Engineering Journal issn: 0125-8281

Article

Efficient Decision Trees for Multi-class Support Vector Machines Using Large Centroid Distance Grouping

Pittipol Kantavat[a], Patoomsiri Songsiri[b], and Boonserm Kijsirikul[c,*]

Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
E-mail: [a]pittipol.k@chula.ac.th, [b]patoomsiri.s@chula.ac.th,[c]boonserm.k@chula.ac.th (Corresponding author)

Abstract. We propose a new technique for support vector machines (SVMs) in tree structures for multiclass classification. For each tree node, we select an appropriate binary classifier using data class centroids and their in-between distances, categorize the training examples into positive and negative groups of classes and train a new classifier. The proposed technique is fast-trained and can classify an output class data with a complexity between $O(\log_2 N)$ and $O(N)$ where N is the number of classes. The 10-fold cross-validation experimental results show that the performance of our methods is comparable to that of traditional techniques and required less decision times. Our proposed technique is suitable for problems with a large number of classes due to its advantages of requiring less training time and computational complexity.

Keywords: Support Vector Machine (SVM), multiclass classification, data centroid, large centroid distance grouping, decision tree.

ENGINEERING JOURNAL Volume 26 Issue 5
Received 13 September 2021
Accepted 10 May 2022
Published 31 May 2022
Online at https://engj.org/
DOI:10.4186/ej.2022.26.5.13

DOI:10.4186/ej.2022.26.5.13

1. Introduction

The support vector machine (SVM) [1, 2] is a statistical learning algorithm that generates a hyperplane to separate the set of positive and negative data and maximize the margin between them. We can use the SVM to solve problems with more than two classes by applying multiclass learning techniques, for which two main approaches are used: solving a single optimization problem [1, 3, 4] and combining a group of several binary SVMs. However, Hsu and Lin [5] indicated that the latter approach is more practical for the real problems.

There are several widely used multiclass techniques. The One-Versus-One technique (OVO) [6] trains $N \times (N-1)/2$ binary classifiers for an N-class problem; each pairwise classifier is constructed from two out of N corresponding classes, and it learns to distinguish the two classes. In the classification phase, the voting process called Max-Wins [7] is conducted by selecting a class with the highest voting score. One-Versus-All (OVA) trains N binary classifiers for an N-class problem, and each classifier is trained with samples of one class as positive samples and all samples from the other classes as negative samples. The output class is determined by choosing the class with the highest classification score. Several modifications/enhancements have been proposed based on OVA and OVO. Hastie and Tibshirani [8] used the joint probability for estimating the pairwise classes and improving the accuracy of the original OVO. Kumar and Gopal [9] proposed a method of reducing the training time of OVA. In this research, our proposed method is also based on the OVO technique.

Among the traditional multiclass techniques, OVO generally provides higher accuracy. However, it consumes more processing time because the required computational time is O(N^2) where N is the number of classes. Many researchers have proposed the alternative techniques based on OVO to reduce the computational time. Decision Directed Acyclic Graph (DDAG) [10], Adaptive Directed Acyclic Graph (ADAG) [11] and Optimized DDAG [12] use only $N-1$ classifiers to perform a classification. Although they require less computational time, the classification accuracy is sacrificed as a trade-off.

The concept of the decision tree structure can be applied to OVO-based multiclass classification. The benefit of the tree structure is that we can eliminate more than one nonanswer class at one processing node of the tree, which will lead to lower than O(N) classification time. The Binary Tree of SVM (BTS) [13] constructs the classification tree using the OVO node selected randomly or by the smallest distance to the centroid of all training data. The information-based dichotomization tree [14] constructs the OVO-based classification tree using entropy. The adaptive binary tree [15] calculates and selects the classifiers with the lowest average number of support vectors in the tree construction. The optimal decision tree-based multiclass SVM [16] applies statistical measurement indicators in the classifier selections at each tree node. The information-based decision tree (IB-DTree) and the information-based and generalization-error estimation decision tree (IBGE-DTree) [17] are constructed by selecting binary classifiers using entropy and generalization error estimation. IB-DTree and IBGE-DTree also proposed a new classifier training method for use in the classification phase called "class-grouping-by-majority", which can provide a decision tree with a small tree-depth.

Many techniques that construct a tree-structure SVM using OVA classifiers or clustering techniques have been proposed. The decision-tree-based SVM [18] groups classes by calculating the Euclidian distance and Mahalanobis distance as a separability measure. Support vector machines with binary tree architectures [19] apply kernel-based self-organizing maps for the tree construction. The SVM binary decision tree [20] constructs a tree by calculating class centroids in the kernel space. The half-against-half multiclass SVM [21] constructs a hierarchy structure for the classification by dividing data classes into two groups equally. The multi-state-mapping multiclass SVM [22] applies kernel functions for constructing a tree SVM using the k-means algorithm. Decision tree based OVA [23] applies probabilistic output of the SVM to determine paths for a constructed tree. The single-space-mapped binary tree SVM [24] and the multi-space-mapped binary tree SVM [24] calculate the Euclidean distance to determine the set of hyper-parameters in the OVA-based approach. An efficient SVM [25] constructs a skewed binary tree to separate one class from other classes by evaluating the SVM function. OAA-SVM-MS [26] is a fast-training OAA-SVM using a generalization error calculated from geometrically ergodic Markov samples.

