INTRODUCING STOCHASTIC PROCESSES WITHIN THE BACKPROPAGATION ALGORITHM FOR IMPROVED CONVERGENCE

ARTHUR SALVETTI AND BOGDAN M. WILAMOWSKI
University of Wyoming
Department of Electrical Engineering
Laramie, WY 82071

ABSTRACT:
The backpropagation algorithm is well known as the best alternative developed so far to train multilayer feedforward neural networks. However, as a deterministic method based on a gradient descendent of an error function, it has some limitations on the convergence speed. This is due to either the presence of large flat plateau or local minima in the topography of the function surface. In order to overcome these limitations, several variations and alternatives to this method have been proposed. For example, a common solution is to reinitialize the training every time a local minimum was found. Stochastic procedures such as the annealing method, and other combination of methods have already been applied. In this paper, two new methods were developed by introducing stochastic elements within the weight update procedure. Rather than being a batch process, the random factor happens on real time, being concomitant to the deterministic process, what may save some important training time. In this way, not only higher but also faster convergency were achieved. The proposed modified algorithms were experimented with the XOR problem and the results obtained are reported and compared to the backpropagation.

INTRODUCTION

The backpropagation (Werbos, 1974)(Rumelhart, McClealland, 1986) is one of the best and most used algorithms developed so far to train feedforward multilayer neural networks. However, as a deterministic gradient descendent algorithm, it has limitations on its speed and quality of convergence. The slow speed has been attributed to the presence of large flat plateaus on the surface of the error function while the quality of convergence is dependable on the presence of several local minima on this same surface (Fahlman, 1988)(Werbos, 1993). These problems become even more evident as the complexity of network and the size of the application increases.

Several modifications have been proposed in order to overcome these obstacles and make the algorithm more efficient. To mention some, we have the momentum method (Silva, Almeida, 1990), quickprop (Fahlman, 1988) and modified derivative (Wilamowski, Torvik, 1993). Some others interesting kind of approaches involve the

of random processes within the backpropagation algorithm. A common way to introduce randomness in the process is to restart the training with a new set of random weights every time it fails to give an appropriate solution (Kolen, 1988). In spite of the fact that this may improve the quality of convergence, it seems unpractical and time consuming when large problem applications are considered. Rather than introducing stochastic processes in a batch mode as suggested by the procedure above, real time introduction of stochastic elements have also been proposed. Some random factor are introduced and they are concomitant with the steps of the deterministic process. Some of the modifications that have been proposed are : addition of noise to the weights during training (Von Lehmen, 1988), stochastic weight adjustment rule (Hanson, 1990) and simulated annealing based modifications (Owen, 1993). The idea is to achieve better solutions by somehow getting out of local minima.

In this paper, some variations of introducing random processes within the backpropagation are proposed and investigated. The resultant algorithms are compared to the standard back propagation in order to evaluate the importance of random factors for improved convergence.

RANDOM SELECTION OF WEIGHTS

Several different modifications were tested and compared with the standard backpropagation. By standard backpropagation we mean that it has no momentum factor and that the patterns were presented always in the same sequence.

One modification consists on randomly changing the sequence in which the patterns are presented during training. This procedure is already widely used but it was here included for comparison.

Other modification consists on randomly select the weights to be updated. Instead of being always updated in each step, weights are updated one by one in a random manner. In this way, each time a pattern is presented not all weights are updated as usual but just the randomly selected ones. In order to keep track of the number of calculations involved in the process, one iteration is considered when the same number of weight updates occur as if it was a standard backpropagation iteration.

The standard backpropagation technique was able to converge only in 60% to 70% of cases depend on the initial randomly chosen weights. With suggested modifications process was able to converge in 85% of cases. The results were observed in 200 trials for each category. The problem used was the classical XOR for a network with two neurons in the hidden layer. Also the number of required iterations were slightly reduced. Fig. 1 shows errors as a function of iterations for standard backpropagation approach. Fig. 2 Shows errors as a function of iterations for case of random pattern presentation, and Fig. 3 shows error as a function of iteration for the case of the random selection of weights. All comparison was done using classical exclusive OR example. The error was defined as the sum of the squares of the errors for each output for each iteration. One iteration was defined as the cycle where all patterns are presented. The terminating error was chosen to be 0.1.

