computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

SELF-ORGANIZING NEURAL GROVE: EFFICIENT NEURAL NETWORK ENSEMBLES USING PRUNED SELF-GENERATIONG NEURAL TREES

Hirotaka Inoue [1], Kyoshiro Sugiyama [2]

[1] Kure National College of Technology,
2-2-11 Agaminami, Kure, Hiroshima 737-8506 Japan, e-mail: hiro@kure-nct.ac.jp
[2] Advanced Engineering Faculty, Kure National College of Technology,
2-2-11 Agaminami, Kure, Hiroshima 737-8506 Japan, e-mail: s201212@sd.kure-nct.ac.jp

Abstract: Recently, multiple classifier systems have been used for practical applications to improve classification accuracy. Self-generating neural networks are one of the most suitable base-classifiers for multiple classifier systems because of their simple settings and fast learning ability. However, the computation cost of the multiple classifier system based on self-generating neural networks increases in proportion to the numbers of self-generating neural networks. In this paper, we propose a novel pruning method for efficient classification and we call this model a self-organizing neural grove. Experiments have been conducted to compare the self-organizing neural grove with bagging and the self-organizing neural grove with boosting, and support vector machine. The results show that the self-organizing neural grove can improve its classification accuracy as well as reducing the computation cost. *Copyright © Research Institute for Intelligent Computer Systems, 2013. All rights reserved.*

Keywords: neural network ensembles; self-organization; improving generalization capability; bagging; boosting.

1. INTRODUCTION

Classifiers need to find hidden information within a large amount of given data effectively and classify unknown data as accurately as possible [1]. Recently, to improve the classification accuracy, multiple classifier systems such as neural network ensembles, bagging, and boosting have been used for practical data mining [2-5]. In general, base classifiers of multiple classifier systems use traditional models such as neural networks (back-propagation network and radial basis function network) [6] and decision trees (CART and C4.5) [7].

Neural networks have great advantages such as adaptability, flexibility, and universal nonlinear input-output mapping capability. However, to apply these neural networks, it is necessary for the network structure and some parameters to be determined by human experts, and it is quite difficult to choose the right network structure suitable for a particular application at hand. Moreover, they require a long training time to learn the input-output relation of the given data. These drawbacks prevent neural networks from being the base classifier of multiple classifier systems for practical applications.

Self-generating neural networks (SGNN) [8] have a simple network design and high-speed learning. SGNN are an extension of the self-organizing maps (SOM) of Kohonen [9] and utilize the competitive learning that is implemented as a self-generating neural tree (SGNT). The abilities of SGNN make it suitable for the base classifier of multiple classifier systems. In order to improve in the accuracy of SGNN, we proposed ensemble self-generating neural networks (ESGNN) for classification [10] as one of multiple classifier systems. Although the accuracy of ESGNN improves by using various SGNN, the computation cost, that is the computation time and the memory capacity increases in proportion to the increase in numbers of SGNN in multiple classifier systems.

In an earlier paper [11], we proposed a pruning method for the structure of the SGNN in multiple classifier systems to reduce the computation cost. In this paper, we propose a novel pruning method for more effective processing and we call this model a self-organizing neural grove (SONG) [12]. This pruning method is constructed in two stages. In the first stage, we introduce an on-line pruning algorithm to reduce the computation cost by using class labels in learning. In the second stage, we optimize the structure of the SGNT in multiple classifier systems to improve the generalization capability by pruning the redundant leaves after

learning. In the optimization stage, we introduce a threshold value as a pruning parameter to decide which subtree's leaves to prune and estimate with 10-fold cross-validation [13]. After the optimization, the SONG improves its classification accuracy as well as reducing the computation cost. We use bagging [2] and boosting [14] as a resampling technique for the SONG.

We compare the SONG with support vector machine (SVM) [15] to investigate the computational cost and the classification accuracy using ten problems in the UCI machine learning repository [16].

