New Clustering Algorithms for the Support Vector Machine Based Hierarchical Classification

Hakan Cevikalp

Electrical and Electronics Engineering Department, Eskisehir Osmangazi University, Meselik 26480 Eskisehir, Turkey.
Email:hakan.cevikalp@gmail.com

Abstract

This study presents two new clustering algorithms for partition of data samples for the Support Vector Machine (SVM) based hierarchical classification. A divisive (top-down) approach is considered in which a set of classes is automatically separated into two smaller groups at each node of the hierarchy. The first algorithm splits the data samples based on a variation of the Normalized Cuts (NCuts) clustering algorithm wherein the weights of adjacency matrix are modified to utilize class membership in the process. The second algorithm also uses the NCuts clustering; however, it considers the involved classes rather than the individual data samples. It uses the minimum distances between the convex hulls of classes as a distance measure for determining the weights of the graph. Splits are determined for both algorithms based on the eigenvector corresponding to the second smallest eigenvalue of a Laplacian matrix, and it is observed that the proposed algorithms generate well-separated and well-balanced clusters. Unlike other clustering methods used for this purpose, the methods in the present study are found to be more suitable when SVMs are used as base classifiers. As demonstrated in the experiments, the proposed clustering algorithms are integrated into the hierarchical SVM classifiers, which results in significantly improved testing times with a negligible decrease in classification accuracies as compared to the traditional multi-class SVMs.

Key words: hierarchical classification, support vector machines, multi-class classification, clustering, normalized cuts

1. Introduction

Automatic classification of data samples is very important for several applications including visual object classification, text classification, speech recognition, etc. This leads to a requirement for efficient and accurate classifiers. For example, in visual object localization problems, hundreds of sub-windows need to be classified. This task will become tedious if the chosen classifier is not fast. It is known that there is a direct relationship between the real-time performance of a classifier and the number of classes. Human beings can classify between 10^4 and 10^5 object categories, and therefore, this seems to be a practical goal for machines as well (Griffin and Perona, 2008). It is thus crucial that classification algorithms must scale well with the number of classes.

The Support Vector Machine (SVM) classifier is a successful method that simultaneously minimizes the empirical classification error and maximizes the geometric margin (Cortes and Vapnik, 1995; Burges, 1998; Schölkopf and Smola, 2002). Essentially, it finds a separating hyperplane that yields the largest margin (separation gap) between two class samples. The SVM formulation was originally designed for binary classification; however, extending this formulation to more than two classes makes it very complex and is therefore generally avoided. Yet, many classification applications have more than two classes. The multi-class SVM problems are dealt with constructing several binary classifiers. There are various strategies to achieve this goal. Among these, the earliest and the most popular are the one-against-rest (OAR) strategy and the one-against-one (OAO) strategy (Hsu and Lin, 2002). For a C-class classification problem, the former strategy trains C binary classifiers, in which each classifier separates one class from the remaining $C-1$ classes. All classifiers are trained on the entire training set, and the class label of a test sample is determined based on the highest output value of the classifier in the ensemble. The latter strategy constructs all possible $C(C-1)/2$ binary classifiers out of C classes. The decision of the ensemble is typically made using the max wins algorithm: Each OAO classifier casts one vote for its preferred class, and the final decision is made for the class with the most votes. The OAO strategy builds more classifiers than the OAR strategy, and in general, it is considerably faster than OAR during training since it operates on a smaller number of training samples (Hsu and Lin, 2002). However, the OAO classifiers grow in size quadratically with the number of classes, which makes the OAO strategy very expensive for applications with large number of classes.

