Advances in Intelligent Systems Research, volume 146

2nd International Conference on Artificial Intelligence: Technologies and Applications (ICAITA 2018)

Research on the Solution of BP Neural Network Training Problem

Junyang Tan[1], Dan Xia[1,*], Shiyun Dong[1], Binshi Xu[1] and Ye Li[2]

[1]Academy of Armored Forces Engineering, China

[2]State Grid Xiamen Power Supply Bureau, China

[*]Corresponding author

***Abstract*—BP neural network is a common kind of neural network for recognition and classification, but it is obvious that some difficulties are find during the training process. In this paper, we introduced the basic principle of BP algorithm and propose a parameter adjustment procedure for activation function to speed up convergence and avoid gradient diffusion, and the feasibility of this method is proved by a basic BP neural network with simple structure. We analyzed this parameter's available value through a group of experiments based on simple three-layer BP full connected network. In addition, this activation function parameter adjustment procedure is also used in is suitable for Multilayer BP neural networks.**

Keywords—neural network; gradient diffusion; activation function; convergence speed; classification accuracy

I. Introduction

Artificial neural network (ANN) is a popular method that has been widely used in image and speech recognition in recent years. The network establishes an abstract, digital model that represents the process of information processing by neurons. Simulating the biological neural structure, the network consists of a large number of nodes, representing neurons connected by weights [1]. The network is trained to determine the weights for recognition and classification. Because of its distinctive topological structure, ANN has a strong fault-tolerant ability and computation speed, so it has a strong performance for classification and recognition.

The BP neural network(BPNN), proposed by Paul Werbos in 1974 and described by Rumelhart and McCelland in their paper as a type of multilayer, fully connected feed-forward neural network, is one of the most widely used neural network models. BPNN first determines a reliable procedure for target weight acquisition [2], that is, during the training process, the information is propagated forward and the error is contrary. The neural network learns the connection weights and iterates using the gradient descent algorithm until the iterations reach a preset number or the error becomes less than the target value.

Although BPNN has a modeling principle with a distinct mathematical meaning, good representation of classification, and wide range of applications, it also has some shortcomings and deficiencies. First, the BP algorithm can take a relatively long time to converge to the global minimum of the error plane [3], and sometimes, the process falls into local extremes. Second, for BPNN, the gradient diffusion phenomenon appears during the process of back propagation as the number of layers increases.

In order to improve the convergence speed, avoid gradient diffusion problems and improve network performance, some studies on activation functions and pre-training method were considered[4]. In 1991, Wang L proposed a new approach for the estimation of the number of hidden nodes and a new activation function which may decrease the nodes of the net for XOR problem[5]. Song W R studied the effect of network activation function on the convergence speed, and selected the best activation function for network training[6].Hu Y G proposed a new neural network multi-parameter adjustable activation function and changed the BP process accordingly. Later, it was used in the example to prove its superiority over the original activation function[7]. In 2006, Mei D F introduced an adjustment function that was acted on the activation function to optimize the gradient and its derivatives and smooth the BP process[8]. Until now, there are still many scholars studying the activation function of neural networks. Some new activation functions and new adjustment methods are constantly being proposed. Some of them are based on sigmoid functions for parameter adjustment and optimization improvements[9-12]. In the other hand, Hinton and Salakhutdinov proposed a stacked auto-encoder(SAE) in 2006, which is trained until the output is just equivalent to the input [13], and they used a restricted Boltzmann machine(RBM)to build this deep auto-encoder model [14]. Since then, unsupervised pre-training has been widely used in BP networks. In 2015, an supervised pre-training method was proposed by Deng L and YU D[15,16].

In this paper, we propose a parameter adjusting procedure for the activation function to make training faster and more effective. Several group of experiments are set up to analyze and determine the suitable parameters.

II. A Parameter Adjusting Procedure and Experiments

This section mainly concerns the problem of parameter adjustment in the process of fully connected network training. Previous research showed that fully connected networks trained by a BP algorithm take a relatively long time to reach convergence and may encounter a problem with gradient diffusion. Therefore, the parameter adjusting procedure is proposed as follows.

A. Fully Connected Neural Network

A fully connected network is the basic model of a BP neural network, and there are several hidden layers between input and output layer. Each layer have many nodes and a activation function.