The rest of the paper is organized as follows. The next section shows how to construct the SONG. Section 3 shows the experimental results. Then section 4 is devoted to some experiments to investigate the incremental learning performance of SONG. Finally we present some conclusions, and outline plans for future work.

2. CONSTRUCTING SELF-ORGANIZING NEURAL GROVE

First, we mention the on-line pruning method in the learning of SGNT. Second, we show the optimization method in constructing the SONG. Finally, we show a simple example of the pruning method for a two dimensional classification problem.

2.1. ON-LINE PRUNING OF SELF-GENERATING NEURAL TREE

SGNN are based on SOM and are implemented as an SGNT architecture. The SGNT can be constructed directly from the given training data without any intervening human effort. The SGNT algorithm is defined as a tree construction problem of how to construct a tree structure from the given data which consists of multiple attributes under the condition that the final leaves correspond to the given data. First, we mention the on-line pruning method in the learning of SGNT. Second, we show the optimization method in constructing the SONG.

Before we describe the SGNT algorithm, we denote some notations.

- input data vector: $\boldsymbol{e}_i \in \mathbb{R}^m$.
- root, leaf, and node in the SGNT: n_j.
- weight vector of n_j: $\boldsymbol{w}_j \in \mathbb{R}^m$.
- the number of the leaves in n_j: c_j.
- distance measure: $d(\boldsymbol{e}_i, \boldsymbol{w}_j)$.
- winner leaf for $\boldsymbol{e}_i$ in the SGNT: n_{win}.

The SGNT algorithm is a hierarchical clustering algorithm. The pseudo C code of the SGNT algorithm is given as follows:

```
Algorithm (SGNT Generation)
Input:
   A set of training examples E = {e_i},
   i = 1, ... , N.
   A distance measure d(e_i,w_j).
 Program Code:
   copy(n_1,e_1);
   for (i = 2, j = 2; i <= N; i++) {
      n_win = choose(e_i, n_1);
      if (leaf(n_win)) {
         copy(n_j, w_win);
         connect(n_j, n_win);
          j++;
      }
      copy(n_j, e_i);
      connect(n_j, n_win);
      j++;
      prune(n_win);
   }
 Output:
   Constructed SGNT by E.
```

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

Table 1. Sub procedures of SGNT algorithm.

Sub procedure	Specification
copy(n_j, $\boldsymbol{e}_i$/$\boldsymbol{w}_{win}$)	Create n_j, copy $\boldsymbol{e}_i$/$\boldsymbol{w}_{win}$ as $\boldsymbol{w}_j$ in n_j.
choose($\boldsymbol{e}_i$, n_1)	Decide n_{win} for $\boldsymbol{e}_i$.
leaf(n_{win})	Check n_{win} whether n_{win} is a leaf or not.
connect(n_j, n_{win})	Connect n_j as a child leaf of n_{win}.
prune(n_{win})	Prune leaves if the leaves have the same class.

In order to decide the winner leaf n_{win} in the sub procedure *choose*($\boldsymbol{e}_i$,n_1), competitive learning is used. This sub procedure is recursively used from the root to the leaves of the SGNT. If an n_j includes the n_{win} as its descendant in the SGNT, the weight w_{jk}(k = 1,2, … , m) of the $n_j$is updated as follows:

$$w_{jk} \leftarrow w_{jk} + \frac{1}{C_j} \cdot \left(e_{ik} - w_{jk}\right), 1 \le k \le m. \qquad (1)$$

In the SGNT, the input vector $\boldsymbol{x}_i$ corresponds to $\boldsymbol{e}_i$, and the desired output y_i corresponds to the network output o_i which is stored in one of the leaf neurons, for (x_i, y_i) ∈D. Here, D is the training data set which consists of data $\{\boldsymbol{x}_i, y_i | \ i=1,\dots, N\}$, $\boldsymbol{x}_i \in \mathbb{R}^m$ is the input and y_i is the desired output. After all training data are inserted into the SGNT as the leaves, the leaves each have a class label as the outputs and the weights of each node are the averages of the corresponding weights of all its leaves. The whole network of the SGNT reflects the given feature space by its topology.

