Int. J. Appl. Math. Comput. Sci., 2018, Vol. 28, No. 4, 705–717
DOI: 10.2478/amcs-2018-0054

EFFICIENT DECISION TREES FOR MULTI–CLASS SUPPORT VECTOR MACHINES USING ENTROPY AND GENERALIZATION ERROR ESTIMATION

PITTIPOL KANTAVAT [a], BOONSERM KIJSIRIKUL [a,*], PATOOMSIRI SONGSIRI [a], KEN-ICHI FUKUI [b], MASAYUKI NUMAO [b]

[a] Department of Computer Engineering
Chulalongkorn University, Pathumwan, Bangkok 10330, Thailand
e-mail: {pittipol.k,boonserm.k,patoomsiri.s}@chula.ac.th

[b] Institute of Scientific and Industrial Research
Osaka University, Osaka 567-0047, Japan
e-mail: {fukui,numao}@ai.sanken.osaka-u.ac.jp

We propose new methods for support vector machines using a tree architecture for multi-class classification. In each node of the tree, we select an appropriate binary classifier, using entropy and generalization error estimation, then group the examples into positive and negative classes based on the selected classifier, and train a new classifier for use in the classification phase. The proposed methods can work in time complexity between O($\log_2 N$) and O(N), where N is the number of classes. We compare the performance of our methods with traditional techniques on the UCI machine learning repository using 10-fold cross-validation. The experimental results show that the methods are very useful for problems that need fast classification time or those with a large number of classes, since the proposed methods run much faster than the traditional techniques but still provide comparable accuracy.

Keywords: support vector machine, multi-class classification, generalization error, entropy, decision tree.

1. Introduction

Support vector machines (SVMs) (Vapnik, 1998; 1999) were originally designed to solve binary classification problems by constructing a hyperplane to separate two-class data with maximum margin. For dealing with multi-class classification problems, there have been two main approaches. One is to solve a single optimization problem (Vapnik, 1998; Bredensteiner and Bennett, 1999; Crammer and Singer, 2002), and the other is to combine several binary classifiers. Hsu and Lin (2002) suggested that the latter approach may be more suitable for practical use.

The general techniques of combining several classifiers are the one-versus-one (OVO) method (Knerr *et al*., 1990) and the one-versus-all (OVA) method. The former requires $N \times (N-1)/2$ binary classifiers for an N-class problem. Usually, OVO determines the target class using the strategy called max-wins (Friedman, 1996) that selects the class with the highest vote of binary classifiers. OVA requires N classifiers for an N-class problem. In the OVA training process, the i-th classifier employs the i-th class data as positive examples and the remaining classes as negative ones. The output classes are determined by selecting the class with the highest classification score. Both techniques are widely used for classification problems.

There are several techniques that enhance the traditional OVO and OVA. Hastie and Tibshirani (1998) improve the accuracy of OVO by using joint probability estimation for pairwise classes. Kumar and Gopal (2011) have proposed reduced one-against-all, a method that decreases the training time of the original OVA. Chmielnicki and Stąpor (2016) employ the OVA strategy with samples balancing to improve pairwise coupling classification. In this paper, we shall base our techniques on the OVO method.

*Corresponding author

Although OVO generally yields high accuracy results, it takes O(N^2) classification time and thus is not suitable for a problem with a large number of classes. To reduce the running time, the decision directed acyclic graph (DDAG) (Platt *et al*., 2000) and the adaptive directed acyclic graph (ADAG) (Kijsirikul *et al*., 2002) techniques have been proposed. Both methods build $N \times (N-1)/2$ binary classifiers but employ only $N-1$ classifiers to determine the output class. Though both methods reduce the running time to O(N), their accuracy is usually lower than that of OVO.

Some techniques apply the decision-tree structure to eliminate more than one candidate class at each node (classifier) of the tree using OVO classifiers. Fei and Liu (2006) proposed the binary tree of SVMs that select tree node classifiers randomly or through the training data centroids. Songsiri *et al*. (2008) proposed the information-based dichotomization tree, which uses entropy for classifier selection. Chen *et al*. (2009) employ the adaptive binary tree, which applies the minimization of the average number of support vectors for tree construction. Bala and Agrawal (2011) proposed optimal decision-tree-based multi-class SVMs that calculate statistical measurements for decision-tree construction. These techniques share a common disadvantage that a selected classifier may not perfectly separate examples of a class to only the positive or negative side, and hence some techniques allow data of a class to be duplicated to more than one node of the tree.

There are also many techniques that construct a hierarchical structure SVM using OVA classifiers or clustering techniques. Takahashi and Abe (2002) proposed a decision-tree-based SVM that calculates the Euclidean and the Mahalanobis distance as a separability measure for class grouping. Madzarov *et al*. (2009) presented an SVM binary decision tree using class centroids in the kernel space. Cheong *et al*. (2004) introduced support vector machines with a binary tree architecture that apply kernel-based self-organizing map in the tree construction. Lei and Govindaraju (2005) proposed a half-against-half multi-class SVM that divides data classes into two groups randomly and constructs a hierarchy structure for the classification. Liu *et al*. (2008) proposed the multi-state-mapping SVM that applies kernel functions to the k-means algorithm for grouping data classes and constructing a hierarchical SVM. Kumar and Gopal (2010) proposed decision-tree-based OVA, a method inspired by BTS that applies the probabilistic output of the SVM to determine the tree structure. Yang *et al*. (2013) presented the single-space-mapped binary tree SVM and the multi-space-mapped binary tree SVM, hierarchical OVA-based techniques that use the Euclidean distance to determine the set of hyper-parameters. Dong *et al*. (2015) introduced a method that uses similarity clustering to group classes for an OVA-based tree structure.

In this paper, we propose two novel techniques for the problem with a large number of classes. One technique is the *information-based decision tree SVM,* which employs entropy to evaluate the quality of OVO classifiers in the node construction process. The other technique is the *information-based and generalization-error estimation decision tree SVM,* which enhances the first technique by integrating generalization error estimation. The key mechanism of both the techniques is the method called *class-grouping-by-majority*: when a classifier of a tree node cannot perfectly classify examples of any class into either only a positive or a negative side of the classifier, the method will group the whole examples of that class into only one side that contains the majority of the examples, and then train a new classifier for the node.

We ran experiments comparing our proposed techniques with traditional techniques using twenty datasets from the UCI machine learning repository, and conducted a significant test using the Friedman aligned rank test (Friedman, 1996) and the Hommel procedure (García *et al*., 2010) as a *post-hoc* procedure. The results indicate that our methods are useful, especially for problems that need fast classification or those with a large number of classes.

This paper is organized as follows. Section 2 discusses tree-structure multi-class SVM techniques. Section 3 proposes our techniques. Section 4 reports experimental details. Section 5 summarizes our work.