Recently, SVM based hierarchical classifiers have gained significant attention for large class problems owing to their capability to scale well with the number of classes (Platt et al., 2000; Vural and Dy, 2004; Casasent and Wang, 2005; Marszalek and Schmid, 2008; Chen et al., 2004; Griffin and Perona, 2008; Zhigang et al., 2005; Liu et al., 2005). Two popular methods fall into this category, namely, Decision Directed Acyclic Graphs (DDAGs) (Platt et al., 2000) and Binary Hierarchical Decision Trees (BHDTs) (Vural and Dy, 2004; Casasent and Wang, 2005; Chen et al., 2004; Griffin and Perona, 2008; Zhigang et al., 2005). The DDAG method first trains $C(C-1)/2$ binary classifiers, and subsequently uses a Directed Acyclic Graph (DAG) during the testing phase. This is equivalent to operating on a list where each node of the DAG eliminates one class from the list. Thus, the method requires only $C-1$ decision nodes to be evaluated for labeling a test sample rather than $C(C-1)/2$ classifier evaluations, which results in a significant speeding up of the testing phase. However, this algorithm makes some unnecessary comparisons which are considered as irrelevant for the classification of a particular test sample. As an example, consider a visual object classification problem: When a test sample belonging to the 'dogs' class arrives, it is rather unnecessary to make comparisons between unrelated classes such as buildings, airplanes, and cars. In this way, the real-time performance could be further improved. BHDT algorithms have been introduced in order to improve the efficiency of SVM classifiers by reducing the unnecessary comparisons while maintaining the high classification accuracy. To reduce unnecessary class comparisons, a BHDT algorithm uses a decision tree that divides the data hierarchically into two subsets until each subset consists of only one class. The SVM classifier is then used for separating those subsets at each node of the binary tree. The data partition is often achieved using a clustering algorithm, and the accuracy of the SVM classifier at each internal node depends on the generated clusters. Different hierarchy strategies (top-down (Vural and Dy, 2004; Casasent and Wang, 2005; Chen et al., 2004; Griffin and Perona, 2008) and bottom-up (Zhigang et al., 2005)) and different clustering algorithms, such as k-means (Vural and Dy, 2004; Zhigang et al., 2005), kernel k-means (Casasent and Wang, 2005), spherical shells (Vural and Dy, 2004) and balanced subset clustering (Vural and Dy, 2004), have been used in the literature. It is known that BHDT algorithms employ various distance measures for partition such as the Euclidean distance between class means (Vural and Dy, 2004; Zhigang et al., 2005), the Kullback-Leibler distance between class densities (Chen et al., 2004), or the number of misclassifications between classes (Griffin and Perona, 2008). In addition to these algorithms, some BHDT methods determine the partitions based on the clustering of data samples rather than the class sets (Marszalek and Schmid, 2008). A well-balanced tree requires approximately $\log_2 C$ classifier evaluations for traversing a path from the top to a bottom decision node. This results in a more efficient structure as compared to the DDAG algorithm in terms of testing time.

The present study is focused on SVM based BHDTs wherein two clustering algorithms are proposed for the partition of classes. Similar to other BHDT algorithms, the main objective is to improve the real-time efficiency (testing time) while maintaining the high classification accuracy. The first clustering algorithm operates on data samples whereas the second algorithm considers the class sets. It is found that both methods yield well-separated and well-balanced partitions, which is compatible with the goal of SVM classification. The remaining sections of this article are organized as follows: In section 2, the proposed clustering methods are introduced. In section 3, the data sets and the experimental procedure are described. Lastly, the conclusion is provided in Section 4.

2. Method

2.1. Design Issues

BHDT methods use clustering algorithms for the partition of data; thus, the classification accuracy and computational efficiency of the hierarchical classification system depend heavily on the generated clusters. More precisely, well-balanced separable clusters at each node of the tree would significantly improve the performance of the overall system. This requires that the employed clustering algorithm must be compatible with the base classifier, that is, the SVM classifier in our study.

To design optimal clustering algorithm, we should first examine the base classifier SVM since the clustering algorithm as well as the base classifier must aim achieving the same goal for a satisfactory performance. The SVM classifier finds a separating hyperplane that maximizes the margin, which is defined as the distance between the hyperplane and the closest samples from the classes. To do so, SVM first approximates each class with a convex hull (Bennett and Bredensteiner, 2000). A convex hull consists of all points that can be written as a convex combination of the points in the original set, and a convex combination of points is a linear combination of data points where all coefficients are nonnegative and sum up to 1. More formally, the convex hull of samples $\{\mathbf{x}_i\}_{i=1,\ldots,n}$ can be written as

$$H_{convex} = \left\{ \mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i | \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0 \right\}. \tag{1}$$

Convex hulls of two classes are illustrated in Fig. 1. Following this approximation, SVM finds the closest points in these convex hulls (Bennett and Bredensteiner, 2000). Then, these two points are connected with a line segment. The plane, orthogonal to the line segment that bisects the line, is selected as the separating hyperplane as shown in Fig. 1. From this geometrical point of view, in a separable case, the two closest points on the convex hulls determine the separating hyperplane and the SVM margin is merely equivalent to the minimum distance between the convex hulls that represent classes.

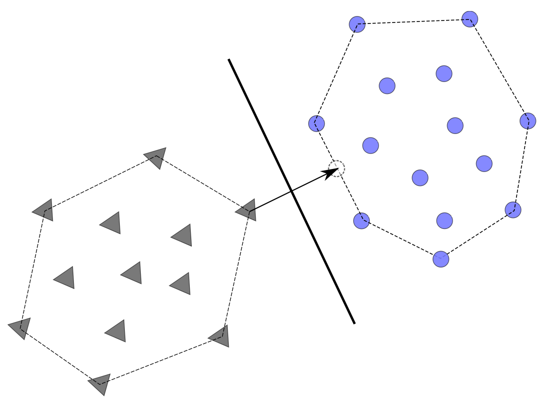

Figure 1: Two closest points on the convex hulls determine the separating hyperplane of the hard-margin SVM classifier.

