Efficient Pruning Method for Ensemble Self-Generating Neural Networks

Hirotaka INOUE
Department of Electrical Engineering & Information Science, Kure National College of Technology
2-2-11 Agaminami, Kure-shi, Hiroshima 737-8506, Japan

and

Hiroyuki NARIHISA
Department of Information & Computer Engineering, Faculty of Engineering,
Okayama University of Science
1-1 Ridai-cho, Okayama-shi, Okayama 700-0005, Japan

ABSTRACT

Recently, multiple classifier systems (MCS) have been used for practical applications to improve classification accuracy. Self-generating neural networks (SGNN) are one of the suitable base-classifiers for MCS because of their simple setting and fast learning. However, the computation cost of the MCS increases in proportion to the number of SGNN. In this paper, we propose an efficient pruning method for the structure of the SGNN in the MCS. We compare the pruned MCS with two sampling methods. Experiments have been conducted to compare the pruned MCS with an unpruned MCS, the MCS based on C4.5, and k-nearest neighbor method. The results show that the pruned MCS can improve its classification accuracy as well as reducing the computation cost.

Keywords: emergent computing, on-line pruning, self-organization, ensemble learning, classification

1. INTRODUCTION

Classifiers need to find hidden information in the given large data effectively and classify unknown data as accurately as possible [1]. Recently, to improve the classification accuracy, multiple classifier systems (MCS) such as neural network ensembles, bagging, and boosting have been used for practical data mining applications [2, 3]. In general, the base classifiers of the MCS use traditional models such as neural networks (backpropagation network and radial basis function network) [4] and decision trees (CART and C4.5) [5].

Neural networks have great advantages of adaptability, flexibility, and universal nonlinear input-output mapping capability. However, to apply these neural networks, it is necessary to determine the network structure and some parameters 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 being the base classifier of the MCS for practical applications.

Self-generating neural networks (SGNN) [6] have simple network design and high speed learning. SGNN are an extension of the self-organizing maps (SOM) of Kohonen [7] and utilize the competitive learning which is implemented as a self-generating neural tree (SGNT). The abilities of SGNN make it suitable for the base classifier of the MCS. In order to improve in the accuracy of SGNN, we proposed ensemble self-generating neural networks (ESGNN) for classification [8] as one of the MCS. 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 number of SGNN in the MCS.

In this paper, we propose an efficient pruning method for reducing the computational cost for ESGNN. This method is constructed from two stages. First, we introduce an on-line pruning algorithm to reduce the computation cost by using class labels in learning. Second, we optimize the structure of the SGNT in the MCS to improve the generalization capability by pruning the tedious 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 [9]. After the optimization, the MCS can improve its classification accuracy as well as reducing the computation cost. We use two sampling methods for the optimizing MCS; shuffling and bagging. Shuffling uses all the training data by changing randomly the order of the training data on each classifier. Bagging [10] is a resampling technique which permits the overlap of the data. We investigate the improvement performance of the pruned MCS by comparing it with the MCS based on C4.5 [11] using ten problems in the UCI repository [12]. Moreover, we compare the pruned MCS with k-nearest neighbor (k-NN) [13] to investigate the computational cost and the classification accuracy. The optimized MCS demonstrates higher classification accuracy and faster processing speed than k-NN on average.

2. PRUNING ESGNN

In this section, we describe how to prune tedious leaves in the MCS. We implement the pruning method as two stages. First, we mention the on-line pruning method in learning of SGNN. Second, we show the optimization method in constructing the MCS. Finally, we show a simple example of the pruning method for a two dimensional classification problem.

Self-Generating Neural Networks

SGNN are based on SOM and implemented as a 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 consist of multiple attributes under the condition that the final leaves correspond to the given data.

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 I shows the sub procedures of the SGNT algorithm and their specifications.

