by attaining the lowest final loss as shown in Figure 6(c).



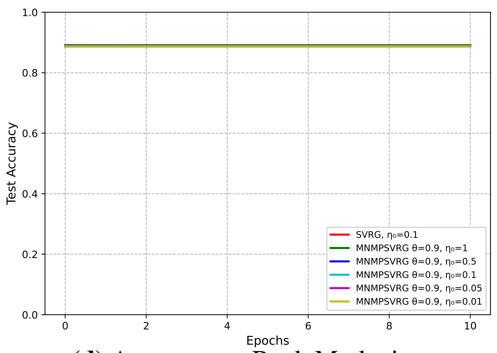
(a) Loss on Adult Income



(b) Accuracy on Adult Income



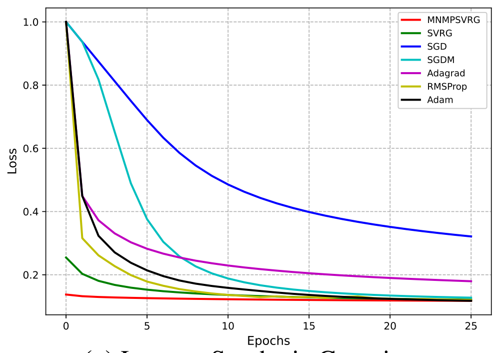
(c) Loss on Bank Marketing



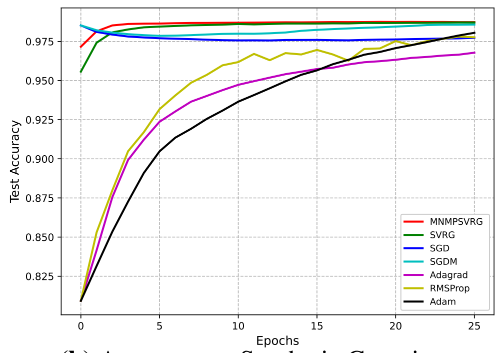
(d) Accuracy on Bank Marketing

Figure 6. Effect of initial step size η_0 with the MNMPSVRG method on Adult Income and Bank Marketing.

Figures 7 and 8 present a comprehensive comparison between the proposed MNMPSVRG method and other modern methods on Model 2. For all compared methods, we adopt their default hyperparameter settings unless otherwise specified. On Synthetic Gaussian, we set the parameter θ of the MNMPSVRG method to 0.1, and use a unified initial step size $\eta = 0.1$ for SGD, Adagrad, SVRG, and MNMPSVRG, while the remaining methods use $\eta = 0.02$. On Bank Marketing, we set $\theta = 0.9$ and $\eta = 0.2$ for the MNMPSVRG, and use $\eta = 0.1$ for SGD, RMSProp, Adam, and SVRG, whereas the other methods are assigned $\eta = 0.5$.

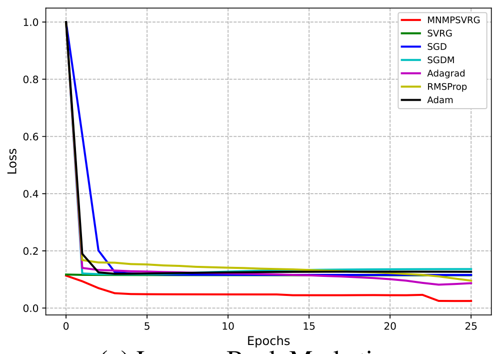


(a) Loss on Synthetic Gaussian

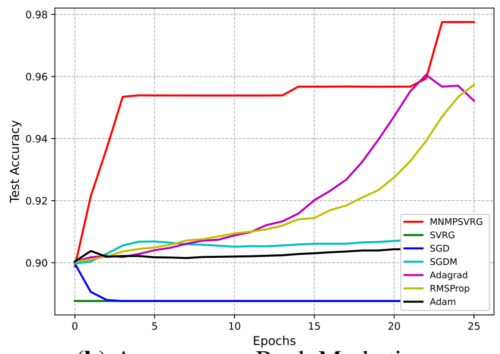


(b) Accuracy on Synthetic Gaussian

Figure 7. Comparison of the MNMPSVRG and other modern methods on Model 2 over Synthetic Gaussian.



