Proceedings of the 8th IEEE Mediterranean Conference
On Control and Automation (MED 2000)
Rio, Patras, GREECE 17-19 July, 2000

# IMPROVED TRAINING OF MULTILAYER FEEDFORWARD NEURAL NETWORKS FOR LARGE INPUT VECTORS

**CATALIN-DANIEL CALEANU †, LYKOURGOS PETROPOULAKIS‡**

† Department of Applied Electronics, Faculty of Electronics and Telecommunications, Politehnica University Timisoara, B-dul V. Parvan, nr.2, 1900 Timisoara, Romania, catalin@earouter.ee.utt.ro

‡ Department of Electronic & Electrical, Engineering, University of Strathclyde, 50 George Street, Glasgow G1 1QE, akis@icu.strath.ac.uk

***Abstract***. In this paper, a new fuzzy controller for inferring multiplayer feedforward neural networks learning rate is presented. The key issue is using *relative* values of a *performance attribute* as fuzzy controller inputs, resulting in an increased generality of fuzzy learning rate adaptation and a faster training algorithm. Experimental results demonstrate improvements in terms of generalization capability and of learning speed in both *large* pattern recognition and data processing tasks.

**Key Words.** Neural nets, fuzzy control, training, algorithm.

## 1. INTRODUCTION

In conventional gradient descend algorithms [5] it is assumed that the learning rate η is fixed and uniform for all weights in a layer. Usually η must be kept very small to prevent parasitic oscillations and ensure convergence. However, a very small fixed value of η slows the learning process considerably. In fact a very large value for the learning rate can result in an unstable learning (optimization) process, whilst a small value η can result in an appreciable and impractically long training periods.

The training time can be considerably reduced by employing an adaptive (variable) learning rate. The adaptation methods attempt to keep the learning rate at each iterative step as large as possible while keeping the process stable.

In this paper we show how a fuzzy controller, using the relative values of performance attribute (e.g. Mean Squared Error (MSE), Sum Squared Error (SSE), etc.) and change of performance attribute, can be used to dynamically alter the learning rate. The final result is a faster and more robust training algorithm, which outperforms classical adaptive learning rate gradient descend training algorithms.

## 2. SYSTEM DETAILS

The fuzzy controller developed for this architecture is based on two observations:
- quasi-totality of classical [3] or fuzzy [2], [6] learning adaptation methods are based on absolute value of change error, as fuzzy controller input. The major drawback of these methods is their dependence on the type of neuron's activation function and error signal value. If we consider a particular neural network architecture and, in the first case, a linear activation function, the range of output error could be $10^2$ times bigger than the case of sigmoid activation function, for the same neural network structure. It is now clear that the design of a universal fuzzy controller relay on the absolute value of error is impossible and a *relative value* of error is most appropriate;

- for increasing the generality of fuzzy learning rate adaptation we deal with the relative value of neural network's *performance attribute* instead of a particular form of this (e.g. MSE, SSE).

In conclusion, we propose as fuzzy Mamdami type controller inputs, where the *relative performance attribute* value is given as:

$$p(n) = \frac{perf(n)}{perf(n-1)} \quad \dots (1)$$

and the *relative change of the relative performance attribute* value is given by:

$$\Delta p(n) = \frac{perf(n)}{perf(n-1)} - \frac{perf(n-1)}{perf(n-2)} \quad \dots (2),$$

and where "n" denotes the current training epoch.

Instead of an incremental value as proposed in ([2], [6]), we propose as output of the fuzzy controller a coefficient "c" which multiplies the learning rate. The result is a faster variation of the learning rate:

$$\eta(n) = c \cdot \eta(n-1) \quad \dots (3)$$

The training algorithm is inspired from classical method of varying learning rate presented in [9]:

$$\eta(k) = \begin{cases} c_1 \cdot \eta(k-1), \text{ if } perf(w(k)) < perf(w(k-1)) \\ c_2 \cdot \eta(k-1), \text{ if } perf(w(k)) \geq c_3 \cdot perf(w(k-1)) \\ \eta(k-1), \text{ otherwise} \end{cases} \quad \dots (4),$$

where typical values of parameters are
$c_1 = 1{,}05$, $c_2 = 0.7$, $c_3 = 1{,}04$.