In order to decide the winner leaf n_{win} in the sub procedure `choose(e_i,n_1)`, competitive learning is used. If an n_j includes the n_{win} as its descendant in the SGNT, the weight w_{jk} $(k = 1, 2, \ldots, m)$ of the n_j is updated as follows:

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

After all training data are inserted into the SGNT as the leaves, the leaves have each 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. For more details concerning how to construct and perform the SGNT, see [6]. Note, to optimize the structure of the SGNT effectively, we remove the threshold value of the original SGNT algorithm in [6] 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 connected to the n_{win}, if those leaves have the same class label, then the parent node of those leaves is given the class label and these leaves are pruned.

TABLE I Sub procedures of the 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 leaves have a same class.

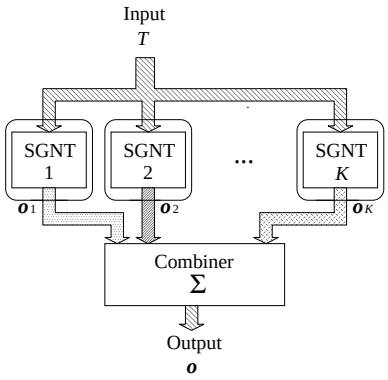

Fig.1 An MCS which is constructed from K SGNTs. The test dataset T is entered each SGNT, the output $\boldsymbol{o}_i$ is computed as the output of the winner leaf for the input data, and the MCS's output is decided by voting outputs of K SGNTs

Optimization of the Multiple Classifier System

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 an MCS by taking the majority of plural SGNT's outputs to improve the accuracy (Figure 1).

Although the accuracy of the MCS is superior or comparable to the accuracy of conventional approaches, the

```
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 ≥ α,
4     then merge all leaves to parent node
5     if all subtrees are traversed in the height j,
6     then j ← j − 1
7   until j = 0
8 end.
```

Fig.2 The merge phase

```
1 begin initialize α = 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     α ← α + 0.05
7   until α = 1
8 end.
```

Fig.3 The evaluation phase

computational cost increases in proportion to the increase in the number of SGNTs in the MCS. In particular, the huge memory requirement prevents the use of MCS for large datasets even with latest computers.

In order to improve the classification accuracy, we propose an optimization method of the MCS 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 (Figure 2). 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 (CV). Figure 3 shows this evaluation phase.

An Example of the Pruning Method

We show an example of the pruning algorithm in Figure 4. 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) (Figure 4(a)). The unpruned SGNT is given in Figure 4(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 memory requirement in the next section. Figure 4(c) shows the pruned SGNT after the on-line pruning stage. In this case, 272 units are pruned away and 48 units remain (the height is 3). The decision boundary is the same as the unpruned SGNT. Figure 4(d) shows the pruned SGNT after the optimization stage in $\alpha = 0.6$. In this case, 27 units are pruned away and 21 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".

To show how well the MCS is optimized by the pruning algorithm, we show an example of the MCS in the same problem used above. Figure 5(a) and Figure 5(b) show the decision region of the MCS in $\alpha = 1$ and $\alpha = 0.6$, respectively. We set the number of SGNTs K as 25. The result of Figure 5(b) is a better estimation of the ideal decision region than the result of Figure 5(a). We investigate the pruning method for more complex problems in the next section.

3. EXPERIMENTAL RESULTS

We investigate the computational cost (the memory capacity and the computation time) and the classification accuracy of MCS based on SGNN with two sampling methods, shuffling and bagging for ten benchmark problems in the UCI repository [12]. We evaluate how the MCS is pruned using 10-fold cross-validation for the ten benchmark problems. In this experiment, we use a modified Euclidean distance measure for the MCS. To select the optimum threshold value α, we set the different threshold values α which are moved from 0.5 to 1; $\alpha = [0.5, 0.55, 0.6, \ldots, 1]$. We set the number of SGNT K in the MCS as 25 and execute 100 trials by changing the sampling order of each training set. All computations of the MCS are performed on an IBM PC-AT machine (CPU: Intel Pentium II 450MHz, Memory: 323MB).