In this paper, we propose a multiclass classification technique using the tree structure inspired by IB-DTree and IBGE-DTree [17]. The key benefit of the new technique is that it consumes less training and classification complexity than traditional techniques. We evaluate our technique using twenty datasets from the UCI machine learning repository [27] and a dataset with a large number of classes, namely, Wikipedia Medium [28], and then compare our results with those of traditional techniques. The results show that our technique is very useful for the problems with a large number of classes or the problems requiring fast-classification speed.

This paper is organized as follows. Section 2 discusses the OVO-based multiclass SVM techniques using

tree structures. Section 3 presents our proposed technique. Section 4 provides the experimental details and results. Section 5 summarizes our research.

2. The OVO-Based Decision Tree SVM

2.1. Binary Tree of SVM

The Binary Tree of SVM (BTS) [13] was proposed by Fei and Liu, and two proposed versions are BTS and c-BTS. At the root node of BTS, an OVO classifier will be randomly selected and used to separate the considered classes in the list into either positive or negative groups. However, examples of a particular class may not be separated to a single side of the chosen hyperplane and will be scattered to both positive and negative groups. In this case, the examples of that class will be duplicated to both left and right child nodes of the root node. At the child nodes, the same process will be continued until each leaf node in the tree contains only one class. For c-BTS, instead of randomly selecting the classifiers, it calculates the centroid of all data and then selects as the separating classifier, the (i vs j) classifier to which the centroids of class i and that of class j are nearest among all data centroids.

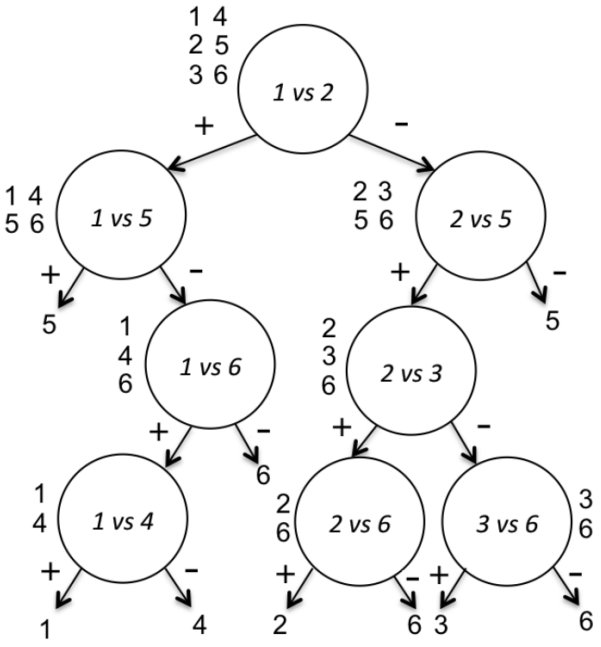


Fig. 1. The Illustration of the binary tree of SVM (BTS). Some data classes may be scattered at some node and appear in more than one leaf node. [13]

A possible example of BTS and c-BTS is illustrated in Fig. 1. At the root node, classifier 1 vs 2 is selected (randomly for BTS and via centroid calculation for c-BTS). Classes 1 and 4 and classes 2 and 3 are grouped together as positive and negative groups, respectively. However, classes 5 and 6 cannot be separated into a single side and are allowed to be duplicated to both positive and negative child nodes. The same process to the root node recursively continues until finished.

According to [13], the final tree structure of BTS and c-BTS depends on threshold configurations. A higher threshold will allow the more duplicating class leaf nodes, resulting in higher classification accuracy but consuming more computational time.

2.2. Information-Based Dichotomization

The information-based dichotomization (IBD) [14] was proposed by Songsiri et al. and is also a multiclass classification tree. Each node of the tree will be selected from OVO classifiers using the entropy values, which are calculated based on the training data. Using entropy information will enable a class with a high probability of occurrence to be classified first in each level. By using this strategy, the expected classification times to derive an answer to the tree will be O($\log_2 N$) in theory.

However, if a selected OVO classifier is not able to perfectly separate all training data of one class into a single side of the hyperplane, then the expected scenario may not be obtained. To address this situation, IBD applies the tree pruning algorithm, which allows for the elimination of some minor examples whose percentage of the minority is less than the optimal pruning threshold P. However, finding the parameter P that can preserve all useful information is not an easy task. In addition to the tree pruning algorithm, IBD also presents another parameter R which is called the optimal range of generalization performance of classifiers to select OVO classifiers with low generalization performance. Both parameters P and R can be acquired using the k-fold cross validation process.

