The Training of Feed-Forward Neural Networks

Golovko V., Dunets A., Savitsky Y.

Brest Polytechnical Institute, Mosckovskaja str., 267
224017, Brest, Republic of Belarus
E-mail: cm@brpi.belpak.brest.by

Abstract

The training of multilayer perceptron is generally a difficult task. Excessive training times and lack of convergence to an acceptable solution are frequently reported. This paper discribes new training methods of feedforward neural networks. In comparison with standard backpropagation algorithm it has smaller time complexity and better convergence. The testing of the proposed algorithm was carried out on a coding information task. The results of experiments are discussed.

Key words: neural networks, back-propagation, training, simulation.

1. Introduction

The basis of multilayer perceptron is back-propagation algorithm. It is characterised by unadaptive rate and instability of training. The instability of training depends on the initial initialisation of weights.

Many researches have devised improvements of and extensions to the basic backpropogation algorithm [1,2]. In paper [3] the backpropogation algorithm with training adaptive rate was offered. As a result the temporary complexity was reduced. However it has not solved the problem of stability of the training algorithm.

This paper describes a new approach for training of the feedforward neural networks. The results of experiments are discussed.

2. Theoretical bases of the method

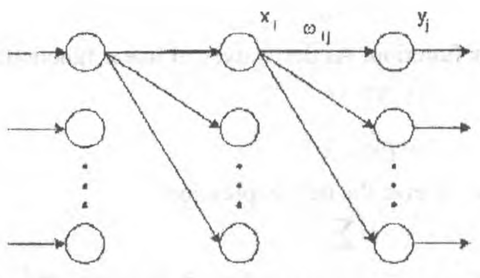

Fig 1. The general structure of the neural network

Hyperbolic tangent is used is this multilayer neural network (fig. 1) as the activation function of neural elements

$$y_j = th(S_j) = \frac{e^{S_j} - e^{-S_j}}{e^{S_j} + e^{-S_j}}, \qquad (1)$$

where S_j characterises the weighed sum j-th of the neural element. It is defined as follows:

$$S_j = \sum x_i \omega_{ij} - T_j, \qquad (2)$$

where T_j - the threshold j-th of the neural element, x_i - the output of the neural element of the previous layer, w_{ij} - the weight between neural elements i and j.

The root-mean-square error of the neural network for one pattern is defined as

$$E=\frac{1}{2}\sum_j (y_j-t_j)^2, \qquad (3)$$

where t_j - the desired target value for neurone j.

The standard method of backpropagation consists in use of gradient descent algorithm in space of weights and thresholds of the neural network:

$$\omega_{ij}(t+1)=\omega_{ij}(t)-\alpha\frac{\partial E}{\partial \omega_{ij}(t)}, \qquad (4)$$

$$T_j(t+1)=T_j(t)-\alpha\frac{\partial E}{\partial T_j(t)}, \qquad (5)$$

where α - the peed of training.

With the use of the steepest descent method it is possible to receive expression for an adaptive step of training of the neural network:

$$\alpha(t)=\frac{\sum_j \gamma_j^2(1-y_j^2)}{(1+\sum_i x_i^2)\sum_j \gamma_j^2(1-y_j^2)^2}, \qquad (6)$$

where γ_j - error j-th neural element. For the output layer

$$\gamma_j=y_j-t_j, \qquad (7)$$

and for other layers it is defined as:

$$\gamma_j=\sum_i \gamma_i(1-y_i^2)\omega_{ji}, \qquad (8)$$

Here i – the number of units of the following layer in relation to the layer j. According to it

$$\omega_{ij}(t+1)=\omega_{ij}(t)-\alpha(t)\gamma_j y_i(1-y_j^2), \qquad (9)$$

$$T_j(t+1)=T_j(t)+\alpha(t)\gamma_j(1-y_j^2), \qquad (10)$$

As it was already marked, the expression (6) allows considerably increasing speed of training. However it does not solved the problem of stability of the algorithm. For the neutralisation of this lack it is offered alongside with the modification of weights and thresholds of neural elements also to carry out the modification of outputs of hidden layer's neurones. So for hidden layer the outputs of the neural elements will change as follows:

$$x_i(t+1)=x_i(t)-\alpha_i(t)\frac{\partial E}{\partial x_i(t)}. \qquad (11)$$

Let's find the derivative of the error function on x_i

$$\frac{\partial E}{\partial x_i}=\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial S_j}\frac{\partial S_j}{\partial x_i}= \qquad (12)$$
$$=\sum_j (y_j-t_j)(1-y_j^2)\omega_{ij}=\gamma_i$$

For calculating of the training adaptive step $\alpha_i(t)$ we shall use the steepest descent method. Then

$$\alpha_i(t)=\min\left\{E\left(x_i-\alpha_i(t)\frac{\partial E}{\partial x_i(t)}\right)\right\}. \qquad (13)$$

Let's define the weighed sum as

$$S_j'=\omega_{ij}(x_i(t)-\alpha_i\gamma_i)+\sum_{k\neq i}\omega_{kj}x_k(t)-T_j, \qquad (14)$$

