An improved algorithm for radar adaptive beamforming based on machine learning

To cite this article: Moyu Bai et al 2019 J. Phys.: Conf. Ser. 1325 012114

View the article online for updates and enhancements.
An improved algorithm for radar adaptive beamforming based on machine learning

Moyu Bai*, Hao Liu, Haochuan Chen, Shengming Gu, Zhenhua Zhang
Beijing Research Institute of Telemetry, Beijing 100076, China
'E-mail: Moyu_Bai@hotmail.com

Abstract. In the field of radar digital signal processing, adaptive beamforming is a widely used technique for suppressing interference and noise. The Least Mean Square Algorithm (LMS) is a simple and easy algorithm for adaptive digital beamforming. However, it has the disadvantage of not achieving a balance between convergence speed and stability. In order to improve the performance of adaptive beamforming, this paper firstly reviews the classical LMS algorithm and then the machine learning optimization algorithm. Improvement effects of the three machine learning methods on the LMS algorithm are analyzed. The results show that the improved LMS algorithm based on AdaGrad exhibits the best performance. The algorithm can independently adjust the adaptive learning rate of different parameter components, making the iterative process of the adaptive beamforming more stable, efficient, and suitable for both theoretical research and engineering practice.

1. Introduction
Adaptive digital beamforming is an important branch of digital signal processing. It is an important technology in array signal processing and smart antenna systems, and is widely used in radar, communication and medical fields, which can significantly improve system performance, suppress interference and noise, increase system capacity, and save power. Therefore, adaptive digital beamforming technology has always been a hot research direction for scholars.

For the radar field, adaptive beamforming technology has very important research value in terms of the current practical environment. One of its main functions is to intelligently adjust the antenna beam according to the change of the electromagnetic environment to achieve the purpose of resisting one or more side lobe interferences while detecting the target. Traditional adaptive beamforming algorithms are roughly divided into three categories: (1) adaptive beamforming algorithm with reference to spatial domain; (2) blind adaptive beamforming algorithm related to DOA estimation; (3) Time-referenced adaptive beamforming algorithm. Based on above algorithms with good stability, neither the direction of the interference signal nor calibration is required. However, such an approach would sacrifice spectrum utilization and have very precise synchronization requirements for the signal, which would be more difficult to use. The LMS algorithm is one of the most common algorithms under the minimum mean square error (MMSE) criterion with reference to time. It is representative and widely used in engineering.

The LMS algorithm was first proposed by Widrow and Hoff in 1960. Its core essence is an algorithm that solves the optimal weight by iteratively. However, due to the fixed step size factor in the iterative process of the algorithm, the convergence speed and stability of the algorithm are mutually constrained, and the application is limited under the condition of limited computing resources and high...
precision requirements. Later, the NLMS algorithm proposed by Zeng Zhaohua et al. [1] in 2003 overcomes the contradiction between the convergence speed and steady-state error caused by the fixed step size in the LMS algorithm, but its step size is affected by signal noise. Yan Jingfan et al. [2] proposed a variable step size LMS algorithm based on sigmoid function in 1996. The algorithm can obtain faster convergence speed and smaller steady-state error, but the algorithm is more complicated, the calculation amount is large, and there is a large step adjustment when the error is close to 0, which is not conducive to the stability of the algorithm. Yang Yi et al. [3] proposed an exponential factor variable step size algorithm in 2012. The principle of the algorithm is similar, and the faster convergence speed can be obtained, but the algorithm is more complicated. In view of the above shortcomings, this paper proposes an improved LMS algorithm based on machine learning.

2. Fixed step size LMS algorithm

The root of the adaptive beamforming algorithm is to generate weighting values in adaptive signal processing. The weighted value is then multiplied by the steering vector of the antenna to obtain the pattern of the array. In some classical adaptive beamforming algorithms, the weighting value is obtained by finding the minimum value of the loss function. The smaller the value of the loss function is, the better the quality of the output signal by signal processing is. Therefore, when the loss function reaches the minimum value, the array output is the best.

