Large Margin Classifier Based on Hyperdisks

Hakan Cevikalp
Electrical and Electronics Engineering Department
Machine Learning and Computer Vision Laboratory, Eskisehir Osmangazi University
Meselik, Eskisehir, 26480 Turkey. Email:hakan.cevikalp@gmail.com

***Abstract*—This paper introduces a binary large margin classifier that approximates each class with an hyperdisk constructed from its training samples. For any pair of classes approximated with hyperdisks, there is a corresponding linear separating hyperplane that maximizes the margin between them, and this can be found by solving a convex program that finds the closest pair of points on the hyperdisks. More precisely, the best separating hyperplane is chosen to be the one that is orthogonal to the line segment connecting the closest points on the hyperdisks and at the same time bisects the line. The method is extended to the nonlinear case by using the kernel trick, and the multi-class classification problems are dealt with constructing and combining several binary classifiers as in Support Vector Machine (SVM) classifier. The experiments on several databases show that the proposed method compares favorably to other popular large margin classifiers.**

***Keywords*-classification; convex hull; hyperdisk; kernel methods; large margin classifier; quadratic programming; support vector machines.**

I. Introduction

Large margin classifiers have recently enjoyed increased attention due to their successful applications in various fields such as computer vision (visual object classification and detection), text classification, biometrics, and genetic microarrays [1,2,3,4]. The most popular large margin classifier, the Support Vector Machines (SVMs) [4], is a binary classification method that simultaneously minimizes the empirical classification error and maximizes the geometric margin, which is defined to be the distance between the best separating hyperplane and closest samples from the classes. If the classes are not linearly separable in the original input space, the data samples are mapped onto a higher-dimensional space where they become linearly separable, and the best separating hyperplane is constructed in the mapped space. Finding such an hyperplane involves the minimization of a convex quadratic function subject to linear inequality constraints, and the quadratic optimization problem can be efficiently solved using sequential minimal optimization [5,6,7] or using minimum enclosing balls [8]. The solution of the quadratic programming problem leads to a sparse solution that enables us to evaluate the decision function by using a small number of samples in the vicinity of the class decision boundaries (more precisely, the training samples that lie on either on the margin or on the "wrong" side of it) called the support vectors. Therefore, SVM classifier is relatively fast compared to the other classification algorithms, which makes it attractive for the most of the pattern classification tasks.

From geometrical point of view, in linearly separable case, SVM classifier approximates each class with a convex hull and finds the closest points in these hulls [9,10]. Then these two closest points are connected with a line segment. The separating hyperplane, that is orthogonal to the line segment and at the same time bisects the line, is chosen to be the best separating hyperplane. In other words, 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. However, convex hulls may not be the best models for approximation of classes especially in high-dimensional spaces. Because, convex hulls approximations tend to be unrealistically tight in high-dimensional spaces since the classes typically extend beyond the convex hulls of their training samples (It should be noted that even if the original dimensionality of the input space is low, the data samples are mapped to a higher-dimensional feature space through kernel mapping during the estimation of the nonlinear decision boundaries with SVMs.) For example, for classes that are ellipsoids or boxes in high dimensions and for any placement of any number of samples sub-exponential in the dimension, the volume of their convex hull is exponentially smaller than the volume of the real class region. Similarly, for the Gaussians, the convex hull of any probable placement of a sub-exponential number of samples contain exponentially little probability mass. Other alternative models to the convex hulls may be affine hulls, hyperspheres, hyperdisks, and hyperellipsoids, which are all convex geometric models: Affine hulls are linear subspaces that have been shifted to pass through the centroids of the classes. The hyperdisk model of a class is the intersection of the affine hull and the smallest bounding hypersphere of its training samples [11]. Hyperellipsoids on the other hand are characterized by the covariance matrix of the class samples as well as their mean. Different studies [12,11,13,1,14,15] show that when such convex models are used in "nearest convex model" type classifiers, convex hulls of samples are often outperformed by simpler convex models such as affine hulls or hyperdisks. These results are not surprising due to the fact that high-dimensional approximations tend to be simple: For a fixed sample size, the amount of geometric details that can be resolved usually decreases rapidly as the dimensionality increases. Note that the methods we just cited are "nearest convex model" type of classifiers rather than a "large margin classifier between the convex models".