2.3. Information-Based Decision Tree

The information-based decision tree (IB-DTree) [17] selects OVO pairwise classifiers with the lowest entropy in the same way as IBD [14]. However, instead of using them for both the training phase and classification phase, IB-DTree uses them only in the training phase as initial classifiers h for the method called "class-grouping-by-majority". This method uses classifier h to separate training examples into groups of positive-class and negative-class. Then, the final classifier h' will be trained using the positive-class group and negative-class group. As a result, a classification tree constructed by IB-DTree contains no duplicated class leaf-nodes and has a small depth.

The IB-DTree can output classification results within a relatively limited decision time. However, it

consumes considerable computation time in the training phase. In the initializing step, all OAO pairwise classifiers are trained for use in the tree. In the construction step, IB-DTree needs to calculate entropy using all training data in K candidate classes for each tree node. Because all binary classifiers (i vs j) in K are considered, the number of calculations per node is $K \times (K-1)/2$ times. As a result, IB-DTree might be an inappropriate option for problems with large a number of classes.

2.4. Information-Based and Generalization-Error Estimation Decision Tree

The information-based and generalization error decision tree (IBGE-DTree) [17] constructs a classification tree using entropy and class-grouping-by-majority, such as IB-DTree. However, instead of selecting only one initial classifier with the lowest entropy, IBGE-DTree sorts all considering classifiers by entropy and selects a number of classifiers with low entropy as initial classifiers h_s. Then, IBGE-DTree applies the class-grouping-by-majority method to all initial classifiers h_s and obtains the final classifiers h'_s. Finally, the classifier with the lowest estimated generalization error [29, 30] will be selected for use in the classification phase.

Regarding classification performance, IBGE-DTree provides better classification outputs than IB-DTree with the same decision time. However, it consumes more computational time than IB-DTree for the training process. IBGE-DTree prepares all OAO pairwise classifiers in the initializing step and calculates entropy for K candidate classes similar to IB-DTree. However, as IBGE-DTree processes the class-grouping-by-majority method many times in one node, it consumes more computational time than IB-DTree in the training phase.

3. Proposed Methods

We propose an OVO-based multiclass classification technique using a tree-structure named the largest-centroid-distance-grouping decision tree (LCDG-DTree). At the beginning of the tree construction, we calculate each data class centroid c_i using Eq. (1).

$$c_i = \frac{1}{\mid X_i \mid} \sum_{x \in X_i} x \tag{1}$$

where X_i is the set of training examples in class i and $\mid X_i \mid$ is the number of examples in class i. Then, all Euclidean distances between each pair of class centroids $d(c_i, c_j)$ are calculated using Eq. (2).

$$d(c_i, c_j) = \parallel c_i - c_j \parallel \tag{2}$$

At each tree node, we select the classifier with the largest centroid distance between the pairwise classes to be the *initial classifier*. For the next step, we use the initial classifier to separate all considered classes into either positive or negative groups. In this step, some considered classes may scatter on both sides of the initial classifier and the tree depth is unnecessarily increased. To avoid this situation, we use the mechanism called the class-grouping-by-majority [17] in Algorithm 1. The class-grouping-by-majority method groups all training data of the considered classes together, either positive or negative group. Then, the final classifier h' is trained using groups of positive and negative classes. Therefore, training a decision tree using this method will prevent the generation of a tree with duplicated class leaf nodes and the tree depth will not be increased unnecessarily. Thus, the decision times required to determine the answer class is decreased and the cumulative errors of the classification are also decreased.

Figure 2 illustrates the construction of LCDG-DTree by applying the same technique as in Fig. 1. At the root node, classifier 1 vs 2 is selected as the initial classifier h and it then is used to label all data of class 1 to class 6. All data of classes 2 and 3 are labeled as positive, while all data of classes 1 and 4 are labeled as negative. In this case, suppose that the majority of class 5 is on the positive side, while the majority of class 6 is on the negative side. Thus, (2, 3, 5) vs (1, 4, 6) is trained as the final classifier h' for the root node. For the next step, classes (2, 3, 5) will be processed as the left child node, and classes (1, 4, 6) will be processed as the right child node. The recursive process continues until finished, and there is no leaf node with a duplicated class in the tree.

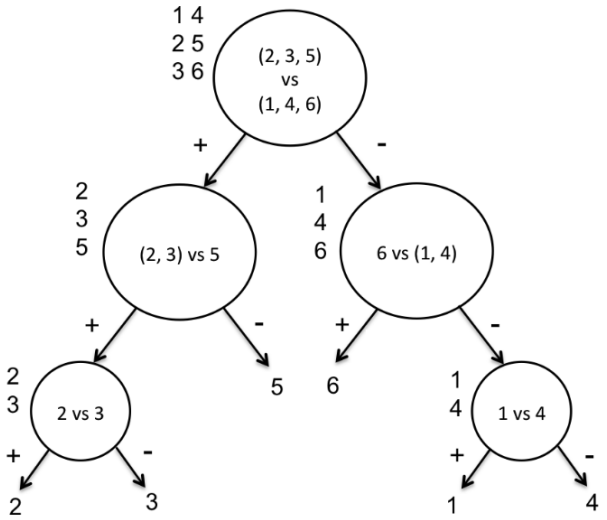


Fig. 2. The Illustration of The Largest-Centroid-Distance-Grouping Decision Tree (LCDG-DTree). Note that there is no leaf node with duplicated class.

