Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.019069

ARTICLE

N-SVRG: Stochastic Variance Reduction Gradient with Noise Reduction Ability for Small Batch Samples

Haijie Pan and Lirong Zheng*

School of Information Science and Engineering, Fudan University, Shanghai, 200433, China

*Corresponding Author: Lirong Zheng. Email: lrzheng@fudan.edu.cn

Received: 01 September 2021 Accepted: 11 October 2021

ABSTRACT

The machine learning model converges slowly and has unstable training since large variance by random using a sample estimate gradient in SGD. To this end, we propose a noise reduction method for Stochastic Variance Reduction gradient (SVRG), called N-SVRG, which uses small batches samples instead of all samples for the average gradient calculation, while performing an incremental update of the average gradient. In each round of iteration, a small batch of samples is randomly selected for the average gradient calculation, while the average gradient is updated by rounding of the past model gradients during internal iterations. By suitably reducing the batch size B, the memory storage as well as the number of iterations can be reduced. The experiments are compared with the state-of-the-art Mini-Batch SGD, AdaGrad, RMSProp, SVRG and SCSG, and it is demonstrated that N-SVRG outperforms SVRG and SASG, and is on par with SCSG. Finally, by exploring the relationship between the small values of different parameters n. B and k and the effectiveness of the algorithm, we prove that our N-SVRG algorithm has some stability and can achieve sufficient accuracy even in the case of small batch size. The advantages and disadvantages of various methods are experimentally compared, and the stability of N-SVRG is explored by parameter settings.

KEYWORDS

Machine learning; SGD; SVRG; memory storage

1 Introduction

The variance problem introduced by the stochastic nature of the SGD algorithm becomes the main problem of optimization algorithms nowadays. The introduction of variance makes SGD reach only sublinear convergence speed with a fixed step size [1], while the stochastic algorithm accuracy is positively related to the sampling variance, and when the variance tends to 0, the deviation of the algorithm will also be 0. In this case, the SGD can still be fast even with a large step size convergence. Therefore, how to reduce the variance of the stochastic gradient in the stochastic algorithm has become an important issue studied by scholars [2].

For SGD variance problem, there are three mainstream methods to reduce the variance of sampling at present that include importance sampling, hierarchical sampling method and control

variable method. The objective function in machine learning is usually solved using the Batch Gradient Descent (BGD) or SGD [3]. BGD algorithm computes the gradients of all samples for each iteration to perform the weight update, and the latter randomly selects one training sample at a time to update the parameters by computing the sample gradients. Then came the improved Mini-Batch SGD (MBGD) algorithm, where MBGD computes the gradient and performs weight update by randomly selecting m data samples in the original data for each iteration. SGD has the advantage that each step relies only on a simple random sample gradient, so the computational consumption is only a fraction of that of the standard GD [4]. However, it has the disadvantage that a constant step size leads to slow convergence in the case of variance introduced by randomness.

In recent years, many scholars have carried out research based on SGD algorithm, for example, momentum algorithm [5], which is based on the idea of gradient momentum accumulation and uses exponential weighted average to reduce the swing amplitude of the gradient. Two concepts of velocity and friction are introduced into momentum algorithm. The function of the gradient is to change velocity, and friction is to gradually reduce the velocity. In the whole training process, the increase and attenuation of momentum are simulated to achieve the purpose of convergence. AdaGrad (Adaptive Gradient) is proposed to learn by gradually decreasing the step size [6]. By accumulating gradients, the learning rate of each weight is related to the value of their previous gradient. However, the consequence of accumulating gradients is that as the training increases, the step size decreases and eventually the training stalls. To address the training stagnation problem that occurs with AdaGrad, Hinton, G. proposed RMSProp (Root Mean Square Prop) to calculate the cumulative gradient using a moving average method that only accumulates the gradient of one window, making the change in step size adapt to the current gradient thus achieving better optimization [7,8]. And for SGD stochasticity introduces the variance problem, where SVRG is used to correct the gradient used for each model update using the global average gradient information [9]. Theoretical analysis and experiments demonstrated that SVRG produced linear convergence with reducing variance. Subsequently, Zhao et al. [10] proposed the SASG (Stochastic Average Gradient Average) algorithm, which has the same thematic idea as SVRG, with the difference that a piece of memory is used to store the original gradients of all samples, and the global average gradient is updated by constantly updating the original gradients during training, which requires a large amount of memory consumption throughout the training [11]. The SCSG (Stochastically Controlled Sto-chastic Gradient) algorithm was proposed, SCSG is to calculate the average gradient by randomly selecting a part of the sample gradient as the global gradient, but when performing the weight update, randomly selecting the number of updates will make the calculation more variable and tedious, and the computation is large [12]. Subsequently, a series of algorithms [13] such as the novel Mini-Batch SCSG [14,15], b-NICE, SAGA [16,17] were generated based on the idea of variance reduction.

However, there is another structural risk minimization problem in machine learning, which is composed of "loss function + regularization term", and different forms of regularization terms lead to different complex problems, such as Overlapping group lasso, Graph-guided fused lasso etc. [18], which are very complex for SGD-based theoretical approaches, while the ADMM algorithm is applied to a wider range of models and its excellent performance proves itself to be an effective optimization tool. Several variance reduction algorithms have been proposed in combination with ADMM, including SAG-ADMM [19], SDCA-ADMM [20], and SVRG-ADMM [21]. All three algorithms are improved algorithms generated based on the update strategy of ADMM. Then [22] proposed the APA-SVRG, which is trained on the basis of SVRG using

proximity averaging, and the experimental results show that the APA-SVRG algorithm is comparable to SVRG-ADMM. The neighborhood stochastic L-BFGS [23] methods, on the other hand, an improved algorithm based on Newton's method, which is trained mainly on loss functions that are smooth functions. Meanwhile, the specular stochastic sub-gradient descent [24] method is used to train on the loss function that is a non-smooth function. However, these schemes compute the gradient unbiased estimation using the average gradient of a small batch of samples, which cannot reduce the total complexity linearly, and the computational cost and memory consumption increase, and the computation and update of the gradient of a small batch of samples increase the computational consumption of the whole algorithm [25].

In order to address the above challenges, we propose a noise reduction method of stochastic gradient method, and then use the idea of small sample average gradient instead of global average gradient to design the algorithm N-SVRG that selects small samples for training while updating the average gradient to achieve variance reduction, and introduce the algorithm flow and convergence analysis of N-SVRG algorithm in detail, and compare it with the mainstream Mini-Batch SGD The N-SVRG algorithm is compared with the mainstream Mini-Batch SGD, AdaGrad, RMSProp, SVRG and SCSG algorithms, and it is proved that the N-SVRG algorithm outperforms SVRG, SASG and other algorithms, and is equal to SCSG. Finally, by exploring the relationship between the small values of different parameters n. B and k and the effectiveness of the algorithm, we prove that the N-SVRG algorithm has some stability and can achieve sufficient accuracy even in the case of low back size. Experimentally comparing the advantages and disadvantages of various methods and exploring the stability of the N-SVRG algorithm through parameter settings.

The contributions of this paper are as follows:

* We propose N-SVRG that uses small batches samples instead of all samples for the average gradient calculation, while performing an incremental update of the average gradient. By suitably reducing the batch size B, the memory storage as well as the number of iterations can be reduced.

* The convergence analysis of the proposed algorithm shows that the algorithm can stably converge to a certain lower limit, and find the saddle point with smooth gradient descent in the whole training process of neural network. It also enables the algorithm in this paper to reduce the computational cost and memory burden.

* The experiments are compared with the state-of-the-art demonstrate that N-SVRG outperforms baseline. Exploring the relationship between the small values of different parameters n. B and k and the effectiveness of the algorithm, we prove that our N-SVRG algorithm has some stability and can achieve sufficient accuracy even in the case of low batch size.

2 Background

For the supervised learning problem in machine learning [2,26]: assume that there is a functional model l in the space L for each model input x, i.e., there is a prediction $l(x)$. That is, given n training data $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ where $x_i \in \mathrm{R}^d$ denotes the input sample data and $y_i \in R$ denotes the corresponding training labels. Our goal is to find the prediction function that minimizes the loss due to inaccurate predictions. However, since the predictions $l(x)$ and y_i will differ each time the model predicts, one uses the loss function to assess the difference between the two. Since each sample corresponds to a loss function, the empirical risk is the average of these n sample loss functions.