This paper introduces a new binary large margin clas-

sifier between the convex models (rather than the nearest convex model classifier) that approximates each class with an hyperdisk model. One motivation for replacing nearest-convex-model approaches with margin-based ones is that for the nearest convex model classifier, the pairwise decision boundaries (surfaces equidistant from the two convex models) are generically at least quadratic or piecewise quadratic in complexity. Such decision boundaries are more flexible than linear ones, but in high dimensions when the training data is scarce this may lead to overfitting, thus damaging generalization to unseen examples. Linear margin based approaches have fewer degrees of freedom, so they are typically less sensitive to the precise arrangement of the training samples. For example for an SVM classifier, motions of the SVM support vectors parallel to the SVM decision surface do not alter the margin and hence do not invalidate the classifier (although they might allow an even better one to be found), whereas they do typically change the piecewise quadratic decision surface of the equivalent nearest-convexhull classifier. The best separating hyperplane between hyperdisks is chosen to be the one that maximizes the distance between them. Finding such an hyperplane was first discussed in [3], and a solution based on a linear system and 2D Newton root-finding process has been given there for linearly separable data. Here this problem is formulated as a quadratically constrained quadratic optimization problem, and it is also extended to the nonlinear case by using the kernel trick. To handle multi-class problems, we construct several binary classifiers and combine them by using different techniques (e.g., one-against-one, one-againts-rest, etc.) as in SVM.

The rest of the paper is organized as follows: In Section 2, we introduce the proposed method. Section 3 describes the experimental results. Concluding remarks are given in Section 4.

II. Method

A. Motivation and Problem Setting

Consider a binary classification problem with the training data given in the form $\{\mathbf{x}_i, y_i\}$, $i = 1, ..., n$, $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \mathbb{R}^d$. The most popular large margin classifier, SVM, finds a separating hyperplane that maximizes the margin, which is defined as the distance between the hyperplane and closest samples from the classes. To do so, SVM first approximates each class with a convex hull [9]. A convex hull consists of all points that can be written as a linear combination of data points where all coefficients are nonnegative and sum up to 1. More formally, the convex hull of samples $\{\mathbf{x}_{ci}\}_{i=1,...,n_c}$ of class c can be written as

$$H_c^{convex} = \left\{ \mathbf{x} = \sum_{i=1}^{n_c} \alpha_i \mathbf{x}_{ci} | \sum_{i=1}^{n_c} \alpha_i = 1, \alpha_i \geq 0 \right\}. \quad (1)$$

Following this approximation, SVM finds the closest points in these convex hulls. Then, these two points are connected with a line segment. The plane, orthogonal to the line segment, that bisects the line is selected to be the separating hyperplane [9,10]. The convex hull model is the tightest possible convex approximation to the class samples, and for classes with more general convex forms, it is typically a substantial under-approximation.

The large margin classifier using affine hulls on the other hand approximates each class with an affine hull [1]. An affine hull of a class c is the smallest affine subspace containing the class samples and the affine hull of samples $\{\mathbf{x}_{ci}\}_{i=1,...,n_c}$ can be written as

$$H_c^{affine} = \left\{ \mathbf{x} = \sum_{i=1}^{n_c} \alpha_i \mathbf{x}_{ci} | \sum_{i=1}^{n_c} \alpha_i = 1 \right\}. \quad (2)$$

This is an unbounded and hence typically rather loose model for a class in contrast to the convex hull approximation. Affine hulls work surprisingly better than convex hulls especially in high-dimensional spaces with limited number of samples [12, 1, 3]. However, one may have problems with affine hull models if the classes have similar or intersecting affine hulls, but very different distributions of samples within their affine hulls.

The hyperdisk is a model between convex and affine hulls, and it captures the best aspects of each model. The hyperdisk of a class is the intersection of the affine hull and the smallest bounding hypersphere of its training samples as illustrated in Fig. 1, and it maintains the stability of the affine hull and hypersphere methods while providing better localization of the training samples and hence potentially a better discrimination. The hyperdisk of a class consists of affine combinations of class samples as before and an additional constraint $||\sum_{i=1}^{n_c} \alpha_i \mathbf{x}_{ci} - \mathbf{s}_c||^2 \leq r_c^2$. Thus, hyperdisk of a class can be written as

$$H_c^{disk} = \left\{ \mathbf{x} = \sum_{i=1}^{n_c} \alpha_i \mathbf{x}_{ci} | \sum_{i=1}^{n_c} \alpha_i = 1, || \sum_{i=1}^{n_c} \alpha_i \mathbf{x}_{ci} - \mathbf{s}_c||^2 \leq r_c^2 \right\}. \quad (3)$$

