Self-Organizing Neural Grove: Efficient Multiple Classifier System with Pruned Self-Generating Neural Trees

Hirotaka Inoue

Abstract. Multiple classifier systems (MCS) have become popular during the last decade. Self-generating neural tree (SGNT) is a suitable base-classifier for MCS because of the simple setting and fast learning capability. However, the computation cost of the MCS increases in proportion to the number of SGNTs. In an earlier paper, we proposed a pruning method for the structure of the SGNT in the MCS to reduce the computational cost. In this paper, we propose a novel pruning method for more effective processing and we call this model self-organizing neural grove (SONG). The pruning method is constructed from both an on-line and an off-line pruning method. Experiments have been conducted to compare the SONG with an unpruned MCS based on SGNT, an MCS based on C4.5, and the *k*-nearest neighbor method. The results show that the SONG can improve its classification accuracy as well as reducing the computation cost.

1 Introduction

Classifiers need to find hidden information in the large amount of given data effectively and must 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, 4, 5]. In general, the base classifiers of the MCS use traditional models such as neural networks (backpropagation network and radial basis function network) [6] and decision trees (CART and C4.5) [7].

Neural networks have great advantages of adaptability, flexibility, and universal nonlinear input-output mapping capability. However, to apply these neural

Hirotaka Inoue
Kure National College of Technology, 2-2-11 Agaminami, Kure,
Hiroshima 737-8506, Japan
e-mail: hiro@kure-nct.ac.jp

L. Franco et al. (Eds.): Constructive Neural Networks, SCI 258, pp. 281–291.
springerlink.com

networks, it is necessary that human experts determine the network structure and some parameters, and it may be quite difficult to choose the right network structure suitable for a particular application at hand. Moreover, a long training time is required 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 trees (SGNTs) [8] have simple network design and high speed learning. SGNTs are an extension of the self-organizing maps (SOM) of Kohonen [9] and utilize competitive learning. The SGNT capabilities make it a suitable base classifier for the MCS. In order to improve the accuracy of SGNN, we propose ensemble self-generating neural networks (ESGNN) for classification [10] as one of the MCS. Although the accuracy of ESGNN improves by using various SGNTs, the computational cost, that is, the computation time and the memory capacity increases in proportion to the increasing number of SGNNs in the MCS.

In an earlier paper [11], we proposed a pruning method for the structure of the SGNN in the MCS to reduce the computational cost. In this paper, we propose a novel MCS pruning method for more effective processing and we call this model a self-organizing neural grove (SONG). This pruning method is comprised of two stages. At the first stage, we introduce an on-line pruning method to reduce the computational cost by using class labels in learning. At the second stage, we optimize the structure of the SGNT in the MCS 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 using 10-fold cross-validation [12]. After the optimization, the SONG can improve its classification accuracy as well as reducing the computational cost. Bagging [2] is used as a resampling technique for the SONG.

In this work, we investigate the improvement performance of the SONG by comparing it with an MCS based on C4.5 [13] using ten problems in a UCI machine learning repository [14]. Moreover, we compare the SONG with *k*-nearest neighbor (*k*-NN) [15] to investigate the computational cost and the classification accuracy. The SONG demonstrates higher classification accuracy and faster processing speed than *k*-NN on average.

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

2 Constructing Self-Organizing Neural Grove

In this section, we describe how to prune redundant leaves in the SONG. First, the on-line pruning method used in learning the SGNT is outlined. 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

SGNT is based on SOM and implemented as a competitive learning algorithm. The SGNT can be constructed directly from the given training data without any human intervention required. 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 explain some notations used.

- input data vector: $e_i \in \mathbb{R}^m$.
- root, leaf, and node in the SGNT: n_j.
- weight vector of n_j: $w_j \in \mathbb{R}^m$.
- the number of the leaves in n_j: c_j.
- distance measure: $d(e_i, w_j)$.
- winner leaf for 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 in Figure 1 where several sub procedures are used. Table 1 shows the sub procedures of the SGNT algorithm and their specifications.

In order to decide the winning 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:

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

```
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.
```

Fig. 1 SGNT algorithm

Table 1 Sub procedures of the SGNT algorithm

Sub procedure	Specification
$copy(n_j, e_i/w_{win})$	Create n_j, copy e_i/w_{win} as w_j in n_j.
$choose(e_i, n_1)$	Decide n_{win} for e_i.
$leaf(n_{win})$	Check n_{win} whether n_{win} is a leaf.
$connect(n_j, n_{win})$	Connect n_j as a child leaf of n_{win}.
$prune(n_{win})$	Prune leaves if they have the same class.

