THE BULLETIN OF OKAYAMA UNIVERSITY OF SCIENCE No.37 A pp.97-105 (2001)

Parallel and Distributed Classification using Ensemble Self-Generating Neural Networks

Hirotaka INOUE and Hiroyuki NARIHISA*

Graduate school of Engineering,
Okayama University of Science,
1-1 Ridai-cho, Okayama 700-0005, Japan.
**Department of Information and Computer Engineering,*
Okayama University of Science,
1-1 Ridai-cho, Okayama 700-0005, Japan.

(Received November 1, 2001)

Abstract

In this paper, we present both improving capability of the accuracy and parallel efficiency of ensemble self-generating neural networks (ESGNNs) for classification on a MIMD parallel computer. Self-generating neural networks (SGNNs) are originally proposed on adopting to classification or clustering by automatic constructing self-generating neural trees (SGNTs) from the given training data. ESGNNs are composed of plural SGNTs each of which is independently generated by shuffling the orders of the given training data, and the output of ESGNNs is computed as the average of all SGNT's outputs. We allocate each of SGNTs to each of processors in the MIMD parallel computer. Experimental results show that the parallel model of ESGNNs improves the classification accuracy than the single SGNNs, goes on maintaining the high speed property of the single SGNNs.

1. Introduction

Neural networks have been widely used in the field of intelligent information processing such as classification, clustering, prediction, and recognition. Generally, for application of these neural networks, it is necessary to determine network structure and some parameters by human experts. It is quite tricky to choose the right network structure suitable for a particular application at hand. Concerning the design of the network structure, the following must be decided, (i) the number of the network layers, (ii) the number of the neurons of each layer, (iii) the weights on connection between consequent layers. During learning iterations, the weights on connections of the given networks are updated so as to converge to the target value conserving the initially decided static network structure. Obtaining the right structure of each network is the most important factor in learning and also the most difficult problem in the design of neural networks.

In order to avoid these tricky and difficult situations, an attention is focused on SGNNs because of their simplicity on networks design[12]. SGNNs are some kinds of extension of the self-organizing maps (SOMs) of Kohonen[6] and utilize the competitive learning algorithm which is implemented as self-generating neural tree.

The SGNT algorithm is proposed [11] to automatically generate a neural tree directly from training data. In our previous study concerning the performance analysis of the SGNT algorithm[4], we showed that the main characteristic of this SGNT algorithm was its high speed convergence in computation time but it was always not the best algorithm in its accuracy comparing with the existing other feed-forward neural networks such as the backpropagation (BP)[9].

In order to improve the generalization capability of SGNNs, we proposed ensemble self-generating neural networks (ESGNNs) for classification[5]. ESGNNs are applied to the ensemble averaging[3] of SGNNs and fully utilized the high speed convergence characteristic of the SGNT algorithm. Although ESGNNs improve a classification accuracy by using various SGNTs, the computation time and the memory capacity increase in proportion to increase in number of SGNTs.

The method of ESGNN has been studied by many researchers of AI and neural networks. Breiman proposed bagging predictors to improve the accuracy of CART[1] and investigated the bagging performance on CART and other methods for classification and regression problems[2]. Since the ensemble learning is a variance-reduction technique, it is well-known that the ensemble learning tends to work well for methods with high variance such as neural networks and tree-based methods.

In this paper, we present the improving capability of the accuracy and the parallel efficiency of ESGNNs for classification on a MIMD parallel computer. We apply ESGNNs to standard classification problems of MONK's[10], Cancer[8], and Card[8], which are given as benchmark problems.

2. Self-Generating Neural Networks

SGNNs are defined as a tree construction problem how to construct a tree structure from the given data which consist of multiple attributes under the condition that leaf neurons correspond to the given data. Before we describe the SGNT algorithm, we denote some notations.