The pseudocode of LCDG-DTree is described in Al-

Algorithm 1 Class-grouping-by-majority [17]

1: **procedure** Class-grouping-by-majority (selected classifier h, candidate classes K)
2: Initialize set of positive classes $P = \varnothing$ and set of negative classes $N = \varnothing$
3: **for** each class $i \in K$:-
4: Label all data of class i to $(+)$ and $(-)$ separated by initial classifier h
5: $p \leftarrow \text{count}(+)$, $n \leftarrow \text{count}(-)$
6: if $(p > n)$ then $P \leftarrow P \cup \{i\}$
7: else $N \leftarrow N \cup \{i\}$
8: **end for**
9: Train final classifier $h' = (P \text{ vs } N)$
10: **return** h', P, N
11: **end procedure**

gorithm 2. At the beginning of the LCDG-DTree algorithm, the centroid of each class and the distances between them are calculated in lines 2-3. Then, tree T with the *Root* node and the set of candidate classes C are initialized in lines 4-5. The recursive process of tree construction starts in line 6 by calling the procedure in lines 9-18. First, the initial classifier h with the largest centroid distance will be selected in line 11. Second, the final classifier h' is trained using the group of positive training data P and negative training data N acquired by class-grouping-by-majority in line 12. Finally, the recursive process for the left/right child node for the positive/negative class groups is called in lines 14-17. The tree construction process is terminated when the stopping conditions hold for all the child nodes in lines 16-17.

The key benefit of LCDG-DTree relative to IB-DTree and IBGE-DTree is that it consumes less computation time in the training phase. There are two main mechanisms that are different from that in the original techniques. First, LCDG-DTree does not need to prepare all $N \times (N-1)/2$ OVO binary classifiers in the initializing step for an N-class problem. Rather, only one binary classifier with the largest centroid distance is created and assigned as the initial classifier h per node. Hence, the total number of OVO pairwise classifiers that will be created in a tree is N-1. Second, calculations are not required for each node of the construction step, because all centroid distances between each pair of classes have already been precalculated in the initializing step. As a result, LCDG-DTree is more appropriate for problems with a large number of classes than IB-DTree and IBGE-DTree.

To demonstrate LCDG-DTree, we show trees constructed by the proposed technique using training data from the Mfeat-Factor and cardiotocography datasets. The Mfeat-Factor is a 10-class dataset with 200 examples in each class. Cardiotocography is a 10-class dataset in which the numbers of examples are 384, 579, 53, 81, 72, 332, 252, 107, 69 and 197. The constructed trees are shown in Fig. 3 and Fig. 4. Note that the tree structures are constructed by the centroid distances between their classes, not the number of examples.

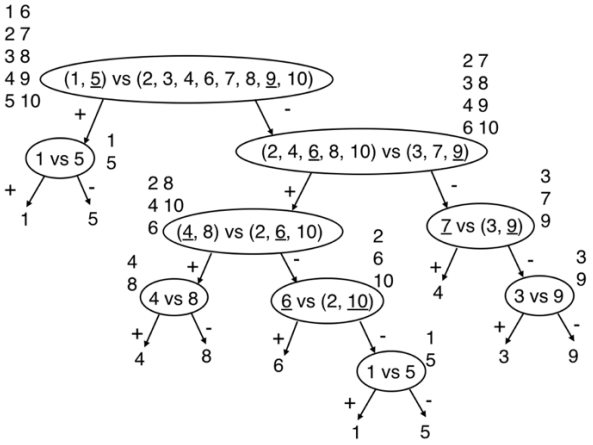


Fig. 3. LCDG-DTree constructed using the Mfeat-Factor dataset. The underlined numbers are classes that are selected for an initial classifier in each node.

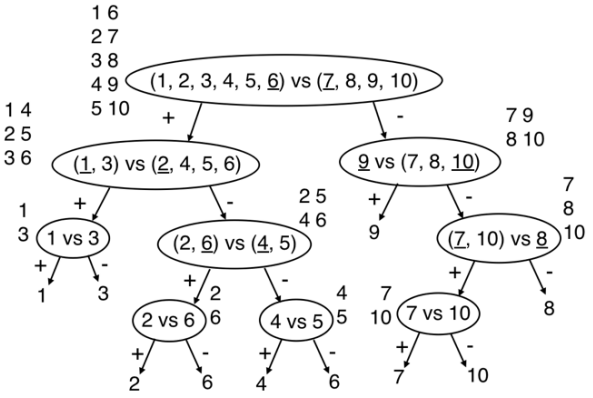


Fig. 4. LCDG-DTree constructed using the cardiotocography dataset. The underlined numbers are classes that are selected for an initial classifier in each node.