The clustering algorithms based on k-means clustering are not good choices for SVM based BHDTs since they do not directly target at margin maximization in the sense described above. In a similar manner, the clustering algorithm maximizing the Kullback-Leibler distance between class densities is also not compatible with SVMs owing to the same reason. When the clustering algorithms using k-means or Kullback-Leibler distance are employed, it may be more difficult to separate the resulting clusters for SVM classifiers, which in turn would degrade both the classification accuracy and the real-time efficiency of the overall classification system. On the other hand, the Normalized Cuts (NCuts) clustering of (Shi and Malik, 2000) is more suitable for partition of data since its objective is similar to that of SVMs. The NCuts clustering algorithm maps the data samples into an infinite-dimensional feature space and cuts through the data by passing a hyperplane through the maximum gap in the mapped data (Rahimi and Recht, 2004). It then labels points that fall on the same side of the hyperplane as belonging to the same cluster. However, NCuts is an unsupervised approach and its split does not guarantee that samples belonging to the same classes are always grouped together in the same node unless class-specific samples are very close to each other and they are far from the other class samples. As a result, NCuts clustering may create overlapping classes that have samples in different clusters. If the created clusters are not compact, the real-time performance of the BHDTs degrades and they behave like K-D trees (Friedman et al., 1977). In the following section, the NCuts algorithm is modified and two variations are proposed that generate well-balanced compact clusters with the maximum margin.

2.2. Sample Based Large Margin Clustering

Let the training samples be $\mathbf{x}_{ci} \in \mathbb{R}^d$, where $c = 1, \ldots, C$ indexes the C classes and $i = 1, \ldots, n_c$ indexes the n_c samples of class c. Prior to introduction of the first proposed method, a summary of the NCuts clustering algorithm is provided since the proposed method is built on this approach.

2.2.1. Normalized Cuts Clustering

Given a dataset of m samples, the NCuts algorithm constructs a weighted graph with m vertices $\{v_1, \ldots, v_m\}$ (one for each sample) and a set of edges containing these vertices. Each edge between vertices v_i and v_j carries a non-negative weight $w_{ij} = w_{ji} \geq 0$ based on the similarity between the samples associated to the vertices. In the present study, a fully connected graph is considered in which all edges are connected. A common choice for weighting the edges, is the heat kernel (Gaussian kernel) $w_{ij} = \exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/t))$, where $d(\mathbf{x}_i, \mathbf{x}_j)$ represents the distance between samples $\mathbf{x}_i$ and $\mathbf{x}_j$ computed using the preferred distance function, and t is the width of the kernel.

The matrix $\mathbf{W} = (w_{ij})_{i,j=1,\ldots,m}$ is the weighted adjacency matrix of the graph. In the case of binary clustering, assigning a label $y_i \in \{-1, +1\}$ to each sample $\mathbf{x}_i$ cuts the graph into set A of the vertices with label $+1$ and set B of vertices with label -1. The cost function of the method is defined as

$$NCut(A, B) = \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right) \sum_{i \in A, j \in B} w_{ij} \tag{2}$$

where Vol is the sum of the weights in a set and $\sum_{i \in A, j \in B} w_{ij}$ is the total weight of the edges that must be removed to make A and B disjoint. This cost function penalizes the cuts that are not well-balanced and ensures that the sets A and B have approximately the same number of elements (Shi and Malik, 2000). However, optimizing the above criterion is NP hard. Thus, by resorting a relaxation, the problem is reduced to minimization of the Laplacian of the graph. If the Laplacian matrix is denoted with $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the diagonal matrix whose entries are the column (or row) sums of $\mathbf{W}$, the algorithm would consist of solving the following generalized eigenvalue problem

$$\mathbf{L} = \lambda \mathbf{D} \mathbf{a}. \tag{3}$$

Subsequently, the components of the eigenvector $\mathbf{a}^*$ corresponding to the second smallest eigenvalue of (3) are thresholded to split data into two sets, i.e.,

$$\begin{cases} y_i = -1 & \text{if } a_i^* > \Delta, \\ y_i = +1 & \text{if } a_i^* \leq \Delta, \end{cases} \tag{4}$$

where Δ is the chosen threshold which is typically equal to zero.

2.2.2. Modified Normalized Cuts Clustering

As mentioned earlier, NCuts clustering is an unsupervised method which does not take class membership information into consideration. Thus, the resulting clusters may have overlapping classes. To overcome this pitfall and reduce the overlapping regions among classes, the following similarity function is adopted to weight the edges

$$w_{ij} = \begin{cases} \exp(-d(\mathbf{x}_{ci}, \mathbf{x}_{\acute{c}j})/\alpha t) & \text{if } c = \acute{c}, \\ \exp(-d(\mathbf{x}_{ci}, \mathbf{x}_{\acute{c}j})/t) & \text{if } c \neq \acute{c}, \end{cases} \tag{5}$$

where $\alpha \geq 1$ is a tuning parameter that is used to change the similarities between class-specific samples. As opposed to the original heat kernel function, this new similarity function has a supervised nature. In particular, if α parameter is equal to 1, the algorithm is equivalent to the original NCuts. However, if α is set to a value higher than 1, the similarity between any two patterns in the same class is artificially increased. As a result, similarities between samples in a same class usually become larger than the similarities between any two patterns belonging to different classes. Thus, samples belonging to the same classes tend to group in the same clusters as illustrated in Fig. 2. This results in improvement of the real-time performance of the BHDT classification system.

