A STUDY CONCERNING THE ELIMINATION OF THE NEURONS OR OF THE LAYERS FROM MULTI-LAYER NEURAL NETWORK

Gheorghe Puşcaşu, Bogdan Codreş

"Dunarea de Jos" University of Galati, Department of Automation and Computer Science,
Fax 40-036-460182, Str. Domneasca, Nr. 111, 800201 Galati, Romania
e-mail: gpuscasu@ugal.ro

Abstract: After a neuronal structure is trained using a training set, an important problem is to generalise the learned. If the system memorises only the data used for the training session it might be possible, the network to give us erroneous results, for another similar set of data.
This paper proposes a study concerning some techniques for the elimination of neurones from the layers of a multi-layer neural network. This procedure is applied after the training stage. The study leads us to a network structure with a smaller number of neurones and layers, structure that approximates in the same manner unlinear function. Another experimental aspect is the fact that there are neurones with outputs which are not modified when at the input of the network is presented a set vector from training sequence. In this case the neurones with the constant output will be eliminated.

Keywords: neural networks, robustness, sensitivity and inactive neurones.

1. INTRODUCTION

In past years utilization of neural networks took a distinct ampleness because of following properties:

•**distributed representation of information [4]:** Information learned by network is stored distributivelly in all network weights.

• **capacity of generalization [3]** in case of uncontained situation in training data set.

•**tolerance to noise [1]:** neural networks have capacity to learn in situation that training set is affected by noise.

•**resistance to partial destruction [5]:** Neural networks can operate also in the case of small region destruction or in the situation when some weights are affected by small errors, because of distributed representation of information.

• **rapidity in calculation [4]:** Neural networks are big time consumers in learning, but once trained, it will calculate very quickly the output for a given input.

The major applications of artificial neural networks involves learning the control mappings between inputs and outputs of complex systems. These mappings can then be used in pattern recognition, in robotics and other applications.

The problem size can be approximately quantified as the dimensionality of the input space. With an increase of the size input space will involves an exponential growth of the network size. In order to acquire a highly systems mapping through learning, a large and complex network is required to approximate the mapping up to a certain degree of accuracy.

The cleaning procedures of neural networks can be divided in two approaches. The first approach [6] consists in the estimation's sensitivity of mean square error, and then the removal of the neurons with least effect upon sensitivity. The second approach [5] consists in the estimation of an objective function that leads us to efficient solutions regarding the neural network's dimension.

Generally the methods based on estimation's sensitivity modify the initial configuration of the network to obtain higher performance. On the other hand, the methods based on penalization algorithms modify the cost function so that the backpropagation through network evaluates other weights than the initial one. The useless weights are equalled with zero and then removed in the training phase. In some situations, even if some weights are not removed the network entails like a system of smaller dimension.

2. ASPECTS REGARDING THE TRAINING OF A MULTILAYER NEURAL NETWORK

To illustrate the training's mechanisms of a multilayer neural network and the mode of data processing it is considered the case of a classifier implemented with a neural network. It is considered this case because in pattern recognition the information is better structured and some aspects can be traced easier.

The recognition problem using a multilayer neural network can be easy exemplified with the case of two patterns represented in the following picture:

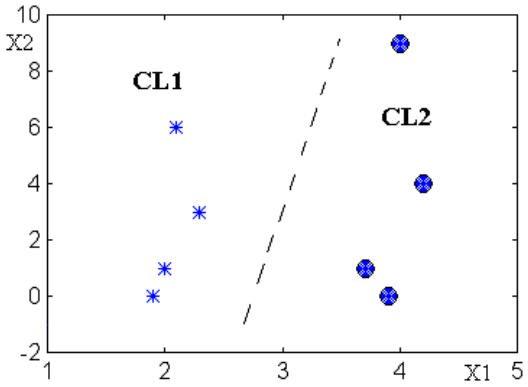

Figure no.1; ✳ - Patterns from class 1; ⊛ - Patterns from class 2

From the picture above results the fact that for the pattern recognition is necessary only the X1 characteristic because the X2 characteristic belongs to the same interval and does not contribute in the recognition's process. This aspect must be encountered in the solving of the problem using neural networks, too.

For the implementation of a classifier it is considered a slightly extradimensioned multilayer neural network with the following structure:

- first layer - three neurones and the tan-sigmoid activation function
- second layer - two neurones and the log-sigmoid activation function

The training set is:

$$P=\begin{bmatrix} 2.0 & 2.1 & 1.9 & 2.3 & 4.0 & 4.2 & 3.7 & 3.9 \\ 1.0 & 6.0 & 0 & 3.0 & 9.0 & 4.0 & 1.0 & 0.0 \end{bmatrix},\quad T=\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

In the following pictures it is represented the evolution of the network's weights in the training session and the contribution of characteristic's vector to the argument of the activation functions.