Experience risk:

$$P_n(w) = \frac{1}{n}\sum_{i=1}^{n} f_i\left(l\left(w; x_i\right), y_i\right) \tag{1}$$

Expected risk:

$$P(w) = E\left[f_i\left(l\left(w; x_i\right), y_i\right)\right] \tag{2}$$

where w represents the parameters in the function model, and $f_i\left(l\left(w; x_i, y_i\right)\right)$ represents the loss function corresponding to sample (x_i, y_i). Formulas (1) and (2) clearly show how the expected risk and empirical risk depend on the error function, sample space, sample set, etc.

Empirical risk minimization is solving for w such that the empirical risk is minimized:

$$w^* \in \arg\min_{w}\left(P(w) : = \frac{1}{n}\sum_{i=1}^{n} f_i\left(w; x_i, y_i\right)\right) \tag{3}$$

However, during the training process, the model may have a strong performance capability because it has a large number of parameters itself, but the data provided for training are insufficient. Insufficient data for training, then the model may overfit during a large number of repeated training sessions, i.e., "the learned model is so well suited to a specific set of data that it does not reliably fit other data or future observations". In order to reduce the impact of overfitting [4,27] to some extent, a regularization term is added to the empirical risk to limit the complexity of the model, which constitutes a structural risk minimization [28] in the form of "loss function + regularization term" problem:

$$w^* \in \arg\min_{w}\left(P(w) : = \frac{1}{n}\sum_{i=1}^{n} f_i\left(w; x_i, y_i\right) + \lambda r(w)\right) \tag{4}$$

where $\lambda > 0$ is a hyperparameter that balances the weight of the regularization term by its own numerical size, and the larger λ is set, the heavier the penalty on the weight. $r(w)$ picks different forms depending on the effect, which include L1 parametrization, L2 parametrization [29], and L$\sim \infty$ parametrization. The most common form is the L2 parametrization, i.e., $r(w) = \|w\|$ 2, which can be calculated using $\sqrt[2]{w_1^2 + w_2^2 + w_n^2}$.

The loss function in machine learning is a non-negative real function $f\left(l(x), y\right)$, where the common loss functions are log Logi Loss, Squared Loss, 0-1 Loss, and Loss and Cross Entropy Error Loss (CEL) [30]. As shown below:

Logarithmic loss function:

$$f\left(l(x), y\right) = \log\left(1 + \exp\left(-yl(x)\right)\right) \tag{5}$$

Mean square error loss function:

$$f\left(l(x), y\right) = \left(y - l(x)\right)^2 \tag{6}$$

0-1 Loss function:

$$f\left(l(x), y\right) = \begin{cases} 1 & l(x) \neq y \\ 0 & l(x) = y \end{cases} \tag{7}$$

Cross-entropy error loss function:

$$f\left(l\left(x\right),y\right)=-y\log\left(l\left(x\right)\right) \tag{8}$$

The use of these loss functions should be chosen according to the specific problem, for example, the cross-entropy error loss function can be used with the Softmax function to form a Softmax-with-Loss layer for training and learning the classification problem [31].

3 Related Concepts

To facilitate the subsequent analysis and study, we give the relevant basic concepts that will be covered in the paper as well as the definitions. There is an iterative sequence x_t, with two adjacent iteration points x_{k+1}, x_k, satisfying the following conditions:

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{x_k-x^*}=\mu \tag{9}$$

1) If $\mu=1$, the algorithm achieves sublinear convergence;

2) If $0<\mu<1$, the algorithm achieves linear convergence [32];

3) If $\mu=0$, the algorithm achieves superlinear convergence.

Let set $C\subset R^d$, any $x_1\in C, x_2\in C$ and $\theta\in[0,1]$ has $(1-\theta)x_1+C, x_2\in C$, the set C is called a convex set.

Let set $C\subset R^d$ be a convex set and f be a function defined on set C. For any set, $x_1\in C, x_2\in C$ exists:

$$f\left(\alpha_1x_1+\alpha_2x_2\right)\le\alpha_1f\left(x_1\right)+\alpha_2f\left(x_2\right)\sum\alpha_i=1,\alpha_i\ge0 \tag{10}$$

$f\ (x)$ is a convex function defined on a convex set C.

For a function $f(y)$ with step $\eta>0$, its proximal operator at the point x [33] (Proximal Operator) is defined as:

$$\text{prox}_f^\eta\left(x\right)=\arg\min_{y\in R^d}\left(f\left(y\right)+\frac{1}{2\eta}\|y-x\|^2\right) \tag{11}$$

If $f=I_C$ is a schematic function on the convex set C

$$I_C\left(\text{x}\right)=\begin{cases}0 & x\in C\\ +\infty & x\notin C\end{cases} \tag{12}$$

Then:

$$\text{prox}_{I_C}^\eta\left(x\right)=\arg\min_{y\in R}\left(\|y-x\|^2\right) \tag{13}$$

That is, the proximity operator of the function $f(y)$ at x can be viewed as the Euclidean projection of x on the set C.

Assume (Lipschitz continuous target gradient) that the objective function F: $R^d \to R$ is continuously differentiable, the gradient function ∇F: $R^d \to R$ of F is Lipschitz continuous and the Lipschitz constant $L > 0$, i.e.,

$$\|\nabla F(w) - \nabla F(\overline{w})\|_2 \le L\|w - \overline{w}\|_2, \quad \{w, \overline{w}\} \subset R^d \tag{14}$$

Ensures that the gradient of F does not change arbitrarily fast with respect to the parameter vector. This assumption is essential for the convergence analysis of most gradient-based methods; without it, the gradient will not be a good indication of the distance to reduce F.

Assumption 1: If the function f_i satisfies L smooth, then for any $x, y \in R^d$, and f_i with gradient $\nabla f_i(x)$ at point x, there exists $L > 0$ such that

$$f_i(y) \le f_i(x) + \langle \nabla f_i(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \tag{15}$$

The constant L is called the Lipschitz constant and is generally used to measure the smoothness of a function. The function satisfies L smoothness in case it itself satisfies the Lipschitz continuity condition. For a fixed f_i, L is a fixed value. This condition places a restriction on the variation of the value of the function.

If the function f_i is strongly convex, then for any $x, y \in R^d$, and f_i with gradient $\nabla f_i(x)$ at point x, there exists $\gamma > 0$ such that

$$f_i(y) \ge f_i(x) + \langle \nabla f_i(x), y - x \rangle + \frac{\gamma}{2}\|y - x\|^2 \tag{16}$$

4 The Proposed Algorithm

4.1 Controlled Variable Method

This section describes the specific implications of the control variables approach [34]. It is assumed that we need to use Monte Carlo [35] sampling method to estimate the expectation μ of the random variable X, i.e., $E[X] = \mu$. Also assume that we have been able to be able to estimate relatively easily the expectation τ of another random variable Y, i.e., $E[Y] = \tau$. We construct a new random variable:

$$\mathrm{Z} = \mathrm{X} + \mathrm{c}(\mathrm{Y} - \tau) \tag{17}$$

where $C \in R^d$ is the corresponding coefficient, it can be seen from Eq. (17) that Z remains an unbiased estimator of μ [16] and that the variance of Z:

$$\mathrm{Var}(\mathrm{Z}) = \mathrm{Var}(\mathrm{X}) + \mathrm{c}^2\mathrm{Var}(\mathrm{Y}) + 2\mathrm{cCov}(\mathrm{X}, \mathrm{Y}) \tag{18}$$

It can be easily obtained by calculation that Var(Z) is minimum at $C = -\frac{\mathrm{Cov(X,Y)}}{\mathrm{Var(Y)}}$, when

$$\mathrm{Var}(\mathrm{Z}) = \mathrm{Var}(\mathrm{X}) - \frac{[\mathrm{Cov}(\mathrm{X}, \mathrm{Y})]^2}{\mathrm{Var}(\mathrm{Y})} = \left(1 - \rho_{X,Y}^2\right)\mathrm{Var}(\mathrm{X}) \tag{19}$$

where $\rho_{X,Y} = \mathrm{Corr}(X, Y)$. It can be seen from (18) and (19). The new random variables Z and X are unbiased estimates of μ as well, but the variance of the random variable Z is smaller than that of X. Therefore, it is better to use the random variable Z as an unbiased estimate of μ than X. From the formula, it can be seen that the variance of Z is sufficiently small as long as the

random variable X is guaranteed to show a certain correlation with Y [3], so Y is also called the control variable of X. This is the control variable method.