- input data vector : $\boldsymbol{e}_i$; $\boldsymbol{e}_i = (e_{i1}, e_{i2}, \ldots, e_{ip})$, e_{ik} means the k-th attribute of $\boldsymbol{e}_i$.
- j-th neuron : n_j; n_j is expressed as ordered pair $(\boldsymbol{w}_j, c_j)$.
- weight vector of n_j : $\boldsymbol{w}_j$; $\boldsymbol{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jp})$.
- the number of the leaf neurons in n_j : c_j.
- tree is expressed as ordered pair $(\{n_j\}, \{l_k\})$, where $\{n_j\}$ is the neuron set and $\{l_k\}$ is the link set of the tree.
- distance measure : $d(\boldsymbol{e}_i, \boldsymbol{w}_j)$; we use Euclidean distance measure.
- winner neuron for $\boldsymbol{e}_i$: n_{win}.

The SGNT algorithm is a hierarchical clustering algorithm. The pseudo C code of the SGNT algorithm is given in Figure 1.

In the SGNT algorithm, some sub procedures are used. Table 1 shows the sub procedures of the SGNT algorithm and their specifications.

A weight w_{jk} of a neuron n_j is updated as follows:

$$w_{jk} = w_{jk} + \frac{1}{c_j + 1} \cdot (e_{ik} - w_{jk}), \tag{1}$$

Table1 Sub procedures of the SGNT algorithm

Sub procedure	Specification
$copy(n_j, \boldsymbol{e}_i/\boldsymbol{w}_{win})$	Create n_j, $\boldsymbol{w}_j \leftarrow \boldsymbol{e}_i/\boldsymbol{w}_{win}$.
$distance(\boldsymbol{e}_i, \boldsymbol{w}_j)$	Compute $d(\boldsymbol{e}_i, \boldsymbol{w}_j)$.
$choose(\boldsymbol{e}_i, n_1)$	Decide n_{win} for $\boldsymbol{e}_i$.
$leaf(n_{win})$	Check n_{win} whether n_{win} is leaf or not.
$connect(n_j, n_{win})$	Connect n_j as child of n_{win}.
$update(\boldsymbol{e}_i, \boldsymbol{w}_j)$	Update $\boldsymbol{w}_j$ of n_j.

Algorithm (SGNT generation) :
Input :
- A set of training examples $E = \{\boldsymbol{e}_i\}, i = 1, \ldots, N$.
- A threshold value $\xi \geq 0$.
- A distance measure $d(\boldsymbol{e}_i, \boldsymbol{w}_j)$.

Method :

```
1  copy(n_1, e_1);
2  for (i = 2, j = 2; i ≤ N; i++) {
3    n_win = choose(e_i, n_1);
4    minDistance = distance(e_i, w_win);
5    if (minDistance > ξ) {
6      if (leaf(n_win)) {
7        copy(n_j, w_win);
8        connect(n_j, n_win);
9        j++;
10     }
11     copy(n_j, e_i);
12     connect(n_j, n_win);
13     j++;
14   }
15   update(e_i, w_win)
16 }
```

Output : Constructed SGNT by E.

Fig.1 SGNT generation algorithm.

here, k is from 1 to p.

After all input data are inserted into the SGNT as its leaf neurons, the weights of each node neuron n_j are the averages of the corresponding weights of all its children. Whole SGNT reflects the given feature space by its topology. A winner leaf neuron n_{win} has the class information as a network output o_{win}.

In the testing process, the test data set T, which consists of data $\{(\boldsymbol{x}_i, y_i), i = 1, \ldots, M\}$, where $\boldsymbol{x}_i$ is the input vector , y_i is the corresponding output label, and M is the number of test data, is entered the root neuron of the SGNT.

Then the input data are reached one of the winner leaf neurons n_{win} of the SGNT through competition, and the desired output y_i is compared with the network output o_{win} in order to evaluate the accuracy of the SGNT. Note that though the competitive learning of the training process is performed between a parent and its children recursively, the competitive learning of the testing process is performed among only children recursively.

3. Ensemble Self-Generating Neural Networks

Though SGNNs have an ability of the fast learning and an applicability of large scale problems, the accuracy is not so good as feed-forward networks which are implemented as a supervised learning method such as BP. In order to acquire more higher performance from given training data, we consider the ensemble averaging of K SGNTs.

