Reproduction of the Improved Powered Stochastic Optimization Algorithm for Large-Scale Machine Learning

Li Jianning[*]

June 15, 2025

Abstract

This paper addresses large-scale machine learning optimization problems, specifically focusing on the logistic function with a non-convex regularizer, where traditional methods often encounter limitations in efficiency and scalability. We reproduce the Improved Powered Stochastic Optimization Algorithm, which integrates the Powerball Stochastic Optimization (PSO) and Stochastic Variance Reduced Gradient (SVRG) methods to develop the PB-SVRGE algorithm. Theoretical analysis demonstrates that PB-SVRGE achieves a faster convergence rate compared to classical PSO-based algorithms. We conduct experiments on the original datasets to investigate the effects of key parameters, including the power coefficient, mini-batch size, and learning rate. The results confirm the effectiveness and robustness of PB-SVRGE, indicating that appropriate parameter selection can further enhance its performance.

[*] Li Jianning, 2401210081, School of Mathematical Sciences

Contents

1 Introduction

The original paper[8] focuses on the improved powered stochastic optimization algorithm, which is designed to address challenges in large-scale machine learning tasks, where traditional optimization methods often struggle with efficiency and scalability. By leveraging stochastic techniques, the algorithm aims to improve convergence rates and handle high-dimensional data effectively.

1.1 Introduction to Optimization Problems

In this project, we mainly focus on optimization problems as follows:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\Sigma_{i=1}^{n} f_i(\mathbf{w}), \tag{1}$$

where n denotes the total number of the instances, w defines the parameter to learn and $f_i : \mathbb{R}^d \to \mathbb{R}$ is a loss corresponding to the i-th instances with d dimnesions. We assume each f_i is a L-smooth function but may be non-convex.

1.2 Notation

In this project, we use the following notations:

- $\mathbb{R}^d$: the set of d-dimensional real number vectors.
- $w \cdot v$: the inner product of two vectors w and v.
- $\|\cdot\|$ and $\|\cdot\|_p$: the Euclidean norm and p-norm of a vector in $\mathbb{R}^d$, respectively.
- $[n]$: the set $\{1, 2, \ldots, n\}$.
- $\nabla f(x)$: the gradient of the function f at the point x.
- $\mathbb{E}[\cdot]$: the expectation operator with respect to the underlying probability space.
- $(\mathbf{x})_i$: the i-th component of the vector $\mathbf{x}$.
- η_t: the learning rate at iteration t.

1.3 Some Classic Optimization Algorithms

Now we introduce some classic and basic optimization algorithms that are well-known and strongly related to the improved powered stochastic optimization algorithm.

Stochastic Gradient Descent (SGD) The update rule for the Stochastic Gradient Descent (SGD) algorithm of iteration t is given by:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_{i_t}(\mathbf{w}_t), \tag{2}$$

where i_t is a randomly chosen index from $[n]$.

Powered Stochastic Optimization (PSO) The Powered Stochastic Optimization (PSO) algorithm is a variant of SGD that incorporates a power term to enhance convergence.[10] The update rule for PSO at iteration t is given by:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \sigma_\gamma(\nabla f_{i_t}(\mathbf{w}_t)), \tag{3}$$

where i_t is a randomly chosen index from $[n]$, the power coefficient $\gamma \in [0, 1)$, and the powerball function σ_γ is defined as

$$\sigma_\gamma(x) = \text{sign}(x)|x|^\gamma \tag{4}$$

where $\text{sign}(x)$ is the sign function defined as

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x = 0, \\ -1, & \text{if } x < 0. \end{cases}$$

When x is a vector, σ_r and the sign function is applied element-wise.

Stochastic Variance Reduced Gradient (SVRG) SVRG is a popular variance-reduced stochastic optimization methods.[4] SVRG maintains a snapshot of the full gradient at each iteration and uses it to reduce the variance of the stochastic gradient. The update rule for SVRG at iteration t is given by:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\nabla f_{i_t}(\mathbf{w}_t) - \nabla f_{i_t}(\tilde{\mathbf{w}}) + \nabla f(\tilde{\mathbf{w}}) \right), \tag{5}$$

where i_t is a randomly chosen index from $[n]$, $\tilde{\mathbf{w}}$ is a snapshot point and $\nabla f(\tilde{\mathbf{w}})$ is the full gradient at that point. The SVRG algorithm typically consists of two loops: an inner loop where the update rule (5) is applied, and an outer loop that stores historical information and updates the snapshot point $\tilde{\mathbf{w}}$, which is a function of $\mathbf{w}_t$s from last inner iteration.