After all training data are inserted into the SGNT as the leaves, each one has a class label as the outputs and the weights of each node are the averages of the corresponding weights of all its leaves. The topology of the whole SGNT network reflects the given feature space. For more details concerning how to construct and perform the SGNT, see [8]. 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} parent node, if all leaves belong to the same class, then these leaves are pruned and the parent node is given the class.

2.2 Optimization of the SONG

The SGNT has a high speed processing capability. However, the accuracy of the SGNT is inferior to the conventional approaches, such as nearest neighbor, because the SGNT cannot guarantee reaching the nearest leaf for unknown data. Hence, we construct the SONG by taking the majority of plural SGNT outputs to improve the accuracy.

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 the SONG for large datasets even with the most advance computers.

In order to improve the classification accuracy, we propose an optimization method of 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 (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 (Figure 3).

```
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 ≥ the threshold value 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 cross validation
4     if the best classification accuracy is obtained,
5     then record the α as the optimal threshold value
6     α ← α + 0.05
7   until α = 1
8 end.
```

Fig. 3 The evaluation phase

2.3 An Example of the Pruning Method for SONG

We show an example of the pruning method for SONG 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 optimization stage in $\alpha = 1$. In this case, 159 leaves and 107 nodes are pruned away and 48 units remain. 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, 182 leaves and 115 nodes are pruned away and only 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.

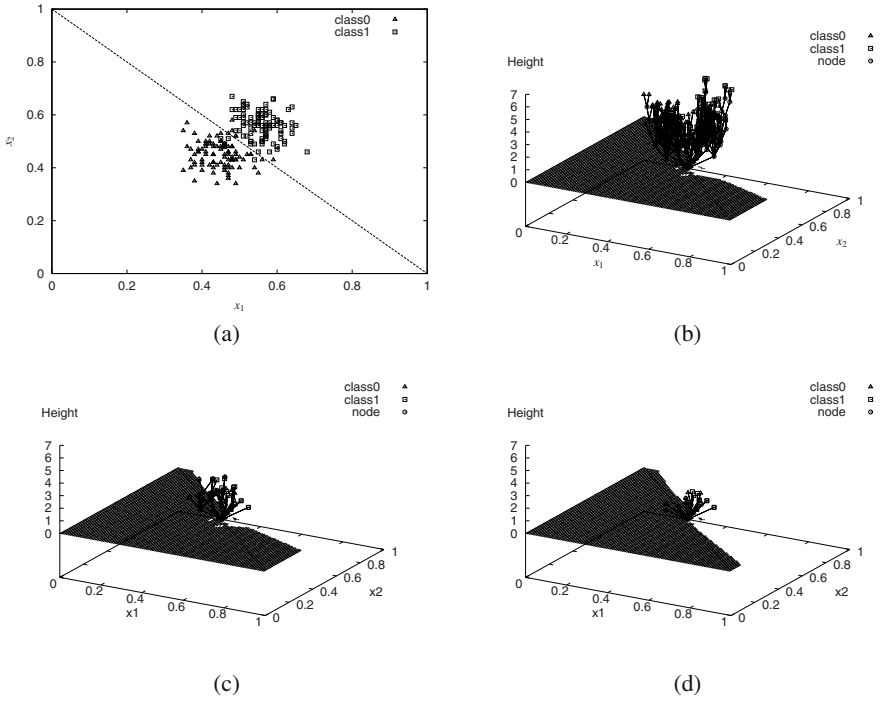

Fig. 4 An example of the SONG pruning algorithm, (a) a two dimensional classification problem with two equal circular Gaussian distribution, (b) the structure of the unpruned SGNT, (c) the structure of the pruned SGNT ($\alpha = 1$), and (d) the structure of the pruned SGNT ($\alpha = 0.6$). The shaded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

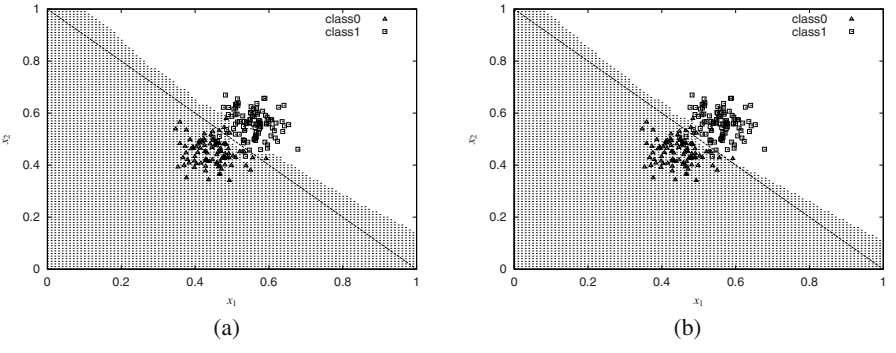

Fig. 5 An example of the SONG's decision boundary ($K = 25$), (a) $\alpha = 1$, and (b) $\alpha = 0.6$. The shaded plane is the decision region of class 0 by the SONG and the dotted line shows the ideal decision boundary

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 SONG from the same training data by changing the input order.

