Mapping Back Propagation Algorithms onto Web-based Parallel Systems for Feed Forward Neural Networks

Joon-Min Gil, Youn-Hee Han, Chong-Sun Hwang, Young-Sik Jeong †

Dept. of Computer Science and Engineering, Korea Univ.
1, 5-Ga, Anam-Dong, SeongBuk-Gu, Seoul 136-701, KOREA
{jmgil, yhhan, hwang}@disys.korea.ac.kr
† Division of Computer and Communication Engineering, WonKwang Univ.
Iksan-Shi, Chollabuk-Do 570-749, KOREA
ysjeong@wonnms.wonkwang.ac.kr

Abstract. Because of the intrinsic high degree of parallelism of the back propagation (BP) algorithms, it is necessary for the BP algorithms to be implemented in parallel environment. Moreover, regardless of the restriction of computational power and the effect on the communication, the parallelism of the BP algorithms must be developed. In this paper, we propose a web-based parallel system which can efficiently reduce the training time by solving the high degree of parallelism of the BP algorithms in web environment. To implement the parallelism of the BP algorithm in web environment, we use farmer-worker model. In order to demonstrate the efficiency of our system, we apply our system to digit recognition problem. Experimental results show that our system has much speedup as compared with the target computer on which sequential BP algorithms run.

1 Introduction

Back propagation (BP) algorithms have been known as a suitable learning algorithm to make a feed-forward neural network. So, they have been successfully applied to various problems such as nonlinear control, signal processing, prediction, pattern recognition, and so on [6].

But, the main difficulty of the BP algorithms is to require much training time. To overcome the difficulty, some approaches have been proposed [2, 3, 7, 8, 10, 11]. They are classified into two categories:

– **Modification of the BP algorithms**: This method is to modify learning mechanism for the BP algorithms itself to reduce the training time. Various BP algorithms have been developed in [2, 3, 6, 10]. They seem to reduce the training time through the modification of architecture of neural network, the modification of learning phase, the reduction of computational complexity for weight-updating, and so on. However, another complexity occurs to maintain the modification of the BP algorithms consistently. Although the exact results are obtained through the modification of the BP algorithms, the minimal training time to make a feed-forward neural network is still required due to the intrinsic property of the BP algorithm.

– **Parallelism of the BP algorithms**: This method is to carry out the BP algorithms on parallel computers. Many algorithms have been developed so as to parallelize the BP algorithm [7, 8, 11]. Because of the intrinsic high degree of parallelism of the BP algorithms, parallel computers seem suitable to accelerate the training and speedup ratio seems to increase as the number of processors increases. However, the speedup ratio strongly depends on both parallelization methods of the BP algorithms and employed parallel computers. So, the BP algorithms on the parallel computers may have different performance according to the interconnection between processors and the number of processors. Consequently, in the parallel computers, machine-independent implementation for the parallization of the BP algorithms is impossible. Moreover, the parallel computers for specific applications are too expensive for end users to use easily.

In order to reduce the training time of the BP algorithms without the sacrifice of the exact results, it is necessary for the BP algorithms to be implemented in general parallel environment efficiently, regardless of the restriction of computational power and the the effect on communication.

Recently, the World-Wide-Web (hereinafter referred as the web) has been the largest virtual system that connects millions of computational nodes [1]. Because the scalability of the web is infinite, the restriction of computational power on the web may be vanished. To get a huge virtual system that can solve the high degree of parallelism of the BP algorithms, it is the web that we choose as platform for the parallel environment of the BP algorithms.

In this paper, we propose a web-based parallel system for the BP algorithms that can efficiently reduce the training time by solving the high degree of parallelism of the BP algorithms. To implement the parallelism of the BP algorithms in web environment, we use farmer-worker model. In the farmer-worker model, a farmer supervises all workers, distributes computational tasks to each worker, and balances the amounts of computations given to workers. The workers take the responsibility of performing the BP algorithms. We implement the web-based parallel system using the JAVA language and automatic mobile codes which have the advantage of such as platform-independent, extending of scalability, easy utilization, and etc.

The remainder of this paper is organized as follows: In section 2, we describe the parallelism of the BP algorithms in web environment. We propose a web-based parallel system for the BP algorithms in section 3. Experiments and results, which shows the efficiency of our system in terms of speedup and the optimization of the number of workers, are presented in section 4. Finally, our conclusions are given in section 5.