We explain the SGNT generation algorithm using an simple example. In this example, m is one and the four training data (x_i, y_i) is (1,1), (2,2), (3,3), and (4,4). Hence, e_{11} =1, e_{21} = 2, e_{31} = 3, and e_{41} = 4. Fig. 1 shows an example of the SGNT generation. First, e_{11} is just copied to a neuron n_1 as the root, and e_{11} is substituted to w_{11} (Fig. 1 (a)). In Fig. 1, the circle is the neuron, the integer in the circle is the number of neuron j, the integer of left-upper of the circle is c_j, and the integer of under the circle is w_{j1}. Next, n_2 and n_3 are generated as the children of n_1 with w_{21}=1, w_{31}=2. w_{11} is updated by e_{21} to 1+1/2(2-1)=1.5 (Fig. 1 (b)). Next, the winner in $\{n_1, n_2, n_3\}$ is n_3 since $d(\boldsymbol{e}_3,\boldsymbol{w}_1)$ = 1.5, $d(\boldsymbol{e}_3,\boldsymbol{w}_2)$ = 2, and $d(\boldsymbol{e}_3,\boldsymbol{w}_3)$ = 1; and thus, n_4 and n_5 are generated as the children of n_3 with w_{41} = 2, w_{51} = 3. w_{31} is updated by e_{31} to 2+1/2(3-2)=2.5 and w_{11} is updated by e_{31} to 1.5+1/3(3-1.5)=2 (Fig. 1 (c)). Finally, n_6 and n_7 are generated as the children of n_5 with w_{61} = 3, w_{71} = 4. w_{51} is updated by e_{41} to 3+1/2(4-3)=3.5, w_{31} is updated by e_{41} to 2.5 + 1/3(4-2.5) = 3, and w_{11} is updated by e_{41} to 2 + 1/4(4-2) = 2.5 (Fig. 1 (d)).

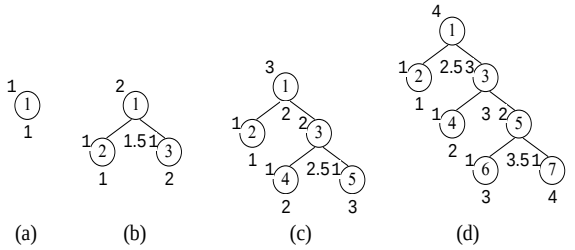


Fig. 1 – An example of the SGNT generation.

Note, to optimize the structure of the SGNT effectively, we remove the threshold value of the original SGNT algorithm in [8] to control the number of leaves based on the distance because of the trade-off between the memory capacity and the classification accuracy. In order to avoid the above problem, we introduce a new pruning method in the sub procedure *prune*(n_{win}). We use the class label to prune leaves. For leaves that have the n_{win}'s parent node, if all leaves belong to the same class, then these leaves are pruned and the parent node is given to the class.

2.2. OPTIMIZATION OF THE SONG

The SGNT has the capability of high speed processing. However, the accuracy of the SGNT is inferior to the conventional approaches, such as nearest neighbor, because the SGNT has no guarantee to reach the nearest leaf for unknown data. Hence, we construct the SONG by taking the majority of multiple SGNT's outputs to improve the accuracy (Fig. 2).

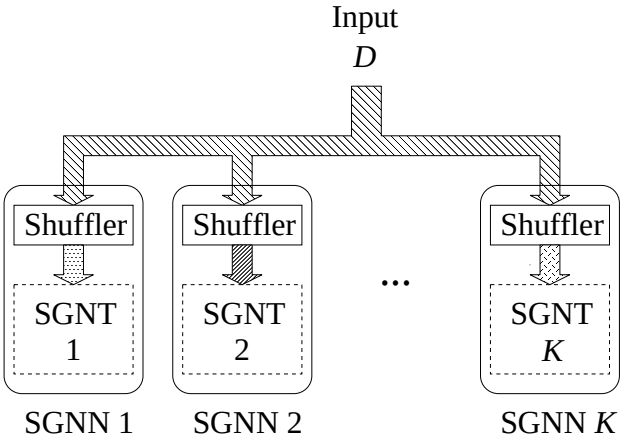


Fig. 2 – The SONG which is constructed from *K* SGNTs. The SONG's output is decided by voting outputs of *K* SGNTs.