The activation function f is defined as sigmoid, as shown in Figure 1, and can be a tangent function or a linear function.

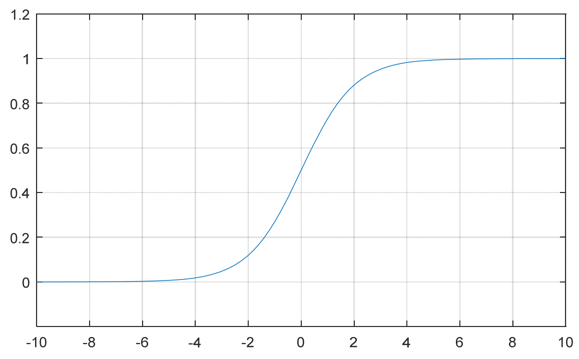


FIGURE I. THE CURVE OF A SIGMOID FUNCTION

B. *Feed-forward Network*

Based on the structure of a fully connected neural network, every input of a node is multiplied by its corresponding weights, and the sum of the results with the bias is considered to be the total input S_j of this node, defined as:

$$S_j = \sum_{i=0}^{k} w_{ij} x_i + b_j \tag{1}$$

$$O_j = f(S_j) \tag{2}$$

where j and k are the number of nodes in layer L and L-1, respectively, so that w_{ij} is the weight connected between the ith node in layer L-1 and jth node in Layer L. The bias and activation of the jth node in layer L are b_j and O_j, respectively, and x_j is the activation of the node in layer L-1. In (2), the activation function f is defined as sigmoid, as shown in Figure 1, and can be a tangent function or a linear function. Following (1) and (2),the activation of nodes in the front layer is used as the input to the nodes in the next layer to perform the iteration.

C. *Back-propagation Network*

Weights and biases are updated and adjusted by the error gradient as follows:

$$E = \frac{1}{2} \sum_{n=0}^{j} e_n^2 \tag{3}$$

where E is the total error of the network (mean square error) and e_j is the output error of the jth node in output layer, which is the difference between the value of the labeled output and actual output.

BPNN uses a gradient descent algorithm for constant training so that the weights can be adjusted to reach error convergence [17,18]. The updated weights are the product of the learning rate and error gradient in (4).

$$\nabla w(i, j) = -\eta \frac{\partial E}{\partial w(i, j)} \tag{4}$$

In this paper, the activation function is defined as sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} \tag{5}$$

Therefore, the derived function of the sigmoid is defined as:

$$f^{'}(x) = f(x)[1 - f(x)] \tag{6}$$

For the weight between the ith node in hidden layer and jth node in output layer:

$$\frac{\partial E}{\partial w_{ij}} = e \cdot f^{'}(S_j) \cdot x_i \tag{7}$$

where we can take node sensitivity, $\delta_{ij} = e \cdot f^{'}(S_j)$, to be similar to the weight of the bias:

$$\frac{\partial E}{\partial b_j} = \delta_{ij} \tag{8}$$

Based on the same procedure, the equation for error back-propagation from a hidden layer to an input layer can be inferred:

$$\frac{\partial E}{\partial w_{ki}} = \delta_{ki} \cdot x_k \tag{9}$$

where δ_{ki} is the weight between the kth node in input layer and the ith node in hidden layer and

$$\delta_{ki} = \sum_{n=0}^{j} \delta_{in} \cdot w_{ij} \cdot f^{'}(S_i) \tag{10}$$

Finally, the equation for weights adjustment is below:

$$w = w - \eta \cdot \frac{\partial E}{\partial w} \tag{11}$$

$$b = b - \eta \cdot \frac{\partial E}{\partial b} \tag{12}$$

The gradient descent algorithm is a common method for training neural networks, but its drawback is obvious: as the number of layers increases, the error gradient obtained by back-propagation will gradually decrease, so that the weights of the first several layers change very slowly. As a result, the learning process leads to failure, and this serious problem is often referred to as gradient diffusion.