4.2 SVRG

SVRG is a new variance reduction method formed based on the control variable method, which itself does not construct control variables from similar samples, but from the perspective of optimizing variables. Its iterative approach is:

$$\begin{aligned} &\widetilde{g}_{j-1} \leftarrow \nabla f_{i_j}\left(w_{j-1}\right)-\left(\nabla f_{i_j}\left(w_{t-1}\right)-\widetilde{\mu}\right) \\ &w_j \leftarrow w_{j-1}-\eta \widetilde{g}_{j-1} \end{aligned} \tag{20}$$

where $w \in R^d$ denotes the model parameters, η denotes the update step, $\widetilde{\mu}=\nabla R_n\left(w_{t-1}\right)=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(\widetilde{w}_{t-1}\right)$ is the global average gradient which is calculated using the model parameters $\widetilde{w}_{t-1}$ obtained at the end of the previous cycle, $\nabla f_i\left(\widetilde{w}_{t-1}\right)$ denotes the gradient of the function f over the sample i_j with respect to parameter $\widetilde{w}_{t-1}$, $\nabla f_i\left(\widetilde{w}_{t-1}\right)-\widetilde{\mu}$ is the deviation of the control variables from the unbiased estimate, and $\widetilde{g}_{j-1}$ is the corrected unbiased estimate, using $\widetilde{g}_{j-1}$ to update w_j.

SVRG itself is a periodic algorithm, with the increase of iterations, $\widetilde{w}_{t-1}$ and w_j-1 gradually converge to w^*, that is: $\widetilde{w}_{t-1}\rightarrow w^*$, $w_{j-1}\rightarrow w^*$, when $t\rightarrow\infty$, so $\nabla f_i\left(w_{j-1}\right)$ and $\nabla f_i\left(\widetilde{w}_{t-1}\right)$ are getting closer and closer, and because $\nabla f_i\left(w_{j-1}\right)$ is the control variable of $\nabla f_i\left(\widetilde{w}_{t-1}\right)$, according to the theory, when and the closer, the higher the correlation between the two, then the $\widetilde{g}_{j-1}$ variance will converge to 0, so that the variance can be reduced to 0 by the control variable method, so the SVRG algorithm achieves linear convergence.

The SVRG algorithm steps are as follows:

Step 1: Given the initialization weight $\widetilde{w}_0$, the number of SGD steps m, the number of outer loop iterations T and the learning step η, initialize $t=0$;

Step 2: Calculate the global average gradient $\widetilde{\mu}\leftarrow\nabla P\left(w_{t-1}\right)=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(w_{t-1}\right)$ and initialize $w_0\leftarrow\widetilde{w}_{t-1}$;

Step 3: $j=0$, select a random sample $i_j\in\{1,\cdots,n\}$, calculate $\widetilde{g}_{j-1}=\nabla f_{i_j}\left(w_{j-1}\right)-\left(\nabla f_{i_j}\left(w_{t-1}\right)-\widetilde{\mu}\right)$, $w_j=w_{j-1}-\eta\widetilde{g}_{j-1}$, $j=j+1$, cycle Step 3 until $j=m$, $\widetilde{w}_t\leftarrow w_j$, go to the next step;

Step 4: t = t + 1, go to Step 2 until t = T, output: Choice 1: $w^*=\widetilde{W}_{\mathrm{t}+1}$; Choice 2: $w^*=\frac{1}{T}\sum_{i=0}^{T}\widetilde{W}_{\mathrm{t}+1}$. From the pseudo-code, we can see that the main calculations of the algorithm are in Steps 2 and 3, where Step 2 is to calculate the average gradient and Step 3 is used to calculate the corrected unbiased estimate of the gradient from $\widetilde{W}_{\mathrm{t}-1}$ to $\widetilde{W}_{\mathrm{t}}$ rounds iteratively at a computational cost of $n+2m$.

Assuming that all f_i are convex functions and satisfy Eqs. (19) and (20) and $\gamma>0$, and assuming that m is large enough, then

$$\alpha=\frac{1}{\gamma\eta\left(1-2\mathrm{L}\eta\right)\mathrm{m}}+\frac{2\mathrm{L}\eta}{1-2L\eta}<1 \tag{21}$$

Expectation of geometric convergence [21] for SVRG:

$$(\mathrm{P}(w_t)) \leq \mathrm{E}\left(\mathrm{P}\left(w^*\right)\right) + \alpha^t \left[P(w_0) - \mathrm{P}\left(w^*\right)\right] \tag{22}$$

Proof: For any i, consider that there is

$$g_i(w) = f_i(w) - f_i\left(w^*\right) - \nabla f_i\left(w^*\right)^T \left(w - w^*\right) \tag{23}$$

science $\nabla g_i(w^*) = 0$, so $g_i(w^*) = \min_w g_i(w)$,

Because $\nabla g_i(w^*) = 0$, therefore $g_i(w^*) = \min_w g_i(w)$, therefore

$$\begin{aligned} &0 = g_i(w^*) \leq \min_\eta \left[g_i(w) - \eta \left\|\nabla g_i(w)\right\|_2^2\right] \\ &\leq \min_\eta \left[g_i(w) - \eta \left\|\nabla g_i(w)\right\|_2^2 + 0.5L\eta^2 \left\|\nabla g_i(w)\right\|_2^2\right] \\ &= g_i(w) - \tfrac{1}{2L} \left\|\nabla g_i(w)\right\|_2^2 \end{aligned} \tag{24}$$

So

$$\left\|\nabla f_i(w) - \nabla f_i\left(w^*\right)\right\|_2^2 \leq 2L\left[f_i(w) - f_i\left(w^*\right) - \nabla f_i\left(w^*\right)^T \left(w - w^*\right)\right] \tag{25}$$

When $i = 1,2, \ldots, n$, the cumulative inequality (21) is added. And using $\nabla P(w^*) = 0$, we obtain

$$\frac{1}{n}\sum_{i=1}^{n} \left\|\nabla f_i(w) - \nabla f_i\left(w^*\right)\right\|_2^2 \leq 2L\left[P(w) - P\left(w^*\right)\right] \tag{26}$$

Give way $\widetilde{g}_{j-1} = \nabla f_{i_j}\left(w_{j-1}\right) - \left(\nabla f_{i_j}\left(w_{t-1}\right) - \widetilde{\mu}\right)$ available

Make $\widetilde{g}_{j-1} = \nabla f_{i_j}\left(w_{j-1}\right) - \left(\nabla f_{i_j}\left(w_{t-1}\right) - \widetilde{\mu}\right)$ available to

$$\begin{aligned} &\mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2 \\ &\leq 2\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right) - \nabla f_{i_j}\left(w^*\right)\right\|_2^2 \\ &+2\mathbb{E}\left\|\left[\nabla f_{i_j}\left(w_{t-1}\right) - \nabla f_{i_j}\left(w^*\right)\right] - \nabla \mathrm{P}\left(w_{t-1}\right)\right\|_2^2 \\ &\leq 2\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right) - \nabla f_{i_j}\left(w^*\right)\right\|_2^2 \\ &+2\mathbb{E}\left\|\left[\nabla f_{i_j}\left(w_{t-1}\right) - \nabla f_{i_j}\left(w^*\right)\right] - \mathrm{E}\left[\nabla f_{i_j}\left(w_{t-1}\right) - \nabla f_{i_j}\left(w^*\right)\right]\right\|_2^2 \\ &\leq 2\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right) - \nabla f_{i_j}\left(w^*\right)\right\|_2^2 + 2\mathbb{E}\left\|\nabla f_{i_j}\left(w_{t-1}\right) - \nabla f_{i_j}\left(w^*\right)\right\|_2^2 \\ &\leq 4\mathrm{L}\left[\mathrm{P}\left(w_{j-1}\right) - \mathrm{P}\left(\mathrm{w}^*\right) + \mathrm{P}\left(w_{t-1}\right) - \mathrm{P}\left(\mathrm{w}^*\right)\right] \end{aligned} \tag{27}$$