Although the accuracy of the SONG is superior or comparable to the accuracy of conventional approaches, the computational cost increases in proportion to the increase in the number of SGNTs in the SONG. In particular, the huge memory requirement prevents the use of SONG for large datasets even with the latest computers.

In order to improve the classification accuracy, we propose an optimization method of the SONG for classification. This method has two parts, the merge phase and the evaluation phase. The merge phase is performed as a pruning algorithm to reduce dense leaves. The merge phase algorithm is given as follows:

Algorithm (Merge phase)
1. begin initialize j = the height of the SGNT
2. do for each subtree's leaves in the height j
3. if the ratio of the most class $\geq \alpha$,
4. then merge all leaves to parent node
5. if all subtrees are traversed in the height j,
6. then $j \leftarrow j-1$
7. until $j = 0$
8. end.

This phase uses the class information and a threshold value α to decide which subtree's leaves to prune or not. For leaves that have the same parent node, if the proportion of the most common class is greater than or equal to the threshold value α, then these leaves are pruned and the parent node is given the most common class.

The optimum threshold values α of the given problems are different from each other. The evaluation phase is performed to choose the best threshold value by introducing 10-fold cross validation. The evaluation phase algorithm is given as follows:

Algorithm (Evaluation phase)

1. begin initialize $\alpha = 0.5$
2. do for each α
3. evaluate the merge phase with 10-fold CV
4. if the best classification accuracy is obtained,
5. then record the α as the optimal value
6. $\alpha \leftarrow \alpha + 0.05$
7. until $\alpha = 1$

end.

2.3. SIMPLE EXAMPLE OF THE PRUNING METHOD

We show an example of the pruning algorithm in Fig. 3. This is a two-dimensional classification problem with two equal circular Gaussian distributions that have an overlap. The shaded plane is the decision region of class 0 and the other plane is the decision region of class 1 by the SGNT. The dotted line is the ideal decision boundary. The number of training samples is 200 (class0: 100, class1: 100) (Fig. 3 (a)).