Here, $\mathbf{s}_c$ is the center of the bounding hypersphere and r_c is the radius. These hypersphere parameters can be found by solving the following quadratic program [16]

$$\begin{aligned} \min_{\mathbf{s}_c, r_c} \ & \left(r_c^2 + \gamma \sum_i \xi_i \right) \\ \text{s.t. } & ||\mathbf{x}_{ci} - \mathbf{s}_c||^2 \leq r_c^2 + \xi_i, \quad i = 1, ..., n_c, \end{aligned} \quad (4)$$

or its dual

$$\begin{aligned} \min_{\alpha} \ & \left(\sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_{ci}, \mathbf{x}_{cj} \rangle - \sum_i \alpha_i \langle \mathbf{x}_{ci}, \mathbf{x}_{ci} \rangle \right) \\ \text{s.t. } & \sum_i \alpha_i = 1, \ \forall i \ 0 \leq \alpha_i \leq \gamma, \ i, j = 1, ..., n_c, \end{aligned} \quad (5)$$

where $\langle . \rangle$ denotes the inner product between samples. Here α_i are Lagrange multipliers and $\gamma \in [0, 1]$ is a ceiling parameter that can be set to a finite value to eliminate overdistant points as outliers. Given the solution, the center of the hypersphere is $\mathbf{s}_c = \sum_i \alpha_i \mathbf{x}_{ci}$ and the radius is $r_c = ||\mathbf{x}_{ci} - \mathbf{s}_c||$ for any $\mathbf{x}_{ci}$ for $0 < \alpha_i < \gamma$.

Our goal is to find the linear separating hyperplane that yields the maximum margin between hyperdisks of classes.

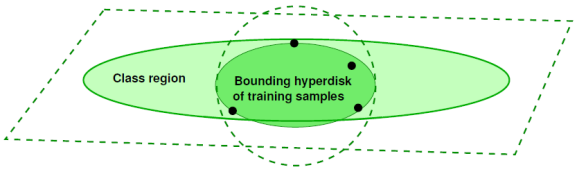

Fig. 1. Hyperdisk model of a class is the intersection of affine hull and bounding hypersphere of class samples.

The points $\mathbf{x}$ which lie on the separating hyperplane satisfy $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, where $\mathbf{w}$ is the normal of the separating hyperplane, $|b|/||\mathbf{w}||$ is the perpendicular distance from the hyperplane to the origin, and $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$. For any separating hyperplane, all points $\mathbf{x}_i$ in the positive class satisfy $\langle \mathbf{w}, \mathbf{x}_i \rangle + b > 0$ and all points $\mathbf{x}_i$ in the negative class satisfy $\langle \mathbf{w}, \mathbf{x}_i \rangle + b < 0$ so that $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$ for all training data points. Finding the best separating hyperplane maximizing the margin between hyperdisks can be solved by computing the closest points on them. The optimal separating hyperplane will be the one that bisects perpendicularly the line segment connecting the closest points as in SVM. The offset (also called threshold), b, can be chosen as the distance from the origin to the point halfway between the closest points along the normal $\mathbf{w}$. Once the best separating hyperplane is determined, a new sample $\mathbf{x}_{test}$ is classified based on the decision function, $f(\mathbf{x}_{test}) = sign(\langle \mathbf{w}, \mathbf{x}_{test} \rangle + b)$.

B. Formulation Based on Quadratically Constrained Quadratic Optimization

In this setup, we formulate the finding the closest points on the hyperdisks as a quadratically constrained quadratic optimization (QCQP) problem. Now let $\mathbf{X}_+$ and $\mathbf{X}_-$ denote the matrices whose columns are the samples belonging to the positive and negative classes, respectively. We first compute the hypersphere center and radius for both classes. Then, finding the closest points on the hyperdisk of classes can be written as the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}_+, \boldsymbol{\alpha}_-} \quad & ||\mathbf{X}_+ \boldsymbol{\alpha}_+ - \mathbf{X}_- \boldsymbol{\alpha}_-||^2 \\ \text{s.t.} \quad & \sum_i \alpha_{+i} = 1, \ \sum_j \alpha_{-j} = 1, \ i(j) = 1, ..., n_+(n_-), \\ & ||\mathbf{X}_+ \boldsymbol{\alpha}_+ - \mathbf{s}_+||^2 \leq r_+^2, \ ||\mathbf{X}_- \boldsymbol{\alpha}_- - \mathbf{s}_-||^2 \leq r_-^2. \end{aligned} \tag{6}$$