(a) Loss on Bank Marketing



(b) Accuracy on Bank Marketing

Figure 8. Comparison of the MNMPSVRG and other modern methods on Model 2 over Bank Marketing.

Figure 7 compares the proposed the MNMPSVRG method with several popular first-order stochastic optimization algorithms on the Synthetic Gaussian dataset under the Model 2 architecture. Figure 7(a) shows that the MNMPSVRG achieves the fastest convergence rate and consistently maintains the lowest training loss throughout all training epochs. While the SVRG remains competitive among the baseline methods, it still results in a higher final loss compared to the MNMPSVRG. In contrast, other methods exhibit significantly slower convergence behavior, with SGD performing the worst in terms of loss minimization. Figure 7(b) confirms the superior generalization capability of the MNMPSVRG. The method achieves near-optimal test accuracy early in the training process and maintains this high level of accuracy consistently. Adaptive methods such as Adam and RMSprop stabilize at noticeably lower accuracy levels.

Figure 8 presents a comprehensive comparison of the MNMPSVRG method with several commonly used optimization algorithms on the Bank Marketing dataset using Model 2. Similar to the results observed in Figure 7, the MNMPSVRG achieves the lowest training loss and the highest test accuracy across all methods.

Table 5 summarizes the detailed results of the MNMPSVRG algorithm on four datasets across two machine learning problems, corresponding to Figures 1, 2, 5, and 6. Table 6 provides a comprehensive comparison between the MNMPSVRG and other baseline algorithms.

Table 5. Loss and accuracy of the MNMPSVRG on four datasets under two problems.

Dataset	**Model 1**			**Model 2**		
	Step size	Loss	Accuracy (%)	Step size	Loss	Accuracy (%)
Synthetic Gaussian ($\theta = 0.1$)	1	0.167628	98.09	0.2	**0.112667**	**98.95**
	0.1	**0.153333**	**99.35**	0.1	0.122569	98.69
	0.01	0.303882	99.12	0.02	0.191507	98.22
	0.001	0.712901	98.49	0.01	0.237625	97.94
	0.0001	0.965833	98.42	0.005	0.306147	97.57
Moons ($\theta = 0.1$)	1	**0.287124**	**87.00**	1	**0.170793**	**86.29**
	0.1	0.305394	86.77	0.5	0.199628	83.41
	0.01	0.432329	83.57	0.2	0.227280	80.72
	0.001	0.801979	79.52	0.1	0.218327	83.08
	0.0001	0.978120	79.26	0.02	0.253557	84.34
Adult Income ($\theta = 0.9$)	0.5	**0.431141**	80.24	0.2	**0.349408**	**71.06**
	0.2	0.462631	**80.49**	0.1	0.386844	68.52
	0.1	0.524256	79.46	0.02	0.440337	66.67
	0.05	0.610234	73.77	0.01	0.473854	66.57
	0.01	0.718122	68.51	0.005	0.524831	66.40
Bank Marketing ($\theta = 0.9$)	1	**0.008464**	**100.00**	1	**0.111526**	**89.10**
	0.5	0.009293	**100.00**	0.5	0.113760	88.93
	0.1	0.019620	99.98	0.1	0.115925	88.77
	0.05	0.037636	99.87	0.05	0.116090	88.77
	0.01	0.158476	93.73	0.01	0.116371	88.77

Table 6. Loss and accuracy of seven algorithms on two datasets under two models.