2. OVO-based decision tree SVM

2.1. Binary tree of the SVM. The binary tree SVM (BTS) (Fei and Liu, 2006) randomly selects binary classifiers to be used as decision nodes of the tree. As mentioned previously, the BTS allows duplicated classes to be scattered in the tree. Basically, data of a class will be duplicated into the left and right child nodes of a decision node when a classifier of the decision node does not completely classify the whole data of the class into only one side (either positive or negative). An alternative version of the BTS, c-BTS, excludes the randomness by using data centroids. In the first step, the centroid of all data is calculated. Then, the centroid of each data class and its Euclidean distance to the centroid of all-data are calculated. Finally, the (i vs j) classifier is selected such that the centroid of class i and the centroid of class j have the nearest distances to the all-data centroid.

Illustrations of the BTS and c-BTS are shown in Fig. 1. At the root node, classifier 1 vs 2 is selected. Classes 1, 4 and classes 2, 3 are separated to positive and negative sides, respectively. However, classes 5 and 6 cannot be separated into only one side. They are reassigned to both positive and negative child nodes. The

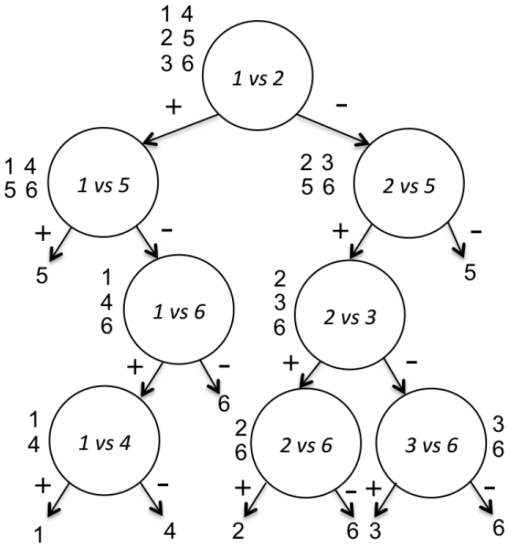


Fig. 1. Illustration of the binary tree of the SVM (BTS). Some data classes may be scattered to several leaf nodes (Fei and Lui, 2006).

recursive process continues until finished. Eventually, the duplicated leaf-nodes of classes 5 and 6 appear more than once in the tree.

The classification accuracy and time complexity of the BTS and c-BTS may vary according to the configuration of a hyper-parameter value. A higher value may produce higher accuracy but will also increase the running time. The time complexity can be O($\log_2 N$) in the best situation. However, the average time complexity was proven to be $\log_{4/3}((N+3)/4)$ (Fei and Liu, 2006).

2.2. Information-based dichotomization. Information-based dichotomization (IBD) (Songsiri *et al*., 2008) employs information theory to create a multi-class classification tree. To construct each node of the tree, IBD prefers to select the OVO candidates with minimum entropy. Therefore, a data class with a high probability of occurrence will be separated first, and this kind of class node tends to be found in very few levels from the root node.

Similarly to the BTS, IBD also faces the problem that a selected OVO classifier may not perfectly classify the examples with the same class label into only a positive or negative side. This leads us to obtain a tree of greater depth. To alleviate this situation, IBD employs a pruning process to remove some minority examples on the side whose percentage of the minority is less than a threshold. IBD then determines the optimal value for the threshold which is referred to as the optimal pruning percentage P. However, it is not always easy to find the parameter P without a risk of losing useful information. Additionally, in IBD the other parameter is also introduced to avoid selecting an OVO classifier with low generalization performance. This parameter is called the optimal range of the generalization performance of classifiers (referred to as parameter R). These two parameters are obtained by using k-fold cross validation.

3. Proposed methods

We propose two novel techniques that are aimed at achieving high classification speed and may sacrifice classification accuracy to some extent. We expect them to work in the time complexity of O($\log_2 N$) in the best case, and thus the proposed techniques are suitable for the problem with a large number of classes that cannot be solved efficiently in practice by the methods with O(N^2) classification time.

3.1. Information-based decision tree. The information-based decision tree (IB-DTree) is an OVO-based multi-class classification technique. It builds a tree by adding decision nodes one by one; it selects the binary classifier with minimum entropy as the *initial classifier* of the decision node. Minimum entropy classifiers will lead to a fast classification time for the decision tree because data classes with high probabilities of occurrence will be found within a few steps from the root node. The initial classifier will be adjusted further to be the *final classifier* for the decision node as described later.

The entropy of a binary classifier h can be calculated as

$$\begin{aligned} Entropy(h) = p^+ \times \Big(\sum_{i=1}^{N} -p_i^+ \log_2 p_i^+\Big) \\ + p^- \times \Big(\sum_{i=1}^{N} -p_i^- \log_2 p_i^-\Big), \end{aligned} \tag{1}$$

where p^+ and p^- are the ratios of positive and negative examples (corresponding to the classifier) to all training examples, respectively. Similarly, p_i^+ is the proportion of positive examples of the class i to all positive examples. By the same token, p_i^- is the proportion of negative examples of the class i to all negative examples. If there is no positive (or negative) example of classifier h for any class, the term ($p_i^+ \log_2 p_i^+$) or ($p_i^- \log_2 p_i^-$) of that class will be defined as 0. From $N \times (N-1)/2$ OVO classifiers, the classifier with minimum entropy is selected using Eqn. (1) as the initial classifier h.

Examples of a specific class may scatter on both positive and negative sides of the initial classifier. The key mechanism of IB-DTree is the class-grouping-by-majority method, presented in Algorithm 1, which groups examples of each class having scattering examples so that they are on the same side containing the majority

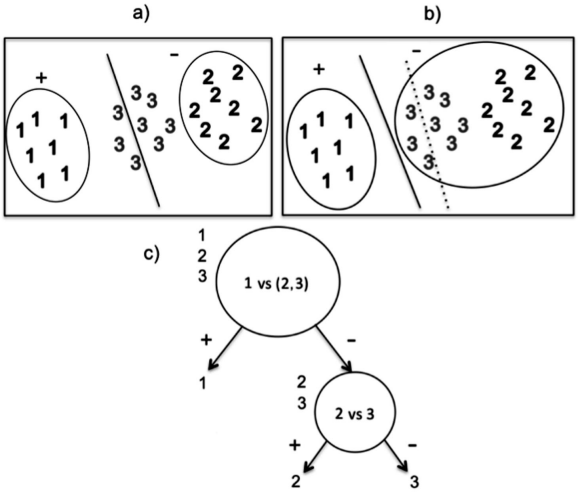


Fig. 2. Example of the class-grouping-by-majority strategy: before grouping (a), after grouping (b), the decision tree after grouping (c).