2 The original Algorithm from the Paper

2.1 Algorithm and Convergence Analysis

The original paper[8] proposes a novel approach that integrates the Powered Stochastic Optimization (PSO) algorithm with the Stochastic Variance Reduced Gradient (SVRG) algorithm to enhance the convergence rate of the optimization process. The resulting algorithm, termed Powerball Stochastic Variance Reduction with Gradient Enhancement (PB-SVRGE)[1], is specifically designed to address large-scale machine learning tasks with improved efficiency and scalability.

[1] The term SVRGE, likely referring to Stochastic Variance Reduced Gradient with Enhancement, is not explicitly defined in the original paper or related literature. We interpret it as incorporating techniques such as the powerball function from PSO, where the suffix ”Enhancement” signifies the modifications introduced.

```
Input: $\tilde{\mathbf{w}}^0 = \mathbf{w}_K^0 = \mathbf{w}^0$, inner loop size $K$, outer loop size $S = \frac{T}{K}$, mini-batch size $b$, learning
       rate $\{\eta_j\}_{j=0}^{K-1}$, power coefficient $\gamma$
Output: $\tilde{\mathbf{w}}_S$
for $s = 0, \ldots, S-1$ do
    $\mathbf{w}_0^{s+1} = \tilde{\mathbf{w}}^s = \mathbf{w}_K^s$;
    $\mathbf{g}^{s+1} = \nabla f(\tilde{\mathbf{w}}^s)$;
    for $k = 0, \ldots, K-1$ do
        Randomly choose $\mathcal{S} \subset [n]$ with $|\mathcal{S}| = b$;
        $\mathbf{v}_k^{s+1} = \nabla f_{\mathcal{S}}(\mathbf{w}_k^{s+1}) - \nabla f_{\mathcal{S}}(\tilde{\mathbf{w}}^s) + \mathbf{g}^{s+1}$;
        $\mathbf{w}_{k+1}^{s+1} = \mathbf{w}_k^{s+1} - \eta_k \sigma(\mathbf{v}_k^{s+1})$;
    end
end
Set $\tilde{\mathbf{w}}^S = \mathbf{w}_K^S$;
```

Algorithm 1: PB-SVRGE Algorithm

Algorithm 1 employs a mini-batch strategy, where the gradient $\nabla f_{\mathcal{S}}(\mathbf{w})$ is computed as $\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla f_i(\mathbf{w})$. The convergence properties of the algorithm have been rigorously established in the original paper under the assumptions outlined in Assumptions 1 and 2. The theoretical guarantee is formalized in Theorem 3, which provides a detailed analysis of the convergence rate and its dependence on algorithmic parameters.

Assumption 1. *Each of the f_i is differential and L-smooth, futhermore, $f(x)$ is differential and L-smooth.*

Assumption 2. *The stochastic gradient oracle is an independent and unbiased estimator of the gradient and satisfies*

$$\mathbb{E}_{\xi_i}\left[\nabla f\left(w, \xi_i\right)\right] = \nabla f(w), \quad \forall w \in \mathbb{R}^d$$
$$\mathbb{E}_{\xi_i}\left[\left\|\nabla f\left(w, \xi_i\right) - \nabla f(w)\right\|^2\right] \leq \hat{\sigma}^2, \quad \forall w \in \mathbb{R}^d$$

where $\nabla f_i(w) = \nabla f\left(w, \xi_i\right)$ and ξ_i denotes a random variable.

Theorem 3. *Set $w^* = \arg\min_w f(w)$, and choose $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = b$. Let T denote the number of total iterations, then, under Assumption 1 and Assumption 2, for any $T \geq 1$, PB-SVRGE (Algorithm 1) can lead to*

$$\mathbb{E}\left[\frac{1}{T}\sum_{s=0}^{S-1}\sum_{k=0}^{K-1} \|\nabla f_{\mathcal{S}}(w_k^{s+1})\|_{1+\gamma}^2\right] \leq \frac{4L\|\mathbf{1}\|_p}{T(1-\theta)}\mathbb{E}[f(\tilde{w}^0) - f(w^*)] + \frac{8\|\mathbf{1}\|_p\hat{\sigma}^2}{b\theta(1-\theta)},$$

where $p = \frac{1+\gamma}{1-\gamma}$, $\theta \in (0,1)$ is an arbitary constant.