The structure of the SGNT dynamically changes in training. The SGNT algorithm decides the structure of the SGNT after all training data are added to the SGNT. A different structure of the SGNT is generated by changing the input order of the training data.

ESGNNs employ not only bootstrap sampling but shuffling the training data to use all training data exactly. ESGNNs can be separated into a training process and a testing process. In the training process, we define "shuffler" to shuffle a set of training data D. Figure 2 shows the structure of the ensemble system including K SGNTs in the training process. The set of all training data D enters each SGNN through each shuffler. The shuffler makes shuffle elements of D at random. All SGNTs are generated by adopting the SGNT algorithm. After training process, various SGNTs are generated independently. We

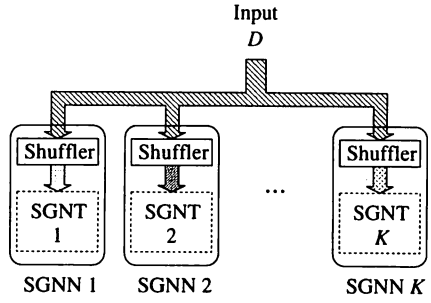


Fig.2 ESGNNs which are constructed from K SGNTs (training process). One expert corresponds to one SGNT, the shuffler makes shuffle elements of input data

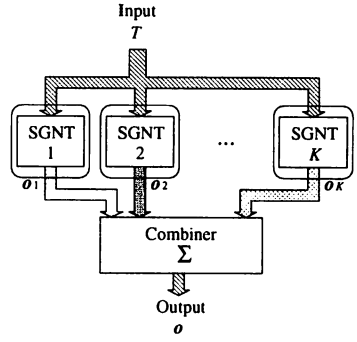


Fig.3 The structure of ESGNNs (testing process)

call a SGNN "expert" in the ensemble system.

In the testing process, the set of test data T enters an ensemble model. Figure 3 shows the structure of the ensemble system including K SGNTs in the testing process. Each output vector $o_k \in \mathbb{R}^M$, where $\mathbb{R}$ is a real number, denotes the output of the expert k for the set of test data T. The output of this ensemble model is computed by averaging the each expert output as follows:

$$o = \frac{1}{K} \sum_{k=1}^{K} o_k. \tag{2}$$

This ensemble model can obtain more sensitive classification than a single SGNN because of its estimation capability of the unknown true probability density. This means improvement of an accuracy of the classification.

In this paper, we adopt the ensemble model to binary classification problems. In order to classify each test data, corresponding output $o_i (i = 1, \ldots, M)$ is evaluated as follows:

$$\begin{aligned} o_i \geq 0.5 &\quad : \quad \text{Class1}, \\ o_i < 0.5 &\quad : \quad \text{Class0}, \\ 0 \leq o_i \leq 1. & \end{aligned} \tag{3}$$

4. Parallel and Distributed Classification

Since each expert make training and test independently, the ESGNNs model has possibility of parallel computation in the training process and the testing process. Hence, we allocate each of experts to each of processors on the MIMD computer. The procedure of the parallelization of ESGNNs is presented as follows:

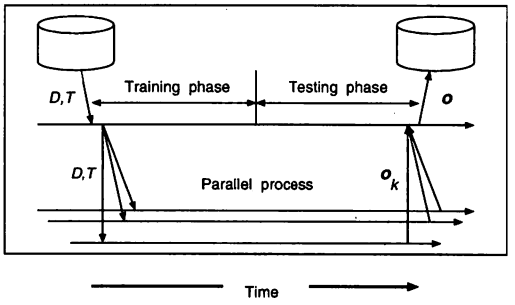


Fig.4 Parallelization of ESGNNs. Horizontal lines are the processors on MIMD parallel computer

Step1: In a master processor, read the training set D and the test set T in the disk.

Step2: In the master processor, broadcast D and T for all $K-1$ slave processors.

Step3: In all processors, generate the SGNT from D, then test the SGNT using T, and compute the o_k independently.

Step4: In all processors, each output o_k for T is collected in the master processor by all-to-one communication.