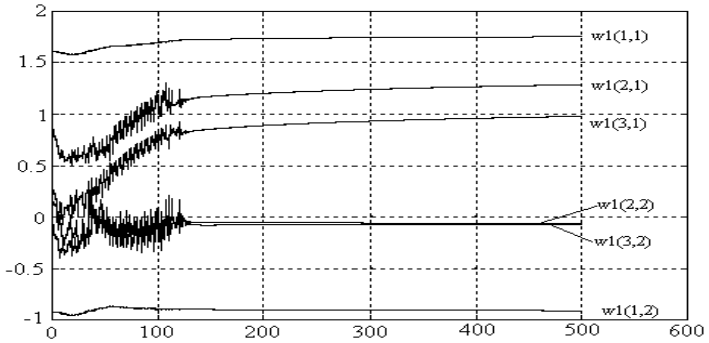

Figure no.2. Evolution of first layer's weights in the training session

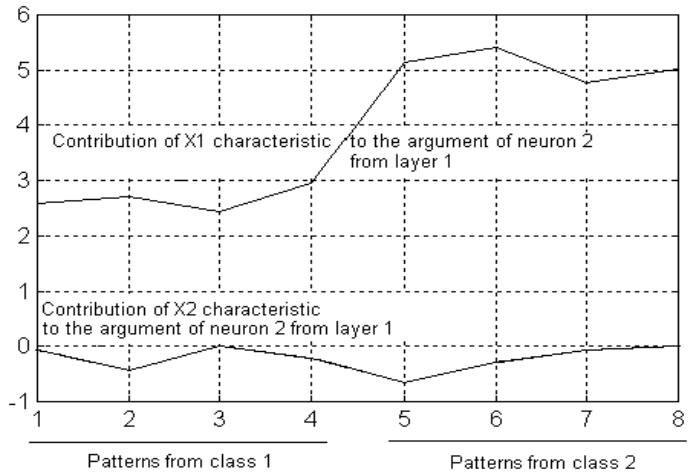

Figure no.3. Contribution of characteristic's vector to the argument of the activation function of neuron 2 from layer 1

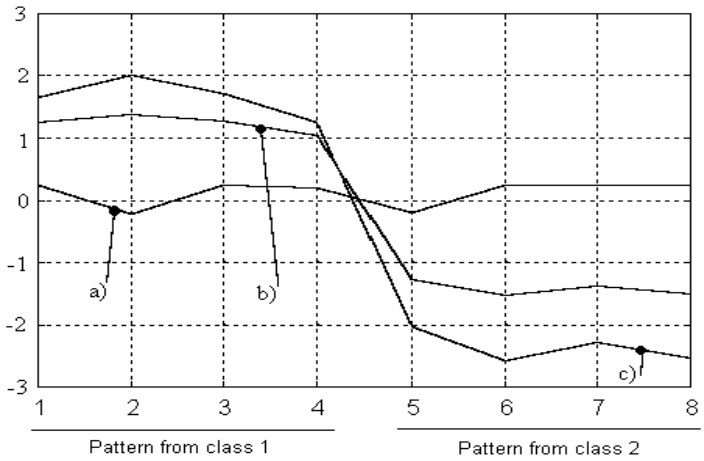

Figure no. 4. Contibutions of neuron's outputs from layer 1 to the argument of the activation function of neuron 1 from layer 2 a)-output of neuron 1 from layer 1, b) - output of neuron 2 from layer 1, c) - output of neuron 3 from layer 1.

From the pictures above one can draw the following conclusions:

- The weights of first neuron from the first layer stay almost unchanged compared to the initial values (figure nr.2), which leads us to the conclusion that these do not contribute to the classification. On the other hand the other weigths suffer changes in the training session.
- The X2 characteristic does not contributes (does not have a decisive part in pattern recognition) to the argument of the activation function of neuron 2 from layer 1. This aspect could represent

a point of departure in the use of neural networks in the selection of relevant characteristics in pattern recognition.

- The first neuron from layer 1 does not contributes to the argument of the activation function of neuron 1 from layer 2.

The results presented in figure 2 presumes the use of an objective function regarding the cleaning methods for neural networks. The aspects illustrated in figure 3 and 4 allow the estimation's sensitivity of the mean square error for the cleaning algorithms.

3. SENSITIVITY OF NEURAL NETWORKS AND PRUNING RULES

3.1. SENSITIVITY AND ROBUSTNESS OF NEURAL NETWORKS

After the multi-layer neural network training, sometimes is necessary to analyse capabilities of neural network as robustness and sensitivity when different types of neurones are eliminated.

