Efficient Learning through Cooperation[*]

R. Venkateswaran
rvenkate@eecs.wsu.edu

Zoran Obradović
zoran@eecs.wsu.edu

School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164-2752

Abstract

A new algorithm has been proposed which uses cooperative efforts of several identical neural networks for efficient gradient descent learning. In contrast to the sequential gradient descent, in this algorithm it is easy to select learning rates such that the number of epochs for convergence is minimized. This algorithm is suitable for implementation on a parallel or distributed environment. It has been implemented on a network of heterogeneous workstations using p4. Results are presented where few learners cooperate and learn much faster than if they learn individually.

1 Introduction

The goal of supervised learning from examples is generalization using some preclassified inputs (training set). Learning in neural networks is achieved by adjusting the connection strengths (*weights*) among processors, so that the outputs reflect the class of the input patterns. One popular method of adjusting the weights is gradient descent learning through back-propagation [8]. Unfortunately, in the back-propagation algorithm, a number of parameters have to be appropriately specified. If parameters are not appropriate, the algorithm can take a long time to converge or may not converge at all [7]. Due to local minimum problem, an appropriate learning rate significantly affects the quality of the generalization and the number of epochs for convergence [2]. Selection of an appropriate learning rate is a computationally expensive experimental problem that can be solved satisfactorily for small networks only [5].

The goal of this paper is to speed-up learning with improved accuracy using systems composed of several neural networks of the same topology that concurrently run the standard back-propagation algorithm. Our approach is different from the approach in [6] where each network learns a subset of training examples. In our system, the various networks periodically communicate with each other and cooperate in learning the entire training set. If any of the processes gets stuck in a local minimum site, the rest of the processes help in moving it out of this predicament. The algorithm also works well if any process gets stuck in a plateau or a ridge.

In Section 2, we propose this new cooperative learning algorithm, followed by experimental results in Section 3 and analysis in Section 4.

2 Cooperative Learning Algorithm

In our algorithm, several processes run the standard back-propagation algorithm concurrently. All processes work on neural networks of identical topology, each using a local copy of the training set. These processes are called the *slave* processes. A *master* process initiates these slaves and controls them. The slaves communicate only with the master. The master initializes its *hypothesis* (the weights and the bias values of the neurons) and broadcasts it to the slaves. The slaves adjust this hypothesis using back-propagation. Each slave uses its own learning rate that is different from the learning rates of other slaves and hence, the adjusted hypothesis in each of the slaves is different.

[*]Research sponsored in part by the NSF research grant NSF-IRI-9308523.

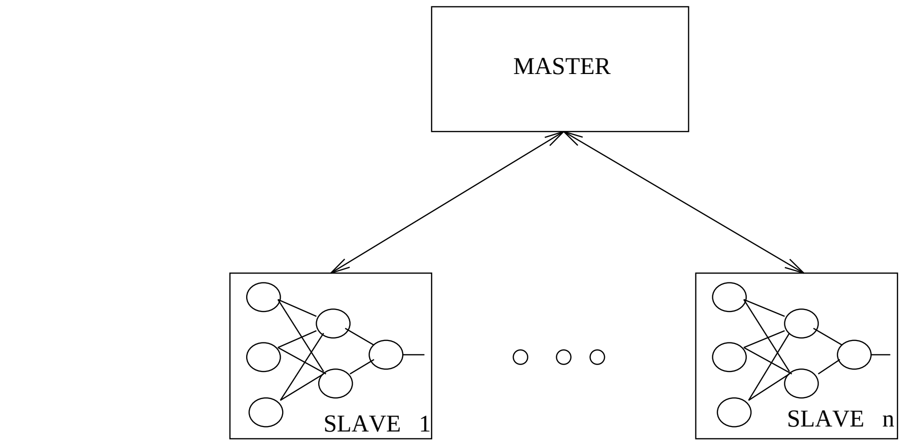

Figure 1: Communication Network Topology for Cooperative Learning Algorithm

Periodically, the slaves cooperate by exchanging information. The period between two cooperations is called an *era*.

The algorithm is suitable for implementation on a distributed platform since the communication graph is simple and the total number of communications is small (Figure 1). Our implementation uses *p*4 which supports parallel programming for both distributed environments and highly parallel computers [1]. It helps to create the master and the slave processes and provides easy means of communication between them. Another advantage of using *p*4 for neural networks implementation is its ability to port directly from a distributed to a highly parallel platform [4].

2.1 Epoch-based Cooperation

In *epoch-based cooperation,* the slaves communicate their learned weights back to the master after a specified number of epochs (one *era*). Since all slaves use the identical topology, the master forms a new hypothesis after each era by averaging these weights. For each link between neurons, the new weight is the average of the weights of that link as computed by the slaves. This hypothesis is broadcast to the slaves and they proceed with back-propagation for the next era starting from this new hypothesis. When any of the slaves has learned the training set to satisfaction, the hypothesis learned by this slave is output and learning is completed.