Step5: In the master processor, compute o by Eq. (2) and write to the disk.

Because the number of the communications between the master processor and each slave processor is only two times (Step2 and Step4), the parallel efficiency is approximately expected the linear speedup (See Figure 4). In our case, all computations are performed on the Intel Paragon (Paragon XP/S15). This is a distributed memory multicomputer, and the architecture is multiple instruction multiple data (MIMD). The Paragon we use has 296 processors. Each processor is Intel i860XP (50MHz). The two-dimensional mesh is adopted as the network of the Paragon.

5. Experimental Details

We allocate a SGNT to each of processors on the Paragon, and compute 100 trials for each single/ensemble model. The number of processors (SGNTs) K for the ensemble averaging is changed from 1 to 30 (1,2,3,4,5,6,7,8,9,10,15,20,25, and 30), and the threshold value ξ is 0 for each SGNT algorithm. In order to reduce the redundant execution, we repeated 100 trials from Step3 to Step5 in prior section continuously. Generally, the parallel efficiency ε is defined as follows:

$$\varepsilon = \frac{S(K)}{K} , \tag{4}$$

where $S(K)$ stands for the speedup, and K is the number of processors. In this paper, we adopt as the scaled speedup one which is given in[7] to evaluate the parallel efficiency as follows:

$$S(K) = P_s + P_p K , \tag{5}$$

where P_s and P_p represent the fraction of the program which is performed in serial and parallel, respectively.

In order to investigate the parallel performance of ESGNNs, we select three typical classification problems which are given as benchmark problems in this classification field. Next, we describe the brief explanation of these problems.

Table2 Six abilities of MONK's problems

x_1: head_shape $\in$ round,square,octagon;
x_2: body_shape $\in$ round,square,octagon;
x_3: is_smiling $\in$ yes, no;
x_4: holding $\in$ sword,balloon,flag;
x_5: jacket_color $\in$ red,yellow,green,blue;
x_6: has_tie $\in$ yes,no;

MONK's : MONK's problems are widely used as the benchmark problems. The learning tasks of MONK's problems are a binary classification task. Table 2 shows six discrete attributes of MONK's problems. Each of them is given by the following logical description of a class.

- Problem M_1: (head_shape = body_shape) or (jacket_color = red). From 432 possible examples, 124 were randomly selected for the training set. No noise was present.
- Problem M_2: Exactly two of the six attributes have their first value. From 432 examples, 169 were selected randomly. No noise was present.
- Problem M_3: (Jacket_color is green and holding a sword) or (jacket_color is not blue and body_shape is no octagon). From 432 examples, 122 were selected randomly. And among them there were 5% misclassification, i.e. noise in the training set.

Cancer : Cancer problems are binary classification task for classify a tumor as either benign or malignant based on cell descriptions gathered by a microscopic examination. Input attributes are :

- the clump thickness,
- the uniformity of cell size,
- cell shape,
- the amount of magical adhesion,
- the frequency of bare nuclei, etc.

Cancer problems have 9 attributes, 699 examples. Each attribute consists of continuous real value. Three problems are given by changing train and test data.

Card : Card problems are binary classification task. Card problems predict the approval or non-approval of a credit card to a customer. Card problems have 51 attributes, 690 examples. This data set has a good mix of attributes:

- continuous,
- nominal with small numbers of values,
- nominal with large numbers of values.

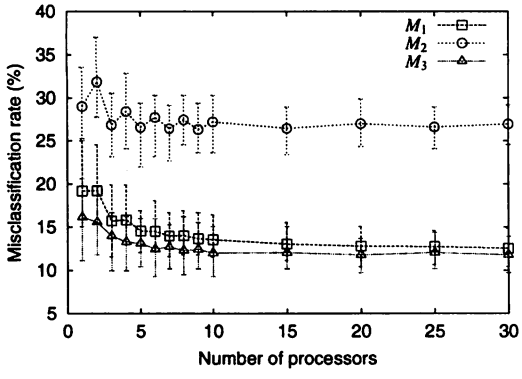
There are a few missing values in 5% of the examples. Three problems are given by changing train and test data.