This elimination means that neuron does not transmit information to the next neurones. Elimination is equivalent with the missing of the same neuron from network structure.

We will define the robustness and sensitivity of a network using mean square error

$$Er = \frac{\sum_{i=1}^{n_p} \cdot \sum_{k=1}^{S_n} \left(A(i,k) - T(i,k) \right)^2}{n_p} \qquad (1)$$

where:

- $A(i,k)$ -represents the k neuron output after training
- n - the number of the network layers
- $T(i,k)$ - the final desired value of the k neurone output after training
- S_n - the number of the neurones from output layer
- n_p - the number of the vectors from training set

Starting from relation (1) we enunciate two definitions.

First definition: The capability of the neural networks to maintain mean square error around a desired value obtained after training algorithm when some different neurones are eliminated from network is called ***robustness***.

Second definition: The capability of the neural networks to modify the mean square error when some different neurones are eliminated is called ***sensitivity***.

Sensitivity problem of the neural networks when the bias and weights values are modified around the final values obtained after training algorithm is show in paper [1]. Hardware implementation of the multi-layer neural networks is accomplished with value deviations comparative with the result values after learning process. Thus it is necessary to analyse the consequences of these deviations regarding the network response comparative with desired response.

3.2. PRUNING NEURONS OF NEURAL NETWORKS WITH INCONSTANT OUTPUT

The study of these capabilities has applications in minimisation of neurones number of the network after training algorithm. As example, we consider two classes represented in figure 5, where are shown through dots the domains of this two classes.

The applications of the multi-layer neural networks are often used without determine the minimum of layers and neurones necessary for a specific application. The study of the mention example may be extended to an approximation of a nonlinear function, too.

The network structure used for implementation of the classifier is:

- first layer - eight neurones and tan-sigmoid activation function
- second layer - twelve neurones and tan-sigmoid activation function
- third layer - one neuron and log-sigmoid activation function

Input vectors, used during learning process corespondent to the dots marked in figure 1 with “0” and “+“, are made from pattern co-ordinates of the both classes. Target vector is '0' for first pattern class and '1' for second pattern class.

In this case we can share the neurones (which have the tan-sigmoid function) according to their activity, when at the entry are set different patterns, in:

- neurones which have an activity at the saturation limit or which have the constant output;
- neurones which have the output between -1 and +1 values;
- neurones with the output in [-1 0) interval or in (0 1] interval.

We notice experimentally that trained network is robust when neurones with output in [-1, 0) interval or in (0 1] interval are eliminated. Sensitivity is observed when neurones with the output in the interval [-1, +1] are eliminated.

The neural network, used in our case, is robust to elimination of the neurones 2, 3, 4, 5, 9, 10 from second layer with the outputs in [-1,0) or in (0,1] and it’s sensitive to elimination of neurones 6, 7, 8 from second layer with the output in interval [-1, +1].

If we look how is calculated the argument of the third layer neuron function, the results are justified. The argument of the third layer neuron function is:

$$S = \sum_{j=1}^{12} w^3(1, j) \cdot A^3(j) + B^3 . \quad (2)$$

The main componets of the sum S are given by the neurones types mentioned above. The argumet of the third layer neurone given by neurones 2,3,4,5,9,10 and neurones 6,7,8 is

$$S_1 = \sum_{j \in \{2,3,4,5,9,10\}} w^3(1, j) \cdot A^3(j) \quad (3)$$

$$S_2 = \sum_{j \in \{6,7,8\}} w^3(1, j) \cdot A^3(j) \quad (4)$$

In figure 6 are represented the S, S1, S2 values when at the network's entry are applied the patterns from training set. The significant value of the function argument for the third layer neurone is given by the S2 sum.

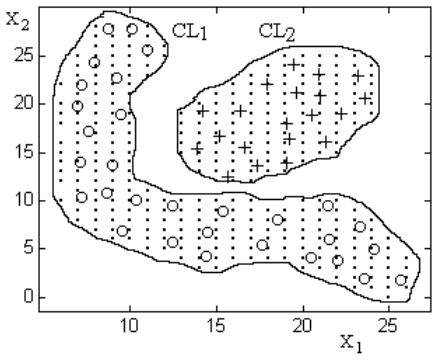

Figure no. 5

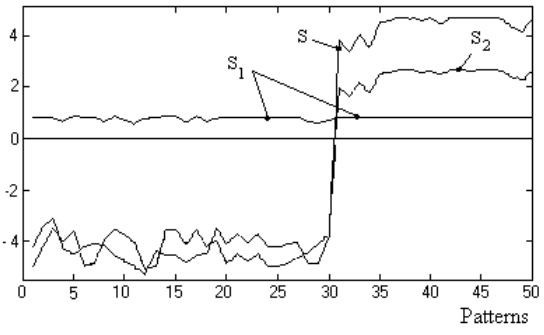

Figure no. 6

3.3. PRUNING INACTIVE NEURONS

In this section is presented a method for the elimination of the neurones from a network, without a modification of the result of the training network. In case of recognition, thanks to the target vector which takes '0' and '1' values, the saturation state of the output neuron activation is forced. In multi-layer networks, for patterns which belong to classes shown in figure 5 are five neurones in first layer and two neurones in second layer, which have the activity at saturation limit or have a constant output. These neurones could be eliminated. In this way, the total number of