Dataset	**Algorithm**	**Model 1**			**Model 2**		
		Step size	Loss	Accuracy (%)	Step size	Loss	Accuracy (%)
Synthetic Gaussian	MNMPSVRG ($\theta = 0.1$)	0.1	0.142895	**99.36**	0.1	**0.117490**	**98.74**
	SVRG	0.1	0.146622	**99.36**	0.1	0.121383	98.71
	SGD	0.1	0.411862	98.84	0.1	0.321237	97.73
	SGDM	0.1	0.162720	99.29	0.02	0.127560	98.58
	Adagrad	0.1	0.225687	97.99	0.1	0.179556	96.78
	RMSProp	0.1	0.143969	98.15	0.02	0.119960	97.77
	Adam	0.1	**0.139270**	98.82	0.02	0.117598	98.05
Bank Marketing	MNMPSVRG ($\theta = 0.9$)	0.5	**0.011710**	**99.98**	0.5	**0.024881**	**97.75**
	SVRG	0.02	0.039367	99.87	0.1	0.114618	88.77
	SGD	0.1	0.203824	90.94	0.1	0.115848	88.77
	SGDM	0.1	0.198930	90.96	0.5	0.136362	90.78
	Adagrad	0.1	0.152026	95.55	0.5	0.087034	95.22
	RMSProp	0.02	0.083980	99.17	0.1	0.095919	95.74
	Adam	0.02	0.202328	90.39	0.1	0.126820	90.55

5.3. *Matrix factorization*

We evaluate the performance of the proposed optimization algorithms on a nonconvex matrix factorization (MF) task using the MovieLens-100K dataset. The goal is to learn low-dimensional user and item embeddings $\mathbf{P} \in \mathbb{R}^{N\times d}$ and $\mathbf{Q} \in \mathbb{R}^{M\times d}$ by minimizing the following regularized squared error objective:

$$\min_{\mathbf{P},\mathbf{Q}} \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} (\langle \mathbf{P}_u, \mathbf{Q}_i \rangle - y_{ui})^2 + \frac{\lambda}{2}\left(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2\right),$$

where Ω is the set of observed (user, item) pairs and y_{ui} is the corresponding rating.

We compare the performance of the proposed MNMPSVRG optimizer (with $\theta = 0.1$ and without proximal thresholding) against standard PyTorch optimizers, including SGD, SGDM, Adagrad, RMSProp, and Adam. All models were configured with the same latent dimension $d = 16$, initial step size $\eta_0 = 0.02$, regularization parameter $\lambda = 10^{-3}$, and a batch size of 64. Each method was trained for 100 epochs, with 70% of the data used for training and the remaining 30% for testing.

In this experiment, the regularizer is the squared Frobenius norm, which is smooth and differentiable. Consequently, the corresponding proximal operator reduces to an identity mapping, and the MNMPSVRG update coincides with a standard gradient-based step. Therefore, explicit proximal thresholding is not required in this setting.

Figure 9 compares the root mean squared error (RMSE) performance of various optimization methods for matrix factorization on the MovieLens-100K dataset. The proposed MNMPSVRG method demonstrates faster convergence and superior generalization, achieving the lowest test RMSE among all methods. In contrast, while standard optimizers such as SGDM and Adagrad exhibit rapid initial descent, their test errors plateau at relatively higher levels. Adaptive methods like RMSProp and Adam show stable training but do not outperform the MNMPSVRG in final test performance. These results highlight the effectiveness of incorporating mini-batch negative momentum and variance reduction in addressing nonconvex matrix factorization problems. Table 7 presents the final training and test RMSE of each algorithm on the MovieLens-100K matrix factorization task, where the MNMPSVRG achieves the lowest values (training: 0.8353, test: 0.9707), consistent with the convergence trend illustrated in Figure 9.

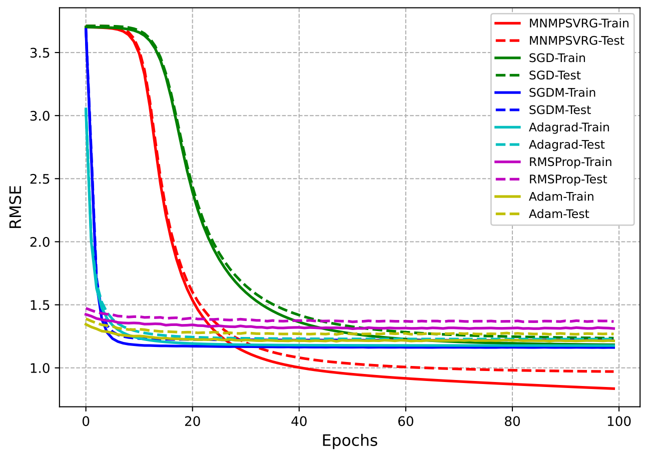


Figure 9. RMSE comparison of the MNMPSVRG and modern methods on MovieLens-100K.