The essence of the LMS algorithm is to construct a loss function based on the MMSE criterion to minimize the loss function. Assuming that there are N array elements in a one-dimensional line array, the expected signal at the moment is \( d(t) \), the data received by the signal processor is \( X(t) \), the weight value of the \( i \) array element is \( W_i \). According to the MMSE guidelines, the goal is to minimize the mean square value of \( y(t) - d_i(t) \), where \( y(t) = W_i^H x(t) - d_i(t) \). Then the loss function is

\[
J(W_i) = E[(y(t)^2 - d_i(t)^2)]
\]

The loss function is the mathematical expectation of the squared error between the output of the \( i \) array element and the expected signal at time \( t \), which is simplified.

\[
J(W_i) = W_i^H E[x(t)x^H(t)] - E[d_i(t)x^H(t)]W_i - W_i^H E[x(t)d_i^*(t)] + E[d_i(t)d_i^*(t)]
\]

(2)

Derived from it, get

\[
\frac{\partial}{\partial W_i} J(W_i) = 2E[x(t)x^H(t)]W_i - 2E[x(t)d_i^*(t)]
\]

(3)

\[
= 2R_x W_i - 2r_{xd}
\]

Where \( R_x = E[x(t)x^H(t)] \) is the autocorrelation matrix of \( x(t) \). \( r_{xd} \) is the cross-correlation between input \( x(t) \) and desired signal \( d_i(t) \). To minimize the loss function, let

\[
\frac{\partial}{\partial W_i} J(W_i) = 0
\]

(4)

Get \( W_i = R_x^{-1} r_{xd} \).

Since the optimal weight is required to get the inverse of the matrix, this is a situation that is very unwilling to see in the calculation, because once the amount of data is too large, it is very difficult to find the inverse of the matrix. Therefore, the conversion idea uses an iterative method to find the optimal value, and the expression of the weight vector update is

\[
W_i(t + 1) = W_i(t) - \frac{1}{\mu} \nabla
\]

(5)

Where \( \mu \) is the step factor used to determine the convergence speed of the algorithm. When the step factor \( \mu \) is constant, there is

\[
\nabla = R_x W_i(t) - r_{xd} = E[x(t)x^H(t)]W_i(t) - E[x(t)d_i^*(t)]
\]

(6)
Expressed by the corresponding instantaneous value, the estimate of the gradient at time $t$ is

$$\hat{\nabla}(t) = x(t)[x^H(t)W_i(t) - d_i^H(t)] = x(t)e_i(t)$$  \hspace{1cm} (7)

Where $e_i(t) = x^H(t)W_i(t) - d_i^H(t)$ represents the instantaneous error of the output and the $i$ response $d_i(t)$. From the mathematical derivation, it is easy to know that $\hat{\nabla}(t)$ is an unbiased estimate of $\nabla$. By the above derivation, the LMS adaptive beamforming algorithm which can obtain the fixed step size is

$$W_{i}(t + 1) = W_{i}(t) - \mu x(t)e_i(t)$$  \hspace{1cm} (8)

The value of $\mu$ is related to the convergence speed. The larger the $\mu$ value is, the smaller the number of iterations is, but it may lead to the inability to converge. The smaller the $\mu$ value is, the more the number of iterations is, but it can guarantee convergence to the optimal value. Since the convergence speed and stability of the algorithm cannot be optimal together, the selection of the $\mu$ value in the algorithm is very important. So this paper proposes a variable step size improvement LMS algorithm based on machine learning.

3. The improved radar adaptive beamforming algorithm based on machine learning

The LMS algorithm is essentially one of the generalized linear models [4]. It assumes that the output error satisfies the normal distribution and estimates the error using the maximum likelihood method, which has the disadvantage of not meeting the convergence speed and convergence. Introducing machine learning theory, improving LMS algorithm, optimizing learning rate, and balancing learning speed and convergence is an important means to achieve high convergence speed and high stability. Several machine learning optimization algorithms commonly used at present are as follows

3.1 Stochastic Gradient Descent algorithm (SGD)

This method uses a random extraction method, each time using a sample data to calculate the gradient, each time a gradient value is calculated and multiplied by the step factor, and then added to the vector to be obtained in the previous step, and then updated. The vector to be sought is solved iteratively by analogy [5]. Suppose you use least squares to find the loss function, then there is.

$$J(w) = \frac{1}{2}(d - y_w(x))^2$$  \hspace{1cm} (9)

$J(w)$ is the loss function that needs to be minimized. Then we use the partial derivative for $w$ to find the gradient value.

$$\frac{\partial J(w)}{\partial w} = -(d - y_w(x))x$$  \hspace{1cm} (10)

Update the value of $w$ based on the gradient value

$$w_{j+1} = w_j + \mu(d - y_w(x))x$$  \hspace{1cm} (11)

