The 4th International Conference on Computing and Data Science (CONF-CDS 2022)
DOI: 10.54254/2755-2721/2/20220622

Research on SGD Algorithm Using Momentum Strategy

Liqi Xue

College of Science, Shanghai University, Shanghai, China, 200444

xuelq_vikki@shu.edu.cn

Abstract. With the continuous development of stochastic gradient descent algorithms, many efficient momentum algorithms have appeared. Stochastic gradient descent(SGD) is one of the classic algorithms in optimization. Its accelerated version, the SGD algorithm with momentum strategy, has been a hot research topic in recent years. Therefore, this paper will analyze and summarize these series of algorithms, starting with the classical momentum algorithm, and introduce some improved versions of the momentum algorithm. Numerical experiments on real problems will also be done to evaluate the performance of these algorithms. It is proved that the addition of momentum and adaptive learning rate effectively improve the performance of these algorithms. In future research, some cutting-edge momentum algorithms and other basic network should be analyzed.

Keywords: momentum algorithms, stochastic gradient descent, sgd, machine learning.

1. Introduction

Gradient descent is widely used in machine learning, such as minimizing loss functions and determining the optimal value of model parameters, etc. However, with the advent of the big data era, some shortcomings, such as low computing efficiency and incompatibility with the computing power of hardware have gradually appeared. Large-scale datasets bring challenges to the iteration speed and computing power of GD. In 1951, the stochastic approximation[1] proposed by Robbins and Monro gave an idea for estimating the gradient from part of the training samples. Estimating the gradient on minibatch datasets greatly reduces the computational cost of the algorithm. Thus, the stochastic gradient descent method (SGD) was born.

Compared with GD, SGD has the advantages of simple calculation and a fast convergence rate. Despite the fact that the mini-batch gradient estimation has noise, the step length needs to decrease with the increase of iteration times to ensure the convergence of the algorithm. This will impact the convergence speed. Therefore, a series of acceleration algorithms, such as the momentum algorithm and the Nesterov algorithm came into being. Besides, since the performance of SGD is also affected by the ill-conditioned Hessian matrix of the cost function, the iterative results fluctuate greatly, which is not conducive to getting the optimal value of parameters. Because of these issues, the addition of momentum can effectively avoid the influence of the ill-conditioned Hessian matrix and variance caused by stochastic gradient estimation on the descent speed and efficiency [2].

In recent years, research on stochastic gradient descent algorithms has become a popular direction in machine learning. Many scholars have improved the original algorithms, promoted convergence, and increased the application scenarios of the algorithms. In terms of algorithm evaluation and comparison,

DOI: 10.54254/2755-2721/2/20220622

Shi, Wang, Shang and Zhang[3] summarized and compared various stochastic gradient descent algorithms, but there are still large gaps in related review articles. There are even no scholars that have conducted a comprehensive evaluation and analysis of the momentum algorithms' development in the last few years. This paper is based on the above background. The rest of the paper is organized as follows: the second part will introduce some fundamental knowledge; the third part will introduce two traditional stochastic gradient descent algorithms using momentum strategies; the main content of the fourth part is the adaptive learning rate algorithms with momentum; the fifth part will evaluate the performance of these algorithms with experiments on classical problems; the last part will summarize and prospect the content of the paper. This paper summarizes the current research results in this field. It provides help for the selection of optimization algorithms for machine learning problems and for future research of momentum algorithms.

2. Problem Description and SGD Algorithm

In machine learning problems, the value of the cost function $J(w)$ is always need to be minimized, where w is the parameters that need to be optimized. During the training period, the cost function is represented by the expectation of the loss function L on training set:

$$J(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x_i}; \boldsymbol{w}), y_i), \tag{1}$$

$$i = 1, \dots, N, \boldsymbol{x_i} \in \mathbb{R}^n, y_i \in R, \boldsymbol{w} \in \mathbb{R}^d$$

In E.q. (1), N is the total number of samples in training set, and x_i denotes the input. y_i, $f(x_i; w)$ are the real output and the predicted output respectively.