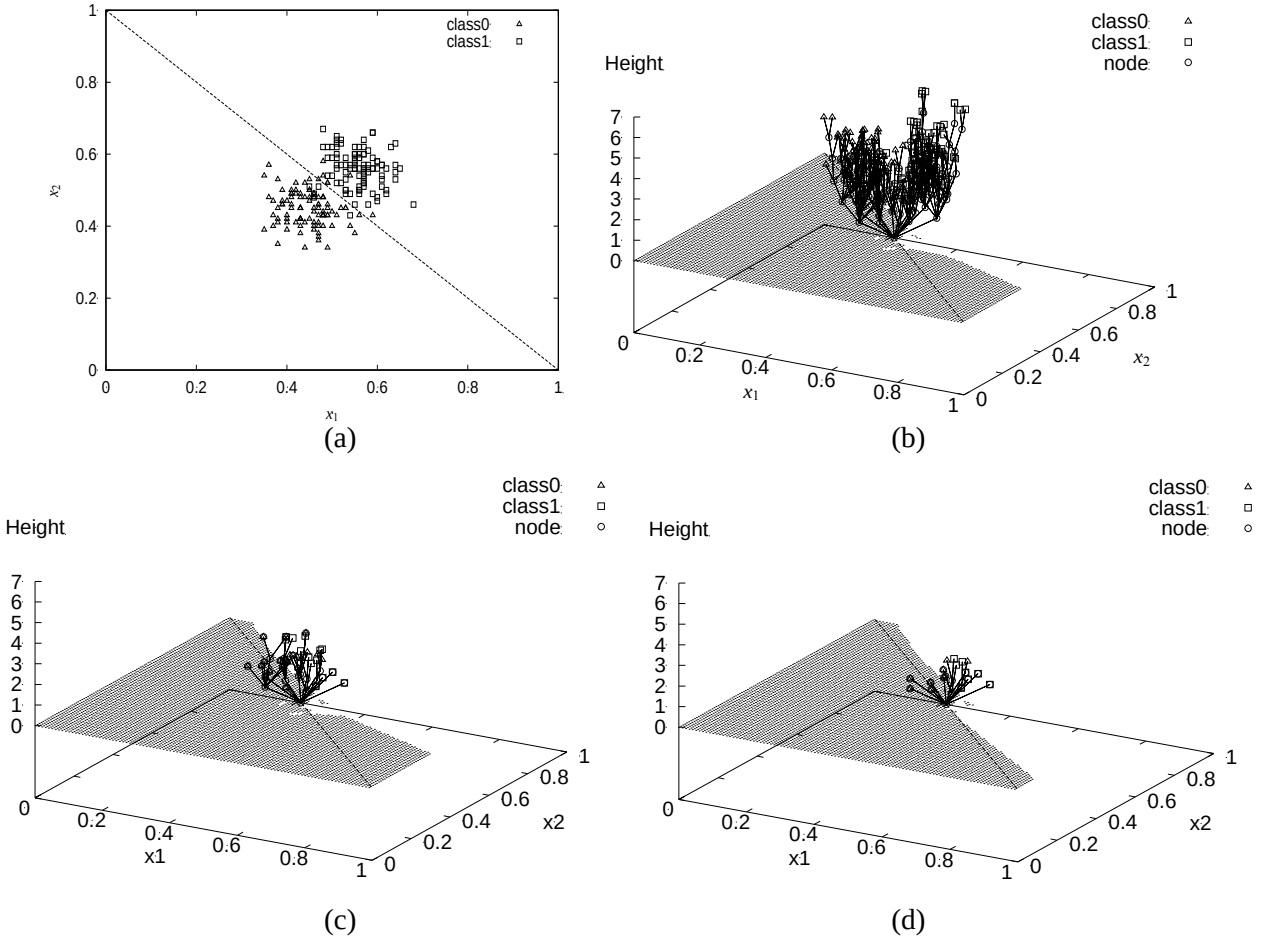


Fig. 3 – An example of the SGNT's pruning algorithm, (a) a two dimensional classification problem with two equal circular Gaussian distribution, (b) the structure of the unpruned SGNT, (c) the structure of the pruned SGNT (α= 1), and (d) the structure of the pruned SGNT (α = 0.6).The shaded plane is the decision region of class 0 by the SGNT and the doted line shows the ideal decision boundary

The unpruned SGNT is given in Fig. 3 (b). In this case, 200 leaves and 120 nodes are automatically generated by the SGNT algorithm. In this unpruned SGNT, the height is 7 and the number of units is 320. In this, we define the unit to count the sum of the root, nodes, and leaves of the SGNT. The root is the node which is of height 0.The unit is used as a measure of The decision boundary is the same as the unpruned SGNT. Fig. 3 (d) shows the pruned SGNT after the merge phase in α = 0.6. In this case, 182 leaves and 115 nodes are pruned away and only 23 units remain. Moreover, the decision boundary is improved more than the unpruned SGNT because this case can reduce the effect of the overlapping class by pruning the SGNT.

In the above example, we use all training data to construct the SGNT. The structure of the SGNT is changed by the order of the training data. Hence, we can construct the MCS from the same training data by changing the input order. We call this approach “shuffling”.

3. EXPERIMENTAL RESULTS

We investigate the computational cost (the memory capacity and the computation time) and the classification accuracy of the SONG with bagging for ten benchmark problems in the UCI machine learning repository [16]. Table 2 presents the abstract of the datasets.

Table 2. The brief summary of the datasets. *N* is the number of instances, *m* is the number of attributes.