DOI:10.4186/ej.2022.26.5.13

Algorithm 2 Largest-Centroid-Distance-Grouping Decision Tree (LCDG-DTree)

procedure LCDG-DTree
    Calculate all of data centroids c_i ; $i \in C = \{1, 2, 3, ..., N\}$
    Calculate distances between class centroids $d(c_i, c_j)$; $i, j \in C$
    Initialize the tree T with root node *Root*
    Initialize the set of candidate output classes C
    Construct Tree (*Root*, S)
    return T
end procedure
procedure Construct Tree (node D, candidate classes K)
    Sort the class centroid distance between each class in K
    initial classifier $h \leftarrow$ create classifier (i vs j) with the largest centroid distance
    final classifier h', positive classes P, negative classes $N \leftarrow$ Class-grouping-by-majority(h, K)
    D.classifier $\leftarrow h'$
    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

4. Experiments and Results

We ran the experiments to compare our proposed method, LCDG-DTree, to the traditional strategies, i.e., OVO, OVA, DDAG, ADAG, IB-DTree, IBGE-DTree and LCDG-DTree. The smallest-centroid-distance-grouping decision tree (SCDG-DTree) is an alternate version of the centroid-based algorithm that constructs a tree in a similar way as LCDG-DTree but selects the pairwise classes with the smallest centroid distance instead of the largest distance.

We ran experiments based on 10-fold cross-validation on twenty datasets from the UCI repository [27], as shown in Table 1. We also conducted an experiment using a dataset with a large number of classes, i.e., Wikipedia Medium [28], to compare the performance of OVA, LCDG-DTree and SCDG-DTree. The original version of Wikipedia medium is a text dataset. We used only the single-label examples and calculated the term frequency-inverse document frequency to represent the text [31]. We evaluated the classification accuracy by running 10-fold cross validation. We merged training data and test data into one group for each dataset and normalized the data to the range [-1, 1]. For twenty datasets from the UCI repository, we used the software package SVM^{light} version 6.02 [32] using the RBF kernel. The kernel parameter (γ) and regularization parameter C were selected from {0.001, 0.01, 0.1, 1, 10} and {1, 10, 100, 1000}. For the Wikipedia Medium dataset, we use a linear kernel with regularization parameter $C = 1$. For DDAG and ADAG, as the initial order of classes affects the classification accuracy, we randomly selected 50,000 initial orders and calculated the average classification accuracy.

The results are shown in Tables 2-7. Table 2 shows the average classification accuracy results of 20 datasets from the UCI repository, while Table 3 shows the average classification accuracy results of Wikipedia Medium. Table 4 shows the win-lose-draw between the techniques under comparison. Table 5 shows the Friedman aligned ranks test [7] and the Hommel procedure [33], which were used to assess the accuracy of our method with the other tree-structure methods. Table 6 shows the Friedman aligned ranks test and the Hommel procedure, which were used to assess the error rate of our method with the other nontree-structure methods. Table 7 shows the average decision times that are used to determine the output class of testing examples.

In Table 2, a bold number indicates the highest accuracy in each dataset. The number in parentheses shows the ranking of each technique. The highest accuracy is obtained by OVO, followed by OVA, ADAG and DDAG. Among the tree structure techniques, IBGE-DTree provides the highest accuracy, while LCDG-DTree and IB-DTree are runners-up in performance. In Table 3, the results for the Wikipedia Medium dataset show that OVA yields higher classification accuracy than the LCDG-DTree and SCDG-DTree methods.

Table 4 shows the pairwise win-lose-draw between the techniques under comparison. The statistical tests indicate that OVO significantly outperforms all other techniques. Among the tree structure techniques, IBGE-DTree provides the highest accuracy results while SCDG-DTree underperforms relative to the other techniques.

Table 5 shows the rankings and adjusted p-values

Table 1. Experimental datasets: Twenty datasets from the UCI repository and Wikipedia Medium (*).

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	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
Wikipedia Medium*	1027	160748	111028

of error rate from Table 2 using the Friedman aligned ranks test and the Hommel procedure between LCDG-DTree as the control algorithm and the traditional tree-structure methods (IB-DTree and IBGE-DTree) as the traditional algorithms. The empirical results show that the ranking performance of IB-DTree and IBGE-DTree is better than that of LCDG-DTree. However, the adjusted p-values indicate that LCDG-DTree is not significantly different from IB-DTree and IBGE-DTree with a significance level less than 0.025.

Table 6 shows rankings and adjusted p-values of the classification error rates using the Friedman aligned ranks test and the Hommel procedure between LCDG-DTree as the control algorithms and the nontree-based methods (OVO, OVA, DDAG and ADAG) as the traditional algorithms. The empirical results show that performance of LCDG-DTree is inferior to that of OVO, OVA, DDAG and DDAG. However, the adjusted p-values indicate that LCDG-DTree is not significantly different from OVA, DDAG and ADAG with significance levels less than 0.025.

Table 7 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. The results show that IB-DTree and LCDG-DTree share the best performance in speed among the UCI datasets, while OVO is the slowest.

The experimental results show that LCDG-DTree is one of the most efficient techniques among the comparison tree-structure techniques and yields classification accuracy comparable to OVA, DDAG, ADAG, IB-DTree and IBGE-DTree. OVO provides the highest accuracy among the comparison techniques but requires a very high running time for classification. OVO is inappropriate for application to the problems with a large number of classes. For example, for Wikipedia Medium, OVO needs 526,851 decision times; hence, it is impractical and is not tested in the experiment. On the other hand, LCDG-DTree needs only 92.3 decision times. Note that IB-DTree and IBGE-DTree are also impractical for Wikipedia Medium; moreover, because they need 526,851 OVO classifiers in the training process, they cannot be tested in the experiment.