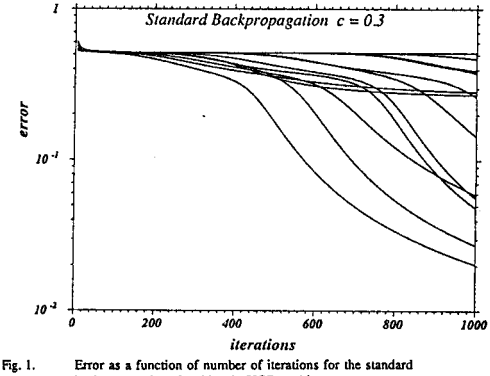

Fig. 1. Error as a function of number of iterations for the standard backpropagation algorithm in XOR problem

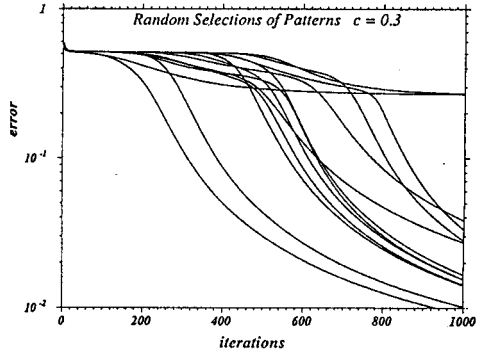

Fig. 2. Error as a function of number of iterations for the backpropagation algorithm with randomly selected patterns in XOR problem

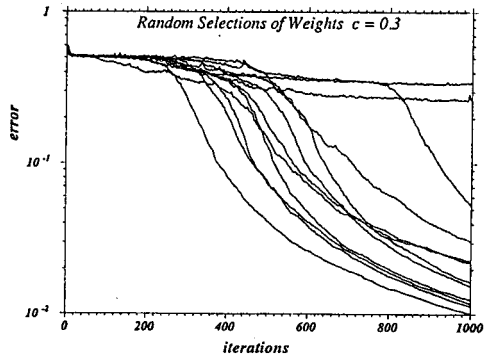

Fig. 3. Error as a function of number of iterations for the backpropagation algorithm with randomly selected weights for updating in XOR problem

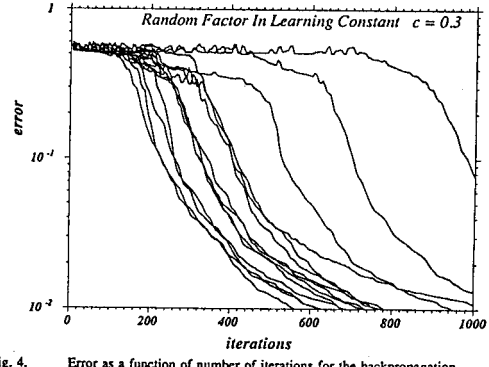

Fig. 4. Error as a function of number of iterations for the backpropagation algorithm with random factor in the learning constant for XOR problem. The average learning constant was set to c = 0.3.

RANDOM FACTOR IN LEARNING CONSTANT

Other modification uses a random factor within the learning constant. Each time the derivative De/Dw factor reaches a low threshold value, the learning constant starts having a uniform random distribution around its central value. The lower the value of De/Dw, the higher the range of the distribution. As the error gets more stable the wider ranger of the learning constant can be assumed. On the other hand, if the derivative De/Dw has a large value than mean value of the learning constant is used. In this way, the average value of the learning constant remains the same, but the probabilistic distribution around this average increases as De/Dw decreases. The learning constant is computed from the following equation :

$$c = c_0 + (\text{random}(2) - 1)\exp\left(-\frac{dE}{dw}\right)\frac{\text{random}\left(\frac{dE}{dw}\right)}{\frac{dE}{dw}} \qquad (1)$$

Fig. 4 shows errors as a function of iterations for the case when learning constant is c=0.3. Figure 5 and 6 show similar results for the average learning constant c=7. One may observe very significant improvement in the learning speed.

Table I show the convergence in percentage and the average speed of convergence in number of iterations. In this table, all four algorithms used a learning constant (or its average) of 0.3.

TABLE I: Convergence (in %) was estimated using 200 trial runs and Iteration required to reach an error below 0.1. This also was estimated based on the 200 runs. When process did not converged 1000 iterations were used for computation of the average. In all cases the learning constant c = 0.3 was used.