Table 7. Final train and test RMSE of different algorithms on MovieLens-100K.

Algorithm	Train RMSE	Test RMSE
MNMPSVRG	**0.8353**	**0.9707**
SGD	1.1831	1.2359
SGDM	1.1613	1.2183
Adagrad	1.1765	1.2278
RMSProp	1.3120	1.3670
Adam	1.2145	1.2684

5.4. Prediction of time series tasks

In the short-term wind power forecasting task, we employed a gated recurrent unit (GRU) model to evaluate the proposed MNMPSVRG algorithm with several defferent θ using the *Wind* dataset, which recorded turbine power generation and related meteorological variables at 10-minute intervals from January 1 to December 31, 2018. All features were normalized, and the training samples were reshaped into the three-dimensional tensor format $(N, 1, d)$ required by recurrent neural networks (RNNs). Supervised learning samples were constructed via a sliding-window approach with input length $n_{in} = 1$ and prediction horizon $n_{out} = 1$. A single-layer GRU with hidden dimension $h = 128$ was adopted. In the experiments, we set the batch size $b = 128$, learning rate $\eta = 0.001$, and the number of inner iterations m equal to the number of batches in the training set.

For several fixed values of theta, the MNMPSVRG algorithm consistently reduced the training mean absolute error (MAE), indicating that the implemented variance reduction strategy effectively stabilizes the error decrease in this GRU-based time series forecasting task.

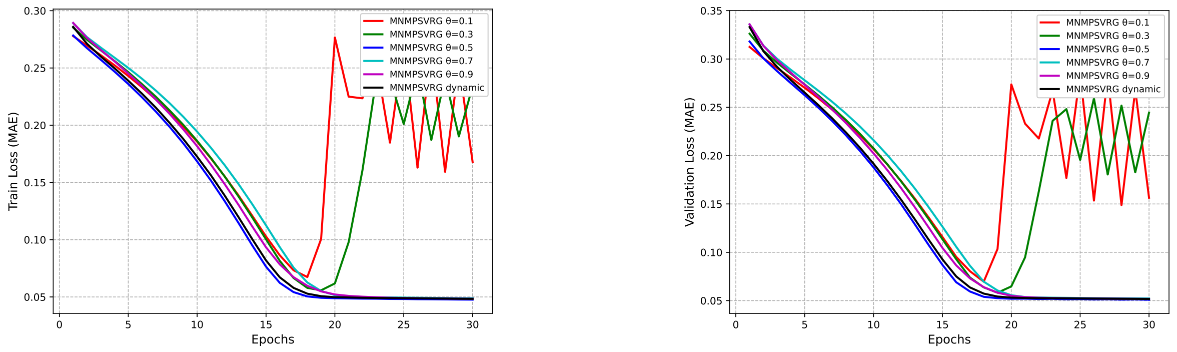
Inspired by the backtracking stepsize rule in the FISTA [19], we introduce a new dynamic adjustment strategy on the negative momentum. For the convex combination $y_t^{s+1} = \theta x_t^{s+1} + (1 - \theta)\tilde{x}^s$ in Algorithm 1, the negative momentum θ is replaced by dynamic θ_t as follows:

$$\theta_t = \begin{cases} \dfrac{t-2}{t+1}, & t > 2, \\ 0, & t \leq 2. \end{cases}$$

This design ensures that θ_t takes a small value in the initial iterations, thereby assigning larger weight to the snapshot $\tilde{x}^s$ in the combination, which enhances the variance reduction effect and stabilizes the optimization path. As the iteration proceeds ($t > 2$), $\theta_t = \frac{t-2}{t+1}$ increases monotonically and gradually approaches 1, and larger weight is assigned to the current iterate x_t^{s+1}. This dynamic adjustment mechanism not only maintains path stability in the early stage but also fully exploits the latest iterate information in the later phase.