The first inequality uses $\|a+b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$, the second inequality uses $\widetilde{\mu} = \nabla P\left(w_{t-1}\right)$

$\mathbb{E}\|\xi - \mathrm{E}\xi\|_2^2 = \mathbb{E}\|\xi\|_2^2 - \|\mathrm{E}\xi\|_2^2 \leq \mathbb{E}\|\xi\|_2^2$, the third inequality uses (27). Existing

$$w_j = w_{j-1} - \eta \widetilde{g}_{j-1}, E\left(\widetilde{g}_{j-1}\right) = \nabla \mathrm{P}\left(w_{j-1}\right), \text{ so}$$

$$\begin{aligned}
&\mathbb{E}\left\|w_j - V\right\|_2^2 \\
&= \left\|w_{j-1} - w^*\right\|_2^2 - 2\eta\left(w_{j-1} - w^*\right)^T \mathbb{E}\widetilde{g}_{j-1} + \eta^2 \mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2 \\
&\leq \left\|w_{j-1} - \mathrm{w}^*\right\|_2^2 - 2\eta\left(w_{j-1} - \mathrm{w}^*\right)^T \nabla P\left(w_{j-1}\right) + 4L\eta^2\left[P\left(w_{j-1}\right) - P\left(\mathrm{w}^*\right)\right] \\
&+ [8 + C(j-1)] L\eta^2 \left[P\left(w_{t-1}\right) - P\left(\mathrm{w}^*\right)\right] \\
&\leq \left\|w_{j-1} - \mathrm{w}^*\right\|_2^2 - 2\eta\left[P\left(w_{j-1}\right) - P\left(\mathrm{w}^*\right)\right] \\
&+ 4L\eta^2\left[P\left(w_{j-1}\right) - P\left(\mathrm{w}^*\right) + P\left(w_{t-1}\right) - P\left(\mathrm{w}^*\right)\right] \\
&= \left\|w_{j-1} - \mathrm{w}^*\right\|_2^2 - 2\eta\left(1 - 2\ \mathrm{L}\eta\right)\left[P\left(w_{j-1}\right) - P\left(\mathrm{w}^*\right)\right] \\
&+ 4\mathrm{L}\eta^2\left[P\left(w_{t-1}\right) - P\left(\mathrm{w}^*\right)\right]
\end{aligned} \tag{28}$$

The first inequality uses the previously obtained inequality $\mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2$, and the second inequality uses the P(w)-convex property. By accumulating the $j = 1, \ldots m$, stages we obtain

$$\begin{aligned}
&\mathbb{E}\left\|w_m - w^*\right\|_2^2 + 2\eta m\left(1 - 2L\eta\right)\mathbb{E}\left[P\left(w_t\right) - P\left(w^*\right)\right] \\
&\leq \mathbb{E}\left\|w_0 - w^*\right\|_2^2 + 4Lm\eta^2\mathbb{E}\left[P\left(w_{t-1}\right) - P\left(w^*\right)\right] \\
&\leq \tfrac{2}{\gamma}\mathbb{E}\left[P\left(w_{t-1}\right) - P\left(w_*\right)\right] + 4Lm\eta^2\mathbb{E}\left[P\left(w_{t-1}\right) - P\left(\mathrm{w}^*\right)\right] \\
&= \left[\tfrac{2}{\gamma} + 2\mathrm{Lm}\eta^2\right]\mathbb{E}\left[P\left(w_{t-1}\right) - P\left(\mathrm{w}^*\right)\right]
\end{aligned} \tag{29}$$

The second inequality uses the strong convexity property, thus obtaining

$$\mathbb{E}\left[P\left(w_t\right) - P\left(w_*\right)\right] \leq \left[\frac{1}{\gamma\,\eta\left(1 - 2L\eta\right)m} + \frac{2L\eta}{1 - 2L\eta}\right]\mathbb{E}\left[P\left(w_{t-1}\right) - P\left(w_*\right)\right] \tag{30}$$

So get $\mathrm{E}\left(\mathrm{P}\left(\widetilde{w}_t\right)\right) \leq \mathrm{E}\left(\mathrm{P}\left(w^*\right)\right) + \alpha^t\left[P\left(w_0\right) - \mathrm{P}\left(w^*\right)\right]$, cite as evidence.

5 N-SVRG Algorithm

The SASG algorithm is a memory-consuming algorithm in the SVRG family of algorithms. Although the computational cost of the SVRG algorithm is huge when computing the average of all sample gradients, it can be computed in parallel during the computation because it can use matrix operations itself. the difference between the SASG and SVRG algorithms is that the SASG algorithm requires one memory block to store all sample gradients, which not only increases the computational cost (because matrix parallelism cannot be used), but also increases the memory burden of the computer. In order to reduce the computational cost and memory burden, we designed the back decimation update gradient [2,33].

5.1 Algorithm Introduction

In the SASG algorithm, a piece of memory is used to store the original gradients of all samples, while a new gradient $\nabla f_{i_j}\left(w_{t-1}\right)$ modeled with the current new weights is computed after passing $\nabla f_{i_j}\left(w_{t-1}\right) \leftarrow \nabla f_{i_j}\left(w_{j-1}\right)$ thereby achieving the purpose of updating $\nabla P_n\left(w_{t-1}\right)$. However, its memory consumption is too large, and the computational cost increases, which makes it impractical to use the SASG algorithm in the face of the complexity of the model due to the large size of the data and the huge parameters, etc. The SCSG algorithm is mainly designed to reduce the computational cost. In the SCSG algorithm, it does not need to calculate the gradient of all samples, but only needs to consistently sample a small batch of samples and calculate the

average gradient instead of the global sample gradient, so that it can obtain the same training accuracy and convergence speed as SVRG [15,34].

The small batch average gradient in SCSG and the SGD update approach in the SASG algorithm inspired our algorithm. n-SVRG algorithm discards the calculation of the global average gradient for all samples and consistently samples a small batch of samples from the sample, and replaces the global average sample gradient for all by calculating the batch average gradient, i.e., $\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(w_{t-1}\right) \rightarrow \frac{1}{B}\sum_{i=1}^{B}\nabla f_i\left(w_{[i]}\right)$, so that we can reduce the computational cost. At the same time, we store the gradients of these samples, as in the SASG algorithm, the batch sample gradients are modified after each update of the weight, and then the average gradient is recalculated. But we change the update form of SASG average gradient from $\nabla f_{i_j}\left(w_{t-1}\right) \leftarrow \nabla f_{i_j}\left(w_{j-1}\right)$ to $\nabla f_{i_j}\left(\widetilde{w}_{t-1}\right) \leftarrow 0$, that is, the sample gradient calculated by the past model is directly rounded off, and the gradient variance is reduced by directly rounding of the sample gradient. From the perspective of the control variables method, it can be interpreted as follows: the mean of the original weight gradient is used as the estimator, and the variance is reduced by gradually decreasing the number of samples. So inevitably, the number of iterations to compute the unbiased estimate of the gradient is correspondingly reduced to less than B, which indirectly reduces the computational cost.