In the algorithm, the step size $\mu$ is fixed. It can be seen that the LMS algorithm is an SGD algorithm in which the loss function is a mean square value. $\mu$ needs fine tuning to achieve better algorithm iteration speed and convergence results. Therefore, in essence, the LMS algorithm in adaptive beamforming is the SGD algorithm in machine learning. The SGD algorithm has the same performance as the LMS algorithm and does not have any improvement on the LMS algorithm. Therefore, the following simulation analysis will no longer consider the SGD algorithm.
3.2 Momentum algorithm

Fig. 1. Momentum algorithm graph

Figure 1. Momentum algorithm graph

Fig. 1 explains the basic principles of the Momentum algorithm intuitively [6]. Where A is the starting point, calculate the gradient $\nabla a$ of point A, and then drop to point B.

$$w_{j+1} = w_j - \alpha \nabla a$$  \hspace{1cm} (11)

Where $\alpha$ is the step size of the gradient $\nabla a$ at point A.

At point B, the gradient of point A is added, and the gradient has an attenuation value $\gamma$, $\gamma < 1$. With this attenuation value, the influence of the early gradient on the current gradient is getting smaller and smaller. If there is no attenuation value, the iteration will eventually oscillate and it will be difficult to converge or even diverge. So the parameter update formula for point B is as follows

$$v_j = \gamma v_{j-1} + \alpha \nabla b$$ \hspace{1cm} (12)

$$w_{j+1} = w_j - v_j$$ \hspace{1cm} (13)

Where $v_{j-1}$ represents the sum of momentum accumulated in all previous steps. Step by step iteration according to the above formula, finally, $w$ can be obtained to minimize the loss function.

$\gamma$ can make the most recent iteration direction occupy a larger proportion in this momentum, which is equivalent to a moving average of momentum. The step size $\alpha$ of the gradient is still a constant. It can be seen from the implementation of the algorithm steps that the Momentum algorithm is suitable for solving the problem where the loss function is a non-convex function, and it has the impulse to make the gradient rush out of the local minimum point to achieve the global best.

3.3 AdaDelta algorithm

The essence of the Adadelta algorithm is to adaptively constrain the step size and use different learning rates for different parameter components [7]. Adadelta performs a moving average of the squared gradient of each parameter component, which is:

$$w_{j+1} = w_j - \frac{\mu}{\sqrt{n_j + \varepsilon}} g_j$$ \hspace{1cm} (14)

$$n_j = \nu^* n_{j-1} + (1 - \nu)^* g_j^2$$ \hspace{1cm} (15)

$g = \frac{\partial J(w)}{\partial w}$ is the gradient of the loss function and $J(w)$ is the loss function. $\nu < 1$, indicating that the step size is less affected by the gradient value of the previous moment. The role of $\varepsilon$ is to avoid the denominator being zero. AdaDelta can prevent the training step from falling to zero during the long training period of the complex model, so that the training stops early, but its disadvantage is that the gradient is smaller close to the extreme point, and the gradient value of the sliding accumulation is smaller. This will increase the learning rate and may lead to non-convergence in a simple convex function optimization problem.

3.4 AdaGrad algorithm

The AdaGrad algorithm is similar to the AdaDelta algorithm in that it sets different learning rates for different parameter components, but unlike AdaDelta, which uses the moving average of the gradient squares, it uses the gradient squared accumulation [8]:
By comparing the AdaGrad algorithm with the LMS algorithm, you can see that the step size $\mu$ is no longer a fixed value, but a value that changes with the iterative update. The algorithm divides the step size by the square root of all the gradients, so that the algorithm learning rate is gradually reduced during the training process. At the same time, it can be seen that the initial squared gradient is small, which can make the step size of the initial iteration large, speed up the gradient descent, and reduce the iteration time. At the same time, in the place where the loss function is relatively flat, the gradient is small, which will increase the step size and speed up the gradient. In the latter part of the iteration, the sum of the squares of the gradient is large, and the rate of decline is reduced, so that it can be better iterated around the most advantageous.

For each optimization algorithm, the hyperparameters of the model must be provided, such as the training step size, the training step convergence factor, and so on. These hyperparameters cannot be dynamically changed in the actual use environment. The adaptability of the same set of hyperparameters to different scale data is also an important evaluation criterion for model robustness. The AdaGrad algorithm is precisely because of its adaptability, it can cope with a wider range of data scale changes. And the detailed test simulation will be given in the following chapters.

