ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 359-364

A New Training Algorithm for Feedforward Neural Networks

Brijesh K. Verma, *Member IEEE*, and Jan J. Mulawka
Warsaw University of Technology
Nowowiejska 15/19, Room 225
00-665 Warsaw, Poland
E-mail: Bverma@koral.ipe.pw.edu.pl

Abstract- This paper introduces an approach for fast training of the Feedforward Neural Networks (FNNs). The approach is based on linearization of nonlinear output activation function by inverting it and calculation of weights using gradient-decent and QR decomposition techniques. The approach called Gradient-descent Orthogonalized Training (GOT) algorithm. The algorithm GOT is applied to some benchmark problems and the results are compared to those obtained using Error Back-Propagation (EBP) algorithm.

I. INTRODUCTION

Many researchers have recognized the potential of FNNs for pattern recognition, speech recognition, system modeling and other tasks [Her91, Ver94c]. Feedforward neural networks using EBP [Rum86] algorithm have been widely used for many applications. Although EBP algorithm has demonstrated great capabilities in learning difficult mappings, it has some drawbacks. First, there is no guarantee that the network will find the global minimum of the cost function. Second, is that the algorithm converges very slowly. Many modifications [Her91, Mul94, Ver94a] have been proposed to improve the training time of the EBP algorithm.

In this paper, another speed-up approach is proposed. The approach combines the gradient descent technique [Rum86] and the Gram_Schmidt orthogonalization method [Kie92].

II. NETWORK ARCHITECTURES

Feedforward neural network is a multilayer network consisting of nodes grouped into layers. We consider a two-layer network which is illustrated in Figure 2. The network has n inputs, m outputs and h hidden units. All neurons represented by the model depicted in Figure 1 are grouped in sequentially connected layers. Each neuron is characterized by one output and many inputs, which are the neuron outputs of the preceding layer. Let $x_1 \ldots x_n$ and o denote the inputs and output of the neuron and w is the weight, then the computation performed by each neuron can be expressed as:

$$net = \sum_{j=1}^{n} x_j w_j \tag{1}$$

$$o = f(net) = \frac{1}{(1+\exp(-net))} \tag{2}$$

III. DESCRIPTION OF THE PROPOSED ALGORITHM

The proposed training algorithm uses iterative error backpropagation and orthogonalized-QR decomposition methods. First an error backpropagation [Rum86, Mul94] is applied to the two-layer FNN. After some iterations, we stop the training process and check the inputs of the output layer, if we get similar vectors then take only one among them for further calculations (use data reduction algorithm) [Ver94c]. Next we convert the output nonlinear activation function as shown below.
Equation (2) can be rewritten as follows:

$$d=\frac{1}{(1+\exp(-net))} \tag{3}$$

where d is the desired output.
In the FNNs, the desired output is always known for each neuron in the output layer. We can easily calculate the net value from Equation (3) as follows:

$$net=\ln(\frac{d}{(1-d)}) \tag{4}$$

Let H be the matrix of selected input vectors and $\underline{w}$ is the weight vector of the output layer. We can write a linear system of equations for the output layer as follows:
$$H\underline{w}=\underline{net} \tag{5}$$
The weights ($\underline{w}$) of the output layer can be calculated from (5) using orthognalized-modified Gram_Schmidt method [Kie92, Ver94a].

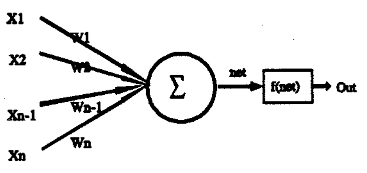

Figure 1. Single neuron model.

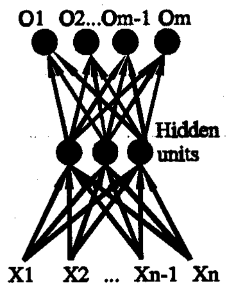

Figure 2. Feedforward neural network.

TRAINING ALGORITHM

Training of the FNNs using proposed approach requires the following steps:

Step 1. Set all weights to small random values.
Step 2. Present inputs and desired outputs.
Step 3. Calculate actual outputs.
Use the formulas as in (1) and (2) to calculate outputs.
Step 4. Adjust weights.
Use a recursive algorithm error backpropagation [Rum86, Mul94] starting at the output nodes and working back to the hidden layer.