The N-SVRG algorithm steps are as follows:

Step 1: Given the initialization model parameters $\widetilde{w}_0$, the number of outer cycles T, the step size η, the number of SGD steps k and the size of each batch B, initialize $t = 0$;

Step 2: Randomly and consistently sample a batch of $\boldsymbol{\mathcal{H}}_t \subset \{1, \cdots, n\}$, where size $|I_t| = B$;

Step 3: Calculate the gradient $\nabla f_i\left(\widetilde{w}_{t-1}\right)$ for all samples in $\boldsymbol{\mathcal{H}}_t$ one by one and deposit it in $\nabla f_i\left(w_{[i]}\right)$, initializing $w_0 \leftarrow \widetilde{w}_{t-1}$, $j = 0$;

Step 4: Calculate $\widetilde{\mu} = \frac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{[i]}\right)$ and select a random sample $i_j \in \boldsymbol{\mathcal{H}}_t$;

Step 5: Compute $\widetilde{g}_{j-1} \leftarrow \nabla f_{i_j}\left(w_{j-1}\right) - \left(\nabla f_{i_j}\left(w_{[i_j]}\right) - \widetilde{\mu}\right), w_j \leftarrow w_{j-1} - \eta\widetilde{g}_{j-1}$ and remove the gradient of sample ij by means of $\nabla f_{i_j}\left(w_{[i_j]}\right) \leftarrow 0$, $j = j+ 1$, looping Steps 4 and 5 until $j = k$, $\widetilde{w}_t \leftarrow w_j$, going to the next step;

Step 6: t = t + 1, go to Step 2 until t = T, output: $w^* = w^T$.

As can be seen from Step 5 to $\nabla f_{i_j}\left(w_{[i_j]}\right) \leftarrow 0$, $\nabla f_{i_j}\left(w_{[i_j]}\right)$ is finite and no longer embraced after deletion, to avoid repeating deletions by mistake, the random sorting of the samples in B will be used in the program programming to draw them sequentially. The algorithm steps show that the main computational cost of the N-SVRG algorithm is in Step 3 and Step 5, where the number of computations in Step 3 is B and Step 5 contains an inner loop with k, because $k < B$, so the maximum computational cost of the algorithm is 2 B.

5.2 Algorithm Convergence Analysis

For simplicity, we only consider the case where each $f_i(w)$ is convex and smooth and $P(w)$ is a strongly convex problem. There are the following assumptions:

f_i is a convex function with L-Lipschitz gradient [35]

$$f_i(w) - f_i\left(w'\right) - \frac{L}{2}\left\|w - w'\right\|^2 \leq \nabla f_i\left(w'\right)\left(w - w'\right) \tag{31}$$

Assume that $P(w)$ is a strongly convex function

$$P(w) - P\left(w'\right) - \frac{\gamma}{2}\left\|\mathrm{w} - w'\right\|_2^2 \geq \nabla P\left(w'\right)\left(\mathrm{w} - w'\right) \tag{32}$$

The expectation in Step 3 of the SVRG algorithm is $\mathrm{E}\left(\nabla f_{i_j}\left(w_{t-1}\right) - \widetilde{\mu}\right) = 0$, but the variance problem makes $\nabla f_{i_j}\left(w_{t-1}\right) - \widetilde{\mu}$ non-zero. The variance of $\nabla f_{i_j}\left(w_{j-1}\right)$ is reduced by using $\nabla f_{i_j}\left(w_{j-1}\right) - \left(\nabla f_{i_j}\left(w_{t-1}\right) - \widetilde{\mu}\right)$ in the update rule of the algorithm.

In the algorithm of this paper, the same SVRG is wanted to be achieved by updating $\widetilde{\mu}$. Consider the jth loop in the jth.

$$\begin{aligned}
&\widetilde{\mu} = \tfrac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{[i]}\right) \\
&= \tfrac{1}{B-j+1}\left(\sum_{i=1}^{B}\nabla f_i\left(w_{t-1}\right) - \textstyle\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)\right) \\
&= \tfrac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{t-1}\right) - \tfrac{1}{B-j+1}\textstyle\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)
\end{aligned} \tag{33}$$

Then

$$\begin{aligned}
&\widetilde{g}_{j-1} = \nabla f_{i_j}\left(w_{j-1}\right) - \left(\nabla f_{i_j}\left(w_{[i_j]}\right) - \widetilde{\mu}\right) \\
&= \nabla f_{i_j}\left(w_{j-1}\right) - \nabla f_{i_j}\left(w_{t-1}\right) + \left(\tfrac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{[i]}\right) - \tfrac{1}{B-j+1}\textstyle\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)\right) \\
&= \nabla f_{i_j}\left(w_{j-1}\right) - \nabla f_{i_j}\left(w_{[i_j]}\right) + \tfrac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{[i]}\right) - \tfrac{1}{B-j+1}\textstyle\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)
\end{aligned} \tag{34}$$

From the literature it is clear that

$$\frac{1}{n}\sum_{i=1}^{n}\left\|\nabla f_i(w) - \nabla f_i\left(w_*\right)\right\|_2^2 \leq 2L\left[P(w) - P\left(w_*\right)\right] \tag{35}$$

We can take the expectation for $\widetilde{g}_{j-1}$ and get

$$\begin{aligned}
&\mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2 \leq 2\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right)-\nabla f_{i_j}\left(w_{t-1}\right)+\frac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{t-1}\right)\right\|_2^2 \\
&+2\mathbb{E}\left\|\frac{1}{B-j+1}\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)\right\|_2^2 \\
&\leq 4\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2 \\
&+4\mathbb{E}\|\nabla f_{i_j}\left(w_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)-\frac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{t-1}\right)\|_2^2 \\
&+2\mathbb{E}\left\|\frac{1}{B-j+1}\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)\right\|_2^2 \\
&\leq 4\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2 \\
&+8\mathbb{E}\left\|\nabla f_{i_j}\left(w_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2 n+8\mathbb{E}\left\|-\frac{1}{B-j+1}\sum_{i=1}^{B}\nabla f_i\left(w_{t-1}\right)\right\|_2^2+2\mathbb{E}\left\|\frac{1}{B-j+1}\sum_k^{j-1}\nabla f_{i_k}\left(w_{[i_k]}\right)\right\|_2^2
\end{aligned}$$

Since $\frac{1}{B}\sum_{i=1}^{B}\nabla f_i\left(w_{t-1}\right)$ is used in the algorithm instead of $\nabla P\left(\widetilde{w}_{t-1}\right)$ and $\nabla P\left(\widetilde{w}_{t-1}\right)=\mathbb{E}\left[\nabla f_{i_j}\left(\widetilde{w}_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)\right]$, so

$$\begin{aligned}
&\mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2 \\
&\leq 4\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2+8\mathbb{E}\left\|\nabla f_{i_j}\left(\widetilde{w}_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2 \\
&+8\left(\frac{B}{B-j+1}\right)^2\left\|\mathbb{E}\left[\nabla f_{i_j}\left(\widetilde{w}_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)\right]\right\|_2^2+2\left(\frac{j-1}{B-j+1}\right)^2||\mathbb{E}\left[\nabla f_{i_j}\left(w_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)\right]||_2^2 \\
&\leq 4\mathbb{E}\left\|\nabla f_{i_j}\left(w_{j-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2 \\
&+\left[8+8\left(\frac{B}{B-j+1}\right)^2+2\left(\frac{j-1}{B-j+1}\right)^2\right]\mathbb{E}\left\|\nabla f_{i_j}\left(\widetilde{w}_{t-1}\right)-\nabla f_{i_j}\left(w_*\right)\right\|_2^2 \\
&\leq 4L\left[P\left(w_{j-1}\right)-P\left(w_*\right)\right]+\left[8+8\left(\frac{B}{B-j+1}\right)^2+2\left(\frac{j-1}{B-j+1}\right)^2\right]L\left[P\left(\widetilde{w}_{t-1}\right)-P\left(w_*\right)\right]
\end{aligned} \tag{36}$$

The first three inequalities are used $\|a+b\|_2^2\leq 2\|a\|_2^2+2\|b\|_2^2$ and the fourth inequality is used $\|\mathbb{E}[\mathrm{x}]\|_2^2\leq\mathbb{E}\|\mathrm{x}\|_2^2$ [26].