The numerical results of the fixed parameters θ (0.1, 0.3, 0.5, 0.7, 0.9) and the dynamic θ_t will be presented separately below. In the experiments, Algorithm 1 with θ_t is denoted as the MNMPSVRG dynamic. The corresonding numerical results are shown in Figures 10 and 11, and Tables 8 and 9.

Figure 10 illustrates the MAE performance of the MNMPSVRG with fixed θ values and the dynamic θ_t variant on both the training and validation sets, while Table 8 reports the detailed MAE and RMSE results. It can be observed that excessively small θ values lead to unstable performance in the later iterations. Among the fixed settings, $\theta = 0.5$ achieves the best overall results, and the dynamic θ_t variant also demonstrates competitive performance.



(**a**) Train loss (**b**) Validation loss

Figure 10. Train loss and validation loss on the Wind dataset.

Table 8. MAE and RMSE of the MNMPSVRG with different θ on training loss and validation loss.

Algorithm	**Training Loss**		**Validation Loss**	
	MAE	**RMSE**	**MAE**	**RMSE**
MNMPSVRG $\theta = 0.1$	0.167824	0.191321	0.156517	0.179369
MNMPSVRG $\theta = 0.3$	0.234277	0.247918	0.244346	0.259946
MNMPSVRG $\theta = 0.5$	**0.047670**	**0.087277**	**0.050800**	**0.080830**
MNMPSVRG $\theta = 0.7$	0.048892	0.090051	0.052144	0.082942
MNMPSVRG $\theta = 0.9$	0.048399	0.088824	0.051700	0.082048
MNMPSVRG dynamic	0.048367	0.089146	0.051606	0.082267

Figure 11 presents the prediction performance of different MNMPSVRG variants under various θ settings for the next 500 hours, while Table 9 reports the corresponding MAE and RMSE between the final predictions and the true values. It can be observed that the dynamic θ_t version of the MNMPSVRG achieves competitive performance compared with the fixed θ counterparts.

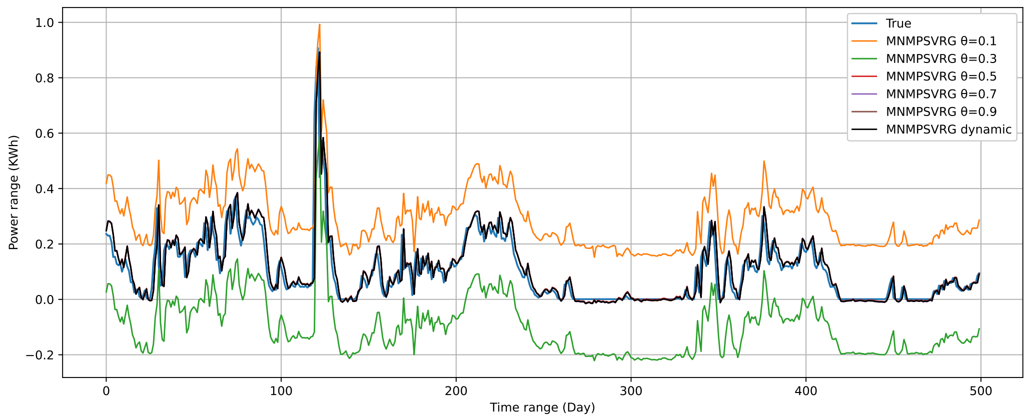


Figure 11. Prediction performance of different MNMPSVRG variants on the Wind dataset.

Table 9. MAE and RMSE of the MNMPSVRG with different θ.

Algorithm	**MAE**	**RMSE**
MNMPSVRG $\theta = 0.1$	0.193093	0.198024
MNMPSVRG $\theta = 0.3$	0.198858	0.204230
MNMPSVRG $\theta = 0.5$	0.028033	0.052458
MNMPSVRG $\theta = 0.7$	0.028247	0.052519
MNMPSVRG $\theta = 0.9$	0.028040	**0.052273**
MNMPSVRG dynamic	**0.027942**	0.052304

6. Conclusions

This paper proposed a first-order accelerated mini-batch stochastic algorithm for addressing nonconvex and nonsmooth optimization problems. Built upon the ProxSVRG framework, the proposed method integrated mini-batching and a negative momentum technique to enhance performance and accelerate convergence. Finally, the experimental results validated our theoretical findings and demonstrated that the proposed algorithm exhibited superior convergence and stability compared with existing methods for nonconvex and nonsmooth optimization problems. In future work, we will focus on extending the theoretical analysis, such as investigating the convergence theory for problems with the regularization term $h(x)$ nonconvex, designing adaptive parameter tuning strategies, and establishing stronger convergence rate guarantees.