SGD (Stochastic gradient descent) is a widely used optimization algorithm in machine learning. It uses the small batches $\{x^{(1)}, \dots, x^{(m)}\}$ that are randomly selected on training set to estimate the real gradient of $J(w)$ on training set. Therefore, the descent direction E.q. (2) and the kth parameter updating function E.q. (3) can be given:

$$\boldsymbol{g} = \frac{1}{m}\sum_{i=1}^{N} \nabla_{\boldsymbol{w}} L(f(\boldsymbol{x_i}; \boldsymbol{w}), y_i) \tag{2}$$

$$\boldsymbol{w}^k \leftarrow \boldsymbol{w}^{k-1} - \epsilon_k \boldsymbol{g}_k \tag{3}$$

In E.q. (3), g_k is the gradient estimation calculated by E.q. (2), and $\epsilon_k \in \mathbb{R}^+$ is the learning rate of the kth iteration.

3. Momentum Algorithms

3.1. CM (Classical Momentum)

The Classical momentum(CM or HB, heavy ball) algorithm[4] was first proposed by Polyak in 1962. It adds the impact of the previous gradient in the descent direction of SGD. Its gradient estimation is the same as SGD, which is E.q. (2). If v denotes the velocity, i.e., the variation of parameters, the iterative formula of v is:

$$\boldsymbol{v}_k = \alpha \boldsymbol{v}_{k-1} - \epsilon \boldsymbol{g}_k, \tag{4}$$

where $\alpha \in [0,1)$, $\epsilon \in \mathbb{R}^+$are the hyper-parameters. Their values generally change as the number of iterations increases. In addition, the effect of the previous gradient $g_1, \dots g_k$ on v is decreasing during the iteration. With the definition of v, the kth parameter updating function is shown:

$$\boldsymbol{w}^k \leftarrow \boldsymbol{w}^{k-1} + \boldsymbol{v}_k \tag{5}$$

DOI: 10.54254/2755-2721/2/20220622

While proposing the CM algorithm, Polyak also pointed out that the method can accelerate convergence to local minimum values[2]. For large-scale data sets, its number of iterations required for convergence is significantly reduced compared with SGD.

3.2. Nesterov's Accelerated Gradient

Proposed by Nesterov in 1983, the NAG(Nesterov's accelerated gradient) algorithm[5] is similar to CM. While calculating the gradient, NAG considers the impact of present velocity v. During the kth iteration, NAG temporarily updates w by present velocity v_k at first, using the updated parameters to calculate current gradient:

$$\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}^k + \alpha \boldsymbol{v_k}$$
$$\boldsymbol{g}_k = \frac{1}{m}\sum_{i=1}^{N} \nabla_{\tilde{\boldsymbol{w}}} L(f(\boldsymbol{x_i}; \tilde{\boldsymbol{w}}), y_i)$$

Then NAG uses the same method E.q. (4) and E.q. (5) as CM to update w.

By analyzing the iterative steps, the author find that NAG introduces the effect of the previous gradient when calculates the current gradient. This makes the algorithm more accurate in estimating the value of the cost function $J(w)$, and gets the local minimum point effectively. When the function is smooth, the global rate of convergence is able to achieve $O(\frac{1}{k^2})$, which is faster than SGD.

4. Adaptive momentum algorithms

The value of the learning rate has a great impact on the convergence of the algorithm. Therefore, the adaptive algorithm emerges. It adapts the learning rate from model parameters and monitors different situations in the running process of the algorithm in real time. In this section, some adaptive momentum algorithms will be introduced, including Adam and AMSGrad.

4.1. Adam

Adam is a kind of adaptive momentum algorithm. It calculates first-order and second-order moment estimations and corrects them through a correction term. Among these estimations, first-order moment estimation is used to get the descent direction and reduce large fluctuations during the updating process. Second-order moment estimation calculates the learning rate of different situations.

