Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement

Jing Li, Ji-hang Cheng, Jing-yuan Shi, and Fei Huang

Aviation University of Air Force
130022, Changchun China
568026388@qq.com

Abstract. The back propagation (BP) neural network algorithm is a multi-layer feedforward network trained according to error back propagation algorithm and is one of the most widely applied neural network models. BP network can be used to learn and store a great deal of mapping relations of input-output model, and no need to disclose in advance the mathematical equation that describes these mapping relations. Its learning rule is to adopt the steepest descent method in which the back propagation is used to regulate the weight value and threshold value of the network to achieve the minimum error sum of square. This paper focuses on the analysis of the characteristics and mathematical theory of BP neural network and also points out the shortcomings of BP algorithm as well as several methods for improvement.

Keywords: Neural network, BP, algorithm.

1 Back Propagation Algorithm (BP Algorithm)

1.1 BP Algorithm

In fact, BP algorithm is a method to monitor learning. It utilizes the methods of mean square error and gradient descent to realize the modification to the connection weight of network. The modification to the connection weight of network is aimed at achieving the minimum error sum of squares. In this algorithm, a little value is given to the connection value of network first, and then, a training sample is selected to calculate gradient of error relative to this sample [1].

1.2 BP Learning Algorithm

The BP learning process can be described as follows:

(1) Forward propagation of operating signal: the input signal is propagated from the input layer, via the hide layer, to the output layer. During the forward propagation of operating signal, the weight value and offset value of the network are maintained constant and the status of each layer of neuron will only exert an effect on that of next layer of neuron. In case that the expected output can not be achieved in the output layer, it can be switched into the back propagation of error signal.

D. Jin and S. Lin (Eds.): Advances in CSIE, Vol. 2, AISC 169, pp. 553–558.
springerlink.com

(2) Back propagation of error signal: the difference between the real output and expect output of the network is defined as the error signal; in the back propagation of error signal, the error signal is propagated from the output end to the input layer in a layer-by-layer manner. During the back propagation of error signal, the weight value of network is regulated by the error feedback. The continuous modification of weight value and offset value is applied to make the real output of network more closer to the expected one [2].

2 Description of BP Algorithm in Mathematics

The ideology guiding the learning rules of BP network is: the modification to the weight value and threshold value of network shall be done along the negative gradient direction reflecting the fastest declining of function.

$$x_{k+1} = x_k - \eta_k g_k$$

In the formula mentioned above, xk represents the matrix of current weight value and threshold value; gk represents the gradient of current function; ηk represents the learning rate. Here, the three-layer BP network is taken as an example to describe the BP algorithm [3] in details.

As for the three-layer BP network, suppose its input node is xi, the node of hide layer is yj, and the node of output layer is zl. The weight value of network between the input node and node of hide layer is wji, and the weight value of network between the nodes of hide layer and output layer is vlj. When the expected value of the output node is tl, f(·) is the active function. The computational formula of the model is expressed as follows:

Forward propagation: output of computer network

Output of the node of hide layer

$$y_j = f(\sum_i w_{ji} x_i - \theta_j) = f(net_j)$$

including

$$net_j = \sum_i w_{ji} x_i - \theta_j$$

Computational output of the output node

$$z_l = f(\sum_j v_{lj} y_j - \theta_l) = f(net_l)$$

$$net_l = \sum_j v_{lj} y_j - \theta_l$$

Including

Error of the output node

$$E = \frac{1}{2}\sum_l (t_l - z_l)^2 = \frac{1}{2}\sum_l (t_l - f(\sum_j v_{lj} y_j - \theta_l))^2$$

$$= \frac{1}{2}\sum_l (t_l - f(\sum_j v_{lj} f(\sum_i w_{ji} x_i - \theta_j) - \theta_l))^2$$

Back propagation: the gradient descent method is adopted to regulate the weight value of all layers, and the learning algorithm of weight value is expressed as follows:

2.1 Modification of Weight Value

1. Derivation of output node by means of error function

$$\frac{\partial E}{\partial v_{lj}} = \sum_{k=1}^{n} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial v_{lj}} = \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial v_{lj}}$$

E is a function containing several z_k, but only one z_l is related with v_{lj} and all the z_k are independent from each other, in this formula,

$$\frac{\partial E}{\partial z_l} = \frac{1}{2} \sum_k [-2(t_k - z_k) \cdot \frac{\partial z_k}{\partial z_l}] = -(t_l - z_l)$$

$$\frac{\partial z_l}{\partial v_{lj}} = \frac{\partial z_l}{\partial net_l} \cdot \frac{\partial net_l}{\partial v_{lj}} = f'(net_l) \cdot y_j$$