D. *Parameter Adjustment*

In this paper, a parameter adjustment method is used to solve gradient diffusion problems to a certain extent. As shown in (5) below, we use a sigmoid function as an activation function, but during the training process, a saturation phenomenon may develop. It is often found that the sigmoidal curve described in Figure 2 is steep from -5 to 5 and almost horizontal within other intervals. Once Si is outside of the range of values from -5 and 5, activation of each node will be indistinctive, and thus, the network will not learn effectively. To prevent activation from approaching 0 or 1, we set an assistant coefficient B as follows:

$$f(x)=\frac{1}{1+e^{-\frac{x}{B}}} \tag{13}$$

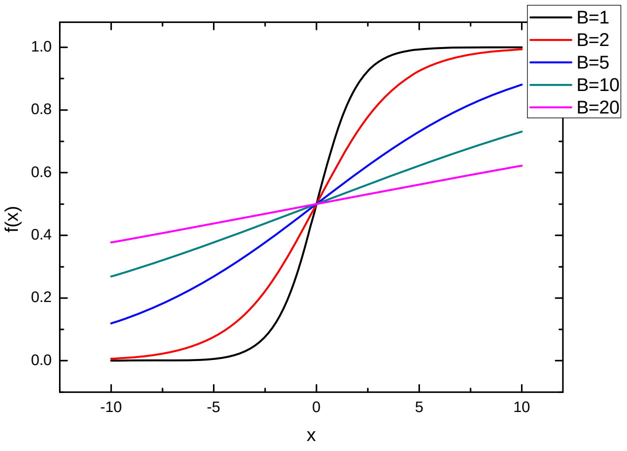


FIGURE II. THE CURVES OF SIGMOID FUNCTIONS WITH DIFFERENT B VALUES

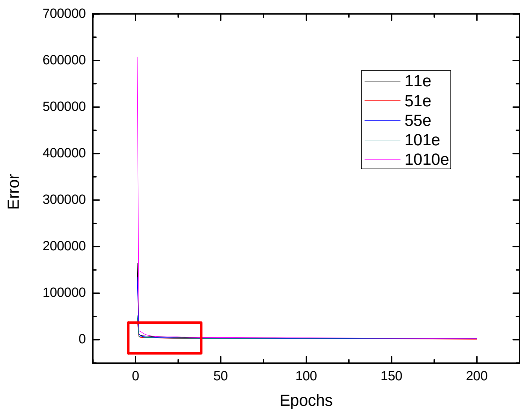
Changes in the sigmoid shape from using different B values are shown in Figure 2, and we can observe that the curve tends to be horizontal as B increases and that the range of slopes becomes larger. Although the slope of points far away from the origin becomes larger, the slope of the central interval is decreased. As a result, the training can converge rapidly first and then slowly. The derivative of the activation in the moment is described as (14):

$$f^{'}(x)=\frac{f(x)[1-f(x)]}{B} \tag{14}$$

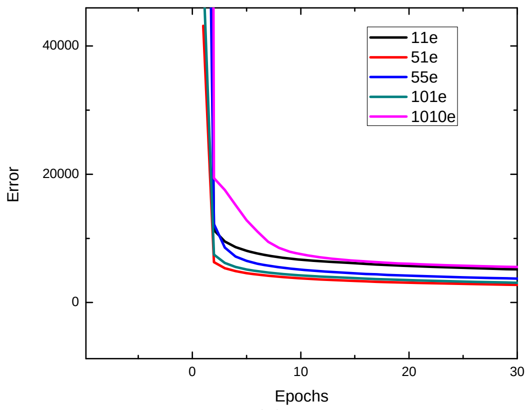
We remove B in (14) to expand the value into B times, and then, the derivative is used to calculate (7) - (12). By this procedure, the error gradient can expand so that the variation in weights can increase. This procedure solves the gradient diffusion problem, allowing the network to be trained normally.

Actually, the gradient is reduced from the back layer to the front layer and finally disappears during the error back-propagation process. Using this method, the gradient expands into B times when propagating over a layer, that is, the gradient of the first layer increases exponentially. Thus, the training tends to finish learning about weights of earlier layers first and later layers second.

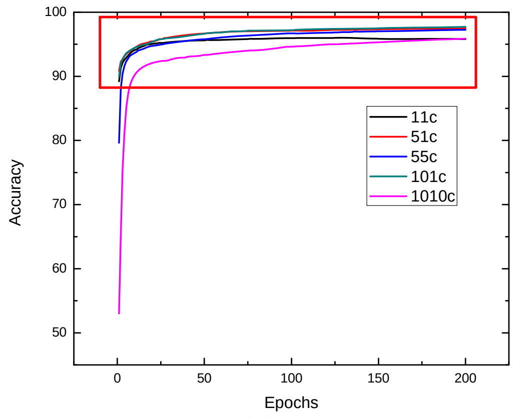
To determine the value of B, some experiments are necessary, and as follows, we use MNIST and a single hidden layer BPNN {784,800,10} with a different B to make a comparison.