TABLE II shows the average memory requirement of 100 trials for the MCS based on shuffled SGNN and the MCS based on bagged SGNN. As the memory requirement, we count the number of units which is the sum of the root, nodes, and leaves of the SGNT. The memory requirement is reduced from 55.4% to 96.2% in shuffling, and from 64.9% to 96.8% in bagging, by optimizing the MCS. It is found that the bagged SGNT can be a higher memory compression than the shuffled SGNN. This supports that the pruned MCS can be effectively used for all datasets with regard to both the computational cost and the clas-

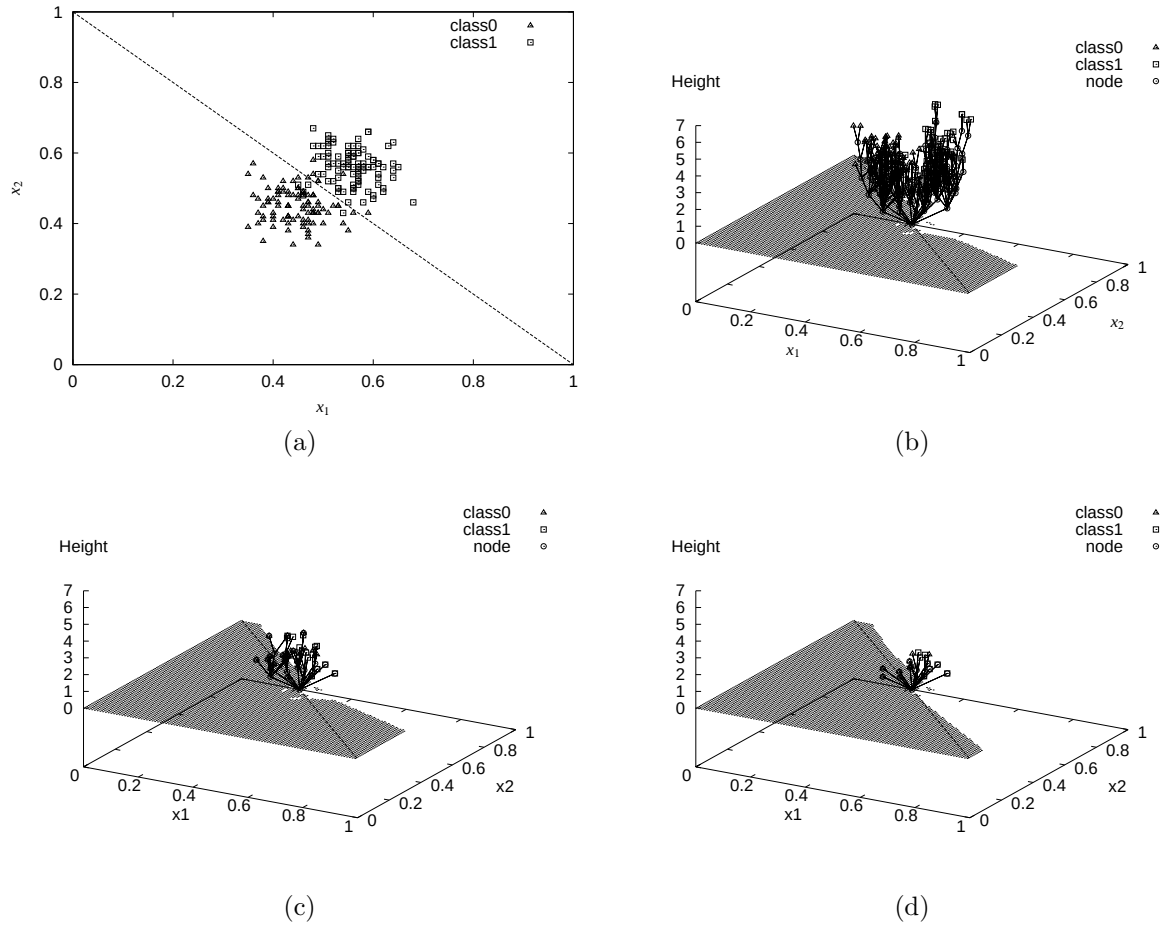

Fig.4 An example of the SGNT's pruning algorithm, (a) a two dimensional classification problem with two equal circular Gaussian distribution, (b) the structure of the unpruned SGNT, (c) the structure of the pruned SGNT ($\alpha = 1$), and (d) the structure of the pruned SGNT ($\alpha = 0.6$). The shaded plane is the decision region of class 0 by the SGNT and the doted line shows the ideal decision boundary

sification accuracy. These results are reduced 18% and 17.4% for shuffing and bagging on average than the previous results in [14].