During the kth iteration, Adam uses E.q. (2) to compute the gradient g_k of current position. Then Adam calculates biased first-order estimation s and second-order moment estimation r through this gradient:

$$\boldsymbol{s}_k = \rho_1 \boldsymbol{s}_{k-1} + (1-\rho_1)\boldsymbol{g}_k = (1-\rho_1)(\boldsymbol{g}_k + \cdots + {\rho_1}^{k-1}\boldsymbol{g}_1), \tag{6a}$$

$$\boldsymbol{r}_k = \rho_2 \boldsymbol{r}_{k-1} + (1-\rho_2)\boldsymbol{g}_k \odot \boldsymbol{g}_k$$
$$= (1-\rho_2)(\boldsymbol{g}_k \odot \boldsymbol{g}_k + \cdots + {\rho_2}^{k-1}\boldsymbol{g}_k \odot \boldsymbol{g}_k), \tag{6b}$$

where $\odot$ represents the Hadamard product, and it has $g_k \odot g_k \in \mathbb{R}^d$. Then these biased estimations are amended:

$$\tilde{\boldsymbol{s}}_k = \frac{\boldsymbol{s}_k}{1-{\rho_1}^k}$$
$$\tilde{\boldsymbol{r}}_k = \frac{\boldsymbol{r}_k}{1-{\rho_2}^k}$$

From the above formulas, the update number of parameters Δw^k and update parameters can be obtained:

DOI: 10.54254/2755-2721/2/20220622

$$\Delta \boldsymbol{w}^k = -\epsilon \frac{\tilde{\boldsymbol{s}}_k}{\sqrt{\tilde{\boldsymbol{r}}_k} + \delta} \tag{7}$$
$$\boldsymbol{w}^k \leftarrow \boldsymbol{w}^{k-1} + \Delta \boldsymbol{w}^k$$

Due to the fact that δ is a scalar quantity, the computational process of Δw^k is performing the E.q.(7) to every 1-D element in Δw^k.

In this iteration process, the initial value of s and r are both 0. At the same time, the initial value w^0 of the parameters w must be given. Moreover, the learning rate ϵ, the parameters in estimation ρ_1, ρ_2 and δ are the hyper-parameters, too. Through analyzing the formulas E.q. (6a) and E.q. (6b), it can be found that the parameters ρ_1 and ρ_2 actually control the disintegration rate of the moment estimates in the first $k-1$ iterations as k increases. Further, because the information of s in the first $k-1$ iterations retain in s_k, Adam can be regarded as an adaptive momentum algorithm. Its iteration process has the same principle as CM, like E.q. (4). Some proofs also confirm the efficiency of Adam. Its individual convergence rate is similar to CM in the worst case, while it has the lower bound of convergence rate[6].

4.2. AMSGrad

Compared with momentum algorithms, Adam performs better in calculating the descent direction and automatically updating the learning rate, which is widely used in various non-convex optimization problems. However, the convergence proof of Adam[7] is flawed since it is proved that Adam does not even convergent when dealing with 1-D convex functions in 2018[8]. Therefore, the improved version of Adam, AMSGrad, was proposed.

The updating formula of AMSGrad is:

$$\boldsymbol{s}_k = \rho_1 \boldsymbol{s}_{k-1} + (1-\rho_1)\boldsymbol{g}_k$$
$$\boldsymbol{r}_k = \rho_2 \boldsymbol{r}_{k-1} + (1-\rho_2)\boldsymbol{g}_k \odot \boldsymbol{g}_k$$

where g_k is the gradient calculated by E.q. (2) of the kth iteration. Then AMSGrad modifies r_k as follows:

$$\tilde{\boldsymbol{r}}_k = \max(\boldsymbol{r}_k, \tilde{\boldsymbol{r}}_{k-1}) \tag{8}$$

Let $R_k = diag(\tilde{r}_k)$, and this paper supposes F is the space that parameter w exists. The temporary update of w is:

$$\widetilde{\boldsymbol{w}}^{k+1} = \boldsymbol{w}^k - \alpha_k \frac{\boldsymbol{s}_k}{\sqrt{\boldsymbol{r}_k}}$$

In the space F of parameter, the updating formula of w is:

$$\boldsymbol{w}^{k+1} \leftarrow min_{\boldsymbol{w}\in \mathrm{F},\ \sqrt{R_k}} \|\boldsymbol{w} - \widetilde{\boldsymbol{w}}^{k+1}\|$$

To explain the corrections made by AMSGrad, the original paper[8] establishes a framework of adaptive momentum algorithms first. During the kth iteration, the algorithm calculates the gradient g_k of current position, and uses the following format to update parameter w:

$$\boldsymbol{s}_k = \boldsymbol{f}_k(\boldsymbol{g}_1, \dots, \boldsymbol{g}_k)$$
$$\boldsymbol{r}_k = \boldsymbol{h}_k(\boldsymbol{g}_1, \dots, \boldsymbol{g}_k)$$