After the transformation of this expression we get

$$S_j'=S_j-\alpha_i\gamma_i\omega_{ij}. \qquad (15)$$

Using Taylor series decomposition for function $\gamma_j'=th(S_j')$ and transforming the received expression, we have

$$y_j'=y_j-\alpha_i\gamma_i\omega_{ij}. \qquad (16)$$

From here the root-mean-square error of the network

$$E=\frac{1}{2}\sum_i (y_i'-t_j)^2. \qquad (17)$$

We find such α_i, for which the root-mean-square error of the network is minimal

$$\frac{\partial E}{\partial \alpha_i}=\sum_i (y_j-\alpha_i\gamma_i\omega_{ij}-t_j)(-\gamma_i\omega_{ij})=0. \qquad (18)$$

Transforming last expression, we find the meaning of the training adaptive step the neurones of the hidden layer

$$\alpha_i=\frac{\sum_j (y_j-t_j)\omega_{ij}}{\sum_j \omega_{ij}^2\sum_j (y_j-t_j)(1-y_j^2)\omega_{ij}}. \qquad (19)$$

Let $y_j-t_j=\gamma_j$. Then

$$\alpha_i=\frac{\sum_j \gamma_j\omega_{ij}}{\gamma_i\sum_j \omega_{ij}^2}, \qquad (20)$$

where γ_j, γ_i - accordingly the error of j-th neurone of the following layer and the error of i-th neurone of the previous layer.

Thus it is supposed to carry out the training of the neural network on three parameters: weights, thresholds and outputs of neural elements. The outputs of the neural elements change with the purpose of minimisation of the neural network root-mean-square error as follows

$$x_i(t+1) = x_i(t) - \alpha_i(t)\gamma_i, \qquad (21)$$

where γ_i - error i-th of the neural element.

As the area of meanings of the activation function of neural elements is segment D = [-1;1], after the reception of desirable neurone's outputs of the hidden layer it is necessary to make normalisation of the received meanings, which belong to the segment D. The normalisation will be carried out for each neurone by the following rule.

1) The pattern is determined, for which the value M of the desirable output for the given neurone is the maximal on the module

$$M = \max_k |x^k| = \overline{1, L}, \qquad (22)$$

where L – the number of patterns.

2) If this value belongs to segment D, the procedure is finished.

3) The desirable outputs for the given neurone for all patterns are recalculated according to the formula

$$\overline{x^k} = \frac{x^k}{M}, k = \overline{1, L} \qquad (23)$$

4) The weights from the given neurone to neurones of the following layer change according to the formula

$$\omega_{ij}^{*} = \omega_{ij} \cdot M, j = \overline{1, J}, \qquad (24)$$

where i – the number of considered neurones, J – the quantity of neurones in the following layer.

3. Algorithm

The algorithm consist of the following steps:

1. The random initial of weights is made and the minimal root-mean-square error of the network E_m is set.
2. The modification of weights and thresholds according to expressions (9) and (10) only for the last layer is made. Simultaneously with each pattern according to (21) the desirable outputs $\overline{x_i}$ of neural elements from the hidden layer are defined.
3. Outputs of hidden layers of network are considered only: as the entrance information the outputs $\overline{x_i}$ of neural elements are used, as target - reference outputs
   3.1. The modification of weights, thresholds and desirable outputs according expressions (9), (10) and (21) accordingly is made for L entrance patterns.
   3.2. Item 3.1 is repeated, while the total root-mean-square error of the examined fragment of the neural network will not become less than E_m.
4. The desirable outputs are normalized according to formulas (22)-(24).
5. The modification on L patterns of weights and thresholds of the following layer of the network is made. Thus the error i-th of the neural element is equal to $\gamma_i = x_i - \overline{x_i}$.
6. The procedure is repeated from step 2, while the total root-mean-square error of the neural network will not become less than E_m.

4. Testing

The experimental check of the received results was carried out on the task of coding of the information. Thus the neural network should on the basis of cyclic coding carry out transformation informational polynomial in a surplus code.

Let word length of informational polynomial be n = 4, and superfluous polynomial m = 7. Then the architecture of the multilayer neural network contains 4 input and 7 output neurones. In the hidden layer we shall use 8 neural elements. Then we have the neural network consisting of 3 layers the with size of training set L = 16.

The experiments have shown that in comparison with other algorithms 100 % stability is got. So with any initialisation of weights the neural network was trained up to the minimal error. With the use of standard backpropagation only in 10 % of all attempts to train the network the acceptable result was reached.

5. Conclusion

The experimental check of algorithm is carried out on the basis of the developed method. It is characterised by independent training of each of layers of a neural network. The experiments have shown that the offered algorithm has greater stability in relation to standard backpropagation. In the offered method the potential opportunities for automatic generation of neural network architecture are incorporated also. Nowadays in this direction the researches will be carried out.

6. References

1. Silra F., Almeida L. Speeding up backpropogation.// Advanced neural computers, North-Holland, pp.151-160, 1990.
2. Saarinen S., Bramley R., Cybenko G. ILL-conditioning in neural network training problems. SIAM Journal on Scientific Computing, 14:693-714, 1993.
3. Golovko V., Savitsky Ju., Gladischuk V. A neural net for predicting problem. Timisoara: University of Timisoara, Romania, 1996.