Set $C(j-1)=8\left(\frac{B}{B-j+1}\right)^2+2\left(\frac{j-1}{B-j+1}\right)^2$ also $\mathbb{E}v_t=\nabla P\left(w_{j-1}\right)$; This has

$$\begin{aligned}
&\mathbb{E}\left\|w_j-w_*\right\|_2^2 \\
&=\left\|w_{j-1}-w_*\right\|_2^2-2\eta\left(w_{j-1}-w_*\right)\mathbb{E}\widetilde{g}_{j-1}+\eta^2\mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2 \\
&\leq\left\|w_{j-1}-w_*\right\|_2^2-2\eta\left(w_{j-1}-w_*\right)\nabla P\left(w_{j-1}\right) \\
&+4L\eta^2\left[P\left(w_{j-1}\right)-P\left(w_*\right)\right]+\left[8+C(j-1)\right]L\eta^2\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right] \\
&\leq\left\|w_{j-1}-w_*\right\|_2^2-2\eta\left[P\left(\widetilde{w}_{j-1}\right)-P\left(w_*\right)\right]+4L\eta^2\left[P\left(w_{j-1}\right)-P\left(w_*\right)\right] \\
&+\left[8+C(j-1)\right]L\eta^2\left[P\left(\widetilde{w}_{t-1}\right)-P\left(w_*\right)\right]n \\
&\leq\left\|w_{j-1}-w_*\right\|_2^2-2\eta\left(1-2\ \mathrm{L}\eta\right)\left[P\left(\widetilde{w}_{j-1}\right)-P\left(w_*\right)\right] \\
&+\left[8+C(j-1)\right]L\eta^2\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right]
\end{aligned} \tag{37}$$

The first inequality uses $\mathbb{E}\left\|\widetilde{g}_{j-1}\right\|_2^2$ and the second inequality uses the $P(w)$-convex property. By accumulating the $j=1, \ldots, k$ stages we get

$$\begin{aligned}
&\mathbb{E}\left\|w_k-w_*\right\|_2^2+2\eta \mathrm{k}\left(1-2\mathrm{L}\eta\right)\mathbb{E}\left[P\left(w_t\right)-P\left(w_*\right)\right] \\
&\leq \mathbb{E}\left\|w_0-w_*\right\|_2^2+\left[8+\sum_{j=1}^k C\left(j\right)\right] L\eta^2\mathbb{E}\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right] \\
&\leq \frac{2}{\gamma}\mathbb{E}\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right]+\left[8+\sum_{j=1}^k C\left(j\right)\right] L\eta^2\mathbb{E}\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right] \\
&\leq \left[\frac{2}{\gamma}+\left[8+\sum_{j=1}^k C\left(j\right)\right] L\eta^2\right]\mathbb{E}\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right]
\end{aligned} \tag{38}$$

The second inequality uses the strongly convex property (38), and we thus obtain

$$\begin{aligned}
&\mathbb{E}\left[P\left(w_t\right)-P\left(w_*\right)\right] \leq \left[\frac{1}{\gamma[\eta \mathrm{k}(1-2\ \mathrm{L}\eta)]}+\frac{\left[8+\sum_{j=1}^k C(j)\right] L\eta}{2\mathrm{k}(1-2\ \mathrm{L}\eta)}\right]\mathbb{E}\left[P\left(w_{t-1}\right)-P\left(w_*\right)\right] \\
&\alpha=\frac{1}{\gamma[\eta \mathrm{k}(1-2\ \mathrm{L}\eta)]}+\frac{\left[8+\sum_{j=1}^k C(j)\right] L\eta}{2\mathrm{k}(1-2\ \mathrm{L}\eta)}
\end{aligned} \tag{39}$$

We have the convergence of N-SVRG

$$\mathbb{E}\left[P\left(w_T\right)-P\left(w_*\right)\right] \leq \alpha^T\mathbb{E}\left[P\left(w_0\right)-P\left(w_*\right)\right] \tag{40}$$

The convergence of the N-SVRG algorithm shows that the convergence rate of the algorithm is related to the selection of parameters B and k.

Regarding the selection of parameters B and k on the stability of the algorithm and the convergence speed exploration we will explain in detail in the experimental section.

6 Experimental Design and Experimental Results

6.1 Experimental Data

MNIST [35] is a handwritten digit image dataset that was collected with the aim of logarithmically enabling the recognition of handwritten digits, as shown in Fig. 1. This dataset has been widely used in machine learning and deep learning since 1998 by Yan LeCun in the LeNet-5 network to test the effectiveness of algorithms such as SVM, Linear Classifiers, Neural Nets, KNN [31,35], etc.

Figure 1: Example of MNIST dataset

The dataset consists of a training set and a test set, where the training set contains 60,000 handwritten digital images and the digital labels corresponding to the digital images. Similarly, the test set contains a total of 10,000 handwritten digital images and digital labels corresponding to the digital images. The data set is composed of digital images from 0 to 9, each of which is a single-channel gray scale image of 28 pixels by 28 pixels, and each pixel is a uint 8 data type, i.e., an 8-bit unsigned integer with a value between 0 and 255. Each image is labeled with "1", "3", "5", etc.

6.2 Experimental Setup

6.2.1 Data Pre-Processing

The experimental dataset is a MNIST dataset. In the experimental preprocessing stage, we need to regularize and expand the images in one dimension so that when we use batch processing, we can transform a batch of digital image samples into a two-dimensional array for computation. At the same time, we encode the data labels as one-hot. One-hot encoding is an array where the correct solution label is 1 and all others are 0. For example, like label '7' in one-hot encoding array is {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0}.

6.2.2 Network Structure

All program networks are structured as $728 \times 40 \times 10$, the input layer is 728 neurons with input function only, the implicit layer is 40 neurons containing the Relu function, and the output layer is a Softmax-with-Loss layer with 10 neurons. In the output layer we use a Softmax-with-Loss layer for learning.

6.2.3 Experimental Basis

The algorithm is done by Python3.6, for the implementation of the algorithm for the structure in [15,17] is consulted. The device used in this paper is CPU: Inter(R) Core(TM) i5-7500 CPU@3.40 GHz with 16.00 GB of RAM.

The experiment consists of two parts in total. The first part is mainly the evaluation comparison between the algorithm of this paper and the current mainstream algorithm; the second part is mainly the exploration of the relationship between the parameters of the algorithm of this paper.

6.3 Algorithm Comparison

The N-SVRG algorithm is programmed in PyCharm using Python3.6, and is compared with the mainstream Mini-Batch SGD, AdaGrad, RMSProp, SVRG and SCSG algorithms.

Parameter setting of each algorithm: Mini-Batch SGD:

Maximum number of iterations 4800, per batch size 250;

AdaGrad: 4800 iterations maximum, 250 per batch size;

RMSProp: 4800 iterations maximum, 250 per batch size;

SVRG: Maximum epoch count 20 times, SGD steps 5000;

SCSG: Maximum number of iterations 4800, batch size 250, number of SGD steps 250;

N-SVRG: Maximum number of iterations 4800, batch size 250, number of SGD steps 249.

The following points can be seen in Fig. 2 and Table 1:

(1) N-SVRG, RMSProp and SCSG all reach around 65% accuracy by the first epoch of iteration, while SVRG, Mini-Batch SGD and AdaGrad only reach below 40%. As the iteration proceeds, N-SVRG, RMSProp and SCSG reach 80% or more accuracy after only 7 epochs, SVRG reaches 65%, and Mini-Batch SGD and AdaGrad reach only 50%. Batch SGD and AdaGrad.

(2) N-SVRG, RMSProp and SCSG all achieve high accuracy smoothly and quickly. In terms of accuracy, the difference between N-SVRG and SCSG is 0.26%, and the difference between RMSProp and SCSG is 1.13%. The detection accuracy of SVRG is slightly lower than the previous three algorithms, but it is still better than Mini-Batch SGD and AdaGrad.