DOI: 10.54254/2755-2721/2/20220622

where F is the space of parameters, $f_k: \mathrm{F}^k \to \mathbb{R}^d$, $d_k: \mathrm{F}^k \to \mathbb{R}^{d+}$. Using these two parameters to temporarily update w:

$$\widetilde{\boldsymbol{w}}^{k+1} = \boldsymbol{w}^k - \alpha_k \frac{\boldsymbol{s}_k}{\sqrt{\boldsymbol{r}_k}}$$

Finally, w is updated by $\widetilde{w}^{k+1}$:

$$\boldsymbol{w}^{k+1} \leftarrow min_{\boldsymbol{w}\in \mathrm{F},\ \sqrt{\boldsymbol{r}_k}} \|\boldsymbol{w} - \widetilde{\boldsymbol{w}}^{k+1}\|$$

Through the framework, it can be found that the Adam actually sets f_k, h_k as follows and lets $\alpha_k \equiv \alpha$:

$$\boldsymbol{f}_k = (1-\rho_1)(\boldsymbol{g}_k + \cdots + {\rho_1}^{k-1}\boldsymbol{g}_1)$$
$$\boldsymbol{h}_k = (1-\rho_2)(\boldsymbol{g}_k \odot \boldsymbol{g}_k + \cdots + {\rho_2}^{k-1}\boldsymbol{g}_k \odot \boldsymbol{g}_k)$$

To compare the correctness of AMSGrad, Γ is assumed as the evaluation quantity of the reciprocal change of the step length, judging the stability of the algorithm in the later learning iterations. Γ has the format:

$$\Gamma_{k+1} = \Delta\frac{\sqrt{\boldsymbol{r}_k}}{\alpha_k} = \frac{\sqrt{\boldsymbol{r}_{k+1}}}{\alpha_{k+1}} - \frac{\sqrt{\boldsymbol{r}_k}}{\alpha_k}$$

During the descent of Adam, Γ sometimes will be positive and other times will be negative. This violates the requirement in the proof of Adam's convergence that Γ must maintain the property of semipositive definite. Therefore, AMSGrad adds the amendment E.q.(8) in the second-order moment estimation calculated by Adam, ensuring the non-negativity of Γ. Theoretically, the AMSGrad algorithm makes up for the shortcomings of Adam. However, further convergence proof and numerical experiments are needed to prove this result.

5. Numerical Experiments

In this section, the above momentum algorithms will be used to optimize some real problems in machine learning. Through analyzing the results, the efficiency of these algorithms can be compared.

5.1. Experiments for Momentum Algorithms

To validate the efficiency and availability of the momentum algorithms, these 3 algorithms are applied to the fundamental machine learning problems: logistic regression, ridge regression, and LASSO. The format of the problems is as follows:

$$\text{Logistic regression: } \min_x \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-y_i x_i^T \boldsymbol{w})) + \lambda\|\boldsymbol{w}\|_2^2$$

$$\text{Ridge regression: } \min_x \frac{1}{2N}\sum_{i=1}^{N} (x_i^T \boldsymbol{w} - y_i)^2 + \lambda\|\boldsymbol{w}\|_2^2$$