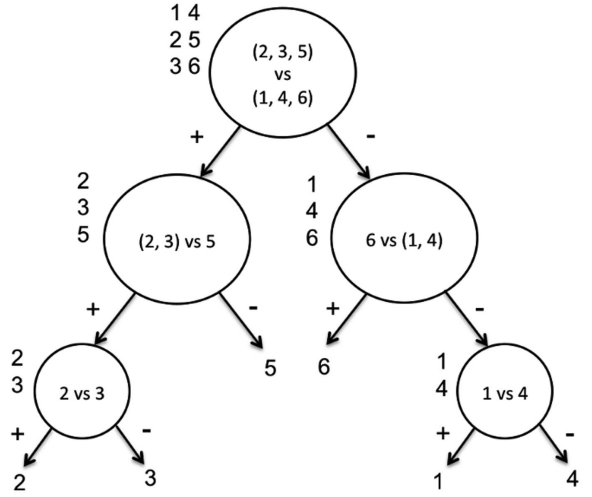


Fig. 3. Illustration of the information-based decision tree (IB-DTree). No duplicated class at the leaf nodes.

of examples. Using the groups of examples labeled by class-grouping-by-majority, IB-DTree then trains the final classifier h' of the decision node. A traditional OVO-based algorithm might face the problem when encountering data of another class k using classifier h = (i vs j) and having to scatter examples of class k to both left and right child nodes. Our method will never face this situation, because data of class k will always be grouped in either a positive or negative class of classifier h' = (P vs N). Hence, there is no need to duplicate class k data to the left or right child node, and the tree depth will not be increased unnecessarily. The tree without unnecessary extra-depth reduces not only decision times to derive the answer, but also the cumulative error of the classification. This is because every classifier along the path to the answer may produce a wrong prediction. Therefore, the deeper the tree, the more cumulative error it produces. However, this is a trade-off with the quality/accuracy of the classifier that is more difficult to reach with a large number of data classes to be separated.

An example of class-grouping-by-majority for a 3-class problem is shown in Fig. 2. Suppose that we select the initial classifier 1 vs 2 as h for the root node. In Fig. 2(a), most of class-3 data are on the negative side of the hyperplane. Therefore, we assign all training data of class-3 as negative examples and train classifier 1 vs (2, 3) as a new classifier h' for use in the decision tree as in Fig. 2(b). As a result, we obtain a decision tree constructed by IB-DTree as shown in Fig. 2(c).

To illustrate IB-DTree, we show in Fig. 3 a decision tree constructed by IB-DTree using the same example as in Fig. 1 of the BTS method. At the root node, classifier 1 vs 2 is selected as h. Most of the training examples of classes 3 and 5 are on the positive side of h, while a majority of training examples of classes 4 and 6 are on the negative side of h. Consequently, (2, 3, 5) vs (1, 4, 6) is trained as classifier h'. For the remaining steps of the tree, the process continues recursively until finished, and there is no duplicated class leaf-node in the tree.

As described in Algorithm 2, IB-DTree constructs a tree using a recursive procedure starting from the root node from lines 1–7 with all candidate classes. The node-adding procedure will be processed from lines 8–19. First, the initial classifier h with the lowest entropy will be selected. Second, data of each class will be grouped to either the positive group (P) or the negative group (N). Then, the final classifier h' will be trained using P and N and will be designed as a decision node. Finally, the algorithm processes the child nodes recursively and stops the process at the leaf nodes when the stopping condition holds.

There are several benefits of using IB-DTree. First, there is no duplicated class leaf-node and thus the depth of the tree is small. Second, no parameter tuning is required as there is no parameter in the class-grouping-by-majority process. Third, as entropy is used in pairwise class selection, the data class with a high probability of occurrence has a higher chance to be selected as the initial classifier and to be found in few levels from the root node, and thus the average decision time is low. Finally, there is no information loss in IB-DTree, as no data pruning is used.

3.2. Information-based and generalization-error estimation decision tree. The information-based and generalization-error estimation decision tree

Algorithm 1. Class-grouping-by-majority.

procedure Class-grouping-by-majority (selected classifier h, candidate classes K)
  Initialize set of positive classes $P = \emptyset$ and set of negative classes $N = \emptyset$
  for each class $i \in K$:-
    Label all data of class i to (+) and (–) separated by initial classifier h
    $p \leftarrow$ count(+), $n \leftarrow$ count(–)
    if ($p > n$) then $P \leftarrow P \cup \{i\}$
    else $N \leftarrow N \cup \{i\}$
  end for
  Train final classifier h' = (P vs N)
  return h', P, N
end procedure

Algorithm 2. Information-based decision tree SVM (IB-DTree).

1: **procedure** IB-DTree
2: Initialize the tree T with root node $Root$
3: Initialize the set of candidate output classes $S = \{1, 2, 3, ..., N\}$
4: Create all binary classifiers (i vs j); $i, j \in S$
5: Construct Tree ($Root$, S)
6: **return** T
7: **end procedure**
8: **procedure** Construct Tree (node D, candidate classes K)
9: **for** each binary classifier (i vs j); $i, j \in K$; $i < j$:-
10: Calculate the entropy using training data of all classes in K
11: **end for**
12: initial classifier $h \leftarrow$ classifier (i vs j) with the lowest entropy
13: final classifier h', positive classes P, negative classes $N \leftarrow$ class-grouping-by-majority(h, K)
14: D.classifier $\leftarrow h'$
15: Initialize new node L; D.left-child-node $\leftarrow L$
16: Initialize new node R; D.right-child-node $\leftarrow R$
17: **if** $|P| > 1$ **then** Construct Tree (L, P) **else** L is the leaf node with answer class P
18: **if** $|N| > 1$ **then** Construct Tree (R, N) **else** R is the leaf node with answer class N
19: **end procedure**

(IBGE-DTree) is an enhanced version of IB-DTree. In the node-building process, IBGE-DTree selects classifiers using both entropy and generalization error estimation.

The details of IBGE-DTree are described in Algorithm 3. The IBGE-DTree algorithm is different from IB-DTree in lines 12–17. Instead of selecting classifiers based only on the lowest entropy, it also considers the generalization error of the classifiers. First, IBGE-DTree ranks the classifiers in ascending order by the entropy. Then, it trains some classifiers using the class-grouping-by-majority technique and selects the classifier with the lowest generalization error. The positive group (P) and negative group (N) for building the child nodes in lines 22–23 are obtained from the classifier with the lowest generalization error in lines 18–19.

Generalization error estimation is the evaluation of a learning model's actual performance on unseen data. For SVMs, a model is trained using the concept of the structure risk minimization principle (Vapnik V.N., 1974). The performance of an SVM is based on the VC dimension of the model and the quality of fitting training data (or the empirical error). The expected risk R (α) is bounded in accordance with the following equation (Burges, 1998; Bartlett and Shawe-Taylor, 1999):

$$R(\alpha) \leq \frac{l}{m} + \sqrt{\frac{c}{m}\Big(\frac{R^2}{\Delta^2}\log^2 m + \log\frac{1}{\delta}\Big)}, \qquad (2)$$

where l, R, Δ and m are respectively the number of labeled examples with margin less than Δ, the radius of the smallest sphere that contains all data points, the distance between the hyperplane and the closest points of the training set (margin size), and the number of training data. The first and second terms of (2) define the empirical error and the VC dimension, respectively.