Theorem 3 further implies that by selecting $b = O(T)$, the PB-SVRGE algorithm achieves a convergence rate of $O\left(\frac{1}{\sqrt{(1+2b)T}}\right)$, which demonstrates better than the convergence rate of $O\left(\frac{1}{\sqrt{T}}\right)$ exhibited by traditional PSO-based algorithms such as pbSGD and pbSGDM[9].

2.2 Experiment Settings

2.2.1 Datasets

The original paper conducts experiments performed on various datasets, which is shown in Table 1, where the downloading links are also provided. The MNIST dataset and the CIFAR-10 dataset are used for other experiments.

Table 1: Summary of data sets

Data set	# examples	# features	url[a]
a8a	22,696	123	https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a8a
covtype[1]	581,012	54	https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#covtype.binary
CIFAR-10[6]	60,000	1,024	https://www.cs.toronto.edu/~kriz/cifar.html
ijcnn1	49,990	22	https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#ijcnn1
MNIST[7]	60,000	784	http://yann.lecun.com/exdb/mnist/[b]
news20.binary	19,996	1,355,191	https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#news20.binary

[a] This column provides the url where the datasets can be downloaded.

[b] The MNIST dataset is officially located at http://yann.lecun.com/exdb/mnist/, but the web page is not available now. We downloaded it from https://github.com/geektutu/tensorflow-tutorial-samples/tree/master/mnist.

[c] All these datasets are downloaded from the LIBSVM website[2], except for the MNIST dataset and CIFAR-10 dataset.

2.2.2 Objective Functions

Given a set of example pairs $\{x_i, y_i\}_{i=1}^{n}$, the goal was to find a solution of the following loss function:

$$\min_{w\in\mathbb{R}^d} f(w) := \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-y_i x_i^T w}) + \lambda r(w), \tag{6}$$

where $r(w) = \Sigma_{i=1}^{d}\frac{w_i^2}{1+w_i^2}$ is a non-convex regularizer.

2.2.3 Experiment Settings

The original experiments were conducted in three distinct parts. Each part involved tuning one of three parameters: the power coefficient γ, the mini-batch size b, and the learning rate η_k. Detailed settings and experimental configurations are provided in Section 4.

3 Enhancement Algorithm

We were unable to propose enhancements to the original algorithm, we focus on reproducing the original algorithm and conducting experiments to verify its effectiveness.

4 Comparison of the Experiments

We use the same datasets and objective functions mentioned in Section 2.2 as the original paper to conduct the experiments.

We take $\lambda = 0.1$ in (6). All the experiments were conducted on an AMD Ryzen 7 5800H CPU. The code is implemented in Jupyter Notebook with a Python 3.9.13 kernel.

As mentioned before, we show how the power coefficient γ, the mini-batch size b, and the learning rate η_k effets on the performance.

4.1 How the epoch are defined

In the original paper, the definition of an epoch is not explicitly clarified with respect to K and S. The authors only state that n stochastic gradient computations (i.e., one full gradient evaluation) constitute a single effective pass, and that $K = O(n)$ is used in the convergence analysis. Further discussion regarding the choice of K can be found in [4], where $K = 5n$ is adopted for non-convex objective functions, and the gradient computation in the outer loop is also included in the count.

In our experiments, we set $K = \frac{3n}{b}, = 10$ and ignore the outer loop computation, resulting in a total of 30 full gradient evaluations.

4.2 Datasets

In our experiments, we attempted to use the same datasets as those in the original paper, namely a8a, ijcnn1, covtype, and news20.binary. However, the covtype dataset contains an excessively large number of examples (n), and the news20.binary dataset has an extremely high dimensionality (d), resulting in substantial memory consumption and prolonged computation time[1]. Consequently, we restricted our experiments to the a8a and ijcnn1 datasets. The original paper also didn't conducted experiments on these two datasets when evaluating the effects of the learning rate η and the mini-batch size b.