2 Parallelism of the BP Algorithms

2.1 BP Algorithms

A theoretical study of a general feed-forward neural network model is proposed in [4]. It shows that the BP algorithms can adapt to feed-forward neural network. The BP algorithms are based on the gradient descent method on the error surface:

$$E(w_{ij}, w_{jk}) = \frac{1}{2}\sum_p \sum_i ({d_i}^p - {o_i}^p)^2 \tag{1}$$
$${o_i}^p = s(\sum_j w_{ij} s(\sum_k w_{jk} {z_k}^p)))$$

where, ${d_i}^p$ and ${z_k}^p$ are the input and the desired output in the training pattern p, and ${o_i}^p$ is the output of neural network for the training pattern p. w_{jk} and w_{ij} are the weights of the neurons in the output and the hidden layers, respectively. Each neuron's output is transformed by the non-linear transformation function s (In general, sigmoid function is used). The weight changes Δw_{jk} and Δw_{ij} are calculated for every training pattern iteratively using the following gradient:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{2}$$

where, η is a learning rate. The weights are updated as follows:

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}. \tag{3}$$

There are two weight-updating methods, i.e., batch training and on-line training. In the on-line training, the equation (3) is calculated for every training pattern. It is more efficient than the batch training because it needs less iterations for the training convergence. Moreover, it can be simply implemented. So, it is preferred over the batch training.

On the contrary, in the batch training, Δw_{ij} and Δw_{jk} are averaged for all the training patterns, and the equation (3) is calculated after all the training patterns are presented in neural network. From the viewpoint of training time, the batch training requires less training time than the on-line training for one epoch because weight-updating is performed only a time for one epoch. The batch training requires the local storage to save the weight changes for each training pattern. It makes for the equation (2) to calculate independently. So, the batch training can be easily used for parallel environment.

To make the BP algorithms to parallel, a tradeoff between on-line training and batch training must be carefully considered. Next, we describe the parallelization of the BP algorithms so that each advantage for two methods is maximized.

2.2 Parallelization of the BP Algorithms

As described in previous subsection, batch training seems to be a suitable method to model the BP algorithms to parallel environment. However, it has the disadvantage of increasing the iteration for the training convergence. Also, it is important to consider the number of message passing which causes to increase communication cost between processors. In particular, in web environment, the communication cost is higher than parallel computers. Therefore, the implementation of parallel computation on the web leads to modify the parallel algorithm implemented on a specific parallel computer or to develop new parallelism method.

The parallelization of the BP algorithms can be divided into two methods [8]:

- **Pattern partitioning**: The batch training can be used in this method. In this method, all the training patterns are divided into some pattern blocks. The pattern blocks are allocated to each processor. All processors execute equation (2) for allocated pattern blocks independently. After one epoch, weights are updated on the average for the weight changes calculated in each processor. This method is a kind of the coarse-grained methods.
- **Network partitioning**: The on-line training can be used in this method. This method is applied to large neural networks. Its efficiency mainly depends on the density of the neural network connections. But, it is difficult to be implemented in parallel environment with the high communication costs because it needs communication between processors for every training pattern. This method is a kind of the fine-grained methods.

In general, the processors interconnected in the web have higher communication cost than that of parallel computers. So, it may be suitable for the pattern partitioning method in the web-based parallel system. However, the pattern partitioning based on batch training requires much training iteration than the on-line training. Therefore, it is requires that the parallelization of the BP algorithms in web environment be more efficient so that the communication between processors does not affect the overall performance and the simple implementation of the BP algorithms itself is maintained.

Consequently, it is necessary to be hybridized for both pattern partitioning and network partitioning so as to parallel the BP algorithms in web environment. To achieve such aim, a single neural network must be divided into multiple neural networks so that an individual neural network takes the responsibility for specializing a specific part among training patterns. And, weight-updating executed in the individual neural network influences only itself and is not averaged for all the training patterns as the batch training. Thus, the hybrid method of both pattern partitioning and network partitioning does not required the communication between processors and has the possibility to remove the communication overhead.