Step 5. Repeat by going to Step 2 (After some iterations go to Step 6).
Step 6. Use data reduction algorithm [Ver94c]; calculate net_1, net_2,..., net_p (p-number of training pairs) for the output layer using Equation (4).
Step 7. Provide a linear system of equations as follows:
$H\underline{w}=\underline{net}$, where H - input matrix for the output layer; $\underline{w}$ - weight vector of the output layer and $\underline{net}$ is calculated from (4).
Step 8. Calculate weights of the output layer using modified Gram-Schmidt method.
Repeat Step 6 through 8 for each neuron of the output layer.

IV. COMPUTER SIMULATIONS

The proposed algorithm (GOT) has been tested on artificial problems such as the parity problem, exclusive nor problem, iris problem, sonar problem, on real problems arising in pattern recognition, and on a variety of other problems from various sources. The algorithm has been implemented in C on an HP-UX 715/50 workstation. The Root-Mean-Square (RMS) error is calculated using following formula.

$$RMS = \frac{\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{m}(d_{ij}-o_{ij})^2}}{m*p} \tag{6}$$

where m is the number of output nodes, p is the number of training pairs, d_{ij} and o_{ij} are the desired and calculated values for the *i*th pair and *j*th output.

Experiment 1. Iris Data Classification Problem
The particular problem is that of classifying examples of different kinds of iris flowers into one of three species of irises: setosa, versicolor, and viginica. There are 150 instances, 50 for each class; instances are described using four features: sepal width, sepal length, petal width, and petal length. The units for all four are centimeters, measured to the nearest millimeter. A learning rate of 0.5 and momentum of 0.4 is used. A number of iterations, training time, RMS error, and average performance for the proposed and error backpropagation algorithms are shown in Table 1. The proposed algorithm shows many times faster learning.

Table 1. Comparative results on the iris data classification problem.

# of hidden units	# of iterations	Training time [s]	Gain term η	Performance on training set [%]		Performance on test set [%]	
				EBP	GOT	EBP	GOT
6	300	8	0.5	95.55	97.77	95.55	95.55
12	100	9	0.5	84.44	98.88	83.33	97.77
12	150	11	0.5	86.66	98.88	86.66	97.77
12	600	45	0.5	95.55	96.66	94.33	96.66

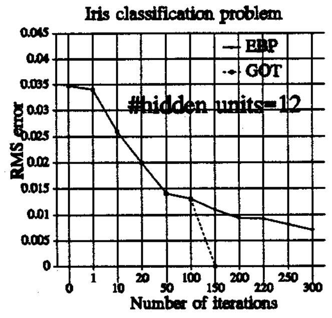

Figure 3. Comparison of rates of learning for iris data classification problem.

Experiment 2. Pattern Recognition Problem

The proposed algorithm has been tested using a pattern recognition problem, consisting of the recognition of character patterns encoded in as 8 x 8 pixel matrix, according to the IBM PC VGA character set. 36 different characters (0..9,A..Z) had to be recognized, corresponding to ASCII codes between 32 and 95. The number of input-output pairs is thus 36, and for each pair the input is a vector of 64 binary values, corresponding to the pixel matrix representing a character, while the output is a vector of 7 binary values, representing its coded ASCII value. Therefore, the used FNN's have 64 inputs and 7 outputs. We set all input-output pairs of values of the training set to 0.1 and 0.9 rather than to zero and one, respectively, to improve convergence. The network architectures with different numbers of hidden units are used for this problem. The results for different numbers of hidden units are presented in Table 2. Figure 4 depicts learning profiles produced for this problem and indicates that the proposed learning algorithm yields many times faster learning.

Table 2. Comparative results on the pattern recognition problem.

Training algorithm	# of hidden units	# of iterations	Training time [s]	Gain term η	Performance on training set [%]	Performance on test set [%]
EBP	12	200	40	0.5	90.90	63.63
	20	100	20	0.5	27.27	21.21
	20	200	90	0.5	100.0	69.69
	20	300	105	0.5	100.0	72.72
GOT	12	200	40	0.5	96.97	57.57
	20	100	40	0.5	100.0	63.63
	20	200	90	0.5	100.0	69.69
	36	1	7	0.5	100.0	19.20

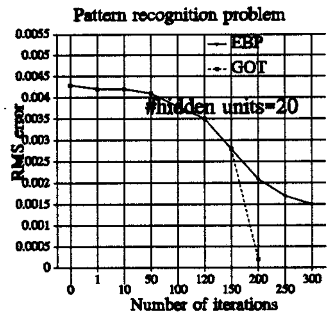

Figure 4. Comparison of rates of learning for pattern recognition problem.