To show how well the SONG is optimized by the pruning algorithm, we show an example of the SONG in the same problem used above. Figure 5(a) and Figure 5(b) show the decision region of the SONG in $\alpha = 1$ and $\alpha = 0.6$, respectively. We set the number of SGNTs K to 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 the SONG with bagging for ten benchmark problems in the UCI machine learning repository [14]. Table 2 presents the abstract of the datasets.

We evaluate how 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 and k-NN. Since the performance of the SONG is not sensitive in the 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 SGNTs K in the SONG to 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 900MHz CPU, 1GB RAM, and Solaris 8.

Table 3 shows the average memory requirement and classification accuracy 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 between 65% to 96.6% and the classification accuracy is improved by 0.1% to 2.9% by optimizing the SONG. This confirms that the

Table 2 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

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

	memory requirement			classification accuracy		
Dataset	pruned	unpruned	ratio	pruned	unpruned	ratio
balance-scale	107.68	861.18	12.5	0.866(6.36)	0.837(7.83)	+2.9
breast-cancer-w	30.88	897.37	3.4	0.97(2.41)	0.966(2.71)	+0.4
glass	104.33	297.75	35	0.714(13.01)	0.709(14.86)	+0.5
ionosphere	50.75	472.39	10.7	0.891(6.75)	0.862(7.33)	+2.9
iris	15.64	208.56	7.4	0.962(6.04)	0.955(5.45)	+0.7
letter	6197.5	27028.56	22.9	0.956(0.77)	0.955(0.72)	+0.1
liver-disorders	163.12	471.6	34.5	0.648(12.89)	0.636(13.36)	+1.2
new-thyroid	49.45	298.21	16.5	0.958(7.5)	0.957(7.49)	+0.1
pima-diabetes	204.4	1045.03	19.5	0.749(7.05)	0.728(7.83)	+2.1
wine	15	238.95	6.2	0.976(4.41)	0.972(5.57)	+0.4
Average	693.88	3181.96	16.9	0.869	0.858	+1.1

Table 4 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	0.779	**0.866**	+8.7	0.795	0.827	+3.2
breast-cancer-w	0.956	**0.97**	+1.4	0.946	0.963	+1.7
glass	0.642	0.714	+7.2	0.664	**0.757**	+9.3
ionosphere	0.852	0.891	+3.9	0.897	**0.92**	+2.3
iris	0.943	**0.962**	+1.9	0.953	0.947	−0.6
letter	0.879	**0.956**	+7.7	0.880	0.938	+5.8
liver-disorders	0.59	0.648	+5.8	0.635	**0.736**	+10.1
new-thyroid	0.939	**0.958**	+1.9	0.93	0.94	+1
pima-diabetes	0.695	0.749	+5.4	0.749	**0.767**	+1.8
wine	0.955	**0.976**	+2.1	0.927	0.949	+2.2
Average	0.823	0.869	+4.6	0.837	**0.874**	+3

SONG can be effectively used for all datasets with regard to both the computational cost and the classification accuracy.

To evaluate SONG's performance, we compare it with an MCS based on C4.5. We set the number of classifiers K in the MCS to 25 and we construct both MCSs by bagging. Table 4 shows the improved performance of the SONG and the MCS based on C4.5. The results of the SGNT and the SONG are the average of 100 trials. The SONG performs better 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, SONG can improve the classification accuracy for all problems. Therefore, SONG is an