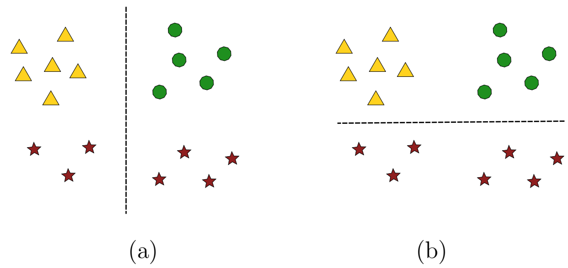

Figure 2: NCuts clustering algorithm splits data samples in order to create maximum gap between the separated samples and it creates overlapping classes as in (a). Modified NCuts algorithm on the other hand takes class membership information into consideration, and samples in the same classes tend to group in the same clusters. So, this process creates more compact clusters as in (b).

Given a set of samples at each internal node of the hierarchical tree, the proposed method applies the modified NCuts clustering algorithm to split data. In some cases, there might still be classes that lie on both sides of the cluster decision boundaries. In such cases, the approach introduced in (Marszalek and Schmid, 2008) is followed. In particular, the overlapping classes are introduced into both clusters if the ratio of the overlapping samples is greater than a selected threshold, and the resulting uncertain classification decisions are postponed until the number of classes is reduced and it becomes tractable for learning good decisions.

2.3. Class Based Large Margin Clustering

In this method, we focus on the separability of classes rather than samples and apply original NCuts clustering for partitioning class sets directly. Given a set of m ($m \leq C$) classes at each internal node of the hierarchical tree, the proposed clustering algorithm would now construct a weighted graph with m vertices (one for each class at a node) and a set of edges containing these vertices. It should be noted that the size of the graph is greatly reduced since the number of classes in a node is considerably smaller than the number of samples in those classes. Since the SVM margin is equivalent to minimum distance between the convex hulls of classes, this distance measure is used during computation of the weights of the edges. As in SVM, the shortest distance between convex hulls of two classes ω_i and ω_j is determined by the two closest points in these convex hulls. The problem of finding these two points can be represented as a quadratic optimization problem

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{v}} \ \frac{1}{2}||\mathbf{X}_i\mathbf{u} - \mathbf{X}_j\mathbf{v}||^2 \\ &s.t. \ \sum_{k=1}^{n_i} u_k = 1, \ \sum_{k=1}^{n_j} v_k = 1, \ \mathbf{u} \geq 0, \ \mathbf{v} \geq 0, \end{aligned} \tag{6}$$

where $\mathbf{X}_i$ represents the matrix whose columns are sample vectors belonging to the class ω_i and $\mathbf{u} \geq 0$ implies that its all elements are greater than or equal to zero. It must be noted that the objective function of this quadratic optimization problem is convex and a global minimum exists. This formulation is also equivalent to the hard-margin SVM formulation (Bennett and Bredensteiner, 2000). Let $\mathbf{u}^*$ and $\mathbf{v}^*$ be an optimal solution of (6). The minimum distance between the convex hulls of classes is then given by $d(\omega_i, \omega_j) = ||\mathbf{X}_i\mathbf{u}^* - \mathbf{X}_j\mathbf{v}^*||$. This distance is equivalent to $2/||\mathbf{w}||$ in SVM formulation where $\mathbf{w}$ represents the normal of the optimal separating hyperplane returned by the SVM algorithm. However, a problem may arise if the convex hulls of classes overlap, i.e., classes are not linearly separable. In this case, the distances between those classes become zero and it may not reflect the actual similarity between classes. If the classes are close to being linearly separable and they overlap because of a few outliers, the influence of those outliers can be reduced by contracting or reducing the convex hulls during distance computation by introducing an upper bound on the coefficients in (6)[1]. In this case, the new optimization problem becomes

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{v}} \ \frac{1}{2}||\mathbf{X}_i\mathbf{u} - \mathbf{X}_j\mathbf{v}||^2 \\ &s.t. \ \sum_{k=1}^{n_i} u_k = 1, \ \sum_{k=1}^{n_j} v_k = 1, \ 0 \leq \mathbf{u} \leq \tau, \ 0 \leq \mathbf{v} \leq \tau, \end{aligned} \tag{7}$$

where $\tau \leq 1$ is the user-chosen positive bound. However, if the classes are not linearly separable, the data can be mapped into a higher-dimensional space where the classes become linearly separable. It should be noted that the objective function of (7) can be written in terms of the dot products of the samples, which allows the use of the kernel trick. As a result, the data can be implicitly mapped into a higher-dimensional space where the convex hulls do not overlap and the distances between classes in the mapped space can be computed.