In this paper, we use below defined misclassification rate as the classification accuracy.

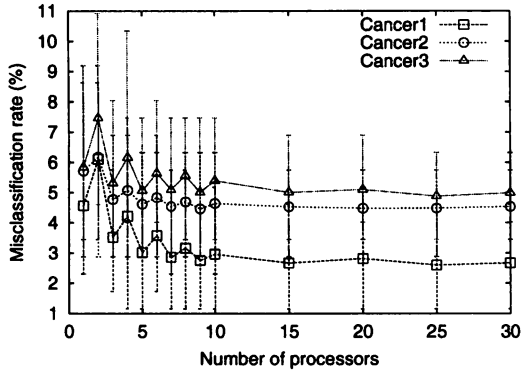
$$\text{misclassification rate} = \frac{\text{number of failures}}{\text{number of test data}} . \tag{6}$$

6. Experimental Results

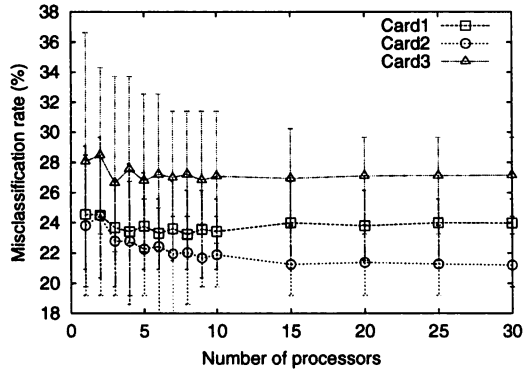
Figure 5(a), Figure 5(b), and Figure 5(c) show the influence of the number of processors on misclassification rate (%) for MONK's (M_1, M_2, and M_3), Cancer (Cancer1, Cancer2, and Cancer3), and Card (Card1, Card2, and Card3) problems respectively. Misclassification rates are improved by computing the ensemble averaging of various SGNTs for all problems. Here, each misclassification rate shows the average of 100 trials and its error-bar. It is shown that the classification accuracies are improved by



(a) MONK's

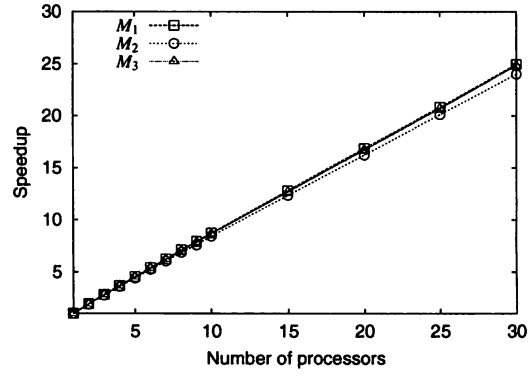


(b) Cancer

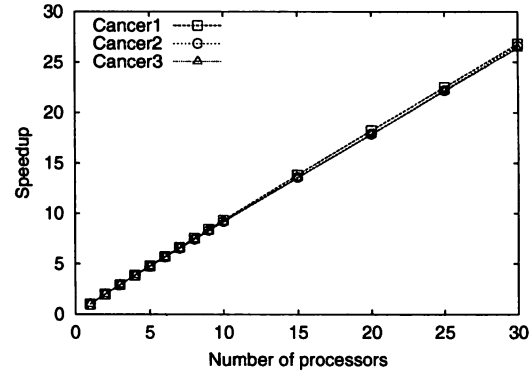


(c) Card

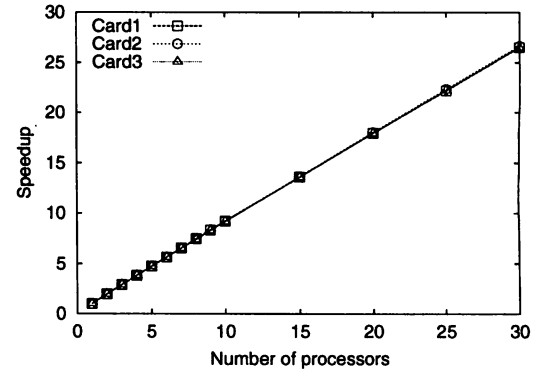
Fig.5 Influence of the number of processors on misclassification rate (%) for (a) MONK's, (b) Ikeda map, and (c) Card