The neural network model described above is known as OCON (One Class One Network) [6, 9]. In the OCON model, the number of neural networks is equal

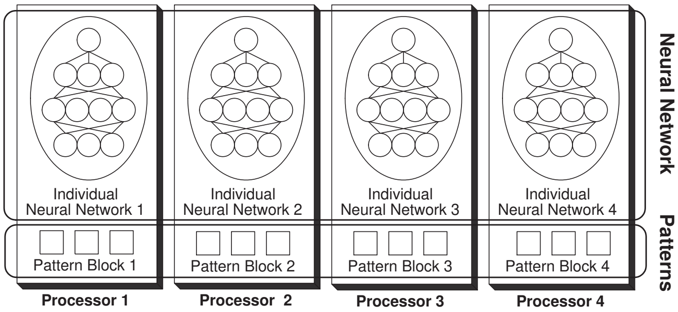

Fig. 1. Parallelization for the BP algorithms

to the number of output neurons. An OCON designed to recognize the digits, for example, has ten neural networks. Kung [6] and Tsay et al. [9] reported that an OCON may even require fewer overall hidden layer nodes. Also, in the simulation for pattern recognition, the OCON performed favorably in terms of training convergence and accuracy. In this paper, we use OCON as the neural network model to parallel the BP algorithms in web environment.

2.3 Mapping of the BP Algorithms onto Parallel Environment

In this subsection, we describe a strategy for mapping the BP algorithms onto parallel environment. Let $N = \{n_1, n_2, \ldots, n_o\}$ be an OCON and n_i be a feed forward neural network in the OCON. Let $D = \{d_1, d_2, \ldots, d_o\}$ be the training patterns and d_i be a pattern block. Let M be a mapping of an OCON, (N, D), onto a processor, p_i. The mapping is defined by

$$p_i = M(n_j, d_j) \tag{4}$$

where, $i = 1, 2, \ldots, P$ and $j = 1, 2, \ldots, o$. P and o are the number of processors and the number of output neurons, respectively.

In equation (4), an individual neural network and a pattern block are allocated to a processor. Each processor performs training for the weight of the allocated neural network using the allocated pattern block. After the weights of all the individual neural networks are trained in each processor, the construction of OCON for a given problem is completed. Figure 1 shows that an individual neural network and a pattern block are mapped onto each processor. Pattern partitioning and network partitioning are combined in this parallelization for the BP algorithms.

3 A Web-based Parallel System for the BP Algorithms

3.1 Architecture

The web-based parallel system for the BP algorithms consists of the following major components:

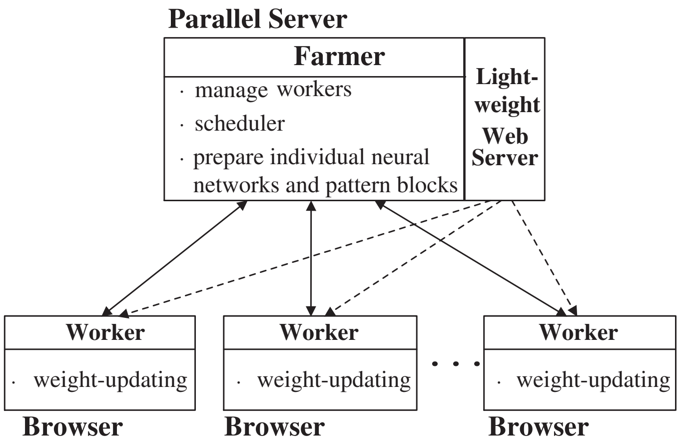

Fig. 2. Web-based parallel system for the BP algorithms

- **Farmer**: It takes the responsibility for registering and managing workers in each browser and prepares both individual neural networks and pattern blocks to be allocated to workers. Also, it maintains the communication relation between the farmer and each worker. A scheduler balances computational loads imposed on each worker.
- **Worker**: It makes the environment for executing the BP algorithms in its browser. Each worker performs weight-updating for an allocated individual neural network.
- **Light-weight Web Server**: It sends the HTML document embedding the worker to browsers on request. It implements the essential functionality needed to serve the worker, but it makes it possible for the worker process to run on any computer connected to the web.

Figure 2 shows what the components do and how they are related. In our web-based parallel system, each individual neural network and pattern block is allocated to a different worker, and weight-updating for the allocated pattern block is performed in each worker independently. In initial step, the farmer prepares individual neural networks and pattern blocks for allocating them to workers. According to the performance of each worker, a different number of individual neural networks and pattern blocks can be allocated to workers. In training step, a scheduler embedded in the farmer monitors the processing time of each worker periodically. After one epoch, the scheduler balances computational loads imposed on each worker by reallocating individual neural networks and pattern blocks to workers. A browser in each worker receives the mobile object, which includes an individual neural network and a pattern block, from the farmer. The browser on each worker performs weight-updating (equation (3)) for allocated individual neural networks and pattern blocks. By this way, training process in a worker is iteratively executed until all mapping examples from the pattern block allocated in the worker are learned within an acceptable overall error. After all workers finish training, the overall training process is completed.