In this way,

$$\frac{\partial E}{\partial v_{lj}} = -(t_l - z_l) \cdot f'(net_l) \cdot y_j$$

Suppose the error of input node is

$$\delta_l = (t_l - z_l) \cdot f'(net_l)$$

In this way

$$\frac{\partial E}{\partial v_{lj}} = -\delta_l \cdot y_j$$

2. Deviation of the node of hide layer by error function

$$\frac{\partial E}{\partial w_{ji}} = \sum_l \sum_j \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_{ji}}$$

E is a function containing several z_l; it is targeted at certain w_{ji}, corresponding to one y_j, and related to all z_l, in this formula,

$$\frac{\partial E}{\partial z_l} = \frac{1}{2} \sum_k [-2(t_k - z_k) \cdot \frac{\partial z_k}{\partial z_l}] = -(t_l - z_l)$$

In this way,

$$\frac{\partial E}{\partial w_{ji}} = -\sum_l (t_l - z_l) \cdot f'(net_l) \cdot v_{lj} \cdot f'(net_j) \cdot x_i = -\sum_l \delta_l v_{lj} f'(net_j) \cdot x_i$$

Suppose the error of node of hide layer is

$$\delta'_j = f'(net_j) \cdot \sum_l \delta_l v_{lj}$$

In this way

$$\frac{\partial E}{\partial w_{ji}} = -\delta'_j x_i$$

As the modification of weight Δv_{lj} and Δw_{ji} is in proportion to the error functions and descends along the gradient, the formula showing the modification of weight of hide layer and output layer is expressed as follows:

$$\Delta v_{lj} = -\eta \frac{\partial E}{\partial v_{lj}} = \eta \delta_l y_j$$

In this formula, η represents the learning rate. The formula showing the modification between the input layer and hide layer is expressed as follows:

$$\Delta w_{ji} = -\eta' \frac{\partial E}{\partial w_{ji}} = \eta' \delta'_j x_i$$

$$\delta'_j = f'(net_j) \bullet \sum_l \delta_l v_{lj}$$

In this formula, η' represents the learning rate; $\sum_l \delta_l v_{lj}$ in the node error of hide layer δ'_j expresses that the error δ_l of output node z_l is back propagated through the weight value v_{lj} to the node y_j to become the error of node of the hide layer.

2.2 Modification of Threshold Value

The threshold value θ is also a variation value and it also needs to be modified while the weight value is modified; the theory applied is the same as that used in the modification of weight value.

(1) Derivation of the threshold of output node by error function

$$\frac{\partial E}{\partial \theta_l} = \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial \theta_l}$$

In this formula

$$\frac{\partial z_l}{\partial \theta_l} = \frac{\partial z_l}{\partial net_l} \cdot \frac{\partial net_l}{\partial \theta_l} = f'(net_l) \cdot (-1)$$

In this way

The formula expressing the modification of threshold value is

$$\Delta \theta_l = \eta \frac{\partial E}{\partial \theta_l} = \eta \delta_l$$

Namely $$\theta_l(k+1) = \theta_l(k) + \Delta \theta_l = \theta_l(k) + \eta \delta_l$$

(2) Derivation of the threshold of node of hide layer by error function

$$\frac{\partial E}{\partial \theta_j} = \sum_l \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial y_j} \cdot \frac{\partial y_j}{\partial \theta_j}$$

In this formula

$$\frac{\partial y_j}{\partial \theta_j} = \frac{\partial y_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial \theta_j} = f'(net_j) \cdot (-1) = -f'(net_j)$$

In this way

$$\frac{\partial E}{\partial \theta_j} = \sum_l (t_l - z_l) \cdot f'(net_l) \cdot v_{lj} \cdot f'(net_j) = \sum_l \delta_l v_{lj} f'(net_j) = \delta'_j$$

The formula expressing the modification of threshold value is

$$\Delta \theta_j = \eta' \frac{\partial E}{\partial \theta_j} = \eta' \delta'_j$$

Namely $$\theta_j(k+1) = \theta_j(k) + \Delta \theta_j = \theta_j(k) + \eta' \delta'_j$$

3 Improvement of BP Algorithm

In the application of artificial neural network, the BP network and its varied pattern are adopted in most of the neural network models; however, this does not mean that BP network is perfect and there are still inevitable defects in its algorithm, for example, falling into the minimum part of local part in the process of training, the convergence rate being rather slow, the network tending to have more redundancy and the samples newly added those may affect the samples learned, or others. The researcher has put forward many improved algorithms [4] to solve these defects. Its improved methods can be generally classified as three categories: one is to improve the speed of neural network training; the second is to improve the accuracy of training; and the third is to avoid dropping into the minimum point of local part. Among these methods, the rather typical ones are the additional momentum method and the variable learning rate method [5].