There is only one design parameter t, the width of the heat kernel, to be fixed. Well chosen values of t generate well-balanced clusters at each node of the decision tree, which is critical for an efficient and reliable classification. On the other hand, if the width is too small, the algorithm will favor separating a single isolated class from the remaining classes. For unbalanced

[1] In fact this is equivalent to soft margin formulation of SVMs. The reader is referred to (Bennett and Bredensteiner, 2000) for more information.

datasets, it is more efficient to separate the large classes (classes with many samples) at the upper levels of the hierarchy, which renders the well-balanced binary decision nodes less efficient. In such cases, the weights of the adjacency matrix and the width of the kernel can be adjusted to accommodate this kind of supervision. More precisely, we can deliberately decrease the values of the edge weights between the large classes and the others by lowering the width to ensure that the large classes will be separated at the upper levels of the hierarchy.

2.4. Removing Outliers

In the proposed clustering method, the presence of data outliers can significantly change the true geometric structure of the convex hulls. It is undesirable to allow a few outlying points to excessively influence the distance computations. Therefore, the influence of data outliers should be restricted in the clustering process. As described earlier, this can be done by contracting or reducing the convex hulls during distance computations by putting an upper bound on the coefficients in (7) (Bennett and Bredensteiner, 2000). Although this procedure reduces the effects of data outliers, it does not allow the identification of all outlier samples. For a fair comparison with the existing methods, the Support Vector Data Description (SVDD) method (Tax and Duin, 2005) was used for identifying and removing the outliers in the experiments. Given a class, the SVDD method finds a compact bounding hypersphere enclosing all samples in that class. In the case of data outliers, the volume of the hypersphere is minimized for detecting those outliers. The class samples that fall outside the bounding sphere are considered as outliers. This method also allows the use of the kernel trick, and thus it is compatible with the proposed clustering method and SVMs.

3. Experiments

The BHDTs using the proposed clustering methods were tested on synthetic and real databases to assess their performance, and they were compared to the OAO, OAR and DAG SVMs as well as the BHDTs using k-means based clustering (Vural and Dy, 2004) in terms of classification accuracy and testing time[2]. We experimented with the SVM classifiers using linear kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, polynomial kernels $k(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)^p$ with degree $p = 2, 3$, and the Gaussian kernel $\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/t))$. For some databases, it was found that the SVM classification algorithms using linear or polynomial kernels either did not converge to a solution or the classification performances were too low since the selected kernels failed to approximate the class decision boundaries correctly. Thus, such results were omitted and only the results for kernels yielding good classification accuracies were reported. The distances between samples were computed using the Euclidean distance, and the heat kernel function was used for weighting the edges of the adjacency matrix of the graph during clustering for the BHDT method using the proposed sample based clustering. However, for the BHDT method using the proposed class based clustering, the selected kernel function (linear, polynomial, or the Gaussian) was used for computing the distances between the convex hulls of classes. Subsequently, the heat kernel function was again used to weight the edges as usual.

[2] All programs can be downloaded at *http://www2.ogu.edu.tr/∼mlcv/softwarelink.htm*

The shape of the decision boundaries and the distances between the samples or classes significantly affect the optimal values of the design parameters, the kernel width t, and α. In other words, choosing the best design parameters is data dependent. Therefore, randomly created training and validation sets were used to fix these parameters. The best values of the design parameters were determined using a global coarse-to-fine search. Specifically, the minimum and maximum values of design parameters that produce acceptable classification rates were first determined by coarsely searching over a wide range of the parameter space. Subsequently, a coarse grid was constructed over the unknown parameters using these computed values and finally a local search was performed near the parameters that yield the best classification rate for determining the final best values. All experiments were conducted in Matlab environment using a 3-GHz machine with 3 GB of RAM.

3.1. Experiments on Synthetic Data

Here we illustrate some properties of the methods on two simple synthetic data sets. For the first database, 3-dimensional samples drawn from normal distributions were used with means $[\mu\ \mu\ \mu]^\top$ where the value of μ is changed between -50 and 48 to create 100 classes. The classes were close to being linearly separable with having small overlaps between them. Thus, it was assumed that k-means based clustering should work well in this case. For each class, we used 20 samples for training and 20 samples for testing.

For the second database we used 2-dimenional samples drawn from two-component mixture models which are typically used in XOR problem. By shifting centers of mixture components, 50 classes were created, each having 40 samples. The first six classes are plotted in Fig. 3. It can be observed that the classes are not linearly separable in this case, and k-means based clustering would not work well since the overall means are near the origin for all classes. A total of 40 samples per class were used for both training and testing.

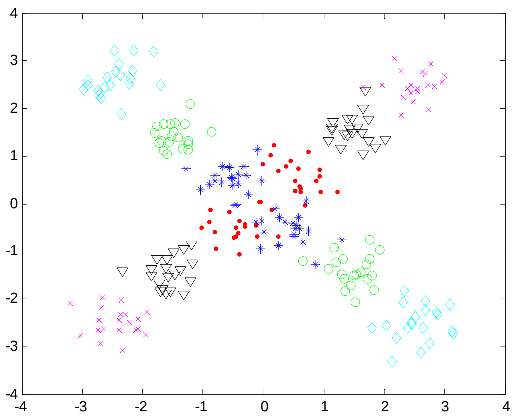

Figure 3: The first six classes having two mixture components from the second synthetic database. The mean of each class is the origin.