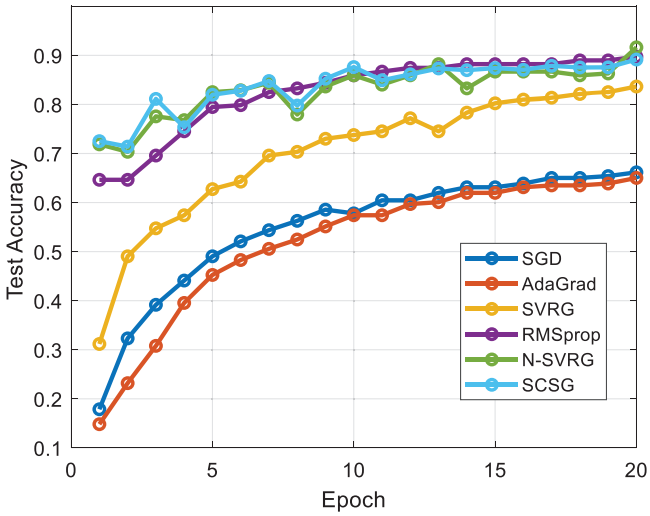

Figure 2: Comparison of algorithm accuracy

Table 1: Algorithm comparison results

	Mini-BatchSGD	AdaGrad	RMSprop	SVRG	SCSG	N-SVRG
Train loss	2.5241	3.1092	0.0546	0.5373	0.0601	0.1259
Test acc	0.6601	0.6607	0.9044	0.8547	0.8957	0.8931

6.4 Stability Exploration

The stability of the N-SVRG algorithm is mainly affected by two factors: the size of each batch B and the number of SGD steps k. The size of B directly affects the computational cost and storage cost of the algorithm, while an appropriate B can enable the algorithm to achieve better accuracy while converging quickly, and the number of SGD steps k has a similar effect with B. Also, the size relationship between k and B is what we need to discuss. Inspired by [24], this paper focuses on two aspects of the experimental investigation, namely, the relationship between the number of training samples B and n ratio and the relationship between B and k the ratio.

B and n ratio exploration:

In the experiment to explore the effect of B and n scaling relationship on the algorithm, in order to make the experimental sample size diversity, we need to take 10,000, 8,000, 4,000 samples from MNIST data one by one for training and $B \in \{n/2, n/4, n/8, n/16, n/32\}$, set $k = B - 1$ for the experiment.

The following can be seen from Fig. 3 and Table 2:

(1) The proportional relationship between B and n has a certain relationship with the effect of the algorithm, the larger B is the smaller the training accuracy, and the longer the training time will be because of the number of substitutions due to the setting of k;

(2) Relatively best accuracy at $B=\frac{n}{8}$ and $B=\frac{n}{32}$;

(3) Although it can be seen from the detection accuracy that the size of B has a relationship with the convergence speed and the accuracy of the algorithm, especially $B=\frac{n}{2}$ and $B=\frac{n}{4}$, it has less impact on the stability of the algorithm and still obtains relatively good results with a simple network structure and strong robustness.

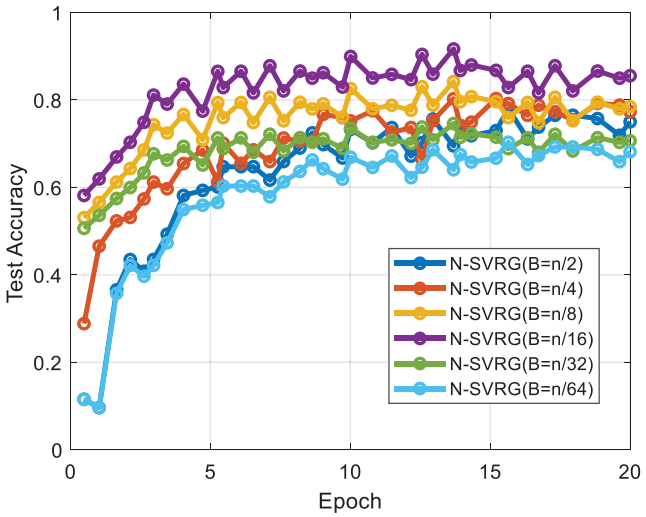

Figure 3: Results of the B *vs.* n ratio investigation

Table 2: Results of the B to n ratio investigation

	n = 10000		n = 8000		n = 4000	
	Train	Test	Train	Test	Train	Test
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
B = n/2	0.4924	0.839	0.4283	0.8433	0.3098	0.8253
B = n/4	0.2630	0.8813	0.3143	0.8564	0.3365	0.8295
B = n/8	0.2196	0.8919	0.325	0.8758	0.2497	0.8481
B = n/16	0.2434	0.9797	0.1781	0.8776	0.3354	0.8462
B = n/32	0.1124	0.894	0.1687	0.897	0.2984	0.8498

In the experiments exploring the effect of the relationship between B and k scaling on the algorithm, we directly use the complete MNIST dataset and consistently sample small batches of samples of sizes 250, 500, and 1000, respectively. while we set the number of SGD steps $k \in \{B\text{-}1, 0.8 \times B\ 0.6 \times B\ 0.4 \times B\ 0.2 \times B\}$ for the experiments.

The following can be seen in Fig. 4 and Table 3:

(1) The relationship between the size of k and B is positively related to the accuracy of the algorithm, the closer k is to B, the higher the testing accuracy of the algorithm, and vice versa, the accuracy decreases, and this phenomenon does not change with the size of B.

(2) From the cross-observation of the data, it is clear that when k is a constant (k = 200), the accuracy decreases with the increase of B, which confirms the above point.

(3) Comparing the results of two experiments, Experiment B with n-proportional probe and Experiment B with k-proportional probe, we can see that the N-SVRG algorithm is robust and maintains high experimental results for general optimization problems when the batch size is too small.

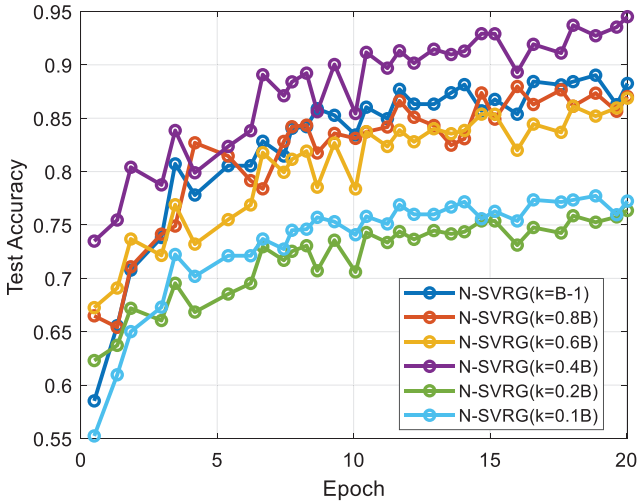

Figure 4: Results of the B *vs.* k ratio exploration

Table 3: Results of the investigation of the ratio of B to k

	B = 250		B = 500		B = 1000	
	Train	Test	Train	Test	Train	Test
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
k = B-1	0.1639	0.8922	0.2331	0.8801	0.2695	0.8745
k = 0.8 ∗ B	0.1435	0.8815	0.1851	0.8812	0.2230	0.8759
k = 0.6 ∗ B	0.1841	0.8722	0.1533	0.8786	0.2436	0.8712
k = 0.4 ∗ B	0.2401	0.8708	0.1899	0.8705	0.2269	0.8614
k = 0.2 ∗ B	0.2991	0.8536	0.3788	0.8449	0.4142	0.8313

6.5 Discussion

Through the experiments we can find that the N-SVRG algorithm is outstanding in convergence speed and accuracy compared with Mini-Batch SGD, AdaGrad and SVRG, and is comparable to RMSProp and SCSG. We can know that N-SVRG algorithm is a relatively stable algorithm, although there are two parameters B and k affect the accuracy of the algorithm, from the experiment we can see that the change of B shows a trend of the smaller the value, the higher

the accuracy of the algorithm, this nature is the algorithm is suitable for general optimization problems.

7 Conclusion

In this paper, we propose a noise reduction method for SVRG, which uses a small batch of samples instead of all samples for average gradient computation and incremental update of the average gradient. The experiments are compared with the mainstream Mini-Batch SGD, AdaGrad, RMSProp, SVRG and SCSG algorithms, and it is proved that the N-SVRG algorithm outperforms SVRG and SASG, and is equal to SCSG. Finally, by exploring the relationship between the small values of different parameters n, B and k and the algorithm effect, we prove that the N-SVRG algorithm has some stability.