TABLE III shows the average classification accuracy of 100 trials for the MCS with shuffling and bagging. It is clear that over 10 datasets, both optimized MCS with shuffled and bagged SGNT lead to more accurate or comparable classifiers than the unoptimized MCS. In comparison with shuffling, bagging is superior or comparable to shuffling on 6 of the 10 datasets. In short, bagging is better than shuffling in terms of the computational cost and the classification accuracy in the MCS.

To evaluate the pruned MCS's performance, we compare the pruned MCS with the MCS based on C4.5. We set the number of classifiers K in the MCS as 25 and we construct both MCS by bagging. TABLE IV shows the improved performance of the pruned MCS and the MCS based on C4.5. The results of the SGNT and the pruned MCS are the average of 100 trials. The pruned MCS has a better performance than the MCS based on C4.5 for 6 of the 10 datasets. Although the MCS based on C4.5 degrades the classification accuracy for iris, the pruned MCS can improve the classification accuracy for all problems. Therefore, the pruned SGNT is a good base classifier for the MCS on the basis of both the scalability for large scale datasets and the robust improving generalization capability for the noisy datasets comparable to the MCS with C4.5.

To show the advantages of the pruned MCS, we compare it with k-NN on the same problems. In the pruned MCS, we choose the best classification accuracy of 100 trials with bagging. In k-NN, we choose the best accuracy where k is 1,3,5,7,9,11,13,15,25 with 10-fold cross-validation. All methods are compiled by using gcc with the optimization level `-O2` on the same computer.

TABLE V shows the classification accuracy, the memory requirement, and the computation time achieved by the pruned MCS and k-NN. Next, we show the results for each category.

First, with regard to the classification accuracy, the pruned MCS is superior to k-NN for 7 of the 10 datasets and gives 1.2% improvement on average. Second, in terms of the memory requirement, even though the pruned MCS

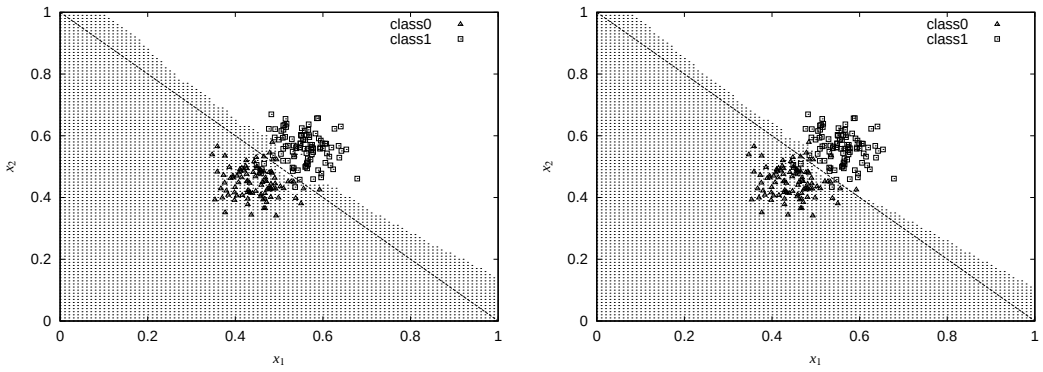

Fig.5 An example of the MCS's decision boundary ($K = 25$), (a) $\alpha = 1$, and (b) $\alpha = 0.6$. The shaded plane is the decision region of class 0 by the MCS and the doted line shows the ideal decision boundary

includes the root and the nodes which are generated by the SGNT generation algorithm, this is less than k-NN for all problems. Although the memory requirement of the pruned MCS is totally used K times in TABLE V, 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. Finally, in view of the computation time, although the pruned MCS consumes the cost of K times of the SGNT, the average computation time is faster than k-NN. In the case of letter, in particular, the computation time of the pruned MCS is faster than k-NN by about 3.8 times. We need to repeat 10-fold cross validation many times to select the optimum parameters for α and k. This evaluation consumes much computation time for large datasets such as letter. Therefore, the pruned MCS based on the fast and compact SGNT is useful and practical for large datasets. Moreover, the pruned MCS has the ability of parallel computation because each classifier behaves independently. In conclusion, the pruned MCS is practical for large-scale data mining compared with k-NN.