The generalization error can be estimated directly using k-fold cross-validation and used to compare the performance of binary classifiers, but it consumes a

Algorithm 3. Information-based and generalization-error estimation decision tree SVM (IBGE-DTree).

procedure IBGE-DTree
  Initialize the tree T with root node $Root$
  Initialize the set of candidate output classes $S = \{1, 2, 3, \ldots, N\}$
  Create the all binary classifiers (i vs j); $i, j \in S$
  Construct Tree ($Root$, S)
  return T
end procedure
procedure Construct Tree (node D, candidate classes K)
  for each binary classifiers (i vs j); $i, j \in K$; $i < j$:-
    Calculate the entropy using training data of all classes in K
  end for
  Sort the list of the initial classifiers (i vs j) in ascending order by the entropy as $h_1, \ldots, h_{all}$
  for each initial classifiers h_s; $s = \{1, 2, 3, \ldots, n\}$, n = number of considering classifiers
    final classifier h'_s, positive classes P_s, negative classes $N_s \leftarrow$ class-grouping-by-majority(h_s, K)
    calculate generalization error estimation of final classifier h'_s
  end for
  D.classifier $\leftarrow$ final classifiers with the lowest generalization error among $h'_1, \ldots, h'_n$
  $P' \leftarrow P_s$ used for training the final classifier with the lowest generalization error estimation
  $N' \leftarrow N_s$ used for training the final classifier with the lowest generalization error estimation
  Initialize new node L; D.left-child-node $\leftarrow L$
  Initialize new node R; D.right-child-node $\leftarrow R$
  if $|P'| > 1$ **then** Construct Tree (L, P') **else** L is the leaf node with answer class P'
  if $|N'| > 1$ **then** Construct Tree (R, N') **else** R is the leaf node with answer class N'
end procedure

high computational cost. Another method to estimate the generalization error is by using the inequality (2) with appropriate parameter substitution (Songsiri *et al*., 2015). Using the latter method, we can compare the relative generalization error on the same datasets and environments. In Section 4, we set the values of $c = 0.1$ and $\delta = 0.01$ in the experiments.

As IBGE-DTree is an enhanced version of IB-DTree, its benefits are very similar to those of IB-DTree. However, as it combines generalization error estimation with entropy, the selected classifiers are more accurate than those of IB-DTree.

The use of the generalization error to enhance the accuracy of OVO-based methods can also be found in the work of Songsiri *et al*. (2015). The focus of the work is to increase the accuracy of the base methods, i.e., DDAG, ADAG, Max-Wins, with the use of more classifiers in some cases. The number of classifiers used by Songsiri *et al*. (2015) is between $N - 1$ and $N \times (N - 1)/2$. On the other hand, here our main objective is to reduce the number of classifiers to the rouge between $\log_2 N$ and $N - 1$, which provides more practical use in the case of datasets with a large number of classes.

IBD (Songsiri *et al*., 2008) also employs entropy and the generalization error to construct tree-based SVMs. IBD shares a similar advantage as our methods, i.e., the use of entropy and the generalization error to select good classifiers for nodes of the tree. However, the process of tree construction in IBD is different from our methods, and IBD has no class-grouping-by-majority mechanism. Therefore, IBD has to duplicate data of some classes which are not perfectly separated into one side, or it may employ a pruning strategy to remove minority examples of a class when the percentage of the number of minority examples is below the threshold (referred to as P in the paper). The duplication then increases the depth of the tree as these classes cannot be immediately removed from the candidate classes. Thus, the depth of the tree and the number of decisions to derive the answer of IBD is larger than those of our proposed methods. In addition, the use of pruning may bring risk of information loss. Another difference between IBD and our methods is that the former has more hyper-parameters, i.e., P and R *(the optimal range of generalization performance)*, than the latter. Thus, IBD needs a more complicated tuning process than required by our methods.

3.3. Examples of IB-DTree and IBGE-DTree. To demonstrate the proposed techniques, we show trees constructed by IB-DTree and IBGE-DTree using training data from Mfeat-Factor and Cardiotocography datasets. Mfeat-Factor is a 10-class dataset with 200 examples in each class. Since every class contains an equal number of examples, a tree of IB-DTree is constructed in a balanced structure, as in Fig. 4.

For IBGE-DTree of the Mfeat-Factor dataset, as it selects the lowest generalization error among low-entropy classifiers, the obtained tree structure is not balanced, as shown in Fig. 5. For the performance, IBGE-DTree yields better accuracy than IB-DTree, but it consumes more decision times, as shown in Tables 2 and 6 in Section 4.

Another example, in Fig. 6, is a tree of IB-DTree constructed using training data from the Cardiotocography dataset. Cardiotocography is a 10-class dataset in which the numbers of examples are 384, 579, 53, 81, 72, 332, 252, 107, 69 and 197, respectively. In the tree construction process, since classes 1, 2, 6, 7 contain more examples than classes 3, 4, 5, 8, 9 and 10, they have a higher chance to be selected for an initial classifier. As shown in Fig. 6, (2 vs 6) is selected at the root node. At the left and right child nodes, (1 vs 2) and (6 vs 7) are selected, respectively. Although the tree structure is imbalanced, it consumes low decision times since data of classes with high probability of occurrence are placed near the root node, as shown in Section 4.

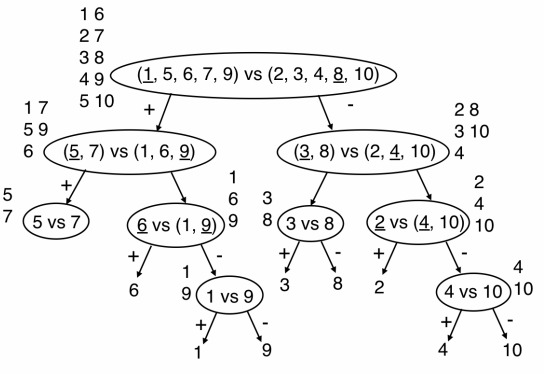


Fig. 4. IB-DTree constructed using the Mfeat-Factor dataset. The underlined numbers are classes that are selected for an initial classifier in each node.

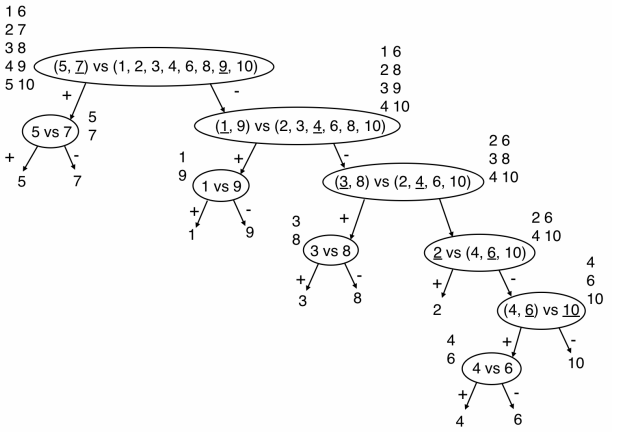


Fig. 5. IBGE-DTree constructed using the Mfeat-Factor dataset. The underlined numbers are classes that are selected for an initial classifier in each node.