Dataset	*N*	*m*	classes
balance-scale	625	4	3
Breast-cancer-w	699	9	2
glass	214	9	6
ionosphere	351	34	2
iris	150	4	3
letter	20000	16	26
liver-disorders	345	6	2
new-thyroid	215	5	3
pima-diabetes	768	8	2
wine	178	13	3

We evaluate how the SONG is pruned using 10-fold cross-validation for the ten benchmark problems. In this experiment, we use a modified Euclidean distance measure for the SONG as follows:

$$d(x,y) = \sqrt{\sum_{i=1}^{m} a_i \cdot (x_i - y_i)^2}, \quad (2)$$

$$a_i = \frac{1}{\max_j - \min_j}, (1 \le j \le N). \quad (3)$$

Since the performance of the SONG is not sensitive to the threshold value α, we set the different threshold values α to vary from 0.5 to 1; α = [0.5,0.55,0.6, … , 1]. We set the number of SGNT *K* in the SONG as 25 and execute 100 trials by changing the sampling order of each training set. All experiments in this section were performed on an UltraSPARC workstation with a 900 MHz CPU, 1 GB RAM, and Solaris 8.

The figures and tables must be numbered, have a self-contained caption. Figure captions should be below the figures; table captions should be above the tables. Also, avoid placing figures and tables before their first mention in the text.

Table 3 shows the average memory requirement and of 100 trials for the SONG. As the memory requirement, we count the number of units which is the sum of the root, nodes, and leaves of the SGNT. The average memory requirement is reduced from 65 % to 96.6 % and the classification accuracy is improved 0.1 % to 2.9 % by optimizing the SONG. Table 4 shows classification accuracy of 100 trials for the SONG. In Table 4, the standard deviation is given inside the bracket ($\times 10^{-3}$). The classification accuracy is improved 0.1 % to 2.9 % by optimizing the SONG. These results support that the SONG can be effectively used for all datasets with regard to both the computation cost and the classification accuracy.

Table 3. The average memory requirement of 100 trials for the bagged SGNT in the SONG.

Dataset	pruned	unpruned	ratio
balance-scale	107.68	861.18	12.5
breast-cancer-w	30.88	897.37	3.4
glass	104.33	297.75	35
ionosphere	50.75	472.39	10.7
iris	15.64	208.56	7.4
letter	6197.5	27028.56	22.9
liver-disorders	163.12	471.6	34.5
new-thyroid	49.45	298.21	16.5
pima-diabetes	204.4	1045.03	19.5
wine	15	238.95	6.2
Average	693.88	3181.96	16.9

Table 4. The classification accuracy of 100 trials for the bagged SGNT in the SONG. The standard deviation is given inside the bracket ($\times 10^{-3}$).

Dataset	pruned	unpruned	ratio
balance-scale	0.866 (6.36)	0.837 (7.83)	+2.9
breast-cancer-w	0.97 (2.41)	0.966 (2.71)	+0.4
glass	0.714 (13.01)	0.709 (14.86)	+0.5
ionosphere	0.891 (6.75)	0.862 (7.33)	+2.9
iris	0.962 (6.04)	0.955 (5.45)	+0.7
letter	0.956 (0.77)	0.955 (0.72)	+0.1
liver-disorders	0.648 (12.89)	0.636 (13.36)	+1.2
new-thyroid	0.958 (7.5)	0.957 (7.49)	+0.1
pima-diabetes	0.749 (7.05)	0.728 (7.83)	+2.1
wine	0.976 (4.41)	0.972 (5.57)	+0.4
Average	0.869	0.858	+1.1

Table 5 shows the average classification accuracy of 10 trials for the SONG with boosting. On boosting, we implement AdaBoost [14] to the SONG. Since original AdaBoost algorithm have been proposed for binary classification problems, we use four binary classification problems. In comparison with boosting, bagging is superior to boosting on all of the 4 datasets. In short,

bagging is better than boosting in terms of the classification accuracy.

Table 5. The average classification accuracy of 10 trials for the SONG with boosting. The standard deviation is given inside the bracket ($\times 10^{-3}$).