(a) MONK's



(b) Cancer



(c) Card

Fig.6 Relation between the number of processors and speedup for (a) MONK's, (b) Cancer, and (c) Card

computing the ensemble averaging of various SGNTs for all problems. Especially, the maximum misclassification rates are largely improved for all problems. Comparison to even and odd numbers of K, odd number ensembles are good accuracy than even number ensembles because even number ensembles has some cases that are divided into the same number of class with respect to binary classification problems. Hence, it is natural consequence that odd number ensembles are better than even number ensembles for binary classification problems.

Figure 6 (a), Figure 6(b), and Figure 6(c) shows the relation between the number of processors and speedup for MONK's , Cancer, and Card respectively. Approximately linear speedup is obtained for all problems.

Consequently, the parallel and distributed classification using ESGNNs can obtain more higher classification accuracy than the single SGNNs by allocating each of SGNTs to each of processors, go on maintaining the high speed processing property of the single SGNNs.

7. Conclusions

In this paper, we presented the parallel and distributed classification using ESGNNs to obtain more effective implementation for classification on the MIMD parallel computer. From the experimental results the following conclusions can be drawn:

- This model can improve the classification accuracy using various SGNTs which are allocated processors on the MIMD computer.
- This model can perform a task with the high parallel efficiency by allocating each of SGNNs to each of processors on the MIMD computer.

In the future, we intend to propose a fast pruning method for more efficient processing for classification.

Acknowledgments

The authors would like to thank the anonymous reviewer for helpful comments and the Information Processing Center in Okayama University of Science for using the Paragon.

References

1) L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

2) Leo Breiman. Bagging predictors. Technical report 421, Department of Statistics, University of California, Sep. 1994.

3) S. Haykin. *Neural Networks: A comprehensive foundation*, chapter 7. Prentice-Hall, Upper Saddle River, NJ, second edition, 1999.

4) Hirotaka Inoue and Hiroyuki Narihisa. Performance of self-generating neural network applied to pattern recognition. In *ISAS'99 (5th International Conference on Information Systems Analysis and Synthesis)*, volume 5, pages 608–614, Orlando, FL, Aug. 1999.

5) Hirotaka Inoue and Hiroyuki Narihisa. Improving generalization ability of self-generating neural networks through ensemble averaging. In Takao Terano, Huan Liu, and Arbee L P Chen, editors, *The Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2000)*, volume 1805 of *LNAI*, pages 177–180, Kyoto, Japan, Apr. 18–20 2000. Springer-Verlag. ISBN: 3-540-67382-2.

6) T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Berlin, 1995.

7) P. M. Pardalos, A. T. Phillips, and J. B. Rosen. *TOPICS IN PARALLEL COMPUTING IN MATHEMATICAL PROGRAMMING.* Applied Discrete Mathematics and Theoretical Computer Science. Science Press, New York, 1992.

8) L. Prehelt. PROBEN1 — a set of neural network benchmark problems and benchmarking rules. Technical report 21/94, Universität Karlsruhe, 1994.

9) David E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 8, pages 318–362. The MIT Press, Cambridge, MA, 1986.

10) S. Thrun et al. The MONK's problems: A performance comparison of different learning algorithms. Technical report CMU-CS-91-197, Carnegie Mellon University, Dec 1991.

11) W. X. Wen, A. Jennings, and H. Liu. Learning a neural tree. In *IJCNN'92 (International Joint Conference on Neural Networks)*, Beijing, China, 1992.

12) W. X. Wen, V. Pang, and A. Jennings. Self-generating vs. self-organizing, what's different? In P. K. Simpson, editor, *Neural Networks Theory, Technology, and Applications*, IEEE Technology Update Series, pages 210–214. IEEE Technical Activities Board, Piscataway, NJ, 1996.