neurones is reduced without modification of the mean square error. We present the elimination possibility of the neurones with the constant output.

It is considerated that the neuron which belongs to k-*1* layer and has α = *constant* output value. The information provided by Nj$\in$*Sk-1* neuron to *Ni*$\in$*Sk is*: $W^k(i, j)=\alpha = ct$. This constant could be taken through a bias modification of the *Ni* neuron. The bias is recalculated with the following relation:

$$B_{imod.}^k = B_i^k + w^k(i,j)\cdot\alpha \qquad (5)$$

Thus activation function of the *Ni* neuron, for tan-sigmoid function becomes:

$$F(x) = \frac{1-e^{-(x+B_i^k+w^k(i,j)\cdot\alpha)}}{1+e^{-(x+B_i^k+w^k(i,j)\cdot\alpha)}}. \qquad (6)$$

This change is done for all neurons, which receive information from the neuron with the constant output.

We present a procedure which eliminates neurones with a constant output during the learning process. The multi-layer network is a $RN^n(S_1,..., S_{k-2}, S_{k-1}, S_k,..., S_n)$ where *n* is the number of layers S_k is the number of neurons from *k*-layer. *j* neuron belonging to *k*-layer is eliminated with this procedure:

1. It is set the counting number for the neuron S_{k-1} layer, $i=2$.
2. The threshold for i neuron from S_{k-1} is modified with relation
   $B_m^k(i) = B^k(i) + w^k(i,j)\cdot\alpha_j$ (4)
3. where α_j is the output *j* neuron from S_{k-1} layer $i=i+1$.
4. If $i \le Sk$ then jump to 2, if not continue.
5. It is eliminated *j* line from W^{k-1} weights matrix with dimensions $(S_{k-1} \times S_{k-2})$.
6. It is eliminated column j from W^k weights matrix with dimensions $(S_k \times S_{k-1})$.
7. It is eliminated *j* element from B^{k-1} weights vector with dimensions S_{k-1} .
8. STOP.

Through a threshold change with a $wk(i,j)*\alpha$ value, are eliminated neurons from network with a constant output ,without a modification of the result of learning .

Using procedure which will be presented, the 1,3,4,5,6 neurones with the constant output, and also the 2,3,4,5,9,10 of second layer with the output in [-1,0) or in (0,1] interval are eliminated.

The network obtained after the elimination mention above is

- first layer - 3 neurones with tan-sigmoid activation function
- second layer - 6 neurones with tan-sigmoid activation function
- third layer - 1 neuron with log-sigmoid activation function

Because it isn't a criterion for determine an optimal structure of the network, which approximates a desire nonlinear function, it is necessary to analyse the neuron's outputs without a contribution to this approximation.

4. CONCLUSIONS

The study of these capabilities has implications in minimising the neurones number of the network, after learning process. During the training algorithm, due the great number of parameters, the network succeeds in the approximation of nonlinear functions using a small number of neurones from the total number. An analysis of the neurones contributions at the total response is necessary for a hardware implementation of the network later on.

5. REFERENCES

[1] Maryhelen Stevenson, Rodney Winter and Bernard Widrow. (1990) Sensitivity of feed-forward neural networks to weight errors. *IEEE Transactions on Neural Networks, Vol.1, pp. 71-80*

[2] Eric Davalo and Patrick Haine (1990) Neural Networks, *Department of Computer Science University of Manchester*

[3] McClelland and D.E. Rumelhart (1986) Parallel Distributed Processing Exploration in The Microstructure of Cognition I, *MIT Press, Cambridge*

[4] David E. Rumelhart and James L. McClelland (1986) Parallel Distributed Processing Exploration, *Psychological and Biological Models, Volume 2. MIT PRESS, Cambridge*

[5] Russell Reed (1993) Pruning algorithms-A Survey, IEEE Transactions on Neural Networks, Vol. 4., pp. 740-747.

[6] Y. Le Cun, J.S. Denker and S. A. Solla, (1990) Optimal brain damage, Advances in neural Information Processing, pp. 598-605;