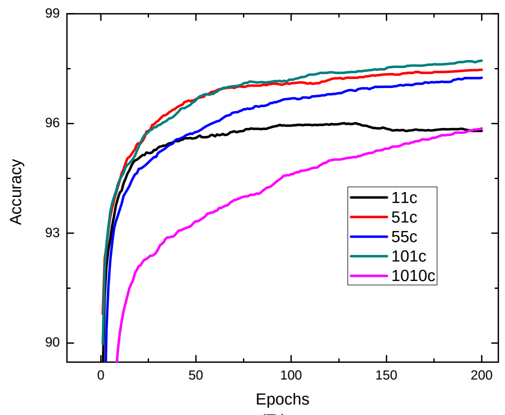
(A)



(B)



(C)



(D)

FIGURE III. THE RESULT OF EXPERIMENTS ON PARAMETER B. THE CURVES IN FIGURE 3(A) SHOW THE RESULTS OF ERROR, AND THE CURVES IN FIGURE 3(B) ARE THE LOCAL OF (A). THE CURVES IN FIGURE 3(C) SHOW THE RESULTS OF ACCURACY, AND THE CURVES IN FIGURE 3(D) ARE THE LOCAL OF (C)

For the experiments, this paper used 5 networks trained for 200 epochs with the same structure and different parameters. The activation function is defined as (13), and its derivative is defined as:

$$f_n^{'}(x) = \frac{f(x)[1 - f(x)]}{A} \tag{15}$$

where A is another parameter that is used to control the size of the gradient. Figure 3 shows that the parameters B and A of these 5 curves are {1, 1}, {5, 1}, {5, 5}, {10, 1}, {10, 10}.

The experimental results show that these five groups have the same trend of error convergence and that group {5, 1} converges fastest. For accuracy, groups {5, 1} and {10, 1} are similar and both are better than the other 3 groups. The cause of these results is considered to be:

When B>1, the curve of the activation function flares out so that the input of the nodes makes activation saturation difficult.

When A<B, the gradient is expanded to speed up learning.

It is not true that the larger B makes training faster because the global $f_n^{'}(x)$ becomes smaller through flareout.

Finally, this paper chooses parameters {5, 1} to perform subsequent experiments. It is worth noting that the learning rate may require an adjustment, along with B; otherwise, the error will oscillate and diverge due to an excessive correction of weights by an oversize gradient.

E. The Performance of Multilayer BP Neural Networks

We trained the multilayer BPNN with MNIST using the following network topology {784, 800, 1500, 1000, 500, 10}, as shown in Figure 4. There is no additional image processing prior to training, such as rotation, deformation, and so on. The parameter B of each layer was determined to be 5, and the learning rate was 0.02.

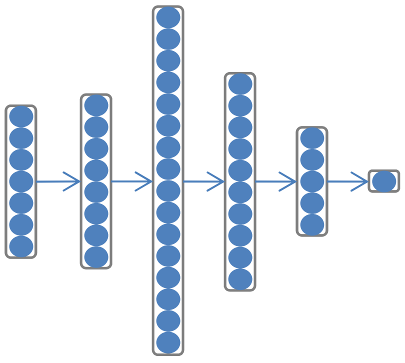


FIGURE IV. THE TOPOLOGY OF BPNN. THE NODES ARE FULLY CONNECTED WITH OTHER NOTES IN THE ADJACENT LAYER. IN FIGURE 4, THE NUMBER OF NODES IN EACH LAYER IS 784, 800, 1500, 1000, 500, AND 10 FROM LEFT TO RIGHT

In this experiment, the network cannot be trained completely if B=1, but the gradient propagation problem is solved when B=5. During the training process, the weights are adjusted once for each sample by stochastic gradient descent. As the experimental results show, when trained for 40 epochs, the accuracy reaches its peak at 98.51%. The classification result is shown in Figure 5.