4.3 Effects on the Parameters

The x-axis of the figures in this section denotes the number of full gradient evaluations, while the y-axis represents the objective gap $f(w_{k+1}^{s+1}) - f(w^*)$ in log scale, where w_{k+1}^{s+1} is the current iterate and w^* denotes the optimal solution, which is approximated by running ADAM[5] for at most 2,000 iterations with a learning rate $\eta = 0.01$ and a stopping criterion $\|\nabla f(w)\| < 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$. The objective gap is recorded every $\frac{n}{10}$ gradient evaluations, resulting in a total of 300 data points.

4.3.1 Effects of Power Coefficient γ

Figure 1 plots the performance of PB-SVRGE when using different power coefficients, where the power coefficient $\gamma \in [0.0, 0.2, 0.4, 0.6, 0.9, 1.0]$. We set the mini-batch size $b = 10$ and the learning rate $\eta = 0.01$.

[1] On a machine with 16GB of memory, processing a single parameter setting for these datasets occupied all available memory for nearly an hour.

4.3.2 Effects of Mini-batch Size b

Figure 2 shows the numerical behavior of PB-SVRGE when we took four different mini-batch sizes. We set $\gamma = 0.9$ and the learning rate $\eta = 0.01$.

4.3.3 Effects of Learning Rate η_k

Figure 3 presented the properties of PB-SVRGE when we employed four different learning rates. We set $b = 10$ and $\gamma = 0.9$.

4.3.4 Analysis of the Results

Some of the curves vanish as the number of passes over the data increases. This phenomenon occurs because w^* is only an approximation obtained by running ADAM, rather than the true optimal solution. When PB-SVRGE converges to a solution better than the approximated w^*, the objective gap $f(w_{k+1}^{s+1}) - f(w^*)$ may become negative.

There are some slight differences between our results and those reported in the original paper[8], but may be attributed to the stochastic nature of the algorithm.

Although some fluctuations are present in our figures due to more sampling points, it can be observed that our implementation of PB-SVRGE is consistent with the results reported in the original paper[8][1]. This consistency demonstrates the correctness and reliability of our experimental setup and implementation.

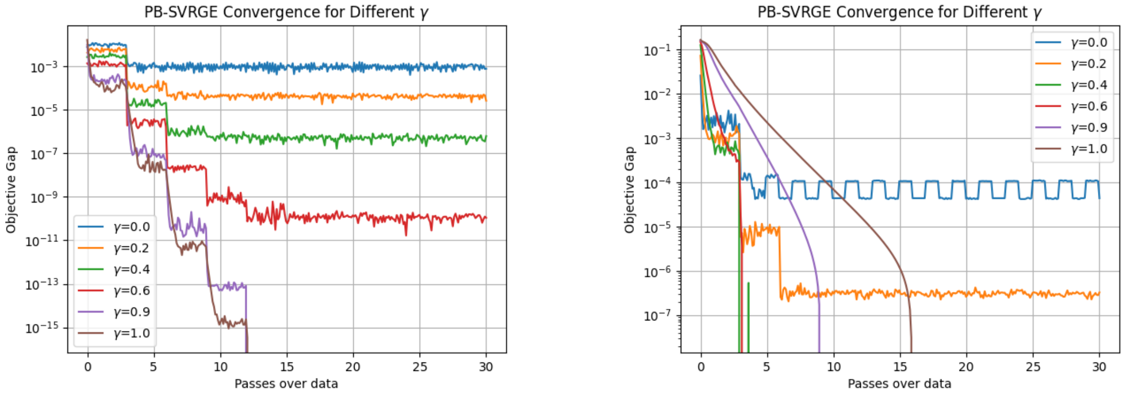
Overall, a better PB-SVRGE performance can be achieved by selecting a larger power coefficient γ, a smaller mini-batch size b, and a moderate learning rate η_k. We set $\gamma = 0.9$, $b = 10$, and $\eta_k = 0.01$ in our comparison.

5 Running the Code

The code for the PB-SVRGE algorithm is implemented in Python and is available at the following GitHub repository: `https://github.com/si11ybear/implementation-of-PB-SVRGE.git`. Detailed instructions for dataset preparation and running the experiments can be found in the `README.md` file within the repository.

Acknowledgements I would like to thank Professor Lin Zhoucheng and the teaching assistants for their guidance and help. The optimization course this semester has been very rewarding. Although I was unable to propose an enhanced algorithm in the final project due to various reasons, I have still learned a lot about the theoretical analysis and implementation of optimization algorithms.