Another advantage of the AdaGrad algorithm is that different components of the parameter vector to be optimized can have different training steps according to the cumulative gradient, and the falling curve can be smoothed and the convergence speed can be improved [9]. The following figure is an iterative diagram of the LMS algorithm and the AdaGrad algorithm in the ascending line. It can be seen that the LMS algorithm has the same learning rate of different parameter components due to the fixed step size, resulting in “oscillation drop” in the iterative process. On the contrary, because the different components of the AdaGrad algorithm have different adaptive learning rates, the AdaGrad algorithm can make the loss function decline process more gradual.

$$w_{j+1} = w_j - \frac{\mu}{\sqrt{\sum_{i=0}^{j} (g_i)^2 + \epsilon}} g_j$$

(16)

For each optimization algorithm, the hyperparameters of the model must be provided, such as the training step size, the training step convergence factor, and so on. These hyperparameters cannot be dynamically changed in the actual use environment. The adaptability of the same set of hyperparameters to different scale data is also an important evaluation criterion for model robustness. The AdaGrad algorithm is precisely because of its adaptability, it can cope with a wider range of data scale changes. And the detailed test simulation will be given in the following chapters.

Another advantage of the AdaGrad algorithm is that different components of the parameter vector to be optimized can have different training steps according to the cumulative gradient, and the falling curve can be smoothed and the convergence speed can be improved [9]. The following figure is an iterative diagram of the LMS algorithm and the AdaGrad algorithm in the ascending line. It can be seen that the LMS algorithm has the same learning rate of different parameter components due to the fixed step size, resulting in “oscillation drop” in the iterative process. On the contrary, because the different components of the AdaGrad algorithm have different adaptive learning rates, the AdaGrad algorithm can make the loss function decline process more gradual.

Figure 2. Iterative diagram of the LMS algorithm and AdaGrad algorithm

The following figure shows the flow chart of the improved LMS algorithm based on AdaGrad.
6

Initialize weight

Input echo signal and desired signal

Calculate the actual output signal

Calculate the gradient

Update weight

Calculate the learning rate

Figure 3. Flow chart of improved the LMS algorithm based on AdaGrad

The LMS algorithm is not optimized using the Momentum method. The reason is that the loss function of the LMS algorithm is a convex function, and the Momentum algorithm is suitable for optimizing the algorithm whose loss function is a non-convex function. The proof that the loss function of the LMS algorithm is a convex function is as follows.

Obtain its Hessian matrix about $W$ according to (1).

$$
\begin{bmatrix}
w_1^2 & w_1 w_2 & \cdots & w_1 w_N \\
w_2 w_1 & w_2^2 & \cdots & w_2 w_N \\
\vdots & \vdots & \ddots & \vdots \\
w_N w_1 & w_N w_2 & \cdots & w_N^2
\end{bmatrix}
$$

(17)

The Hessian matrix is semi-definite, so the loss function of the LMS algorithm is a convex function [9].

In summary, in order to improve the performance of the LMS algorithm, four optimization algorithms in machine learning are considered. For the SGD algorithm, since it is essentially the same as the performance of the LMS algorithm, this optimization method is not considered. For the Momentum algorithm, since the loss function of the LMS algorithm is a convex function, the optimization algorithm does not need to consider jumping out of the local minimum point, and the Momentum algorithm is suitable for optimizing the algorithm whose loss function is a non-convex function. Therefore, the optimization of the LMS algorithm using the Momentum algorithm is no longer considered. For the AdaGrad algorithm and the AdaDelta algorithm, the Adagrad algorithm can effectively reduce the training step size with the training compared to the AdaDelta algorithm. For the convex optimization model of LMS, it has better convergence. The optimization of LMS algorithm by AdaDelta algorithm may result in the algorithm not converging, and the improved LMS algorithm based on AdaGrad can improve the convergence speed and stability of LMS algorithm. Therefore, for the comprehensive consideration of these four optimization algorithms, the AdaGrad method is used to optimize the LMS algorithm, which can improve the convergence speed and stability of the algorithm.

The following table compares the performance of the LMS algorithm with the four optimization algorithms. It can be seen that the improved algorithm based on AdaGrad performs best.
Table 1. Performance comparison of five algorithms

| Algorithm type | Convex function convergence stability | Algorithm learning speed under convergence condition | Whether adaptive gradient |
|----------------|---------------------------------------|-----------------------------------------------------|--------------------------|
| AdaGrad        | Good                                  | Fast                                                | Yes                      |
| AdaDelta       | General                               | Fast                                                | Yes                      |
| SGD            | Good                                  | Slow                                                | No                       |
| LMS            | Good                                  | Slow                                                | No                       |
| Momentum       | Poor                                  | Slow                                                | No                       |