2.2 Time-based Cooperation

One disadvantage of the epoch-based cooperation is that the slaves on faster machines finish their era earlier but they have to wait for the slowest slave to finish its era. So, in a heterogeneous environment, the slowest machine is a bottle-neck and one cannot take advantage of faster machines. For such heterogeneous environments, we propose another approach called the *time-based cooperation.* Here, the era is specified as a duration of time rather than number of epochs. Since all slaves run for the same duration, no machine will be idle.

2.3 Cooperation with Dynamic Learning Rates

In this approach, we start with the cooperative algorithm (epoch or time based) using initial learning rates spread uniformly in (0,1) range. After few eras, the range of the learning rates is reduced. New values for the learning rates are chosen uniformly around the value of the learning

Problem :	To classify 'A', 'I' and 'O'.		
Dimensionality :	2	Number of Classes :	3
Architecture :	2-9-3	Size of Training Set :	16
Percentage Learned :	100%	Era :	50 epochs

One Node	η	0.01	0.05	0.1	0.125	0.15	0.175	0.2
	Epochs	8708	1819	1295	1179	1419	>10000	>10000

Two Nodes	$\eta1$, $\eta2$	0.05 , 0.15	0.1 , 0.15	0.01 , 0.15	0.05 , 0.175
	Epochs	1099	1040	1374	985

Three Nodes	$\eta1$,$\eta2$,$\eta3$	0.05 , 0.10 , 0.15	0.01 , 0.1 , 0.2
	Epochs	1149	1148

Four Nodes	$\eta1$, $\eta2$, $\eta3$, $\eta4$	0.05 , 0.1 , 0.15 , 0.2
	Epochs	946

Figure 2: Epoch-based Cooperation for Pattern Classification Problem

rate of the slave which currently generalizes the best. The advantage of this approach is that the selection of an optimal learning rate becomes completely automatic.

3 Results

Two benchmark problems are used for experimentation. The experiments are performed by varying the number of slaves from one to four. Both the epoch-based and the time-based cooperation are tested.

3.1 Pattern Classification Problem

The problem is to classify three patterns, 'A', 'I' and 'O', formed in a 4-by-4 grid, using a feedforward network. Figure 2 gives the results of the epoch-based cooperation for this problem. In this figure, One Node table gives the number of epochs required to learn the training set using sequential back-propagation algorithm with various learning rates. The number of epochs to learn the training set using the cooperative system of two slaves with various pairs of learning rates is given in Two Nodes table. Here, one slave uses the learning rate $\eta1$ and the other uses $\eta2$. Similarly, other tables show results for cooperative systems of three and four slaves respectively.

3.2 Two-Spirals Problem

This hard benchmark problem consists of two classes of points arranged in two interlocking spirals that go around the origin [3]. The goal is to develop a feed forward network that classifies all the training points correctly. The results of the epoch-based cooperative learning algorithm on a training set of 40 points are given in Figure 3.

Figure 4 gives results of the time-based cooperative algorithm. In one experiment, the cooperative algorithm using two slaves is run on a homogeneous system consisting of two DEC3100

Problem :	Two-Spirals Problem.		
Dimensionality :	2	Number of Classes :	2
Architecture :	2-5-1	Size of Training Set :	40
Percentage Learned :	100%	Era :	100 epochs

One	η	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.5
Node	Epochs	>30000	8799	3683	2593	2238	1727	>30000	>30000	>30000

Two	$\eta 1$, $\eta 2$	0.05 , 0.35	0.15 , 0.35	0.25 , 0.35	0.05 , 0.5
Nodes	Epochs	2691	2122	1777	1920

Three	$\eta 1$,$\eta 2$,$\eta 3$	0.1 , 0.3 , 0.5	0.15 , 0.25 , 0.35
Nodes	Epochs	1775	2103

Four	$\eta 1$, $\eta 2$, $\eta 3$, $\eta 4$	0.1 , 0.2 , 0.3 , 0.4
Nodes	Epochs	2498

Figure 3: Epoch-based Cooperation for Two-Spirals Problem

Problem :	Two-Spirals Problem.		
Dimensionality :	2	Number of Classes :	2
Architecture :	2-5-1	Size of Training Set :	40
Percentage Learned :	100%	Era :	400 msec

Two	η_{DEC1} , η_{DEC2}	0.05 , 0.35	0.15 , 0.35	0.25 , 0.35	0.05 , 0.5
Nodes	Cooperations	40	34	28	27

(a)

Two	η_{DEC} , η_{HP}	0.05 , 0.35	0.15 , 0.35	0.25 , 0.35	0.05 , 0.5
Nodes	Cooperations	6	6	6	5

(b)

Two	η_{HP} , η_{DEC}	0.05 , 0.35	0.15 , 0.35	0.25 , 0.35	0.05 , 0.5
Nodes	Cooperations	22	6	7	21

(c)

Figure 4: (a) Homogeneous and (b,c) Heterogeneous System for Cooperative Learning

workstations. The slave on one of the workstations uses learning rate η_{DEC1} while the other slave on the other workstation uses η_{DEC2}. The number of cooperations required for convergence using various pairs of learning rates are given in Figure 4 (a). In the other experiment two slaves are run on a heterogeneous system consisting of the faster HP9000/735 and the slower DEC3100 workstation. In the pair of learning rates given in Figure 4 (b) the left value is used by the slave on the DEC3100 and the right value by the slave on the HP9000/735.