Author contributions

Weihao Cui: conceptualization, writing–original draft, methodology, experimental implementation, writing–review and editing; Chongyang He: writing–original draft, data analysis, result verification; Mingyuan Cao: writing–original draft, supervision, funding acquisition, writing–review and editing; Yueting Yang: methodology, supervision, funding acquisition, project administration, writing–review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors would like to thank the General Program of Science and Technology Department of Jilin Province (20260102232JC), the Youth Science and Technology Talent Cultivation Project of Jilin Province (20250602005RC), and the Graduate Innovation Project of Beihua University (2024005).

Conflict of interest

The authors declare no conflicts of interest.

References

1. R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, *Proceedings of the 25th International Conference on Machine Learning*, 2008, 160–167. https://doi.org/10.1145/1390156.1390177
2. R. F. Conchas, A. G. Loukianov, E. N. Sanchez, A. Y. Alanis, Finite time convergent recurrent neural network for variational inequality problems subject to equality constraints, *J. Franklin I.*, **361** (2024), 583–597. https://doi.org/10.1016/j.jfranklin.2023.11.041
3. P. Guo, Z. Ye, K. Xiao, W. Zhu, Weighted aggregating stochastic gradient descent for parallel deep learning, *IEEE Trans. Knowl. Data Eng.*, **34** (2022), 5037–5050. https://doi.org/10.1109/tkde.2020.3047894
4. H. Kasai, Stochastic variance reduced multiplicative update for nonnegative matrix factorization, *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, 6338–6342. https://doi.org/10.1109/ICASSP.2018.8461325
5. A. E. Hoerl, R. W. Kennard, Ridge regression: biased estimation for nonorthogonal problems, *Technometrics*, **12** (1970), 55–67. https://doi.org/10.1080/00401706.1970.10488634
6. R. Tibshirani, Regression shrinkage and selection via the lasso, *J. R. Stat. Soc. B*, **58** (1996), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x
7. J. Yin, C. Tang, J. Jian, Q. Huang, A partial Bregman ADMM with a general relaxation factor for structured nonconvex and nonsmooth optimization, *J. Glob. Optim.*, **89** (2024), 899–926. https://doi.org/10.1007/s10898-024-01384-2
8. Y. Yang, H. Lan, L. Jiang, Novel inertial stochastic Bregman inexact ADMMs for solving large-scale nonconvex and nonsmooth optimization without relying on the Kurdyka-Łojasiewicz property, *AIMS Mathematics*, **10** (2025), 24804–24835. https://doi.org/10.3934/math.20251099
9. H. Mine, M. Fukushima, A minimization method for the sum of a convex function and a continuously differentiable function, *J. Optim. Theory Appl.*, **33** (1981), 9–23. https://doi.org/10.1007/bf00935173
10. S. Ghadimi, G. Lan, H. Zhang, Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization, *Math. Program.*, **155** (2016), 267–305. https://doi.org/10.1007/s10107-014-0846-1
11. H. Robbins, S. Monro, A stochastic approximation method, *Ann. Math. Stat.*, **22** (1951), 400–407. https://doi.org/10.1214/aoms/1177729586
12. R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2013, 315–323.
13. A. Defazio, F. Bach, S. Lacoste-Julien, SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives, *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2014, 1646–1654.
14. L. M. Nguyen, J. Liu, K. Scheinberg, M. Takáč, SARAH: a novel method for machine learning problems using stochastic recursive gradient, *Proceedings of the 34th International Conference on Machine Learning*, 2017, 2613–2621.