4. Experiments and results

We performed some experiments to compare the proposed methods, IB-DTree and IBGE-DTree, with the traditional strategies, i.e., OVO, OVA, DDAG, ADAG, BTS-G and c-BTS-G.

We ran the experiments based on 10-fold cross-validation on twenty datasets from the UCI repository (Blake and Merz, 1998), as shown in Table 1. For the datasets containing both training and test data, we merged the data into a single set, and then we used 10-fold cross validation to evaluate classification accuracy. We normalized the data to the range $[-1, 1]$. We used the software package SVM*light*, version 6.02 (Joachims, 1999). The binary classifiers were trained using the RBF kernel. A suitable kernel parameter (γ) and regularization parameter C for each dataset were selected from {0.001, 0.01, 0.1, 1, 10} and {1, 10, 100, 1000}, respectively.

To compare the performance of IB-DTree and IBGE-DTree with the other tree-structure techniques, we also implemented BTS-G and c-BTS-G, which are our enhanced versions of BTS and c-BTS (Fei and Liu, 2006) by applying class-grouping-by-majority to improve efficiency of the original BTS and c-BTS. As a result, BTS-G and c-BTS-G select the pairwise classifiers in the same way as the original versions, but they employ classifiers that are trained using positive and negative groups of classes instead of employing OVO classifiers that are trained using only two classes. Compared with the original version, BTS-G and c-BTS-G contain no duplicated class leaf node. This enhancement is aimed to reduce the decision times as well as the cumulative error in obtaining the answer. For BTS-G, we selected the

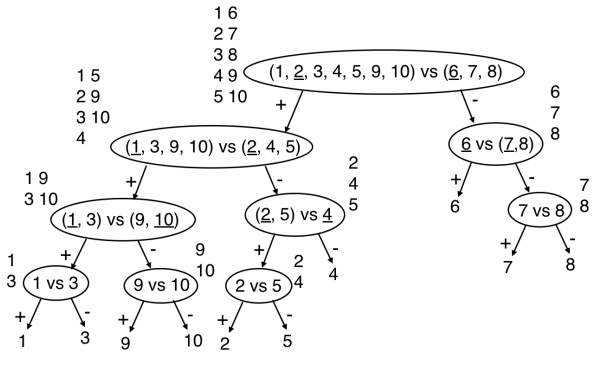


Fig. 6. IB-DTree constructed using the Cardiotocography dataset. The underlined numbers are classes that are selected for an initial classifier in each node.

Table 1. Experimental dataset.

Dataset Name	#Classes	#Attributes	#Examples
Page Block	5	10	5473
Segment	7	18	2310
Shuttle	7	9	58000
Arrhyth	9	255	438
Cardiotocography	10	21	2126
Mfeat-Factor	10	216	2000
Mfeat-Fourier	10	76	2000
Mfeat-Karhunen	10	64	2000
Optdigit	10	62	5620
Pendigit	10	16	10992
Primary Tumor	13	15	315
Libras Movement	15	90	360
Abalone[1]	16	8	4098
Krkopt	18	6	28056
Spectrometer	21	101	475
Isolet	26	34	7797
Letter	26	16	20052
Plant Margin	100	64	1600
Plant Shape	100	64	1600
Plant Texture	100	64	1599

classifier for each node randomly 10 times and calculated the average results. For c-BTS-G, we selected the pairwise classifiers in the same way to the original c-BTS.

For DDAG and ADAG where the initial order of classes affected the final classification accuracy, we examined all datasets by randomly selecting 50,000 initial orders and calculated the average classification accuracy. For IBGE-DTree, we set n (the number of considering classifiers in line 13 of Algorithm 3) to 20 percent of all possible classifiers. For example, if there are 10 classes to be determined, the number of all possible classifiers will be 45. Thus, the value of n will be 9.

The experimental results are shown in Tables 2–6. Table 2 presents the classification accuracies and Table 3 shows the win-lose-draw between the techniques under comparison. Table 4 shows the Friedman aligned ranks test (Friedman, 1996) and the Hommel procedure (García *et al.*, 2010) to assess the accuracy of our methods compared with the other tree-structure methods. Table 5 shows the Friedman aligned ranks test and the Hommel procedure to assess the error rate of our methods compared with the other non-tree-structure methods. Table 6 shows the average decision times that are used to determine the output class of a test example.

In Table 2, the bold number indicates the highest accuracy in each dataset. The number in the parentheses shows the ranking of each technique. The highest accuracy is obtained by OVO, followed by ADAG, OVA and DDAG. Among the tree structure techniques, IBGE-DTree yields the highest accuracy, followed by IB-DTree, BTS-G and c-BTS-G.

Table 3 shows the pairwise win-lose-draw between the techniques under comparison. The results indicate that OVO outperforms all other techniques. Among the tree structure techniques, IBGE-DTree provides the highest accuracy results. BTS-G and c-BTS-G underperform the other techniques.

Table 4 shows rankings and adjusted p-values of the classification accuracies from Table 2 using the Friedman aligned ranks test and the Hommel procedure for the proposed methods (IB-DTree and IBGE-DTree) as control algorithms and the tree-structure methods (BTS-G and c-BTS-G) as traditional algorithms. The empirical results show that the rankings of IB-DTree and IBGE-DTree are better than those of BTS-G and c-BTS-G. The adjusted p-values also indicate that our proposed techniques significantly outperform the comparison techniques, with a significance level less than 0.05.

Table 5 shows rankings and adjusted p-values of error rates using the Friedman aligned ranks test and the Hommel procedure to test a significant difference between error rates of our techniques and those of the traditional non-tree-based techniques, i.e., OVO, OVA, DDAG and ADAG. The results show that IB-DTree significantly underperforms the non-tree-based techniques, in terms of accuracy or the error rate, while IBGE-DTree is insignificantly different from OVA, DDAG and ADAG, with a significance level less than 0.05.

Table 6 shows the average number of decisions required to determine the output class of a test example. The lower the average number of decisions, the faster the classification speed. IB-DTree and IBGE-DTree are the fastest among the techniques compared, while OVO is the slowest one.

The experiments show that IBGE-DTree is the most efficient technique among the tree-structure methods. It outputs the answer very fast and provides accuracy comparable to that of OVA, DDAG and ADAG. IBGE-DTree also performs significantly better than BTS-G and c-BTS-G. OVO yields the highest accuracy among the techniques compared. However, it consumes a very high running time for classification, especially when applied to problems with a large number of classes. For example, for the datasets Plant Margin, Plant Shape, and Plant Texture, OVO needs the decision times of 4,950, while IBGE-DTree requires the decision times of only 7.4 to 8.3.

IB-DTree is also a time-efficient technique that yields the lowest average decision times but gives lower classification accuracy than IBGE-Tree. The classification accuracy of IB-DTree is significantly better than that of BTS-G and c-BTS-G, but it significantly underperforms

[1]For the dataset Abalone, the information gathering process is complex and time consuming, thus it contains a lot of noise and affects classification accuracy.