Similar results are obtained for training set of 80 points. Here, the range of good learning rates for the sequential algorithm is smaller than for 40 points.

4 Analysis of Experimental Results

4.1 Epoch-based experiments

Let η_{min} be the learning rate that minimizes the number of epochs for convergence in standard back-propagation. From the experiments it can be observed that if the learning rates for the slaves in the cooperative algorithm are chosen such that $\eta < \eta_{min}$ for some slaves, and $\eta > \eta_{min}$ for the remaining slaves, then, in general, the cooperative algorithm needs significantly smaller number of epochs to converge. For instance, suppose that there are two slaves using learning rates η_1 and η_2. In order to get a performance better than the sequential algorithm, we choose the learning rates η_1 and η_2 so that $\eta_1 < \eta_{min} < \eta_2$. For the pattern classification problem, it is easy to see from One Node table in Figure 2 that the fastest convergence for the sequential algorithm takes 1179 epochs with $\eta = 0.125$. By setting $\eta_1 = 0.05$ and $\eta_2 = 0.175$, the cooperative learning algorithm takes only 985 epochs for convergence. Without any cooperation, the algorithm takes 1819 and 10000 epochs for convergence for $\eta = 0.05$ and $\eta = 0.175$ respectively. Similarly, in Figure 3, the fastest convergence for the sequential algorithm takes 1727 epochs for $\eta = 0.3$. With the learning rate set to 0.25 and 0.35 the non-cooperative algorithm takes 2238 and 30000 epochs respectively. But, with cooperation, the convergence takes 1777 epochs, which is very close to the fastest sequential convergence.

In sequential back-propagation, learning rates less than and greater than η_{min} exist if the number of epochs for convergence is a non-monotonic function of the learning rate, which is true for many real-life problems. For these problems, cooperative algorithm will work better, provided appropriate learning rates are selected. The XOR problem is an example where the number of epochs is a monotone decreasing function of the learning rate. So, for this problem, cooperative learning does not give a better performance.

4.2 Time-based experiments

Here, the time between two cooperations (one era is 400ms in our experiments) is fixed. So, the total time for convergence of the time-based cooperation is proportional to the product of the number of cooperations and the execution time of one era. From the Figure 4, it can be observed that the time-based cooperation executed on a heterogeneous system with one fast and one slower machine converges much faster than on a homogeneous system with two slower machines. Also, the algorithm is more efficient if the slave with the higher learning rate is assigned to the faster machine (see Figure 4 b,c). It is clear that the slave on the faster machine executes more epochs per era than the slave on the slower machine. So, if the slave with the smaller learning rate is assigned to the faster machine, the weights computed by the two slaves are not very far apart. Consequently, averaging is not so beneficial in this case.

5 Conclusion

The cooperative learning algorithm proposed here has given promising results. In general, for the back-propagation algorithm, it is very hard to find learning rates for which the algorithm converges in minimum number of epochs. In our algorithm, we can easily select the learning rates such that the number of epochs for convergence is close to this minimum or even better. This approach can be used to improve any gradient descent algorithm. It can be easily implemented on a parallel machine or a network of heterogeneous workstations using *p*4.

The experimentation using cooperation with dynamic learning rates is still under investigation with promising preliminary results. We are also experimenting with a more sophisticated way of combining slave hypotheses (instead of averaging), which might further improve the performance.

References

[1] R.Butler and E.Lusk,"*User's Guide to the p4 Parallel Programming System*," Argonne National Lab., November, 1992.

[2] J.P.Cater, "Successfully Using Peak Learning Rates of 10 (and greater) in Back-Propagation Networks with the Heuristic Learning Algorithm," *IEEE First Int. Conference on Neural Networks,* vol. 2, pp. 645-651, 1987.

[3] S.Fahlman and C. Labiere, "The Cascade Correlation Learning Architecture," *Advances in Neural Information Processing Systems*, vol. 2, Morgan Kaufmann, pp. 524-532, 1990.

[4] J. Fletcher and Z. Obradovic, "Parallel and Distributed Systems for Constructive Neural Network Learning," *IEEE Second Intl. Sym. on High Performance Distributed Computing,* pp. 174-178, 1993.

[5] J.Hertz et al, *"Introduction to the Theory of Neural Computation,"* Addison Wesley, 1991.

[6] R.Jacobs et al, "Adaptive Mixture of Local Experts," *Neural Computation,* vol. 3, no. 1, pp. 79-87, 1991.

[7] S.J.Orfanidis, "Gram-Schmidt Neural Nets," *Neural Computation*,vol.2, pp. 116-126, 1990.

[8] D.E.Rumelhart, G.E.Hilton and R.J.Williams, "Learning Internal Representations by Error Propagation," *Parallel and Distributed Processing*, Eds. D.E.Rumelhart and J.L.McClelland, Cambridge, MA, MIT Press, 1986.