Dataset	SGNT	SONG	ratio
breast-cancer-w	0.96 (6.47)	0.957 (4.13)	-0.3
ionosphere	0.854 (18.26)	0.773 (17.4)	-8.1
liver-disorders	0.588 (17.9)	0.572 (24.3)	-1.6
pima-diabetes	0.696 (12.2)	0.722 (6.82)	+2.6
Average	0.775	0.756	-1.9

To show the advantages of the SONG, we compare it with SVM on the same problems. In the SONG, we choose the best classification accuracy of 100 trials with bagging. In SVM, we use *C*-SVM in libsvm [17] with radial basis function kernel. We select the parameters of SVM, the cost parameters C and the kernel parameters γ, from $15 \times 15 = 225$ combinations by 10-fold cross validation; $C = [2^{12},2^{11},2^{10}, \dots , 2^{-2}]$ and $\gamma = [2^{4},2^{3},2^{2}, \dots , 2^{-10}]$.We normalize the input data from 0 to 1 for all problems in SONG and SVM. All methods are compiled by using gcc with the optimization level -O2 on the same workstation.

Table 6, Table 7, and Table 8 show the classification accuracy, the memory requirement, and the computation time achieved by the SONG and SVM respectively. Next, we show the results for each category.

Table 6. The classification accuracy of 10 trials for the best pruned SONG and SVM.

Dataset	SONG	SVM
balance-scale	0.885	**0.992**
breast-cancer-w	**0.976**	0.973
glass	**0.758**	0.738
ionosphere	0.912	**0.954**
iris	**0.973**	0.96
letter	0.958	**0.977**
liver-disorders	0.685	**0.73**
new-thyroid	0.972	**0.977**
pima-diabetes	0.764	**0.766**
wine	0.983	**0.989**
Average	0.887	**0.904**

First, in view point of the classification accuracy, the SONG superior to SVM 3 of the 10 datasets and degrade 1.7 % in the average in Table 6.

Second, in terms of the memory requirement, even though the SONG includes the root and the nodes which are generated by the SGNT generation algorithm, this is less than SVM for 8 of the 10 datasets. Although the memory requirement of the SONG is totally used K times in Table 7, we release the memory of SGNT for each trial and reuse the memory for effective computation. Therefore, the memory requirement is suppressed by the size of the single SGNT.

Table 7. The memory requirement of 10 trials for the best pruned SONG and SVM.

Dataset	SONG	SVM
balance-scale	109.93	**60.6**
breast-cancer-w	**26.8**	79.6
glass	**91.33**	132.4
ionosphere	**51.38**	147.9
iris	**11.34**	51.3
letter	**6208.03**	7739.7
liver-disorders	**134.17**	214.5
new-thyroid	45.74	**44.1**
pima-diabetes	**183.57**	363.5
wine	**11.8**	62.2
Average	**687.41**	889.58

Table 8. The computation time (in sec.) of 10 trials for the best pruned SONG and SVM.

Dataset	SONG	SVM
balance-scale	**0.82**	4.77
breast-cancer-w	1.18	**0.64**
glass	**0.36**	0.61
ionosphere	1.93	**1.25**
iris	0.13	**0.06**
letter	**208.52**	2359.39
liver-disorders	**0.54**	2.07
new-thyroid	0.23	**0.22**
pima-diabetes	**1.72**	5.63
wine	0.31	**0.15**
Average	**21.57**	236.88

Finally, in view of the computation time, although the SONG consumes the cost of K times of the SGNT to construct the model and test for the unknown dataset, the average computation time is faster than SVM in Table 8. The SONG is slower than SVM for small datasets such as glass, ionosphere, and iris. However, the SONG is faster than SVM for large datasets such as balance-scale, letter, and pima-diabetes. Especially, in letter, the computation time of the SONG is faster than SVM about 11 times. We need to repeat 10-fold cross validation many times to select the optimum parameter for α, k, C, and γ. This evaluation consumes much computation time for large datasets such as letter. Therefore, the SONG based on the fast and compact SGNT is useful and practical for large datasets. Moreover, the SONG has the ability

of parallel computation because each classifier behaves independently. In conclusion, the SONG is a practical method for large-scale data mining compared with SVM.