Table 2. Average classification accuracy results and their standard deviation. The bold number indicates the highest accuracy in each dataset. The numbers in the parentheses show the accuracy ranking.

Datasets	OVA	OVO	DDAG	ADAG
Page Block	**96.857 ± 0.478 (1)**	96.735 ± 0.760 (3)	96.729 ± 0.764 (4)	96.740 ± 0.757 (2)
Segment	97.359 ± 1.180 (5)	97.431 ± 0.860 (3)	**97.442 ± 0.848 (1)**	97.436 ± 0.854 (2)
Shuttle	99.914 ± 0.053 (5)	**99.920 ± 0.054 (1)**	**99.920 ± 0.054 (1)**	**99.920 ± 0.054 (1)**
Arrhyth	72.603 ± 7.041 (2)	**73.146 ± 6.222 (1)**	67.375 ± 7.225 (8)	67.484 ± 7.318 (7)
Cardiotocography	83.208 ± 1.661 (5)	**84.431 ± 1.539 (1)**	84.241 ± 1.609 (3)	84.351 ± 1.607 (2)
Mfeat-Factor	**98.200 ± 1.033 (1)**	98.033 ± 0.908 (3)	98.011 ± 0.941 (5)	98.019 ± 0.919 (4)
Mfeat-fourier	84.850 ± 1.528 (6)	**85.717 ± 1.603 (1)**	85.702 ± 1.589 (3)	85.708 ± 1.585 (2)
Mfeat-Karhunen	**98.000 ± 0.943 (1)**	97.913 ± 0.750 (3)	97.894 ± 0.726 (5)	97.900 ± 0.722 (4)
Optdigit	99.324 ± 0.373 (2)	**99.964 ± 0.113 (1)**	99.288 ± 0.346 (3)	99.288 ± 0.346 (3)
Pendigit	99.554 ± 0.225 (4)	**99.591 ± 0.203 (1)**	99.569 ± 0.213 (3)	99.574 ± 0.211 (2)
Primary Tumor	46.667 ± 7.011 (3)	**50.212 ± 7.376 (1)**	39.278 ± 6.419 (8)	39.486 ± 6.483 (7)
Libras Movement	**90.000 ± 2.986 (1)**	89.074 ± 3.800 (2)	89.034 ± 3.729 (3)	89.017 ± 3.687 (4)
Abalone[1]	16.959 ± 2.388 (8)	**28.321 ± 1.516 (1)**	24.093 ± 3.044 (7)	24.258 ± 3.154 (6)
Krkopt	**85.750 ± 0.769 (1)**	82.444 ± 0.628 (2)	81.952 ± 0.643 (4)	82.235 ± 0.634 (3)
Spectrometer	51.579 ± 6.256 (8)	**68.421 ± 5.007 (1)**	68.052 ± 4.706 (4)	68.392 ± 4.796 (2)
Isolet	**94.947 ± 0.479 (1)**	94.898 ± 0.648 (2)	94.872 ± 0.631 (4)	94.885 ± 0.643 (3)
Letter	97.467 ± 0.305 (4)	**97.813 ± 0.382 (1)**	97.746 ± 0.357 (3)	97.787 ± 0.360 (2)
Plant Margin	82.875 ± 2.655 (4)	**84.401 ± 2.426 (1)**	84.238 ± 2.516 (3)	84.341 ± 2.607 (2)
Plant Shape	70.938 ± 2.783 (3)	**71.182 ± 3.295 (1)**	70.922 ± 3.393 (4)	71.090 ± 3.313 (2)
Plant Texture	**87.179 ± 2.808 (1)**	86.387 ± 2.374 (2)	86.173 ± 2.519 (4)	86.259 ± 2.510 (3)
Avg. Rank	3.35	1.70	4.00	3.15

Datasets	BTS-G	c-BTS-G	IB-DTree	IBGE-DTree
Page Block	96.622 ± 0.812 (5)	96.565 ± 0.884 (8)	96.565 ± 0.779 (7)	96.620 ± 0.852 (6)
Segment	97.273 ± 1.076 (7)	97.100 ± 0.957 (8)	97.316 ± 0.838 (6)	97.403 ± 1.100 (4)
Shuttle	99.914 ± 0.050 (5)	99.914 ± 0.053 (5)	99.916 ± 0.050 (4)	99.910 ± 0.053 (8)
Arrhyth	71.918 ± 5.688 (4)	71.918 ± 5.189 (4)	71.005 ± 5.836 (6)	72.146 ± 4.043 (3)
Cardiotocography	83.048 ± 2.147 (7)	82.926 ± 2.106 (8)	83.819 ± 1.710 (4)	83.161 ± 2.490 (6)
Mfeat-Factor	97.810 ± 0.882 (8)	98.000 ± 0.888 (6)	98.000 ± 0.768 (6)	**98.200 ± 0.816 (1)**
Mfeat-fourier	84.235 ± 1.636 (8)	84.350 ± 1.700 (7)	85.200 ± 1.605 (4)	85.150 ± 1.717 (5)
Mfeat-Karhunen	97.450 ± 0.832 (6)	97.050 ± 0.725 (8)	97.450 ± 0.879 (6)	97.950 ± 1.141 (2)
Optdigit	99.002 ± 0.266 (8)	99.039 ± 0.308 (7)	99.164 ± 0.288 (5)	99.093 ± 0.395 (6)
Pendigit	99.442 ± 0.184 (7)	99.427 ± 0.201 (8)	99.445 ± 0.198 (6)	99.454 ± 0.318 (5)
Primary Tumor	43.016 ± 3.824 (5)	40.635 ± 4.813 (6)	47.937 ± 4.567 (2)	44.762 ± 5.478 (4)
Libras Movement	87.861 ± 4.151 (8)	88.611 ± 3.715 (5)	88.056 ± 3.479 (6)	88.056 ± 3.057 (6)
Abalone[1]	26.635 ± 1.236 (3)	26.013 ± 1.218 (4)	25.281 ± 0.904 (5)	26.745 ± 0.809 (2)
Krkopt	77.137 ± 0.880 (8)	78.190 ± 0.783 (7)	79.006 ± 0.792 (6)	80.610 ± 1.039 (5)
Spectrometer	59.432 ± 5.563 (6)	52.421 ± 5.192 (7)	68.211 ± 3.397 (3)	67.789 ± 6.296 (5)
Isolet	92.850 ± 0.799 (7)	92.677 ± 0.702 (8)	93.639 ± 0.261 (6)	94.011 ± 0.640 (5)
Letter	96.174 ± 0.321 (7)	96.369 ± 0.423 (6)	96.135 ± 0.312 (8)	96.409 ± 0.344 (5)
Plant Margin	77.994 ± 1.946 (8)	78.188 ± 2.739 (7)	80.563 ± 3.638 (5)	79.313 ± 2.863 (6)
Plant Shape	63.219 ± 1.808 (7)	61.750 ± 3.594 (8)	67.000 ± 2.853 (5)	66.750 ± 2.408 (6)
Plant Texture	78.893 ± 2.547 (7)	78.174 ± 3.997 (8)	80.425 ± 3.602 (6)	80.863 ± 2.828 (5)
Avg. Rank	6.55	6.85	5.35	4.75

Table 3. Win-lose-draw between the techniques under comparison.