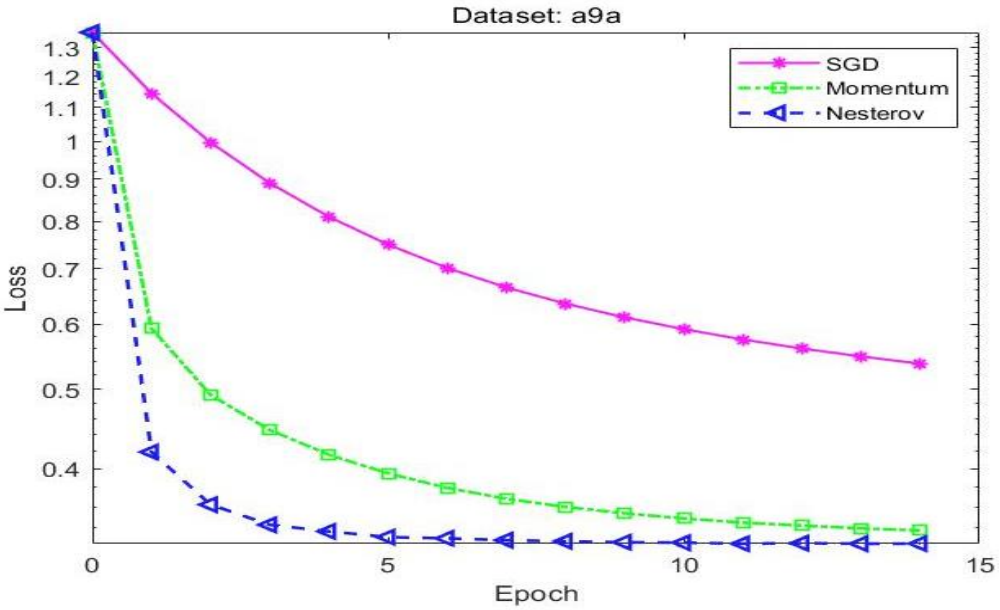
$$\text{LASSO: } \min_x \frac{1}{2N}\sum_{i=1}^{N} (x_i^T \boldsymbol{w} - y_i)^2 + \lambda\|\boldsymbol{w}\|_1$$

where (x_i, y_i) is the data that need to be classified, i.e., x_i is the characteristics and y_i is the label of data point. N is the number of points in the whole dataset. λ is the regularization coefficient to prevent overfitting, which depends on the specific problem. Four datasets are chosen based on each problem. All of them are from the UCI Machine Learning Repository [9], and the details are shown in the following table:

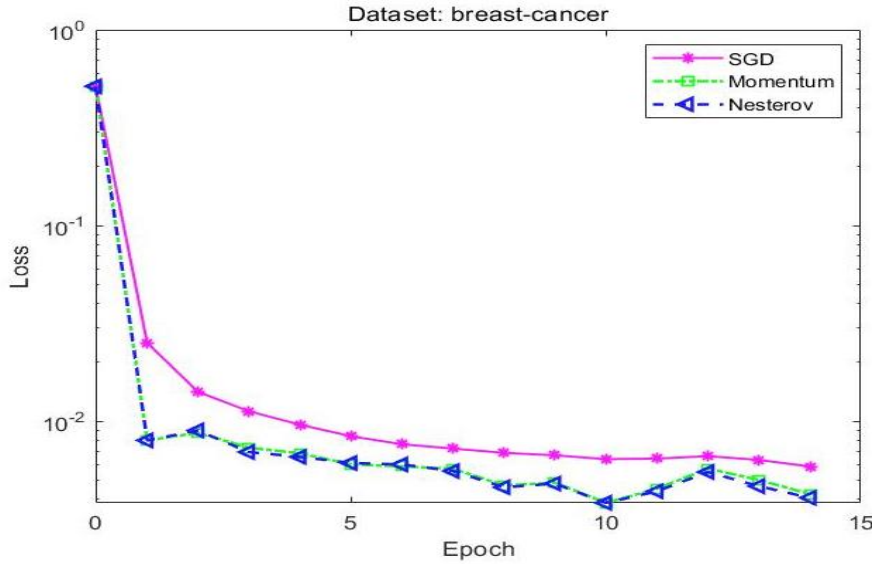
DOI: 10.54254/2755-2721/2/20220622

Table 1. The details of datasets.

Data set	Data points (N)	Features	Model of problem	The value of λ
a9a	16281	122	Logistic regression	1×10^{-2}
Breast cancer	683	10	Logistic regression	1×10^{-3}
Ionosphere	351	34	LASSO	1×10^{-5}
Heart	270	13	Ridge regression	1×10^{-5}