3.2 Optimization of the Number of Workers

In the parallelization for the BP algorithms shown in figure 1, the speedup depends on the number of workers. The time saving rate through the parallel processing increases as the number of workers increases. In digit recognition, for example, if ten workers take part in training, our parallelization method can save ten times of the sequential processing.

The web-based parallel system is intended for enlarging overall computational performance using the workers, that have a different computational performance and are used with a low cost. It provides us with a virtual computational system that has the same computational performance as a target computer with powerful computational capability. To maximize the efficiency of the web-based parallel system, it is important to measure how many workers are used for obtaining the same performance as the target computer.

To compare the processing time of our system with that of the target computer, we consider the following parameters:

- W: The number of workers.
- T_w: The processing time for one pattern in the worker w.
- T_t: The processing time for one pattern in a target computer.
- I_w: The number of iterations for the convergence of an individual neural network in the worker w.
- I_t: The number of iterations for the convergence of a neural network in target computer.
- P_w: The number of patterns in the pattern block allocated to the worker w.
- P_t: The number of all the training patterns. Here, $P_t = \sum_{w=1}^{W} P_w$.

The condition that the web-based parallel system for the BP algorithms has less processing time than the target computer, is defined by

$$\frac{\sum_{w=1}^{W}(T_w \cdot P_w)}{W} \cdot I_w \leq T_t \cdot P_t \cdot I_t. \tag{5}$$

To simplify the equation (5), we assume that $I_w = I_t$. Also, if the number of patterns in all pattern blocks is the same, we can obtain the following relation: $P_t = W \cdot P_w$. Under these assumptions, we can obtain the number of workers which is correspondent to the performance of the target computer as follows:

$$W \geq \sqrt{\frac{\sum_{w=1}^{W} T_w}{T_t}}. \tag{6}$$

In web environment, each worker has various processing time. In this case, the overall performance of our system depends largely on the worker which

Table 1. Parameters

Parameters	Value
Number of neurons in input, first and second hidden layer	96, 200, 40
Tolerant error	0.1×10^{-5}
Learning rate	0.1
Number of learning iteration	2000
Number of training patterns	333
Number of tasks (output neurons)	10

has maximal processing time. Thus, the number of workers is optimized by the following equation:

$$W \geq \frac{\max_{w=1}^{W}\{T_w\}}{T_t} \geq \sqrt{\frac{\sum_{w=1}^{W} T_w}{T_t}}. \tag{7}$$

To predict the number of works through equation (7), the processing times of both target computer and workers must be measured. However, it is impossible to measure the processing times previously. In this paper, we obtain approximative processing time of both target computer and workers through benchmark program, which is similar in computation to the training of neural network.

3.3 Load Balancing

In the web, workers may have a different performance. Thus, the web-based parallel system for the BP algorithm must have the load balancing scheme so that the various computing power of workers does not affect overall performance. In our system, the farmer distributes tasks to workers in proportion to the performance of workers and the number of patterns in pattern blocks. Through the load balancing scheme, the different number of tasks may be distributed to a worker. So, each worker has almost the identical processing time for training, regardless of the different performance of workers.

4 Experiments and Results

In order to show the performance of the web-based parallel system, we applied our system to digit recognition problem, in which an input pattern consist of 8 by 16 bitmaps. The parameters used in the experiments are summarized in table 1.

The farmer, which was run on Pentium II 300, divides training pattern into ten pattern blocks corresponding to ten digits. It prepares an individual neural network per pattern block. The workers run in Internet Explorer 5.0 on heterogeneous computer, such as Pentium 166, 233, and 266. For the comparison of the performance, we used Pentium II 333 as target computer.