4. Simulation and analysis
Since the model input, output and parameters are all complex numbers in the radar system, this experiment uses the strategy of splitting the LMS optimization algorithm into real-domain and complex-domain respectively to realize various optimization algorithms. With this splitting method, the original complex value function needs to perform four real multiplications, three additions and subtractions each time. However, if the split is divided into the real number field and the complex number field, only two real multiplication calculations are needed, which improves the operation speed and saves the operation time. In addition, since the SGD algorithm and the LMS algorithm are essentially identical algorithm, the SGD algorithm will not be considered in the following simulation analysis. Without loss of generality, a one-dimensional line array is used as an example for simulation. The antenna array element number is 16, the array element spacing is half wavelength, the signal to noise rate is 20 dB, the interference to noise rate is 10 dB, the incoming wave direction is 0°, and the interference direction is -40° direction. All simulation diagrams below are based on this condition for simulation analysis.

4.1 Adaptive beamforming performance of the LMS algorithm and improved algorithm based on AdaGrad

![Antenna pattern](image)

Figure 4. Simulated antenna patterns of the LMS algorithm and improved algorithm based on AdaGrad

The figure above shows the antenna pattern of the LMS algorithm and the improved LMS algorithm based on AdaGrad. It can be seen that both algorithms can be aligned in the main lobe direction, and the interference direction reaches more than 40 dB, and the improved algorithm based on AdaGrad has a deeper rake than the LMS algorithm in the -40° interference direction. Therefore, it can be seen that the improved algorithm based on AdaGrad has good accuracy.
4.2 Convergence Simulation for the Four Algorithms with Different Learning Rate

Figure 5. The loss of the LMS algorithm varying with iterations under different learning rates

Figure 6. The loss function of Momentum algorithm with iterations under different learning rates

Figure 7. The loss function of AdaDelta algorithm with iterations under different learning rates
Figure 8. The loss function of AdaGrad algorithm with iterations under different learning rates

Fig. 5 is a comparison of the convergence of the LMS algorithm under different step sizes lengths. It can be seen that for the convex function, the LMS algorithm can effectively converge under the condition of reasonable selection of hyper parameters, and only need to iterate 10 times to reduce the mean square error of the sample to a reasonable interval. However, the super-parameter selectable interval is small, and it cannot be effectively converged when the gradient descent step is set to 0.00025 during the simulation.

Fig. 6 is the convergence of Momentum at different learning rates. It can be seen that, similar to the LMS algorithm, the learning rate setting of the Momentum algorithm is also relatively small, but the convergence stability is not as good as the traditional LMS algorithm, which is caused by the gradient disturbance that caused by the momentum. For the loss function of this problem, the gradient disturbance caused by Momentum will have the opposite effect.

As can be seen from Fig. 7, AdaDelta can also converge to an acceptable range in a short period of time, but also requires fine adjustment of hyper parameters, just like the LMS algorithm. During the experiment, the AdaDelta algorithm could not effectively converge when the hyper parameter was set to 0.0003. Since AdaDelta uses weighted moving averages when accumulating gradient squares, the convergence stability of the simple model of convex functions is not as good as that of AdaGrad algorithm.

It can be seen from the Fig. 8 that the AdaGrad algorithm has the convergence stability and convergence speed of the LMS algorithm after the fine tuning, and the range of the hyper parameter is relatively large. When the $\mu$ is set to 10 during the experiment, it can still effectively converge, which proves the excellent robustness of the AdaGrad algorithm.

In summary, the comparison analysis of the above four simulation figures can be obtained. Under the same data range, the improved LMS based on AdaGrad has a broader range of hyper parameters, that is, it has better stability under the same conditions.
4.3 Convergence Rate Simulation for the Four Algorithms under Optimal Hyper parameters

Figure 9. The loss function of LMS algorithm, Momentum algorithm, AdaDelta algorithm and AdaGrad algorithm with the optimal learning rate

Figure 10. The loss function of LMS algorithm and AdaGrad algorithm with the optimal learning rate

It can be seen from the Fig 9, 10 that for the four algorithms, the AdaGrad optimization algorithm has the best performance and the fastest convergence rate under its optimal learning rate. Fig. 10 is a further comparison of the performance of the AdaGrad and LMS algorithms in terms of convergence. It can be seen that even after fine tuning, AdaGrad has an advantage over the LMS algorithm in convergence stability. The reason is that AdaGrad can take different gradient descent steps for different components of the parameter vector. For the iterative process, the component with larger gradient reduces the learning speed, while for the component with smaller cumulative gradient, the learning speed is improved. This makes the "trajectory" of the gradient drop smoother. It can be concluded that the AdaGrad optimization algorithm in the four algorithms has the fastest iteration and the best performance.