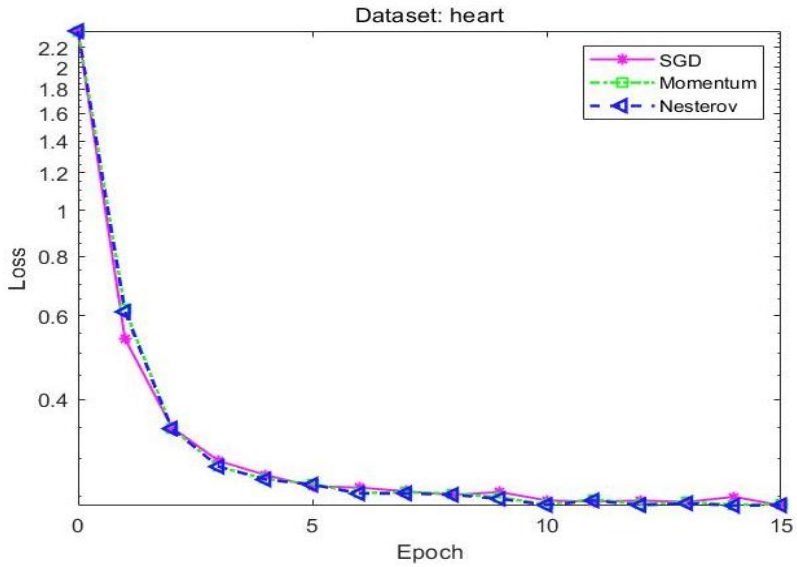


(a)

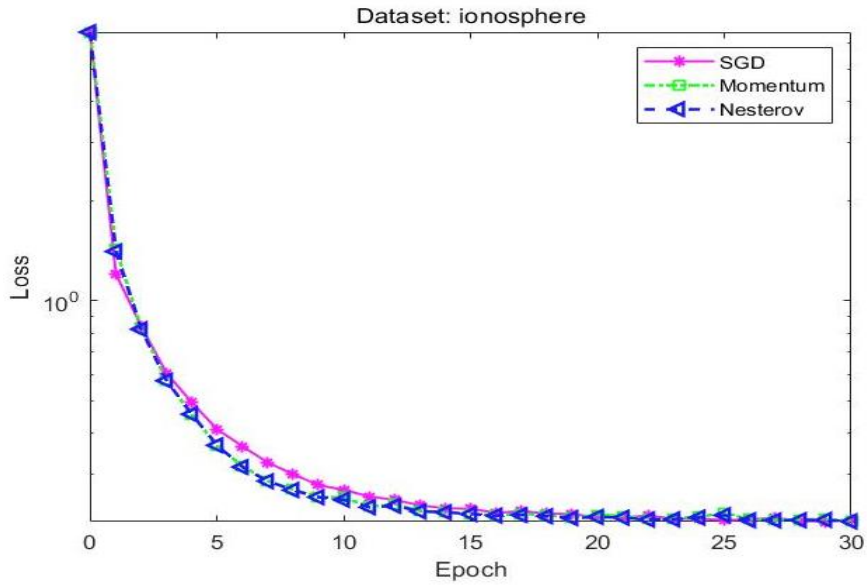


(b)

DOI: 10.54254/2755-2721/2/20220622



(c)



(d)

Figure 1. Performance analysis of momentum algorithms.

MATLAB and C language is used for mixed programming in section 5.1. Grid search is applied to obtain the most suitable learning rate for each momentum algorithm. The initial values of parameter w are set to 0, and the batch size m is set to 10. The results are shown in figure 1.

Through analyzing the results, the author finds that the efficiency of SGD is worse than CM and Nesterov when dealing with logistic regression problems, especially on the a9a dataset. The convergence rate of CM and Nesterov is similar, and Nesterov is slightly faster than CM on the a9a dataset. However, when dealing with ridge regression and LASSO problems, the convergence rate of these 3 algorithms is almost the same. SGD even reduces more function value in the first epoch. This slight difference may be caused by the small scale of the dataset. Moreover, it can be concluded that CM and Nesterov perform better than SGD in the big dataset. The addition of momentum makes them converge faster and approximate the optimal point with fewer iterations.

DOI: 10.54254/2755-2721/2/20220622

4.3. Experiments for Adaptive Momentum Algorithms

The adaptive momentum algorithms are applied to the optimization part of the loss function in mainstream neural networks, analyzing the performance of each algorithm. To compare the modifications made by adaptive momentum algorithms, momentum algorithms are also added to the optimizers.