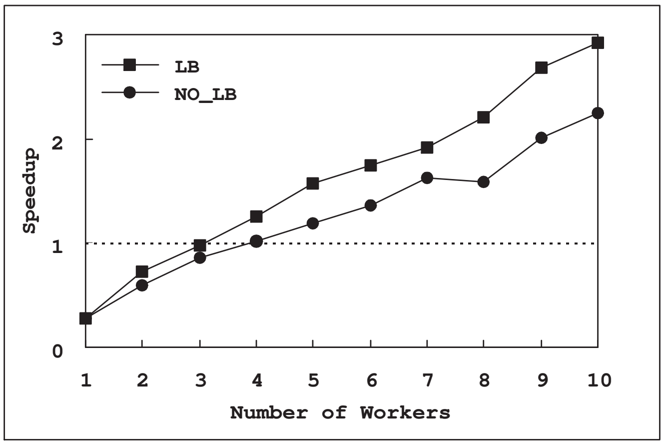

Fig. 3. Speedup of our system for target computer

Figure 3 shows the speedup of our systems for the target computer according to the change of the number of workers. In this paper, the speedup is defined as the processing time of the target computer divided by the processing time of our system using the given number of workers. The dashed line in figure 3 represents the threshold of the speedup. As shown in figure 3, as the number of workers increases, the speedup become more enhanced on the whole and the speedup curve become more remote from the dashed line.

Another goal of this paper is to predict the number of workers to be used in the web-based parallel system. In order to measure appoximative processing times of both target computer and workers, LINPACK benchmark [5] is used in our experiment. From the execution results of the LINPACK, we obtained that the approximative processing times of both target computer and workers are 0.0728 and 0.2474 seconds, respectively. Using equation (7), the number of workers which is correspondent to the performance of target computer, is calculated as

$$W \geq \frac{0.2474}{0.0728} = 3.398. \tag{8}$$

From the result of equation (8), we can obtain the computational systems corresponding to the performance of target computer if more than four workers are participated in training. This result closely agrees with the result of figure 3 (see the dashed line in figure 3).

In figure 3, NO_LB represents the web-based parallel system which does not use load balancing. On the contrary, LB represents the web-based parallel system which balances loads for given tasks in proportion to the performance of workers and the number of patterns in pattern blocks. As shown in figure 3, the LB has a steady increase. However, the NO_LB does not always increase as the number of workers increase. The reason is that the performance of the NO_LB depends on the worker whose processing time is maximal.

5 Conclusions

In this paper, we proposed a web-based parallel system for the BP algorithms that can efficiently reduce the training time by mapping the BP algorithms onto parallel environment. The farmer-worker model was used as the parallelism of the BP algorithms in web environment. Based on this model, we drove the number of workers which is correspondent to the performance of target computer. Also, in the web-based parallel system, the mapping of the tasks onto workers and the load balancing scheme were presented. Experiments for digit recognition problem were carried out on various computational resources in the web and the speedup of the proposed system for target computer increased steadily as the number of workers increases. Additionally, the number of worker obtained from the experiment well agreed with that obtained from theoretical derivation.

References

1. T. Brecht, H. Sandhu, M. Shan, and J. Talbot. ParaWeb: Towards world-wide supercomputing. *In Proc. of the 7th ACM SIGOPS European Workshop*: 181–188, 1996.
2. W.L. Buntine and A.S. Weigend. Computing 2nd derivatives in feed-forward networks - A review. *IEEE Tran. on Neural Networks*, 5(3): 480–488, 1994.
3. L. Fletcher, V. Katkovnik, F.E. Steffens. Optimizing the number of hidden nodes of a feedforward artificial neural network. *In 1998 IEEE Int'l Joint Conf. on Neural Networks Proc.*: 1608–1612, 1998.
4. R. Hecht-Nielsen. Theory of the backpropagation neural network. *In Proc. Int'l Joint Conf. on Neural Networks*, 1: 593–611, 1989.
5. K. Kant and M.M. Srinivasan. *Introduction to Computer System Performance Evaluation*. McGraw-Hill, 1992.
6. S.Y. Kung. *Digital neural networks*. Prentice Hall, 1993.
7. S.B. Nikola. Simulating artificial neural networks on parallel architectures. *IEEE Computer*, 29(3): 56–63, 1996.
8. A. Pétrowski. Choosing among several parallel implementations of the backpropagation algorithm. *In Proc. ICNN*: 1981–1986, 1994.
9. S. Tasy, P. Hong, and B. Chieu. Handwritten digit recognition via OCON neural network by selective pruning. *IEEE Proc. 11th Int'l Conf. Pattern Recognition*: 656–659, 1992.
10. N. Weymaere and J.-P. Matens. A fast and robust learning algorithm for feedforwad neural networks. *Neural Networks*, 4(3), 1991.
11. M. Yasunaga and E. Yoshida. Optimization of parallel BP implementation: Training speed of 1,056 MCUPS on the massively parallel computer CP-PACS. *In 1998 IEEE Int'l Joint Conf. on Neural Networks Proc.*: 563–568, 1998.