In the last case the new weights and the error are discarded. The disadvantage of this method is that parameters $c_1$ and $c_2$ are constant during the training phase making impossible a rapid change in learning rate.

Our algorithm offers different fuzzy degrees for these coefficients resulting in an increase in convergence speed.

The Membership functions for input and output variables are shown in fig. 1, 2, and 3. The fuzzy-rule set for controlling the learning rate parameter is presented in Table 1 and the output variable versus two input variables (control surface) is depicted in fig. 4.

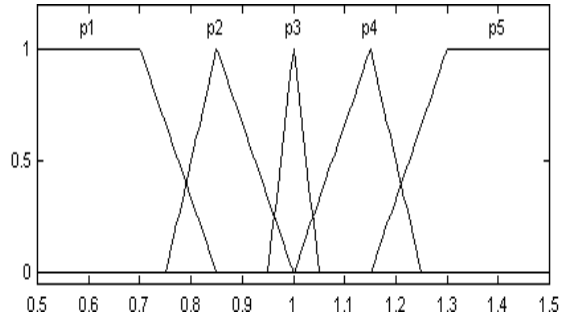

Fig. 1. The membership functions for the relative performance attribute

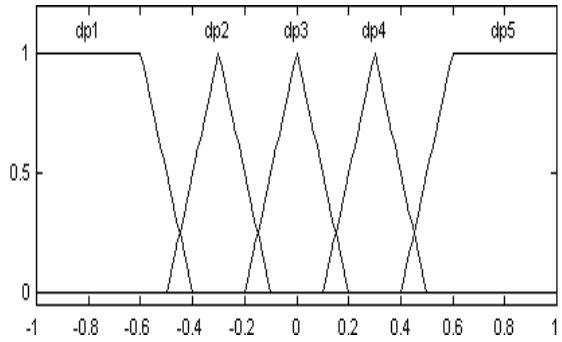

Fig. 2. The membership functions for the change of relative performance attribute

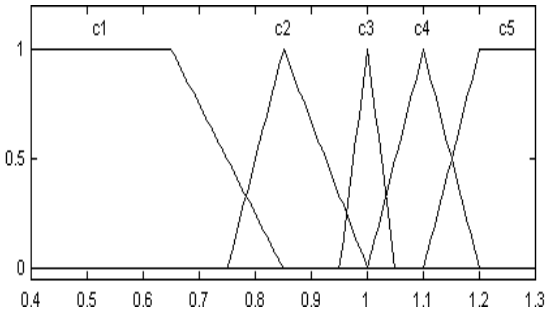

Fig. 3. Membership function for output variable "c".

Table 1. The fuzzy-rule set for controlling the learning rate parameter.

| dp \ p | p1 | p2 | p3 | p4 | p5 |
|---|---|---|---|---|---|
| dp1 | c5 | c5 | c4 | c3 | c2 |
| dp2 | c5 | c4 | c4 | c2 | c2 |
| dp3 | c4 | c4 | c4 | c2 | c1 |
| dp4 | c3 | c2 | c2 | c3 | c1 |
| dp5 | c2 | c2 | c1 | c1 | c1 |

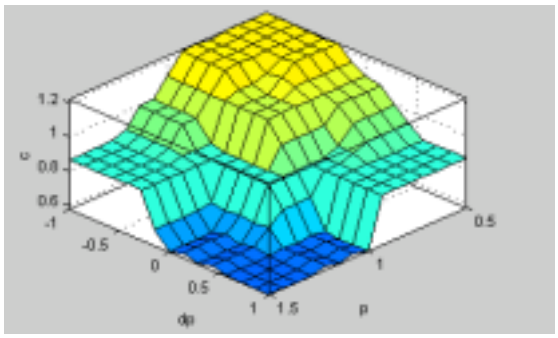


Fig. 4. Fuzzy control surface

## 3. EXPERIMENTAL RESULTS

The proposed method has been tested on several problems such as the parity problem, exclusive NOR problem, function approximation problem and real problems arising in pattern recognition.