This paper compares the efficiency of algorithms in the VGG19 neural network model. VGG 19 is a deep convolutional neural network that contains 16 convolution layers and 3 fully connected layers. It uses smaller convolution layers to guarantee the depth of the network and get better performance. In the experiment, VGG19 is used to classify the CIFAR-100 dataset, which contains 60000 pictures that can be sorted into 20 major categories and 100 subcategories. In each subcategory, 50 pictures are randomly chosen as the training set. The size of each image is $32 \times 32 \times 3$. Some data augmentation techniques are also applied in the training process to improve the robustness of the whole network.

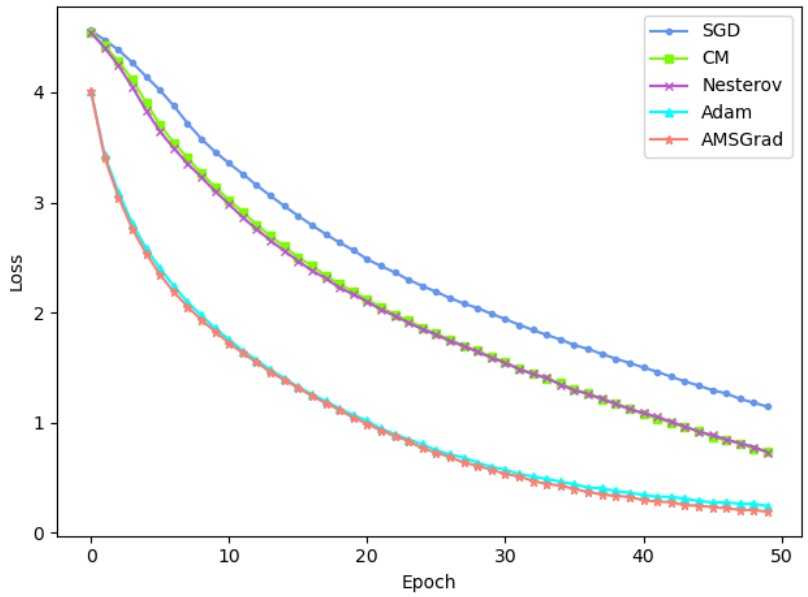
In this section, the experiment is programmed in Python 3.9 and based on the NVIDIA RTX 2080 Ti. When training the network, the cross-entropy loss is used as the loss function, and the batch size m is set to 128. In addition, the value of learning rate depends on specific optimizers. The experimental results of the training phase are shown in the following picture.

Figure2(a) shows the change of loss function values during the iterations, and Figure2(b) shows the time cost of training in each epoch. It can be found that the network that uses SGD has the biggest loss, and the loss function also has the slowest rate of descent. This shows that the addition of momentum accelerates the convergence of iterations to a certain extent. In addition, the curves of CM and Nesterov and the curves of Adam and AMSGrad are almost the same, which means that both algorithms have an almost similar convergence rate in this case. In this experiment, the corrections made by AMSGrad are minimal in promoting the algorithm's efficiency. Furthermore, the decline of loss function value is relatively gentle for momentum algorithms, whereas the rate of decline is fast in the first 10 epochs and then flattens out for adaptive momentum algorithms. The latter is also more efficient than the former, meaning that the addition of the moment estimation idea effectively improves the performance of the algorithm. Therefore, the adaptive algorithms have better performance than general momentum algorithms. Comparing the training time, the single-epoch training time of these algorithms is mainly concentrated within 50s-60s. The time cost of Adam fluctuates greatly, which may also be caused by the chance of an experiment. From this result, AMSGrad has made some improvements to the instability of Adam. SGD also has less training time for each epoch compared to adaptive algorithms. This situation shows that the addition of moment estimation increases the computational complexity at the same time.