Algorithm	Convergence	Iterations required
Standard Backpropagation	61%	971
Random Selection of Patterns	90%	627
Random Selection of Weights	85%	650
Random Factor in Learning Constant	99%	500

CONCLUSION

It can be noted that all three modifications improved either the convergence quality or speed. The random presentation of patterns seems important to improve the probability of convergence and speed. The random selection of weights to be updated also seems to have influence on both. Certainly, it is the proposed introduction of the random factor within the learning constant that caused the most important changes in both the convergence and the speed.

The results suggest that random factors, when properly designed to operate within the backpropagation, may not only improve the quality of convergence but also speed up the process. This suggests that random factors also have an important role on the plateau regions as well as in the local minima. Further research is

suggested in order to define how properly these modifications should be designed.

REFERENCES

Fahlman, S., (1988). "Faster-Learning Variations on Back-Propagation: an Empirical Study," Proceedings of the 1988 Connectionist Models Summer School.

Hanson,S.J., (1990). "Behavioral Diversity, Search and Stochastic Connectionist Systems," Quantitative Analysis of Behavior: Neural Network Models of Conditioning and Action. Harvard Press.

Kolen, J.F., (1988). "Faster Learning Through a Probabilistic Approximation Algorithm," IEEE International Conference on Neural Networks, San Diego, CA.

Owen, C.B., (1993). "A Stochastic Approach to Backpropagation Using Simulated Annealing,"

Rumelhart, D.E. and McClelland, J.L., (1986). Parallel Distributed Processing: "Explorations in the Microstructure of Cognition," vol. 1, The MIT Press.

Silva, F.M., Almeida, L.B, (1990). "Acceleration Technique for the Backpropagation Algorithm." in Neural Networks EURASIP workshop, Sesimbra, Portugal, February .

Von Lehmen,A., Paek,E.G., Liao,P.F., Marrachi,A. and Patel,J.S. (1988). "Factors Influencing Learning by Backpropagation," IEEE International Conference on Neural Networks, San Diego, Sciences," Ph.D. thesis, Harvard University.

Werbos, P., (1993). "Supervised Learning: Can It Escape Its Local Minimum ?," Proceedings of the 1993 World Congress on Neural Networks, San Diego, CA.

Wilamowski, B.M., Torvik, L.M., (1993). "Modification of Gradient Computation in the Back Propagation Algorithm." Proceedings of ANNIE'93 - Artificial Neural Networks in Engineering, #: 175-180, ASME Press 1993.

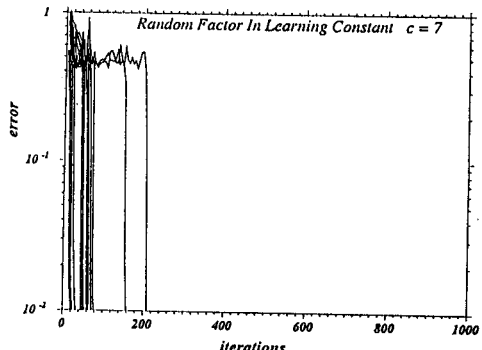

Fig. 5. Error as a function of number of iterations for the backpropagation algorithm with random factor in the learning constant for XOR problem. The average learning constant was set to c = 7.

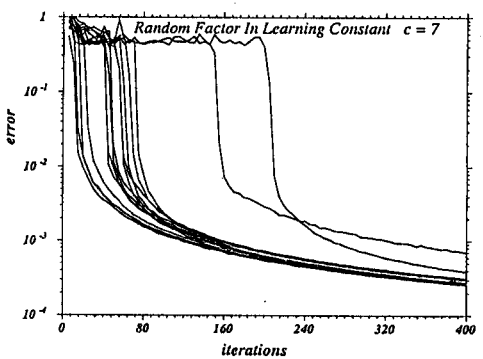

Fig. 6. Error as a function of number of iterations for the backpropagation algorithm with random factor in the learning constant for XOR problem. The average learning constant was set to c = 7. Drawn in extended scale.

INTELLIGENT ENGINEERING SYSTEMS THROUGH ARTIFICIAL NEURAL NETWORKS

VOLUME 4

Editors:

Cihan H. Dagli
Benito R. Fernández
Joydeep Ghosh
R. T. Soundar Kumara