The method (trainfuzzy) has been implemented in Matlab v.5.2 using Neural Network Toolbox v.3.0 and Fuzzy Logic Toolbox v.2.0, and

compared with gradient descend with momentum (traingdm) and gradient descend with momentum and variable learning rate (traingdx) predefined methods. The training times given below are CPU times required for training on a PC INTEL PII, 333MHz. In our experiments, iteration is said to be completed when all training patterns are presented and weights of the MLP are modified.

*Experiment 1- Approximation of a Function:* The task of training a function is a stringent one. In this experiment an application of the proposed approach to build a network, which approximates the following function, is presented:

$$f(x) = 0.2 + 0.8[x + 0.7\sin(2\pi x)] \quad ...(5)$$

We assume $0 \leq x \leq 1$. The training data are taken at intervals of 0.1; thus we have 11 data points. We used 101 evaluation points taken at intervals of 0.01. The evaluation data is used to verify the interpolative power of the network.

The results for the best numbers of hidden units are presented in Table 2. The training process was stopped when MSE reached the value of 0.003.

Fig. 5 depicts learning profiles produced for this problem and indicates that the proposed learning method yields much faster learning.

Table 2. Comparative results on the approximation of a function problem.

| Train. Meth. | hidden units | Itera-tions | Train. time [s] | Init. Lear. rate | Train. set [%] | Test set [%] |
|---|---|---|---|---|---|---|
| traingdm | 9 | 150 | 18.89 | 0.65 cons. | 3.560 | 2.360 |
| traingdx | 9 | 59 | 5.50 | 0.65 | 2.785 | 0.936 |
| trainfuzzy | 9 | 34 | 3.91 | 0.65 | 2.482 | 0.009 |

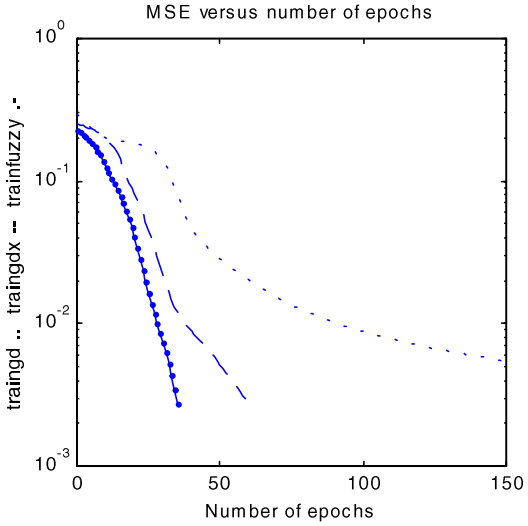


Fig. 5. MSE versus number of iterations for approximation of a function problem.

*Experiment 2 – Pattern Recognition Problem:* The problem consists of the recognition of 100 facial images (10 subjects x 10 images/person) from AT&T Database of Faces (formerly The ORL Database of Faces) [4]. The images are divided into 50 training images and 50 test images. The images format is 56x46 pixels. We carry two types of experiments:

a) The images are applied without any pre-processing to one hidden layer MLP.

The size of the neural network will be:
56 x 46 = 2576 input neurons, with 50 hidden neurons and 10 output neurons. The problem is quite difficult because of the large number of input neurons and it cannot always be easily tackled by other fast training algorithms (Fletcher-Powell conjugate gradient, Polak-Ribiere, Levenberg-Marquardt). The results for the best numbers of hidden units are presented in Table 3. The training process was stopped after

80 epochs or a MSE of 0.03. Figure 6 illustrates the training process.