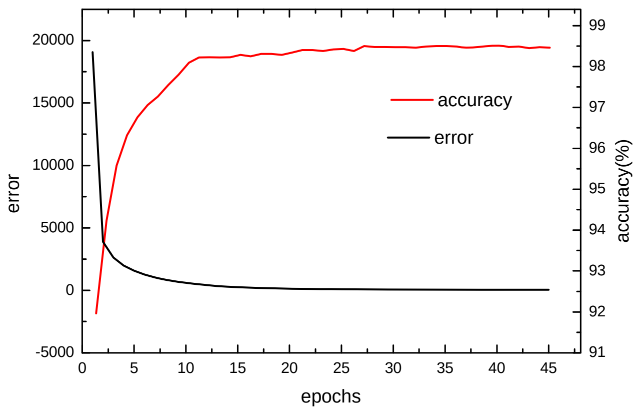


FIGURE V. THE RESULTS OF BPNN WITHOUT PRE-TRAINING

III. CONCLUSION

In this paper, we use a new a parameter adjustment procedure for activation function to solve the BP neural network training problem. Based on back-propagation and the gradient descent algorithm, we determine the value of this parameter through several group of experiments. Finally we use this method to train a Multilayer BP neural networks, and the result proves its effectiveness and feasibility.

ACKNOWLEDGMENT

The reported research was partially supported by the National Key Research and Development Program of China, no. 2016YFB1100205.

REFERENCES

[1] Simon Haykin.. Neural network and learning machines. Pearson, 2018, 7-8.

[2] Rumelhart, D.E., Hinton, G.E., &Williams, R.J.. Learning representations by back-propagating errors. Nature, 323(6088), 1986, pp. 533- 536.

[3] Arel L, Rose D.C., & Karnowski T.P. Deep machine learning a new frontier in artificial intelligence research. Computational Intelligence Magazine, 5(4), 2010, pp. 13-18.

[4] Erhan D, Bengio Y, Courville A, et al. Why Does Unsupervised Pre-training Help Deep Learning. Journal of Machine Learning Research, 11(3), 2010, pp. 625-660.

[5] Wang L, Zheng N, Wang C. The hidden nodes of BP neutral nets and the activation functions[C]// International Conference on Circuits and Systems, 1991. Conference Proceedings, China. IEEE Xplore, 1991:77-79.

[6] Song W R, Weng G R. Studies on the activation functions of BP algorithm[J]. Journal of Suzhou University, 2002.

[7] Hu Y G, Li W, Hu J M. An Artificial Neural Network with Improved Activation Function and Its Application [J]. Geomatics and Information Science of Wuhan University, 2004, 29(10):916-919.

[8] Mei D F. Improvement of BP Algorithm with Adjustment Function Optimized Gradient[J]. Modern Electronics Technique, 2006, 29(16):102-105.

[9] Hu J, Zeng X. An Efficient Activation Function for BP Neural Network[C]// International Workshop on Intelligent Systems and Applications. IEEE, 2009:1-4.

[10] Chun-Dong X, Wei L, Mu-Gui Z, et al. A BP neural network activation function used in exchange rate forecasting[J]. Advances in Intelligent & Soft Computing, 2012, 134:69-76.

[11] Wang L, Zhang X, Li J, et al. BP neural network activation function's selection and application to runoff forecasting model[J]. Journal of Hydroelectric Engineering, 2014, 33(1):29-36.

[12] LI E Y, Yang P X, Sun X B. Improved Algorithm of BP Neural Networks Based on the Activation Function with Four Adujstable Parameters[J]. Microelectronics & Computer, 2008, 25(11):89-93.

[13] Hinton, G.E., & Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. Science, 313(5786),2006, pp. 504-507.

[14] Hinton G.E, Osinder S, & Teh Y.W. A fast learning algorithm for deep belief nets. Neural Computation,18(7), 2006, pp. 1527-1554.

[15] Deng L., & Yu D. Deep learning: Method and application. China machine press, 2015, pp. 43-45.

[16] Yu D., Deng L., Seide F.T.B., et al. Discriminative pretraining of deep neural networks: US, US9235799, 2016.

[17] Hecht-Nielsen. Theory of the backpropagation neural network. International Joint Conference on Neural Networks 1, 1989, pp. 593-605.

[18] LeCun Y., Bottou L., Bengio Y., and Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 1998, pp. 2278-2324.