Table 5 The classification accuracy, the memory requirement, and the computation time of ten trials for the best pruned SONG and *k*-NN

	classification acc.		memory requirement		computation time (s)	
Dataset	SONG	*k*-NN	SONG	*k*-NN	SONG	*k*-NN
balance-scale	0.878	**0.888**	**109.93**	562.5	**0.82**	1.14
breast-cancer-w	**0.974**	0.969	**26.8**	629.1	**1.18**	1.25
glass	**0.758**	0.701	**91.33**	192.6	0.36	**0.08**
ionosphere	**0.912**	0.866	**51.38**	315.9	1.93	**0.2**
iris	**0.973**	0.96	**11.34**	135	0.13	**0.05**
letter	0.958	**0.96**	**6208.03**	18000	**208.52**	503.14
liver-disorders	**0.685**	0.653	**134.17**	310.5	**0.54**	0.56
new-thyroid	**0.972**	**0.972**	**45.74**	193.5	0.23	**0.05**
pima-diabetes	**0.764**	0.751	**183.57**	691.2	**1.72**	2.49
wine	**0.983**	0.977	**11.8**	160.2	0.31	**0.15**
Average	**0.885**	0.869	**687.41**	2119.1	**21.57**	50.91

efficient 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 SONG, we compare it with *k*-NN on the same problems. The best classification accuracy of 100 trials with bagging were chosen. In *k*-NN, we choose the best accuracy where *k* is 1,3,5,7,9,11,13,15, and 25 with 10-fold cross-validation. All methods are compiled using gcc with the optimization level `-O2` on the same workstation.

Table 5 shows the classification accuracy, the memory requirement, and the computation time achieved by the SONG and *k*-NN. Although there are compression methods available for *k*-NN [16], they take enormous computation time to construct an effective model. We use the exhaustive *k*-NN in this experiment. Since *k*-NN does not discard any training sample, the size of this classifier corresponds to the training set size. The results of *k*-NN correspond to the average measures obtained by 10-fold cross-validation, the same experimental procedure adapted in SONG. Next, we show the results for each category.

First, with regard to the classification accuracy, SONG is superior to *k*-NN for 8 of the 10 datasets and gives 1.6% improvement on average. 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 *k*-NN for all problems. Although the memory requirement of the SONG is totally used *K* times in Table 5, 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 SONG consumes the cost of *K* times the SGNT to construct the model and test for the unknown dataset, the average computation time is faster than *k*-NN. The SONG is slower than *k*-NN for small datasets such as glass, ionosphere, and iris. However, it is faster than *k*-NN for large datasets such as balance-scale, letter, and

pima-diabetes. In the case of letter, in particular, the computation time of the SONG is faster than k-NN by about 2.4 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 SONG based on the fast and compact SGNT is useful and practical for large datasets. Moreover, the SONG is capable parallel computation because each classifier behaves independently. In conclusion, the SONG is a practical method for large-scale data mining compared with k-NN.

4 Conclusions

In this paper, we proposed a new pruning method for the MCS based on SGNT, which is called SONG, and evaluated the computation cost and the accuracy.We introduced an on-line and off-line pruning method and evaluated the SONG by 10-fold cross-validation. Experimental results showed that the memory requirement is significant reduce, and by using the pruned SGNT as the base classifier of the SONG, accuracy is increased. The SONG is a useful and practical MCS to classify large datasets. In future work, we will study an incremental learning and a parallel and distributed processing of the SONG for large scale data mining.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2000)
2. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
3. Schapire, R.E.: The strength of weak learnability. Machine Learning 5(2), 197–227 (1990)
4. Quinlan, J.R.: Bagging, Boosting, and C4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR, August 4-8, 1996, pp. 725–730. AAAI Press, The MIT Press (1996)
5. Rätsch, G., Onoda, T., Müller, K.R.: Soft margins for AdaBoost. Machine Learning 42(3), 287–320 (2001)
6. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons Inc., New York (2000)
8. Wen, W.X., Jennings, A., Liu, H.: Learning a neural tree. In: the International Joint Conference on Neural Networks, Beijing, China, November 3-6, 1992, vol. 2, pp. 751–756 (1992)
9. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1995)
10. Inoue, H., Narihisa, H.: Improving generalization ability of self-generating neural networks through ensemble averaging. In: Terano, T., Liu, H., Chen, A.L.P. (eds.) PAKDD 2000. LNCS, vol. 1805, pp. 177–180. Springer, Heidelberg (2000)

11. Inoue, H., Narihisa, H.: Optimizing a multiple classifier system. In: Ishizuka, M., Sattar, A. (eds.) PRICAI 2002. LNCS (LNAI), vol. 2417, pp. 285–294. Springer, Heidelberg (2002)
12. Stone, M.: Cross-validation: A review. Math. Operationsforsch. Statist. Ser. Statistics 9(1), 127–139 (1978)
13. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
14. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
15. Patrick, E.A., Frederick, P., Fischer, I.: A generalized k-nearest neighbor rule. Information and Control 16(2), 128–152 (1970)
16. Zhang, B., Srihari, S.N.: Fast k-nearest neighbor classification using cluster-based trees. IEEE Transactions on Pattern and Machine Intelligence 26(4), 525–528 (2004)