4. CONCLUSIONS

In this paper, we proposed an efficient pruning method for the MCS based on SGNN and evaluated the computation cost and the accuracy. We introduced an on-line and off-line pruning method and evaluated the pruned MCS by 10-fold cross-validation. We investigated the difference of two sampling methods; shuffling and bagging. Experimental results showed that the memory requirement reduces remarkably, and the accuracy increases by using the pruned SGNT as the base classifier of the MCS. Bagging is better than shuffling in view of the memory reduction and the improvement to the classification accuracy. The pruned MCS comparable to the MCS based on C4.5 and superior to k-NN. The pruned MCS is a useful and practical tool to classify large datasets. In future work, we will study an incremental learning and a parallel and distributed processing of the MCS for large scale data mining.

5. ACKNOWLEDGEMENT

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 15700206, 2003.

6. REFERENCES

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

[2] J. R. Quinlan. Bagging, Boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, Portland, OR, 1996.

[3] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons Inc., New York, 2nd ed., 2000.

[6] W. X. Wen, A. Jennings, and H. Liu. Learning a neural tree. In *the International Joint Conference on Neural Networks*, Beijing, China, 1992.

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

[8] H. Inoue and H. Narihisa. Improving generalization ability of self-generating neural networks through ensemble averaging. In T. Terano, H. Liu, and A. L P Chen, eds, *The Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, vol. 1805 of *LNAI*, pages 177–180, Springer-Verlag, Berlin, 2000.

[9] M. Stone. Cross-validation: A review. *Math. Operationsforsch. Statist., Ser. Statistics*, 9(1):127–139, 1978.

[10] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[11] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[12] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, University of California, Irvine, Dept of Information and Computer Science, 1998. Datasets is available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[13] E. A. Patrick and F. P. Fischer. A generalized k-nearest neighbor rule. *Information and Control*, 16(2):128–152, 1970.

[14] H. Inoue and H. Narihisa. Optimizing a multiple classifier system. In M. Ishizuka and A. Sattar, eds, *PRICAI2002: Trends in Artificial Intelligence*, volume 2417 of *LNAI*, pages 285–294, Springer-Verlag, Berlin, 2002.

TABLE II THE AVERAGE MEMORY REQUIREMENT OF 100 TRIALS FOR THE SHUFFLED SGNT AND THE BAGGED SGNT IN THE MCS

	shuffled SGNT			bagged SGNT		
Dataset	pruned	unpruned	ratio	pruned	unpruned	ratio
balance-scale	133.42	846.43	15.7	113.62	860.61	13.2
breast-cancer-w	34.65	889.05	3.8	28.9	897.81	3.2
glass	125.66	295.64	42.5	104.77	297.95	35.1
ionosphere	68.32	454.16	15	54.52	472.18	11.5
iris	16.59	207.7	7.9	14.65	208.7	7
letter	7312.68	26537.91	27.5	6213.52	27052.43	22.9
liver-disorders	207.37	464.3	44.6	155.17	471.71	32.8
new-thyroid	54.41	296.52	18.3	49.6	298.4	16.6
pima-diabetes	263.06	1023.88	25.6	212.81	1045.4	20.3
wine	18.34	229.19	8	14.69	239.21	6.1
Average	823.45	3124.48	20.8	696.22	3184.44	16.8

TABLE III THE AVERAGE CLASSIFICATION ACCURACY OF 100 TRIALS FOR THE MCS WITH SHUFFLING AND BAGGING. THE STANDARD DEVIATION IS GIVEN INSIDE THE BRACKET ($\times 10^{-3}$)