4.4 Stability Simulation for the Four Algorithms under Different Data Scales

Under the same set of optimal hyper parameters, the adaptability of the optimization algorithm to different scale data is an effective index to evaluate the robustness of the algorithm. The experimental method adopted in this thesis is to multiply the original data by a scale expansion variable alpha, and then retrain the model to study its convergence and the adaptability of the optimization algorithm. The experimental results are as follows.
Figure 11. The loss function of the LMS algorithm with different data scales under the optimal learning rate

Figure 12. The loss function of the Momentum algorithm with different data scales under the optimal learning rate

Figure 13. The loss function of the AdaDelta algorithm with different data scales under the optimal learning rate
Figure 14. The loss function of the AdaGrad algorithm with different data scales under the optimal learning rate

It can be seen from the above four simulation diagrams that under the fine-tuned LMS, when the data scale is expanded to two times, the LMS algorithm cannot converge. It shows that the convergence of the LMS algorithm is very demanding, and only a small learning rate can be used, which makes the learning speed too slow and reduces the performance of the LMS algorithm. Similarly, it can be seen that the Momentum algorithm and the AdaDelta algorithm cannot converge after the data scale is expanded to twice the original. For the AdaGrad algorithm, even if the data is expanded by 10 times, the original hyperparameter setting can still effectively converge the AdaGrad algorithm. It shows that the AdaGrad algorithm has strong data adaptability and satisfies the application requirements of the LMS algorithm in actual scene. It can converge on most data scales without setting the learning rate to a low level, and achieves a balance between learning speed and convergence.

In order to measure the adaptability of AdaGrad, AdaDelta, Momentum and LMS algorithm, this paper takes the hyperparameter range of the model convergence under the same data set as the evaluation index, and concludes that the larger the hyperparameter range is, the better the robustness of the model is. The figure below shows the robustness of the four algorithms.

5. Conclusion

In the field of radar digital signal processing, adaptive beamforming technology is a valuable research, where the LMS algorithm is a simple and easy to implement algorithm in adaptive digital beamforming. The LMS algorithm essentially uses the traditional SGD algorithm in machine learning. Although the model can be effectively converged by fine-tuning the learning rate, it is quite difficult to balance the convergence and learning rate for the same set of hyperparameters for different scales of data. In this paper, the AdaGrad algorithm is selected in the current machine learning first-order optimization
algorithm by analyzing the optimization function. The algorithm can make the independent learning rate adjustment of different parameter components independently, so that the improved LMS algorithm is more stable and efficient in the iterative process compared with the SGD algorithm and other first-order optimization algorithms. After fixing hyperparameters, it shows better performance compared to the traditional LMS algorithm in the convergence of different data sets, implying that the improved LMS algorithm can be applied to the actual scene. Therefore, the improved LMS algorithm based on AdaGrad has faster convergence speed and better stability, which is quite meaningful in theoretical research and engineering practice.

References

[1] Zeng Zhaohua, Liu Guizhong, Zhao Jianping. LMS and normalized LMS algorithm convergence threshold and step size determination [J]. Journal of Electronics and Information, 2003, 25(11): 1469-1474.

[2] Yan Jingfan, Wei Gang. Variable step size LMS adaptive filtering algorithm based on S-type function [J]. Radio engineering, 1996(4): 44-47.

[3] Yang Yi, Cao Xiangyu, Yang Qun. Normalized variable step size LMS algorithm based on exponential function [J]. computer engineering, 2012, 38(10): 134-136.

[4] Emptage M. An Introduction to Generalized Linear Models [J]. Journal of the Royal Statistical Society, 1992, 34(2):224-225.

[5] Paras. Stochastic Gradient Descent [J]. Optimization, 2014.

[6] Phansalkar V V, Sastry P S. Analysis of the back-propagation algorithm with momentum [J]. IEEE Transactions on Neural Networks, 1994, 5(3):505-506.

[7] Zeiler M D. ADADELTA: An Adaptive Learning Rate Method [J]. Computer Science, 2012.

[8] Wilson A C, Roelofs R, Stern M, et al. The Marginal Value of Adaptive Gradient Methods in Machine Learning [J]. 2017.

[9] Yurii Nesterov. Lectures on Convex Optimization [J]. IEEE Transactions on Automatic Control, 2006, 51(11):1859-1859.