The classification rates for the different kernels on the first and the second synthetic databases are given in Tables 1- 3. The Proposed Method1 refers to the BHDT classifier using the modified NCuts clustering, and the Proposed Method2 denotes the BHDT classifier using the class based NCuts clustering. The design parameters were set by repeating the above described procedures 5 times and the final reported classification accuracies are averages over 10 repetitions. Asterisks in the table indicate the performance differences that are statistically significant at 5% level between the given method and the corresponding best result indicated in bold (statistical significany tests are determined based on the two-sample t test procedure (Devore, 2004) for all

Table 1: Classification Rates(%) for the Linear Kernel on the First Synthetic Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	87.27 ± 1.4	12.27
Proposed Method2	87.25 ± 1.3	**1.73**
BHDT of (Vural and Dy, 2004)	87.25 ± 1.3	1.76
DAG SVM	$\mathbf{87.38 \pm 1.3}$	28.48
OAO SVM	87.24 ± 1.3	1280.00
OAR SVM	NA	NA

experiments), and the testing time indicates the time consumed for classifying all test points. It should be noted that the testing time for a single sample is fixed for multi-class SVMs using the OAO and OAR approaches; however, it changes for BHDTs and DDAGs. An exact number of $C-1$ classifiers must be evaluated in order to label a test sample for DDAGs. For BHTDs, the best case occurs if the predicted class is found at the first node, and the worst case occurs if the predicted class is found after applying all $C-1$ decision functions.

For the first database, only linear and the Gaussian kernels produced satisfactory results where the Gaussian kernel yielded better results than the linear kernel. For both kernels, the classification rates of all tested methods are very similar except for OAR SVM, which yields a very poor classification accuracy with respect to the other methods for the Gaussian kernel and does not converge to a solution for the linear kernel (it is because all pairwise

Table 2: Classification Rates(%) for the Gaussian Kernel on the First Synthetic Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	90.23 ± 0.4	14.84
Proposed Method2	90.21 ± 0.4	**2.08**
BHDT of (Vural and Dy, 2004)	90.26 ± 0.5	2.10
DAG SVM	90.23 ± 0.4	33.67
OAO SVM	$\mathbf{90.30 \pm 0.5}$	1489.80
OAR SVM	$71.88^{*} \pm 0.4$	37.97

classes are close to being linearly separable, but classes are no longer linearly separable when the OAR scheme is used). It can be found that BHDTs using the proposed class based clustering and BHDTs using k-means clustering offer the best performance in terms of testing time for both kernels, and the BHDT classifier using the proposed class based large margin clustering is considerably faster than the one using the proposed modified NCuts clustering.

For the second database it was found that only the Gaussian kernel worked well since the decision boundaries between classes are nonlinear and highly complex. For all other tested kernels, either the SVM classification algorithm did not converge to a solution or the classification accuracies were too low. For the Gaussian kernel, the BHDT classifier using the proposed class based clustering achieved the best performance in terms of testing time among all the tested methods. Both of our proposed methods, DAG and

Table 3: Classification Rates (%) for the Gaussian Kernel on the Second Synthetic Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$\mathbf{96.98 \pm 0.4}$	17.53
Proposed Method2	96.97 ± 0.4	**1.81**
BHDT of (Vural and Dy, 2004)	$73.73^{*} \pm 6.3$	2.67
DAG SVM	96.74 ± 0.9	16.82
OAO SVM	96.95 ± 0.4	373.66
OAR SVM	$94.12^{*} \pm 2.8$	15.92

OAO SVMs yield the best classification accuracies where the BHDT classifier using modified NCuts clustering wins with a slight edge. As expected, the BHDT classifier using k-means clustering has the worst performance in terms of classification accuracy; its testing time is also low as compared to the second proposed method. This is because all class means are near the origin and k-means based clustering fails to measure the actual similarities among classes, and thus it becomes very difficult to separate the resulting classes using the SVM classifier. As in the first synthetic database, the BHDT classifier using the class based clustering is much faster than the one using the modified NCuts clustering.

3.2. Experiments on Coil-100 Object Database

Here we test methods on the Coil-100 objects database[3]. The Coil-100 database includes 72 view-images of 100 different objects taken at 5-degree-apart orientations. The size of each image is 128×128. All images were converted to gray scale and Principal Component Analysis (PCA) was applied to reduce the dimensionality to 100. A total of 36 samples were used from each class for training, and the remaining 36 samples were used for testing. The design parameters were set using 5 random training/test splits and the final reported classification accuracies were averages over 10 random training/test splits. Data outliers in the training sets were removed using the SVDD method prior to the application of the classifiers.