	MCS with shuffled SGNT			MCS with bagged SGNT		
Dataset	optimized	unoptimized	ratio	optimized	unoptimized	ratio
balance-scale	0.864(6.19)	0.837(7.45)	+2.7	0.869(5.68)	0.848(7.93)	+2.1
breast-cancer-w	0.972(1.89)	0.968(2.05)	+0.4	0.972(2.45)	0.968(2.66)	+0.4
glass	0.728(11.87)	0.717(11.95)	+1.1	0.721(11.77)	0.716(13.73)	+0.5
ionosphere	0.912(5.63)	0.882(5.53)	+3	0.893(8.24)	0.868(7.79)	+2.5
iris	0.964(4.05)	0.964(3.94)	0	0.965(4.83)	0.961(4.74)	+0.4
letter	0.959(0.69)	0.958(0.68)	+0.1	0.956(0.76)	0.956(0.75)	0
liver-disorders	0.621(12.69)	0.605(13.03)	+1.6	0.624(14.88)	0.608(17.01)	+1.6
new-thyroid	0.958(5.19)	0.957(5.92)	+0.1	0.952(6.32)	0.949(6.76)	+0.3
pima-diabetes	0.747(6.28)	0.72(8.18)	+2.7	0.749(7.34)	0.730(8.71)	+1.9
wine	0.964(4.08)	0.958(5.29)	+0.6	0.965(4.73)	0.96(4.2)	+0.5
Average	0.868	0.856	+1.2	0.866	0.856	+1

TABLE IV THE IMPROVED PERFORMANCE OF THE PRUNED MCS AND THE MCS BASED ON C4.5 WITH BAGGING

	MCS based on SGNT			MCS based on C4.5		
Dataset	SGNT	MCS	ratio	C4.5	MCS	ratio
balance-scale	0.781	**0.869**	+8.8	0.795	0.827	+3.2
breast-cancer-w	0.957	**0.972**	+1.5	0.946	0.963	+1.7
glass	0.641	0.721	+8	0.664	**0.757**	+9.3
ionosphere	0.853	0.894	+4.1	0.897	**0.92**	+2.3
iris	0.949	**0.965**	+1.6	0.953	0.947	−0.6
letter	0.879	**0.956**	+7.7	0.880	0.938	+5.8
liver-disorders	0.58	0.624	+4.4	0.635	**0.736**	+10.1
new-thyroid	0.935	**0.952**	+1.7	0.93	0.94	+1
pima-diabetes	0.699	0.749	+5	0.749	**0.767**	+1.8
wine	0.95	**0.965**	+1.5	0.927	0.949	+2.2
Average	0.822	0.866	+4.4	0.837	**0.874**	+3

TABLE V THE CLASSIFICATION ACCURACY, THE MEMORY REQUIREMENT, AND THE COMPUTATION TIME OF TEN TRIALS FOR THE BEST PRUNED MCS AND k-NN

	classification acc.		memory requirement		computation time (s)	
Dataset	MCS	k-NN	MCS	k-NN	MCS	k-NN
balance-scale	0.882	**0.899**	**100.41**	562.5	**1.27**	2.52
breast-cancer-w	**0.977**	0.973	**26.7**	629.1	1.69	**1.31**
glass	**0.756**	0.706	**115.97**	192.6	0.48	**0.04**
ionosphere	**0.915**	0.875	**25.62**	315.9	1.7	**0.25**
iris	**0.973**	0.960	**10.9**	135	0.18	**0.05**
letter	0.958	**0.961**	**6273.15**	18000	**220.52**	845.44
liver-disorders	**0.666**	0.647	**150.28**	310.5	0.77	**0.6**
new-thyroid	**0.968**	**0.968**	**53.57**	193.5	0.34	**0.05**
pima-diabetes	**0.768**	0.753	**204.11**	691.2	**2.47**	3.41
wine	**0.978**	0.977	**12.2**	160.2	0.36	**0.13**
Average	**0.884**	0.872	**697.29**	2119.1	**22.98**	85.38