15. L. Xiao, T. Zhang, A proximal stochastic gradient method with progressive variance reduction, *SIAM J. Optimiz.*, **24** (2014), 2057–2075. https://doi.org/10.1137/140961791

16. F. Pedregosa, R. Leblond, S. Lacoste-Julien, Breaking the nonsmooth barrier: a scalable parallel method for composite optimization, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, 55–64.

17. N. Pham, L. Nguyen, D. Phan, Q. Tran-Dinh, ProxSARAH: an efficient algorithmic framework for stochastic composite nonconvex optimization, *J. Mach. Learn. Res.*, **21** (2020), 1–48.

18. Y. Nesterov, A method for solving the convex programming problem with convergence rate O (1/k2), *Proceedings of the USSR Academy of Sciences*, **269** (1983), 543–547.

19. A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM J. Imaging Sci.*, **2** (2009), 183–202. https://doi.org/10.1137/080716542

20. B. Wen, X. Chen, T. K. Pong, Linear convergence of proximal gradient algorithm with extrapolation for a class of nonconvex nonsmooth minimization problems, *SIAM J. Optimiz.*, **27** (2017), 124–145. https://doi.org/10.1137/16m1055323

21. Z. Wu, M. Li, General inertial proximal gradient method for a class of nonconvex nonsmooth optimization problems, *Comput. Optim. Appl.*, **73** (2019), 129–158. https://doi.org/10.1007/s10589-019-00073-1

22. L. He, J. Ye, J. E, Nonconvex optimization with inertial proximal stochastic variance reduction gradient, *Inform. Sciences*, **648** (2023), 119546. https://doi.org/10.1016/j.ins.2023.119546

23. Z. Allen-Zhu, L. Orecchia, Linear coupling: an ultimate unification of gradient and mirror descent, *Proceedings of 8th Innovations in Theoretical Computer Science Conference*, 2017, 3:1–3:22. https://doi.org/10.4230/LIPIcs.ITCS.2017.3

24. S. Bubeck, Convex optimization: algorithms and complexity, *Found. Trends Mach. Le.*, **8** (2015), 231–357. https://doi.org/10.1561/2200000050

25. W. Su, S. Boyd, E. J. Candes, A differential equation for modeling Nesterov's accelerated gradient method: theory and insights, *J. Mach. Learn. Res.*, **17** (2016), 5312–5354.

26. A. Nitanda, Stochastic proximal gradient descent with acceleration techniques, *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2014, 1574–1582.

27. Z. Yang, SARAH-M: a fast stochastic recursive gradient descent algorithm via momentum, *Expert Syst. Appl.*, **238** (2024), 122295. https://doi.org/10.1016/j.eswa.2024.122295

28. Z. Allen-Zhu, Katyusha: the first direct acceleration of stochastic gradient methods, *J. Mach. Learn. Res.*, **18** (2018), 1–51.

29. L. He, J. Ye, J. E, Accelerated stochastic variance reduction for a class of convex optimization problems, *J. Optim. Theory Appl.*, **196** (2023), 810–828. https://doi.org/10.1007/s10957-022-02157-1

30. J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.*, **12** (2011), 2121–2159.

31. T. Tieleman, Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude, *COURSERA: Neural Networks for Machine Learning*, **4** (2012), 26–31.

32. D. P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv: 1412.6980. https://doi.org/10.48550/arXiv.1412.6980

33. A. Beck, *Introduction to nonlinear optimization: theory, algorithms, and applications with MATLAB*, Philadelphia: Society for Industrial and Applied Mathematics, 2014. https://doi.org/10.1137/1.9781611973655

34. S. J. Reddi, S. Sra, B. Poczos, A. J. Smola, Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization, *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, 2016, 1145–1153.

35. Z. Wang, B. Wen, Proximal stochastic recursive momentum algorithm for nonsmooth nonconvex optimization problems, *Optimization*, **73** (2024), 481–495. https://doi.org/10.1080/02331934.2022.2112191

36. B. T. Polyak, Some methods of speeding up the convergence of iteration methods, *USSR Comput. Math. Math. Phys.*, **4** (1964), 1–17. https://doi.org/10.1016/0041-5553(64)90137-5