5. Conclusions

We proposed the LCDG-DTree technique for calculating the data class centroids and selecting the classifier with the largest distance in the tree construction. Our technique has the advantage of training faster than the traditional tree-structure techniques IB-DTree and IBGE-DTree and presents a comparable classification performance. The classification complexity of LCDG-DTree is close to $O(\log_2 N)$.

To examine our technique, we ran the experiments on twenty datasets from the UCI repository and a dataset with a large number of classes, Wikipedia Medium. In summary, LCDG-DTree is the most ef-

DOI:10.4186/ej.2022.26.5.13

Table 2. Average classification accuracy results and their standard deviations by twenty datasets from the UCI repository. Numbers in bold indicate the highest accuracy in each dataset, and 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	(6)	97.431 ± 0.860	(4)	97.442 ± 0.848	(2)	97.436 ± 0.854	(3)
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	(2)	98.033 ± 0.908	(4)	98.011 ± 0.941	(6)	98.019 ± 0.919	(5)
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	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	3.40		1.70		4.10		3.25	
Datasets	IB-DTree		IBGE-DTree		SCDG-DTree		LCDG-DTree	
Page Block	96.565 ± 0.779	(8)	96.620 ± 0.852	(6)	96.620 ± 0.742	(6)	96.656 ± 0.780	(5)
Segment	97.316 ± 0.838	(7)	97.403 ± 1.100	(5)	97.143 ± 0.821	(8)	**97.662 ± 0.796**	**(1)**
Shuttle	99.916 ± 0.050	(8)	99.910 ± 0.053	(6)	99.910 ± 0.053	(6)	99.884 ± 0.060	(8)
Arrhyth	71.005 ± 5.836	(6)	72.146 ± 4.043	(4)	72.374 ± 4.883	(3)	72.146 ± 5.858	(4)
Cardiotocography	83.819 ± 1.710	(4)	83.161 ± 2.490	(6)	82.973 ± 3.019	(8)	83.161 ± 1.974	(6)
Mfeat-Factor	98.000 ± 0.768	(7)	98.200 ± 0.816	(2)	98.000 ± 1.000	(7)	**98.250 ± 0.791**	**(1)**
Mfeat-fourier	85.200 ± 1.605	(4)	85.150 ± 1.717	(5)	84.550 ± 1.165	(7)	84.200 ± 1.874	(8)
Mfeat-Karhunen	97.450 ± 0.879	(6)	97.950 ± 1.141	(2)	97.450 ± 0.985	(6)	97.400 ± 0.775	(8)
Optdigit	99.164 ± 0.288	(5)	99.093 ± 0.395	(7)	99.021 ± 0.484	(8)	99.146 ± 0.418	(6)
Pendigit	99.445 ± 0.198	(7)	99.454 ± 0.318	(5)	99.381 ± 0.226	(8)	99.454 ± 0.172	(5)
Primary Tumor	47.937 ± 4.567	(2)	44.762 ± 5.478	(4)	43.810 ± 8.373	(6)	44.762 ± 8.531	(4)
Libras Movement	88.056 ± 3.479	(6)	88.056 ± 3.057	(6)	86.667 ± 4.864	(8)	88.889 ± 2.928	(5)
Abalone	25.281 ± 0.904	(5)	26.745 ± 0.809	(3)	26.403 ± 0.823	(4)	28.306 ± 1.661	(2)
Krkopt	79.006 ± 0.792	(8)	80.610 ± 1.039	(5)	80.596 ± 0.840	(6)	79.502 ± 1.045	(7)
Spectrometer	68.211 ± 3.397	(3)	67.789 ± 6.296	(5)	65.474 ± 4.790	(6)	64.211 ± 5.244	(7)
Isolet	93.639 ± 0.261	(6)	94.011 ± 0.640	(5)	93.151 ± 0.832	(8)	93.587 ± 0.878	(7)
Letter	96.135 ± 0.312	(8)	96.409 ± 0.344	(5)	96.140 ± 0.282	(7)	96.225 ± 0.515	(6)
Plant Margin	80.563 ± 3.638	(5)	79.313 ± 2.863	(6)	75.563 ± 3.947	(8)	78.188 ± 1.754	(7)
Plant Shape	67.000 ± 2.853	(5)	66.750 ± 2.408	(6)	64.063 ± 2.504	(8)	64.438 ± 2.771	(7)
Plant Texture	80.425 ± 3.602	(6)	80.863 ± 2.828	(5)	78.111 ± 3.588	(7)	77.423 ± 4.438	(8)
Avg. Rank	5.60		4.90		6.75		5.60	

Table 3. Average classification accuracy results and their standard deviation by the Wikipedia Medium. Numbers in bold indicates the highest accuracy in each dataset.