Experiment 3. Sonar Problem, Mines vs. Rocks

This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. The data set, is in the standard CMU Neural Network Benchmark format. The data has 60 continuous inputs and 2 enumerated outputs. This data set can be used in a number of different ways to test learning speed, quality of ultimate learning, ability to generalize, or combinations of these factors. There are 208 patterns in total with 111 belonging to the "metal" class and 97 belonging to the "rock" class. These 208 patterns are divided between the 104-member training set and the 104-member test set.

The purpose of this experiment is to compare the performances of the proposed algorithm and the conventional EBP with different numbers of hidden units. A learning rate of 0.2 and momentum of 0.5 is used. Errors less than 0.2 are treated as zero. Initial weights are uniform random values in the range -0.3 to +0.3.

As shown in Table 3 the performance of the proposed algorithm and conventional EBP is best for 22 and 12 hidden units respectively and the proposed algorithm shows better generalization and faster learning.

Table 3. Comparative results on the sonar problem, Mines vs. Rocks

Training algorithm	# of hidden units	# of iterations	Training time [s]	Gain term η	Training set % right on	Test set % right on
EBP	12	300	90	0.2	100.0	93.26
	22	150	125	0.2	100.0	86.53
	22	300	260	0.2	100.0	84.50
	104	300	830	0.2	81.73	47.11
GOT	12	300	90	0.2	100.0	94.23
	22	150	125	0.2	100.0	95.19
	22	300	260	0.2	100.0	95.19
	104	1	20	0.2	100.0	81.73

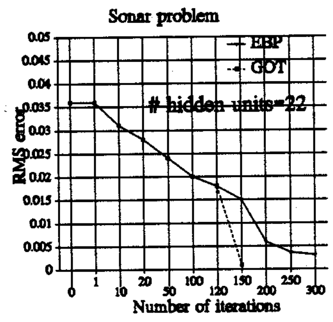

Figure 5. Comparison of learning profiles for sonar problem.

V. CONCLUSIONS

We have shown how to train a feedforward neural network by combining gradient-descent and QR-decomposition modified Gram_Schmidt (MGS) methods. The experiments show that our algorithm (GOT) achieves recognition accuracy as good as or better than FNNs trained using error backpropagation algorithm (EBP), and the training process is much faster than EBP. This is true even if various modifications are made to speed up the convergence of EBP. Also there is no chance for local minima because direct methods used in GOT for training the output layer does not have such problems as local minima.

REFERENCES

[Her91] Hertz, J., Krogh, A. and Palmer, R. (1991). Introduction to the Theory of Neural Computation. Addison-Wesley Publishing Company, USA.

[Kie92] Kielbasinski, A. and Schwetlick, H. (1992). Numerical Linear Algebra, Warsaw.

[Mul94] Mulawka, J.J. and Verma, B.K. (1994). Improving the Training Time of the Backpropagation Algorithm, International Journal of Microcomputer Applications, vol. 13, no.2, pp.85-89, 1994, Canada.

[Rie93] Riedmiller, M. and Braun, H. (1993). A Direct Method for Faster Backpropagation Learning: The RPROP Algorithm, Proceedings IEEE International Conference on Neural Networks, San Francisco, California, March 28-April 1, vol.1, pp.586-591.

[Rum86] Rumelhart, D.E., Hinton, C.E. and Williams, R.J. (1986). Learning International Representations by error propagation, In parallel distributed processing: Explorations in the microstructures of cognition, Cambridge: MIT press.

[Ver94a] Verma, B.K. and Mulawka, J.J. (1994). Training of the Multilayer Perceptron using Direct Solution Methods, Proceedings of the Twelfth IASTED International Conference, Applied Informatics, pp. 18-24, May 17-20, 1994, Annecy, France.

[Ver94b] Verma, B.K. and Mulawka, J.J. (1994). A Modified Backpropagation Algorithm, Proceeding of the World Congress on Computational Intelligence, vol.2, pp.840-846 26 June-2 July 1994, Orlando, Florida, USA.

[Ver94c] Verma, B.K. (1994). New Methods of Training the Multilayer Perceptrons, Ph.D. Dissertation, Warsaw University of Technology, Warsaw, Poland.