3.1 Additional Momentum Method

The additional momentum method is formed by introducing the momentum coefficient α based on the gradient descent algorithm. The formula that shows the adjustment of weight value and includes the additional momentum coefficient is expressed as follows:

$$\Delta w(t+1) = \alpha \Delta w(t) + \eta(1-\alpha)\frac{\partial E}{\partial w}$$

In this formula, $\Delta w(t+1)$ and $\Delta w(t)$ represent the weight corrections after the $(t+1)^{th}$ and t^{th} iteration; the value of the momentum coefficient α must be selected between 0 and 1, and generally 0.9 is selected. $\frac{\partial E}{\partial w}$ represents the negative gradient of the error sum of squares to weight in the BP algorithm.

The process of networked learning is that of weight modification; in this algorithm, the correction result of last time is used to affect the corrections of this time; when the correction of last time is oversized, the symbol of the second item in this formula will be contrary to that of correction of last time, in order to reduce the correction of this time and lower the oscillation. When the correction of last time is undersized, the symbol of the second item in this formula will be the same with that of correction of last time, in order to amplify the correction of this time and speed up the correction. It is thus clear that the application of additional momentum method always tries to increase the corrections those are in the same direction of gradient. This method has accelerated the convergence rate and reduced to a certain extent the probability of falling into minimal local part.

3.2 Variable Learning Rate Method

The error surface of network varies dramatically according to the variable parameter; the larger learning rate shall be selected for the areas whose error surfaces are very smooth; the smaller learning rate shall be selected for the areas whose error surfaces are very precipitous. The variable learning rate method is used for the self-adaptive

adjustment of learning rate according to the change of error. The formula below shows the adjustment of learning rate:

$$\eta(t+1) = \begin{cases} k_{inc}\eta(t) & E(t+1) < E(t) \\ k_{dec}\eta(t) & E(t+1) > E(t) \\ \eta(t) \end{cases}$$

In this formula, the incremental factor of learning rate $k_{inc}>1$, often 1.05; the reduction factor of learning rate $0<k_{dec}<1$, often 0.7; $E(t+1)$ and $E(t)$ represent the total error sum of squares after the $(t+1)^{th}$ and the t^{th} iteration respectively; η represents the learning rate, and in standard BP algorithm, it is always a constant value between 0 and 1. If $E(t+1)<E(t)$, it represents that the t^{th} iteration is effective, then multiplying the incremental factor to increase the learning rate; if $E(t+1)>E(t)$, it represents the t^{th} iteration is ineffective, then multiplying the reduction factor to reduce the learning rate in order to reduce the ineffective iteration and accelerate the learning rate of network.

3.3 Optimization of Initial Weight

On one side, several local minimal points exist in the error surface of BP network; on the other side, the error gradient descent algorithm is adopted to adjust the weight of network; these two sides have caused the results of training of network easily to fall into the minimal point. Hence, the initial weight of network has exerted an enormous effect on the final result of training of network and it is one of the important factors those affect the possibility of network to achieve certain acceptable accuracy.

The initial weight of network is generally generated at random in certain interval; the training starts with an initial point and reaches gradually to a minimum of error along the slope of error function; that is to say, once the initial value is defined, the convergence direction of network is determined. In case that a bad initial weight is selected, the convergence direction of network can be towards the direction of divergence and this causes the non-convergence of concussion in network. In addition, the convergence rate of network training is also related to the selection of initial weight. As a result of that, it seems to be of great significance in selecting an appropriate initial weight, so as to accelerate the convergence rate of network training and avoid the concussion occurred in the process of studying. The genetic algorithm can be used to optimize the initial weight of neural network and enables the initial weight to jump out of the local extremum, speed up the convergence of BP network and improve the convergence accuracy of network.

References

1. Yan, P., Huang, R.: Artificial Neural Network — Model, Analysis and Application. Anhui Educational Publishing House, Hefei
2. Zhou, K., Kang, Y.: Neural Network Models and MATLAB Simulation Program Design. Tsinghua University Press, Beijing
3. Gao, J.: Artificial Neural Network Theory and Simulation Examples. China Machine Press, Beijing
4. Liu, J.: Intelligent Control. Electronic Industry Press, Beijing
5. Wang, X., Cao, L.: Genetic Algorithm – Theory, Application and Software Implementation. Xi'an Jiaotong University Press