Funding Statement: This work was supported by the National Natural Science Foundation of China under Grant 62076066.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Jain, P., Nagaraj, D. M., Netrapalli, P. (2021). Making the last iterate of SGD information theoretically optimal. *SIAM Journal on Optimization, 31(2),* 1108–1130. DOI 10.1137/19M128908X.
2. Hu, B., Seiler, P., Lessard, L. (2020). Analysis of biased stochastic gradient descent using sequential semidefinite programs. *Mathematical Programming, 187(5),* 1–26. DOI 10.1007/s10107-020-01486-1.
3. Prashanth, L. A., Korda, N., Munos, R. (2021). Concentration bounds for temporal difference learning with linear function approximation: The case of batch data and uniform sampling. *Machine Learning, 110(3),* 559–618. DOI 10.1007/s10994-020-05912-5.
4. Pan, H., Zheng, L. (2021). DisSAGD: A distributed parameter update sheme based on variance reduction. *Sensors, 21(15),* 5124. DOI 10.3390/s21155124.
5. Xie, Y., Li, P., Zhang, J., Ogiela, M. R. (2021). Differential privacy distributed learning under chaotic quantum particle swarm optimization. *Computing, 103(3),* 449–472. DOI 10.1007/s00607-020-00853-2.
6. Yao, J., Wang, J., Tsang, I. W., Zhang, Y., Sun, J. et al. (2019). Deep learning from noisy image labels with quality embedding. *IEEE Transactions on Image Processing, 28(4),* 1909–1922. DOI 10.1109/TIP.2018.2877939.
7. Yang, Z. (2021). Variance reduced optimization with implicit gradient transport. *Knowledge-Based Systems, 212,* 106626. DOI 10.1016/j.knosys.2020.106626.
8. Khamaru, K., Pananjady, A., Ruan, F., Wainwright, M. J., Jordan, M. I. (2021). Is temporal difference learning optimal? An instance-dependent analysis. *SIAM Journal on Mathematics of Data Science, 3(4),* 1013–1040. DOI 10.1137/20M1331524.
9. Zhang, C., Xie, T., Yang, K., Ma, H., Xie, Y. et al. (2019). Positioning optimisation based on particle quality prediction in wireless sensor networks. *IET Networks, 8(2),* 107–113. DOI 10.1049/iet-net.2018.5072.
10. Zhao, H., Wu, D., Su, H., Zheng, S., Chen, J. (2021). Gradient-based conditional generative adversarial network for non-uniform blind deblurring via DenseResNet. *Journal of Visual Communication and Image Representation, 74(10),* 102921. DOI 10.1016/j.jvcir.2020.102921.
11. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S. et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research, 17(1),* 1235–1241.
12. Loey, M., Manogaran, G., Taha, M. H. N., Khalifa, N. E. M. (2021). Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. *Sustainable Cities and Society, 65(3),* 102600. DOI 10.1016/j.scs.2020.102600.

13. Duchi, J. C. (2018). Introductory lectures on stochastic optimization. *The Mathematics of Data, 25,* 99–185. DOI 10.1090/pcms/025.
14. Xie, T., Zhang, C., Zhang, Z., Yang, K. (2018). Utilizing active sensor nodes in smart environments for optimal communication coverage. *IEEE Access, 7,* 11338–11348. DOI 10.1109/ACCESS.2018.2889717.
15. Gower, R. M., Richtárik, P., Bach, F. (2021). Stochastic quasi-gradient methods: Variance reduction via Jacobian sketching. *Mathematical Programming, 188(1),* 135–192. DOI 10.1007/s10107-020-01506-0.
16. Garcia, N., Kawan, C., Yüksel, S. (2021). Ergodicity conditions for controlled stochastic nonlinear systems under information constraints: A volume growth approach. *SIAM Journal on Control and Optimization, 59(1),* 534–560. DOI 10.1137/20M1315920.
17. Metel, M. R., Takeda, A. (2021). Stochastic proximal methods for non-smooth non-convex constrained sparse optimization. *Journal of Machine Learning Research, 22(115),* 1–36.
18. Yang, Z., Wang, C., Zhang, Z., Li, J. (2019). Mini-batch algorithms with online step size. *Knowledge-Based Systems, 165,* 228–240. DOI 10.1016/j.knosys.2018.11.031.
19. Gower, R. M., Richtárik, P., Bach, F. (2021). Stochastic quasi-gradient methods: Variance reduction via Jacobian sketching. *Mathematical Programming, 188(1),* 135–192. DOI 10.1007/s10107-020-01506-0.
20. Zhang, J., Tu, H., Ren, Y., Wan, J., Zhou, L. et al. (2018). An adaptive synchronous parallel strategy for distributed machine learning. *IEEE Access, 6,* 19222–19230. DOI 10.1109/ACCESS.2018.2820899.
21. Vlaski, S., Sayed, A. H. (2021). Distributed learning in non-convex environments—Part II: Polynomial escape from saddle-points. *IEEE Transactions on Signal Processing, 69,* 1257–1270. DOI 10.1109/TSP.2021.3050840.
22. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M. et al. (2012). Large scale distributed deep networks. *Advances in Neural Information Processing Systems, 25,* 1223–1231.
23. Wei, J., Zhang, X., Ji, Z., Li, J., Wei, Z. (2021). Deploying and scaling distributed parallel deep neural networks on the Tianhe-3 prototype system. *Scientific Reports, 11(1),* 1–14. DOI 10.1038/s41598-021-98794-z.
24. Wang, X., Fan, N., Pardalos, P. M. (2017). Stochastic subgradient descent method for large-scale robust chance-constrained support vector machines. *Optimization Letters, 11(5),* 1013–1024. DOI 10.1007/s11590-016-1026-4.
25. Xing, E. P., Ho, Q., Dai, W., Kim, J. K., Wei, J. et al. (2015). Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data, 1(2),* 49–67. DOI 10.1109/TBDATA.2015.2472014.
26. Pu, S., Nedić, A. (2021). Distributed stochastic gradient tracking methods. *Mathematical Programming, 187(1),* 409–457. DOI 10.1007/s10107-020-01487-0.
27. Zhou, Q., Guo, S., Lu, H., Li, L., Guo, M. et al. (2021). A comprehensive inspection of the straggler problem. *Computer, 54(10),* 4–5. DOI 10.1109/MC.2021.3099211.
28. Skoraczyński, G., Dittwald, P., Miasojedow, B., Szymkuć, S., Gajewska, E. P. et al. (2017). Predicting the outcomes of organic reactions via machine learning: Are current descriptors sufficient? *Scientific Reports, 7(1),* 1–9. DOI 10.1038/s41598-017-02303-0.
29. Nguyen, L. M., Scheinberg, K., Takáč, M. (2021). Inexact SARAH algorithm for stochastic optimization. *Optimization Methods and Software, 36(1),* 237–258. DOI 10.1080/10556788.2020.1818081.
30. Lu, H., Freund, R. M. (2021). Generalized stochastic Frank-Wolfe algorithm with stochastic substitute gradient for structured convex optimization. *Mathematical Programming, 187(1),* 317–349. DOI 10.1007/s10107-020-01480-7.
31. Schmidt, M., Le Roux, N., Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming, 162(1–2),* 83–112. DOI 10.1007/s10107-016-1030-6.
32. Konečný, J., Liu, J., Richtárik, P., Takáč, M. (2015). Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing, 10(2),* 242–255. DOI 10.1109/JSTSP.2015.2505682.
33. Xin, R., Khan, U. A., Kar, S. (2021). An improved convergence analysis for decentralized online stochastic non-convex optimization. *IEEE Transactions on Signal Processing, 69,* 1842–1858. DOI 10.1109/TSP.2021.3062553.

34. Guo, Y., Liu, Y., Bakker, E. M., Guo, Y., Lew, M. S. (2018). CNN-RNN: A large-scale hierarchical image classification framework. *Multimedia Tools and Applications, 77(8),* 10251–10271. DOI 10.1007/s11042-017-5443-x.
35. Shetty, A. B., Ail, N. N., Sahana, M., Bhat, V. P. (2021). Recognition of handwritten digits and English texts using MNIST and EMNIST datasets. *International Journal of Research in Engineering, Science and Management, 4(7),* 240–243.