Table 4: Classification Rates (%) for Polynomial Kernel with $p = 2$ on Coil-100 Objects Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$93.93^{*} \pm 0.6$	17.29
Proposed Method2	94.32 ± 0.4	**15.75**
BHDT of (Vural and Dy, 2004)	94.26 ± 0.4	17.90
DAG SVM	94.14 ± 0.5	64.24
OAO SVM	94.15 ± 0.5	2710.02
OAR SVM	$\mathbf{94.50 \pm 0.5}$	122.72

[3]Available at *http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php*

Table 5: Classification Rates (%) for Polynomial Kernel with $p = 3$ on Coil-100 Objects Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$89.21^{*} \pm 0.4$	28.71
Proposed Method2	$89.70^{*} \pm 0.3$	**25.32**
BHDT of (Vural and Dy, 2004)	$89.70^{*} \pm 0.4$	25.67
DAG SVM	$88.82^{*} \pm 0.3$	64.89
OAO SVM	$88.80^{*} \pm 0.3$	2698.90
OAR SVM	$\mathbf{91.86 \pm 0.2}$	650.96

The results for different kernels are given in Tables 4- 6. It can be seen from the tables that the BHDT classifier using the proposed class based clustering is the most efficient method in terms of testing time in all cases. However, its classification rate is slightly lower than some other traditional multi-class SVMs. Nevertheless, the classification performance of the proposed method is still satisfactory for most of the cases. It is found that the BHDT of (Vural and Dy, 2004) using k-means clustering has worse performance than the second proposed method in terms of classification accuracy for polynomial kernel with degree 2 whereas it has better performance for the Gaussian kernel. Overall the best classification accuracies are obtained using the Gaussian kernels, and the worst results are obtained for the polynomial kernel with degree 3.

Table 6: Classification Rates (%) for the Gaussian Kernel on Coil-100 Objects Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$94.97^* \pm 0.2$	104.69
Proposed Method2	$96.45^* \pm 0.3$	**58.03**
BHDT of (Vural and Dy, 2004)	$96.97^* \pm 0.3$	77.78
DAG SVM	$\mathbf{97.39 \pm 0.3}$	209.96
OAO SVM	97.33 ± 0.3	9819.70
OAR SVM	$96.28^* \pm 0.3$	515.26

3.3. Experiments on the AR Face Database

The AR face database includes 26 frontal images with different facial expressions, illuminations conditions, and occlusions for 126 subjects. Images were recorded in two different sessions 14 days apart. Thirteen images were recorded under controlled circumstances in each session. The size of the images is 768×576. A total of 50 individuals (30 males and 20 females) were randomly selected for experiment. The images were aligned and scaled so that the centers of the two eyes always fall on fixed coordinates. The pre-processed images of a person are illustrated in Fig. 4. As in the previous experiment, PCA was applied and the dimensionality was decreased to 200. A total of 20 samples were randomly selected for training for each individual while keeping the remaining six for testing. This process was repeated 10 times and the final classification rates were obtained by averaging the results obtained in each run.

Figure 4: Some pre-processed images from the AR face database.

The results for the different kernels are given in Tables 7- 9. It can be seen that the best classification accuracies are obtained using the linear kernel for all methods except OAR SVM, which achieves the best classification accuracy for the polynomial kernel with degree 2. For the linear kernel, the best classification accuracy is obtained using DAG SVM and OAO SVM, with DAG SVM being more successful owing to a slight advantage. It can be seen that both of these methods significantly outperform the other methods. It is found that the BHDT of (Vural and Dy, 2004) is the fastest in terms of testing time followed by the Proposed Method2. For both the polynomial and the Gaussian kernels, the best classification accuracy is obtained by OAR SVM. The BHDT classifier using the proposed class based clustering and the BHDT of (Vural and Dy, 2004) are the most efficient methods in terms of testing time for the polynomial and Gaussian kernel functions whereas the Proposed Method2 outperforms the BHDT of (Vural and Dy, 2004) in terms of classification accuracy. The Proposed Method1 also outperforms the BHDT of (Vural and Dy, 2004) for the Gaussian kernel; however, its classification accuracy is slightly lower for the polynomial kernel.

Table 7: Classification Rates (%) for Linear Kernel on AR Face Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$96.87^{*} \pm 1.0$	1.58
Proposed Method2	$97.47^{*} \pm 0.7$	0.65
BHDT of (Vural and Dy, 2004)	$96.47^{*} \pm 0.9$	**0.53**
DAG SVM	$\mathbf{99.20 \pm 0.4}$	2.95
OAO SVM	99.17 ± 0.3	64.37
OAR SVM	$97.10^{*} \pm 0.5$	5.01

4. Summary and Conclusion

In this study, two new clustering algorithms were proposed for the partition of data samples for SVM based BHDTs. The proposed methods have two major advantages over the traditional clustering algorithms used for this purpose. Firstly, the proposed methods are suitable when SVMs are used as the base classifier. On the other hand, the most commonly employed k-means clustering algorithm may not be compatible with the SVM classifier as demonstrated in the synthetic database experiments. It must be noted that the k-means clustering is based on the assumption that the class densities are Gaussian and isotropic. However, in general, this assumption is not true for most of the classification problems in the real world, and k-means clustering algorithm might produce clusters that are difficult to separate using the SVM classifier. Secondly, the proposed methods allow the use of kernel functions as opposed to the other clustering algorithms such as Kullback-Leibler distance