	OVO	DDAG	ADAG	BTS-G	c-BTS-G	IB-DTree	IBGE-DTree
OVA	7-13-0	11-9-0	10-10-0	17-2-1	17-2-1	14-6-0	15-4-1
OVO	–	11-8-1	17-2-1	20-0-0	20-0-0	20-0-0	18-2-0
DDAG	–	–	2-16-2	17-3-0	17-3-0	16-4-0	15-5-0
ADAG	–	–	–	17-3-0	17-3-0	17-3-0	15-5-0
BTS-G	–	–	–	–	12-7-1	4-15-1	2-18-0
c-BTS-G	–	–	–	–	–	4-15-1	2-18-0
IB-DTree	–	–	–	–	–	–	8-11-1

Table 4. Rankings and adjusted p-values of classification accuracies using the Friedman aligned ranks test and the Hommel procedure for the proposed methods (IB-DTree and IBGE-DTree) as control algorithms and the tree-structure methods (BTS-G and c-BTS-G) as traditional algorithms. The bold number means that the result is a significant difference.

		Ranking	Adjusted p-values
Traditional methods	BTS-G	36.250	**0.0013939**
	c-BTS-G	36.650	**0.0010818**
Control method	IB-DTree	18.600	–
Traditional methods	BTS-G	54.1	**1.5320E-5**
	c-BTS-G	55.3	**7.2122E-6**
	IB-DTree	30.275	0.2793156
Control method	IBGE-DTree	22.325	–

OVO, OVA, DDAG and ADAG. Although in the general case IBGE-DTree is more considerable than IB-DTree because it yields better classification accuracy, IB-DTree is an interesting option when the training time is limited.

5. Conclusions

In this paper, we proposed IB-DTree and IBGE-DTree, techniques that combine entropy and generalization error estimation for classifier selection in tree construction. Using entropy, the class with high a probability of occurrence will be placed near the root node, resulting in reduction of decision times for that class. The lower the number of decision times, the smaller the cumulative error of the prediction because every classifier along the path may produce a wrong prediction. Generalization error estimation is a method for evaluating the effectiveness of the binary classifier. It enables our algorithm to select accurate classifiers for decision tree construction. Class-grouping-by-majority is also a key mechanism to the success of our methods, which is used to construct the tree without duplicated class scattering in the tree. Both IB-DTree and IBGE-DTree classify the answer in the time complexity of $O(\log_2 N)$ in the best case, and no more than $O(N)$ in the worst one.

We performed experiments comparing our methods with some traditional techniques on twenty datasets from the UCI repository. We can conclude that IBGE-DTree is the most efficient technique that gives the answer very fast, provides accuracy comparable to OVA, DDAG, and ADAG, and yields better accuracy than the other tree-structured techniques. IB-DTree also works fast and provides accuracy comparable to that IBGE-DTree, and could be considered when training time is a crucial factor.

Acknowledgment

This research was supported by the Royal Golden Jubilee PhD Program, the Thailand Research Fund and the Rachadapisek Sompote Fund for Postdoctoral Fellowship, Chulalongkorn University.

References

Bala, M. and Agrawal, R.K. (2011). Optimal decision tree based multi-class support vector machine, *Informatica* **35**(2): 197–209.

Bartlett, P.L. and Shawe-Taylor, J. (1999). Generalization performance of support vector machines and other pattern classifiers, *in* B. Schölkopf *et al.* (Eds.), *Advances in Kernel Methods*, MIT Press, Cambridge, MA, pp. 43–54.

Blake, C.L. and Merz, C.J. (1998). *UCI Repository of Machine Learning Databases*, University of California, Irvine, CA, `http://archive.ics.uci.edu/ml/`.

Bredensteiner, E.J. and Bennett, K.P. (1999). Multicategory classification by support vector machines, *Computational Optimization* **12**(1–3): 53–79.

Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* **2**(2): 121–167.

Chen, J., Wang, C. and Wang, R. (2009). Adaptive binary tree for fast SVM multiclass classification, *Neurocomputing* **72**(13–15): 3370–3375.

Table 5. Rankings and adjusted p-values of error rates using the Friedman aligned ranks test and the Hommel procedure for the proposed methods (IB-DTree and IBGE-DTree) as control algorithms and the non-tree-structure methods (OVO, OVA, DDAG and ADAG) as traditional algorithms. The bold number means that the result is a significant difference.

		Ranking	Adjusted p-values
Traditional methods	OVO	68.300	**6.4727E-5**
	OVA	50.150	**0.0437461**
	DDAG	50.025	**0.0451890**
	ADAG	52.375	**0.0238807**
Control method	IB-DTree	31.650	–
Traditional methods	OVO	66.150	**6.6516E-4**
	OVA	50.975	0.0802108
	DDAG	49.275	0.1177794
	ADAG	51.175	0.0765168
Control method	IBGE-DTree	34.925	–

Table 6. Average number of decision times. The bold number indicates the lowest decision times in each dataset.

Datasets	OVA	OVO	DDAG	ADAG	BTS-G	c-BTS-G	IB-DTree	IBGE-DTree
Page Block	5	10	4	4	**3.628**	3.801	3.790	3.831
Segment	7	21	6	6	3.630	3.882	**2.858**	3.009
Shuttle	7	21	6	6	**4.703**	5.370	5.000	5.019
Arrhyth	9	36	8	8	6.434	5.473	**5.258**	5.418
Cardiotocography	10	45	9	9	4.993	3.698	**3.490**	3.807
Mfeat-factor	10	45	9	9	4.224	3.643	**3.473**	3.754
Mfeat-fourier	10	45	9	9	4.512	3.796	**3.522**	3.786
Mfeat-karhunen	10	45	9	9	4.322	4.561	**3.435**	3.859
Optdigit	10	45	9	9	4.503	4.470	**3.399**	4.566
Pendigit	10	45	9	9	4.031	3.494	**3.487**	3.491
Primary Tumor	13	78	12	12	6.672	6.476	**5.391**	7.610
Libras Movement	15	105	14	14	5.493	5.114	**4.325**	4.411
Abalone	16	120	15	15	9.242	8.540	8.768	**7.626**
Krkopt	18	153	17	17	6.743	4.847	**3.957**	5.083
Spectrometer	21	210	20	20	6.728	6.080	**4.411**	4.613
Isolet	26	325	25	25	6.865	6.015	**5.064**	5.323
Letter	26	325	25	25	6.771	7.104	**4.922**	5.910
Plant Margin	100	4950	99	99	11.338	8.600	**6.973**	7.576
Plant Shape	100	4950	99	99	11.935	9.653	**6.965**	7.446
Plant Texture	100	4950	99	99	12.230	9.618	**7.022**	8.329

Cheong, S., Hoon Oh, S. and Lee, S.-Y. (2004). Support vector machines with binary tree architecture for multi-class classification, *Neural Information Processing Letters* **2**(3): 47–51.