Datasets	OVA	SCDG-DTree	LCDG-DTree
Wikipedia Medium	65.289 ± 0.381	54.109 ± 0.356	57.500 ± 0.581

Table 4. Win-lose-draw comparisons between the row techniques (on the side) to the column techniques (on the top)

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

Table 5. Rankings and adjusted *p*-values of error rates using the Friedman aligned ranks test and the Hommel procedure between LCDG-DTree as the control algorithm and the tree-structure methods (IB-DTree and IBGE-DTree) as the traditional algorithm. Numbers in bold mean that the result is a significant difference.

		Ranking	Adjusted *p*-values
Traditional methods	IB-DTree	32.375	0.2001653
	IBGE-DTree	33.825	0.1226769
Control Method	LCDG-DTree	25.300	-

Table 6. Rankings and adjusted *p*-values of error rates using the Friedman aligned ranks test and the Hommel procedure between LCDG-DTree as the control algorithm and nontree-structure methods (OVO, OVA, DDAG and ADAG) as the traditional algorithm. Numbers in bold mean that the result is a significant difference.

		Ranking	Adjusted *p*-values
Traditional methods	OVO	67.550	**1.9735E-4**
	OVA	52.900	0.0335435
	DDAG	48.400	0.1020463
	ADAG	50.250	0.0662594
Control Method	LCDG-DTree	33.400	-

Table 7. Average number of decision times. Numbers in bold indicates the lowest decision times in each dataset.

Datasets	OVA	OVO	DDAG	ADAG	IB-DTree	IBGE-DTree	SCDG-DTree	LCDG-DTree
Page Block	5	10	4	4	3.790	3.831	3.832	**2.963**
Segment	7	21	6	6	**2.858**	3.009	3.564	3.748
Shuttle	7	21	6	6	4.998	5.019	4.667	**2.430**
Arrhyth	9	36	8	8	**5.258**	5.418	5.475	6.219
Cardiotocography	10	45	9	9	**3.487**	3.807	3.813	4.531
Mfeat-factor	10	45	9	9	**3.473**	3.754	3.818	3.492
Mfeat-fourier	10	45	9	9	**3.522**	3.786	3.545	3.704
Mfeat-karhunen	10	45	9	9	**3.435**	3.859	4.035	3.455
Optdigit	10	45	9	9	**3.399**	4.566	4.045	3.814
Pendigit	10	45	9	9	3.487	3.491	3.785	**3.413**
Primary Tumor	13	78	12	12	5.391	7.610	7.181	**4.844**
Libras Movement	15	105	14	14	4.325	4.411	5.197	**4.267**
Abalone	16	120	15	15	8.768	7.626	9.561	**5.368**
Krkopt	18	153	17	17	**3.957**	5.083	4.449	5.349
Spectrometer	21	210	20	20	**4.411**	4.613	5.000	5.162
Isolet	26	325	25	25	**5.064**	5.323	5.668	5.022
Letter	26	325	25	25	**4.922**	5.910	5.100	5.198
Plant Margin	100	4950	99	99	**6.973**	7.576	7.922	7.464
Plant Shape	100	4950	99	99	**6.965**	7.446	8.496	7.505
Plant Texture	100	4950	99	99	**7.022**	8.329	8.038	7.567
Wikipedia Medium	1027	-	-	-	-	-	**71.772**	92.345

ficient technique that provides a solution quickly and with comparable accuracy to OVA, DDAG, ADAG IB-DTree and IBGE-DTree.

Acknowledgments

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

References

[1] V. N. Vapnik, *Statistical Learning Theory*, 1998.

[2] ——, "An Overview of Statistical Learning Theory." *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–99, 1999.

[3] E. J. Bredensteiner and K. P. Bennett, "Multicategory Classification by Support Vector Machines," *Computational Optimization*, vol. 12, no. 1-3, pp. 53–79, 1999.

[4] K. Crammer and Y. Singer, "On The Learnability and Design of Output Codes for Multiclass Problems," *Machine Learning*, vol. 47, no. 2-3, pp. 201–233, 2002.

[5] C. Hsu and C. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, 2002.

[6] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer Learning Revisited: A Stepwise Procedure for Building and Training A Neural Network," *Neurocomputing*, no. 68, pp. 41–50, 1990.

[7] J. Friedman, "Another approach to polychotomous classification," *Technical Report*, 1996.

[8] T. Hastie and R. Tibshirani, "Classification by Pairwise Coupling," *Annals of Statistics*, vol. 26, no. 2, pp. 451–471, 1998.

[9] M. A. Kumar and M. Gopal, "Reduced one-against-all method for multiclass svm classification," *Expert Systems with Applications*, vol. 38, pp. 14 238–14 248, 2011.

[10] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large Margin DAGs for Multiclass Classification," *Advances in Neural Information Processing Systems*, pp. 547–553, 2000.

[11] B. Kijsirikul, N. Ussivakulz, and P. Road, "Multiclass Support Vector Machines Using Adaptive Directed Acyclic Graph," *International Joint Conference on Neural Networks*, vol. 2, no. 6, pp. 980–985, 2002.