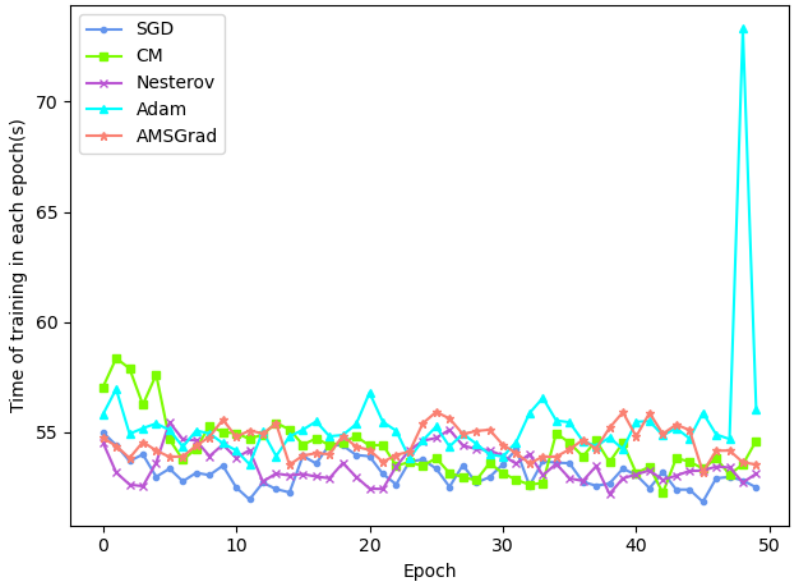
For future research, since the stochastic gradient descent algorithms only use the information in the first-order gradient, some second-order approximation methods can be added, such as the Newton method and quasi-Newton method, to calculate the cost function and improve the processing speed. In addition, for some functions with poor properties, such as non-smooth and non-convex functions, the iteration will be affected by the saddle points when using these algorithms. The application of these kinds of functions is also a worthwhile direction for future research. In terms of adaptive momentum algorithms, the convergence proof of Adam and AMSGrad still needs to be improved. At present, some scholars believe that the AMSGrad algorithm only performs well on some datasets, but does not solve the shortcomings of Adam[10]. Some other revised versions of Adam, such as AdamW[11], NAdam[12], etc., have also been proposed, but the performance of these algorithms has not been fully

DOI: 10.54254/2755-2721/2/20220622

verified. There is no authoritative mathematical proof of their convergence, which is also one of the key points worthy of attention.



(a)



(b)

Figure 2. Performance analysis of algorithms on neural network.

6. Conclusion

This paper starts with CM and the Nesterov algorithm, then introduce two adaptive momentum algorithms: Adam and AMSGrad. These adaptive algorithms can automatically calculate the learning rate according to the current point position and the iteration times, which avoids the oscillation caused by the improper value of step length. This enables the algorithm to efficiently converge to the minimum point and obtains the optimal value for the parameters. Numerical experiments have proved that the

DOI: 10.54254/2755-2721/2/20220622

addition of momentum and adaptive learning rate effectively improve the efficiency of iteration. Due to the fast update speed of related algorithms, this paper dose not consider some cutting-edge algorithms. Moreover, the experiment does not add some popular neural network frameworks. In future research, these new momentum algorithms and basic network should be analyzed.

References

[1] Robbins, H. , & Monro, S. . A stochastic approximation method. Annals of Mathematical Statistics, 22(3), 400-407. (1951).
[2] Goodfellow, I., Bengio, Y., & Courville, A.Deep learning. Cambridge, MA: MIT press. (2016).
[3] Jiarong, S., Dan, W., Fanhua, S. & Heyu, Z. Research progress of stochastic gradient descent algorithm. Acta Automatica Sinica (09), 2103-2119. (2021).
[4] Polyak, B. T. Some methods of speeding up the convergence of iteration methods. Ussr computational mathematics and mathematical physics, 4(5), 1-17. (1964).
[5] Nesterov, Y. . A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Mathematics Doklady. (1983).
[6] Jianzhi, H., Chengcheng, D., Wei, T. & Qing, T. Optimal individual convergence rate of Adam type algorithm in non-smooth convex case. CAAI Transactions on Intelligent Systems (06), 1140-1146. (2020).
[7] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. doi: https://arxiv.org/abs/1412.6980.
[8] Reddi, S. J., Kale, S., & Kumar, S. (2019). On the convergence of adam and beyond. doi: https://arxiv.org/abs/1904.09237.
[9] UCI Machine Learning Repository. https://archive-beta.ics.uci.edu/. 2022.
[10] Tran, P. T.On the convergence proof of amsgrad and a new version. IEEE Access, 7, 61706-61716. (2019).
[11] Loshchilov, I., & Hutter, F. Fixing weight decay regularization in adam. doi: https://openreview.net/forum?id=rk6qdGgCZ (2018).
[12] Dozat, T. Incorporating nesterov momentum into adam. doi: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ (2016).