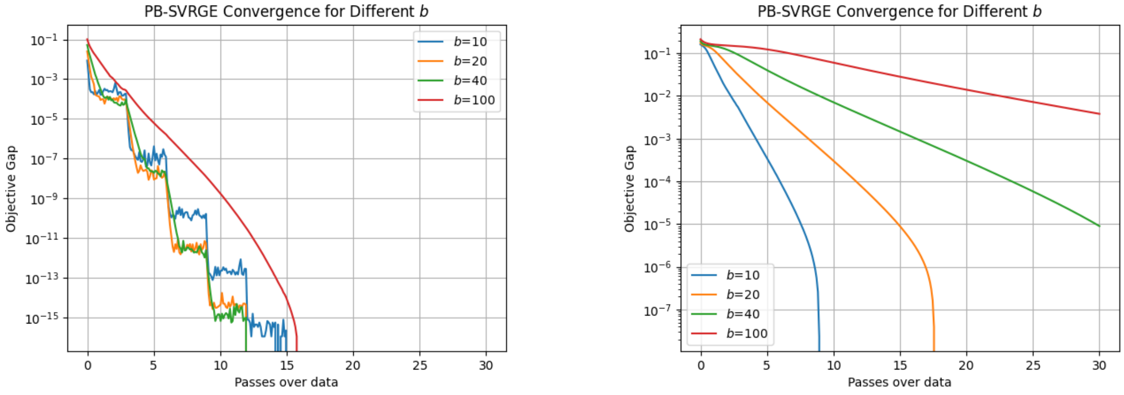
[1] Similar results are provided in the supplementary materials.



(a) a8a dataset

(b) ijcnn1 dataset

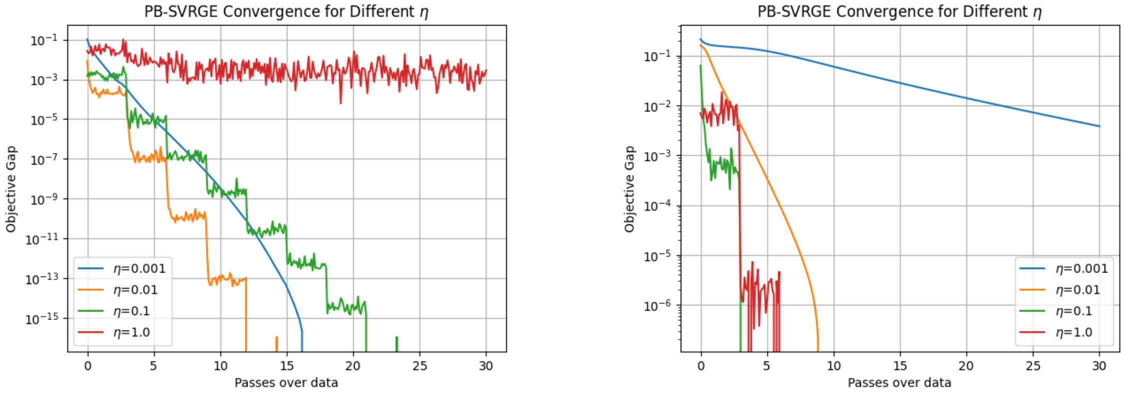
Figure 1: Performance of PB-SVRGE with different power coefficients γ on various datasets.



(a) a8a dataset

(b) ijcnn1 dataset

Figure 2: Performance of PB-SVRGE with different mini-batch sizes b on various datasets.



(a) a8a dataset

(b) ijcnn1 dataset

Figure 3: Performance of PB-SVRGE with different learning rates η_k on various datasets.

References

[1] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), May 2011.

[3] Evgenii Chzhen and Sholom Schechtman. Signsvrg: fixing signsgd via variance reduction, 2023.

[4] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 315–323, Red Hook, NY, USA, 2013. Curran Associates Inc.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[6] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[7] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[8] Zhuang Yang. Improved powered stochastic optimization algorithms for large-scale machine learning. *J. Mach. Learn. Res.*, 24(1), January 2023.

[9] Zhuang Yang and Xiaotian Li. Powered stochastic optimization with hypergradient descent for large-scale learning systems. *Expert Systems with Applications*, 238:122017, 2024.

[10] Ye Yuan, Mu Li, Jun Liu, and Claire Tomlin. On the powerball method: Variants of descent methods for accelerated optimization. *IEEE Control Systems Letters*, 3(3):601–606, July 2019.