[12] F. Takahashi and S. Abe, "Optimizing Directed Acyclic Graph Support Vector Machines," *Proceedings of Artificial Neural Networks in Pattern Recognition (ANNPR 2003)*, no. 1, pp. 166–70, 2003.

[13] B. Fei and J. Liu, "Binary Tree of SVM: A New Fast Multiclass Training and Classification Algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 696–704, 2006.

[14] P. Songsiri, B. Kijsirikul, and T. Phetkaew, "Information-based dichotomization: A method for multiclass support vector machines," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 3284–3291.

[15] J. Chen, C. Wang, and R. Wang, "Adaptive binary tree for fast svm multiclass classification," *Neurocomputing*, vol. 72, no. 13-15, pp. 3370–3375, 2009.

[16] M. Bala and R. K. Agrawal, "Optimal Decision Tree Based Multi-class Support Vector Machine," *Informatica*, vol. 35, pp. 197–209, 2011.

[17] P. Kantavat, B. Kijsirikul, P. Songsiri, K.-I. Fukui, and M. Numao, "Efficient decision trees for multi-class support vector machine using entropy and generalization error estimation," *International Journal of Applied Mathematics and Computer Science*, vol. 28, no. 4, pp. 705–717, 2018.

[18] F. Takahashi and S. Abe, "Decision-tree-based Multiclass Support Vector Machines," *In Neural Information Processing IEEE, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, vol. 3, pp. 1418–1488, 2002.

[19] G. Madzarov, D. Gjorgjevikj, and I. Chorbev, "A Multi-class SVM Classifier Utilizing Binary Decision Tree Support Vector Machines for Pattern Recognition," *Electrical Engineering*, vol. 33, no. 1, pp. 233–241, 2009.

[20] S. Cheong, S. Hoon Oh, and S.-Y. Lee, "Support vector machines with binary tree architecture for multi-class classification," *Neural Information Processing Letter*, vol. 2, no. 3, pp. 47–51, 2004.

[21] H. Lei and V. Govindaraju, "Half-against-half multi-class support vector machines," in *MCS'05 Proceedings of the 6th international conference on Multiple Classifier*, Seaside, CA, 2005.

[22] B. Liu, L. Cao, P. S. Yu, and C. Zhang, "Multi-space-mapped svms for multi-class classification," in *Proceedings of Eighth IEEE International Conference on Data Mining*, vol. 8, 2008, pp. 911–916.

[23] M. A. Kumar and M. Gopal, "Fast multiclass svm classification using decision tree based one-against-all method," *Neural Processing Letters*, vol. 32, no. 3, pp. 311–323, 2010.

[24] B. Liu, Y. Xiao, and L. Cao, "Svm-based multi-state-mapping approach for multi-class classification," *Knowledge-Based Systems*, vol. 129, pp. 79–96, 2017.

[25] P. S. Bogawar and K. K. Bhoyar, "An improved multiclass support vector machine classifier using reduced hyper-plane with skewed binary tree," *Applied Intelligence*, 2018.

[26] Y. Duan, B. Zou, J. Xu, F. Chen, J. Wei, and Y. Y. Tang, "Oaa-svm-ms: A fast and efficient multi-class classification algorithm," *Neurocomputing*, vol. 454, pp. 448–460, 2021.

[27] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning Databases," *University of California*, p. http://archive.ics.uci.edu/ml/, 1998.

[28] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari, "Lshtc: A benchmark for large-scale text classification," *CoRR abs/1503.08581*, 2015.

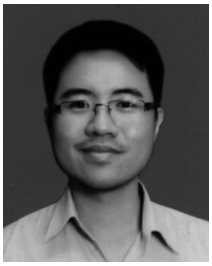
[29] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[30] P. L. Bartlett and J. Shawe-Taylor, "Generalization performance of support vector machines and other pattern classifiers," *Advances in Kernel Methods*, pp. 43–54, 1999.

[31] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Stanford University, 2011.

[32] T. Joachims, "SVM Light," 2008.

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



Pittipol Kantavat Pittipol Kantavat received the BEng, MEng and PhD degrees in computer engineering from Chulalongkorn University, Thailand, in 2004, 2008 and 2018, respectively. He is currently a lecturer at the Department of Computer Engineering, Chulalongkorn University, Thailand. His research interests include artificial intelligence, pattern recognition and machine learning.

Patoomsiri Songsiri received the B.Sc. degree in Computer Science (First class honor) from Prince of Songkla University, Thailand, in 2001. She received the M.Sc. degree in Computer Science and the Ph.D. degree in Computer Engineering from Chulalongkorn University, Thailand, in 2006, and in 2015, respectively. She had conducted research supported by the Rachadapisek Sompote Fund for Postdoctoral Fellowship at Chulalongkorn University, Thailand, from 2015 to 2017.

Boonserm Kijsirikul received the B.Eng. degree in Electronic and Electrical Engineering, the M.Sc. degree in Computer Science, and the Ph.D. in Computer Science from 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 current research interests include Machine Learning, Artificial Intelligence, Natural Language Processing, and Speech Recognition.