Chmielnicki, W. and Stąpor, K. (2016). Using the one-versus-rest strategy with samples balancing to improve pairwise coupling classification, *International Journal of Applied Mathematics and Computer Science* **26**(1): 191–201, DOI: 10.1515/amcs-2016-0013.

Crammer, K. and Singer, Y. (2002). On the learnability and design of output codes for multiclass problems, *Machine Learning* **47**(2–3): 201–233.

Dong, C., Zhou, B. and Hu, J. (2015). A hierarchical SVM based multiclass classification by using similarity clustering, *International Joint Conference on Neural Networks, Killarney, Ireland*, pp.1–6.

Fei, B. and Liu, J. (2006). Binary tree of SVM: A new fast multiclass training and classification algorithm, *IEEE Transactions on Neural Networks* **17**(3): 696–704.

Friedman, J. (1996). Another approach to polychotomous classification, *Technical report*, Stanford University, Stanford, CA.

García, S., Fernández, A., Luengo, J. and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, *Information Sciences* **180**(10): 2044–2064.

Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling, *Annals of Statistics* **26**(2): 451–471.

Hsu, C. and Lin, C. (2002). A comparison of methods for multiclass support vector machines, *IEEE Transactions on Neural Networks* **13**(2): 415–425.

Joachims, T. (1999). Making large-scale SVM learning practical, *in* B. Schölkopf *et al.* (Eds.), *Advances in Kernel Methods—Support Vector Learning*, MIT Press, Cambridge, MA.

Kijsirikul, B., Ussivakulz, N. and Road, P. (2002). Multiclass support vector machines using adaptive directed acyclic graph, *International Joint Conference on Neural Networks, Honolulu, HI, USA*, pp. 980–985.

Knerr, S., Personnaz, L. and Dreyfus, G. (1990). Single-layer learning revisited: A stepwise procedure for building and training a neural network, *Neurocomputing* **68**(68): 41–50.

Kumar, M.A. and Gopal, M. (2010). Fast multiclass SVM classification using decision tree based one-against-all method, *Neural Processing Letters* **32**(3): 311–323.

Kumar, M.A. and Gopal, M. (2011). Reduced one-against-all method for multiclass svm classification, *Expert Systems with Applications* **38**(11): 14238–14248.

Lei, H. and Govindaraju, V. (2005). Half-against-half multi-class support vector machines, *in* N.C. Oza *et al.* (Eds.), *Multiple Classifier Systems, MCS 2005*, Lecture Notes in Computer Science, Vol. 3541, Springer, Berlin/Heidelberg, pp. 156–164.

Liu, B., Cao, L., Yu, P.S. and Zhang, C. (2008). Multi-space-mapped SVMs for multi-class classification, *Proceedings of 8th IEEE International Conference on Data Mining, Washington, DC, USA*, Vol. 8, pp. 911–916.

Madzarov, G., Gjorgjevikj, D. and Chorbev, I. (2009). A multi-class SVM classifier utilizing binary decision tree support vector machines for pattern recognition, *Electrical Engineering* **33**(1): 233–241.

Platt, J., Cristianini, N. and Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification, *in* S.A. Solla *et al.* (Eds.), *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, pp. 547–553.

Songsiri, P., Kijsirikul, B. and Phetkaew, T. (2008). Information-based dicrotomizer: A method for multiclass support vector machines, *IEEE International Joint Conference on Neural Networks, Hong Kong, China*, pp. 3284–3291.

Songsiri, P., Phetkaew, T. and Kijsirikul, B. (2015). Enhancement of multi-class support vector machine construction from binary learners using generalization performance, *Neurocomputing* **151**(P1): 434–448.

Takahashi, F. and Abe, S. (2002). Decision-tree-based multiclass support vector machines, *Proceedings of the 9th International Conference on Neural Information Processing, ICONIP'02, Singapore, Singapore*, Vol. 3, pp. 1418–1488.

Vapnik, V.N. (1998). *Statistical Learning Theory*, John Wiley & Sons, New York, NY.

Vapnik, V.N. (1999). An overview of statistical learning theory, *IEEE Transactions on Neural Networks* **10**(5): 988–99.

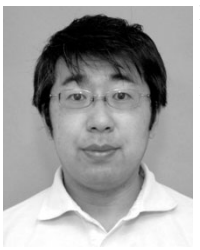
Vapnik V.N., C.A. (1974). *Teoriya Raspoznavaniya Obrazov: Statisticheskie Problemy Obucheniya (Theory of Pattern Recognition: Statistical Problems of Learning)*, Nauka, Moscow.

Yang, X., Yu, Q., He, L. and Guo, T. (2013). The one-against-all partition based binary tree support vector machine algorithms for multi-class classification, *Neurocomputing* **113**(3): 1–7.

Pittipol Kantavat received the BEng and MEng degrees in computer engineering from Chulalongkorn University, Thailand, in 2004 and 2008, respectively. He is currently a PhD candidate in computer engineering at Chulalongkorn University. His study has been supported by the Royal Golden Jubilee PhD Program. His research interests include artificial intelligence, pattern recognition and machine learning.

Boonserm Kijsirikul received the BEng degree in electronic and electrical engineering, and the MSc and PhD degrees in computer science from the Tokyo Institute of Technology, Japan, in 1986, 1990, and 1993, respectively. He is currently a professor at the Department of Computer Engineering, Chulalongkorn University, Thailand. His research interests include machine learning, artificial intelligence, natural language processing, and speech recognition.

Patoomsiri Songsiri received the BSc degree in computer science (first class honor) from the Prince of Songkla University, Thailand, in 2001. She received the MSc degree in computer science and the PhD degree in computer engineering from Chulalongkorn University, Thailand, in 2006 and 2015, respectively. She has conducted research supported by the Rachadapisek Sompote Fund for Postdoctoral Fellowship at Chulalongkorn since 2015. Her research interests include pattern recognition and machine learning.



Ken-ichi Fukui has been an associate professor in the Institute of Scientific and Industrial Research (ISIR), Osaka University, since 2015. He received a Master's degree from Nagoya University in 2003, and a PhD in information science from Osaka University in 2010. He was a specially appointed assistant professor from 2005 to 2010 and an assistant professor from 2010 to 2015 at the ISIR, Osaka University. His research interests include machine learning and data mining algorithms as well as their environmental contribution.

Masayuki Numao is a professor in the Department of Architecture for Intelligence, Osaka University. He received a BEng degree in electrical and electronics engineering in 1982 and a PhD degree in computer science in 1987 from the Tokyo Institute of Technology. He worked in the Department of Computer Science, Tokyo Institute of Technology, from 1987 to 2003, and was a visiting scholar at CSLI, Stanford University, from 1989 to 1990. His research interests include artificial intelligence, machine learning, affective computing and empathic computing.

Received: 3 October 2017
Revised: 21 April 2018
Accepted: 18 May 2018