4. CONCLUSION

In this paper, we proposed a new pruning method for the multiple classifier system based on self-generating neural trees, which is called the self-organizing neural grove, and evaluated the computation cost, that is the computation time and the memory capacity, and the classification accuracy. We introduced an on-line and off-line pruning methods and evaluated the self-organizing neural grove by 10-fold cross-validation. Experimental results showed that the memory requirement reduced remarkably, and the accuracy increased by using the pruned self-generating neural tree as the base classifier of the self-organizing neural grove. The self-organizing neural grove is a useful and practical multiple classifier system to classify large datasets. In future work, we will study a parallel and distributed processing of the self-organizing neural grove for large scale data mining.

6. REFERENCES

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[2] L. Breiman, Bagging predictors, *Machine Learning*, (24) (1996), pp. 123-140.

[3] R. E. Schapire, The strength of weak learnability, *Machine Learning*, (5) 2 (1990), pp. 197-227.

[4] J. R. Quinlan, Bagging, Boosting, and C4.5, *the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, (August 4-8, 1996), pp. 725-730.

[5] G. Ratsch, T. Onoda, K. R. Muller, Soft margins for AdaBoost, *Machine Learning*, (42) 3 (2001), pp. 287-320.

[6] S. Haykin, *Neural Networks: A Comprehensive Foundation*, second ed., Prentice-Hall, Upper Saddle River, NJ, 1999.

[7] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, second ed., John Wiley & Sons Inc., New York, 2000.

[8] W. X. Wen, A. Jennings, H. Liu, Learning a neural tree, *the International Joint Conference on Neural Networks*, Beijing, China, (November 3-6, 1992), Vol. 2, pp. 751-756.

[9] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, 1995.

[10] H. Inoue, H. Narihisa, *Improving generalization ability of self-generating neural networks through ensemble averaging*, in: T. Terano, H. Liu, A.L.P. Chen (Eds.), PAKDD 2000, Lecture Notes in Computer Science, Vol. 1805, Springer, Heidelberg, 2000, pp. 177-180.

[11] H. Inoue, H. Narihisa, *Optimizing a multiple classifier system*, In: M. Ishizuka, A. Sattar (Eds.), PRICAI 2002, Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), Vol. 2417, Springer, Heidelberg, 2002, pp. 285-294.

[12] H. Inoue, K. Sugiyama, Self-organizing neural grove, *the 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Berlin, Germany, (September 12-14, 2013), pp. 319-323.

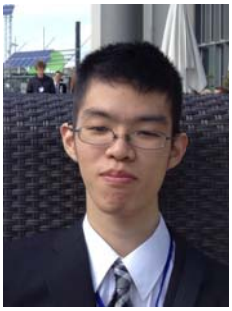
[13] M. Stone, Cross-validation: A review, *Math. Operations for sch. Statist. Ser. Statistics*, (9) 1 (1978), pp.127-139.

[14] Y. Freund and R. E. Schapire, *Boosting: Foundations and Algorithms*, MIT Press, Cambridge, MA, 2012.

[15] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, (2) 27 (2011), pp. 1–27, software available at http://www.csie.ntu.edu.tw/cjlin/libsvm

[16] C. Blake, C. Merz, UCI repository of machine learning databases, 1998. [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html

***Hirotaka Inoue**, received his MSc degree in Okayama University of Science in 1999 and PhD degree in the same university in 2002, both in engineering. His research interests are in neural networks, pattern recognition, and combinational optimization. He is currently an associate professor at Kure National College of Technology in Japan.*



***Kyoshiro Sugiyama**, is a student at Advanced Engineering Faculty, Kure National College of Technology. He will be a graduate student at Nara Institute of Science and Technology from April 2014. His research interests are in neural networks and pattern recognition.*