Table 3. Comparative results on the pattern recognition problem.

| Train. Method | hidden units | Itera-tions | Train time [s] | Init. Learn. rate | Train. set [%] | Test set [%] |
|---|---|---|---|---|---|---|
| traing dm | 50 | 80 | 105.73 | 0.01 cons. | 90 | 90 |
| traing dx | 50 | 80 | 90.63 | 0.01 | 4 | 10 |
| train fuzzy | 50 | 76 | 84.53 | 0.01 | 2 | 4 |

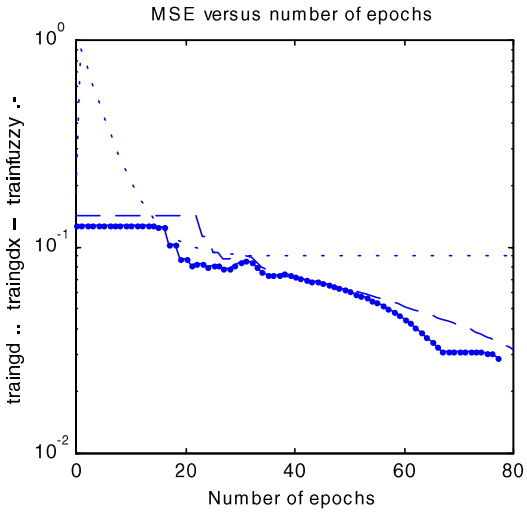


Fig.6. MSE versus number of iterations for pattern recognition problem.

b) Feature vectors are extracted from three image regions (see details in [1]) through interest operator method [7]. Then feature vectors are classified with the same type of neural network. In this case only 3 pattern/class are required for a 100% recognition rate (Table 4).

Table 4. Recognition rate using feature extraction technique.

| Number of pattern/class | Correct test set recognition rate [%] |
|---|---|
| 1 | 84,4 |
| 2 | 98,7 |
| 3 | 100 |

## 4. CONCLUDING REMARKS

In this paper, we propose a method for fast training of multilayer perceptrons (MPL). A number of experiments for classification as well as for approximation have been conducted and some of them are presented. The experiments show that the proposed method is able to train MPL's much faster than traditional gradient descend algorithms. We conclude therefore that owing to this fast learning training rate and the simplicity of the approach, this method would be useful in several practical implementations of real-world problems.

## 5. REFERENCES


[1] Caleanu C.D. – "Parallel neural processing of interest regions in facial recognition", Transactions on Electrotechnics, Electronics and Communication, pp.129, Timisoara, Romania,1999

[2] Choi J., Arabshahi P., Marks R., Caudell T., - "*Fuzzy parameter adaptation in neural system*", International Joint Conference on Neural Networks, Vol.1, pp. 232-238, Baltimore, 1992.

[3] Cichocki A., Unbehauen R., - "*Neural Networks for Optimization and Signal Processing*", John Wiley & Sons, pp.146-157, 1993.

[4] ftp://ftp.uk.research.att.com:pub/data/att_faces.tar.Z

[5] Haykin, S., "*Neural Networks. A Comprehensive Foundation.*", MacMillan Publishing Company, NY, 1994.

[6] Hertz D., Qing H., "*Fuzzy-neuro controller for backpropagation networks*", Proceedings of the Simulation technology and Workshop on Neural Networks conference, Houston, TX, pp. 570-574, 1992.

[7] Moravec H., "*Robot Rover Visual Navigation*", Univ.of Michigan Research Press, 1981.

[8] Verma B., - "*Fast Training of Multilayer Perceptrons*", IEEE Trans. Neural Networks, vol.8, pp.1314-1319, Nov., 1997.

[9] Vogl T., Mangis J., - "*Accelerating the convergence of the backpropagation method*", Biological Cybernetics, vol. 59, p 257-263, 1998.