Table 8: Classification Rates (%) for Polynomial Kernel with $p = 2$ on AR Face Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$90.67^{*} \pm 1.2$	12.14
Proposed Method2	$91.12^{*} \pm 1.5$	**2.74**
BHDT of (Vural and Dy, 2004)	$90.83^{*} \pm 1.5$	**2.74**
DAG SVM	$93.00^{*} \pm 1.6$	3.68
OAO SVM	$93.00^{*} \pm 1.6$	67.96
OAR SVM	$\mathbf{99.14 \pm 0.2}$	65.52

based clustering or the k-means algorithm. This is a significant advantage since the use of different kernel functions in SVM can significantly change the decision boundaries.

In the proposed methods, the most important design parameter is the width of the heat kernel function. Selection of the best value of this parameter is data dependent since the distances between the convex class sets or data samples significantly affect the optimal values of the parameter. Moreover, for unbalanced data sets it must be specially fixed so that the larger classes will be separated at the upper levels of the hierarchical tree. Thus, it is better to fix this parameter based on a cross-validation scheme as done in the present study. The use of the proposed clustering algorithms with well-chosen parameters in BHDTs will generate well-balanced separable clusters at each internal node of the decision tree, which is crucial for a reliable and efficient classification. Although both of the proposed clustering methods

Table 9: Classification Rates (%) for the Gaussian kernel on the AR Face Database.

Methods	Classification Rate	Testing Time (secs)
Proposed Method1	$92.00^{*} \pm 1.6$	15.36
Proposed Method2	$91.90^{*} \pm 1.2$	**6.07**
BHDT of (Vural and Dy, 2004)	$90.87^{*} \pm 1.5$	6.45
DAG SVM	$94.83^{*} \pm 1.5$	16.97
OAO SVM	$94.90^{*} \pm 1.4$	408.17
OAR SVM	$\mathbf{97.93 \pm 0.8}$	102.05

worked well in the experiments, the proposed class based NCuts clustering method seemed to be more efficient than the proposed sample based clustering method. The average testing time for labeling an unseen sample using the proposed methods is $O(\log_2 C)$. Thus, it can be concluded that the proposed methods can considerably increase the speed of classification with a small decrease in the classification accuracies when the number of classes is large.

References

Bennett, K. P. and Bredensteiner, E. J., 2000. Duality and geometry in SVM classifiers, International Conference on Machine Learning.

Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery, pp. 121-167.

Casasent, D. and Wang, Y.-C., 2005. A hierarchical classifier using new support vector machines for automatic target recognition, Neural Networks, vol. 18, pp. 541-548.

Chen, Y., Crawford, M.M., and Ghosh, J., 2004. Integrating support vector machines in a hierarchical output space decomposition framework, Proc. 2004 International Geoscience and Remote Sensing Symposium, pp. 949-952.

Cortes, C. and Vapnik, V., 1995. Support vector networks, Machine Learning, vol. 20, pp. 273-297.

Devore, J. L., 2004. Probability and Statistics for Engineering and the Sciences, Brooks/Cole-Thomson Learning, Belmont CA.

Friedman, J. H. and Bentley, J. I. and Finkel, R. A., 1977. An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math Software, vol. 3, pp. 209-226.

Griffin, G. and Perona, P., 2008. Learning and using taxonomies for fast visual categorization, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Hsu, C. and Lin, C., 2002. A comparison of Methods for Multi-Class Support Vector Machines, IEEE Transactions on Neural Networks, vol. 13, pp. 415-425.

Liu, S., Yi, H., Chia, L.T., and Rajan, D., 2005. Adaptive Hierarchical Multi-Class SVM Classifier for Texture-Based Image Classification, IEEE International Conference on Multimedia and Expo.

Marszalek, M. and Schmid, C., 2008. Constructing category hieararchies for visual recognition, European Conference on Computer Vision.

Martinez, A. M. and Benavente, R., 1998. The AR Face Database, CVC Tech. Report 24.

Platt, J. C., Cristianini, N., and Shawe-Taylor, J., 2000. Large margin dags for multiclass classification, Advances in Neural Information Processing Systems, pp. 547-553.

Rahimi, A. and Recht, B., 2004. Clustering with Normalized Cuts is Clustering with a Hyperplane, Statistical Learning in Computer Vision.

Schölkopf, B., Smola, A., 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.

Shi J., and Malik, J., 2000. Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 888-905.

Tax, D. M. J., and Duin, R. P. W., 2005. Support Vector Data Description, Machine Learning, vol. 54, pp. 45-66.

Vural V. and Dy, J.G., 2004. A hierarchical method for multi-class support vector machines, Proceedings of the twenty-first International Conference on Machine Learning.

Zhigang, L., Wenzhong, S., Qianqing, Q., Xiaowen, L., and Donghu, X., 2005. Hierarchical support vector machines, Geoscience and Remote Sensing Symposium.