If we let $\boldsymbol{\alpha} \equiv \begin{pmatrix} \boldsymbol{\alpha}_+ \\ \boldsymbol{\alpha}_- \end{pmatrix}$, $\mathbf{y}$ be a column vector of combined labels, and $\mathbf{e}$ be a column vector of ones of arbitrary dimension, the optimization problem can be written as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top \mathbf{G} \boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}_+^\top \mathbf{e}_+ = 1, \ \boldsymbol{\alpha}_-^\top \mathbf{e}_- = 1, \\ & \boldsymbol{\alpha}_+^\top \mathbf{G}_+ \boldsymbol{\alpha}_+ - 2\boldsymbol{\alpha}_+^\top \mathbf{X}_+^\top \mathbf{s}_+ + \mathbf{s}_+^\top \mathbf{s}_+ \leq r_+^2, \\ & \boldsymbol{\alpha}_-^\top \mathbf{G}_- \boldsymbol{\alpha}_- - 2\boldsymbol{\alpha}_-^\top \mathbf{X}_-^\top \mathbf{s}_- + \mathbf{s}_-^\top \mathbf{s}_- \leq r_-^2, \end{aligned} \tag{7}$$

where $\mathbf{G} = (\mathbf{y}\mathbf{y}^\top) \circ \big([\mathbf{X}_+ \ \mathbf{X}_-]^\top [\mathbf{X}_+ \ \mathbf{X}_-]\big)$, $\mathbf{G}_+ = \mathbf{X}_+^\top \mathbf{X}_+$, and $\mathbf{G}_- = \mathbf{X}_-^\top \mathbf{X}_-$. Here $\circ$ denotes the element-wise (Hadamard) multiplication of matrices. This is a quadratically constrained quadratic programming problem and it is convex since the Hessian matrix $\mathbf{G}$ of the objective function and the other two Hessian matrices $\mathbf{G}_+$ and $\mathbf{G}_-$ of constraints are positive semi-definite matrices.

QCQP problems can be transformed into semi-definite programming (SDP), that is the optimization problem over the intersection of an affine set and cone of the positive semi-definite matrices [17]. CVX software[1] uses this approach. However, in our simulations with synthetic data, CVX sometimes failed to find a solution or returned a wrong solution. Therefore we used MOSEK software[2] in the experiments since it always successfully returned correct solutions with the simulated data. MOSEK transforms the QCQP problem into a second-order cone programming (SOCP) problem, and SOCP problems can be solved in polynomial time by interior points methods more efficiently than SDP [18]. Recently more efficient algorithms have been introduced for solving QCQP problems [19,20].

Given optimal $\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_+ \\ \boldsymbol{\alpha}_- \end{pmatrix}$, the closest pair of points in the two disks and the normal of the maximum margin separating hyperplane can be found by using the following equation

$$\mathbf{w} = \frac{1}{2}(\mathbf{x}_+ - \mathbf{x}_-) = \frac{1}{2}(\mathbf{X}_+ \boldsymbol{\alpha}_+ - \mathbf{X}_- \boldsymbol{\alpha}_-), \tag{8}$$

where $\mathbf{x}_+$ and $\mathbf{x}_-$ denote the closest points on the hyperdisks of the positive and negative classes, respectively. The offset b of the separating hyperplane will be

$$b = -\frac{1}{2}\mathbf{w}^\top (\mathbf{x}_+ + \mathbf{x}_-). \tag{9}$$

If the hyperdisks are close to being linearly separable and they overlap because of a few outliers, there are several ways to overcome this problem. Firstly, ceiling parameter γ can be set to a value smaller than 1 to find a more smaller compact hypersphere that does not include outliers. If this does not solve the overlapping problem between the hyperdisks, we can introduce lower and upper bounds on Lagrange coefficients α_i in (7) to reduce hyperdisks so that they do not overlap anymore as in reducing convex hulls introduced in [9].

In case of linearly inseparable hyperdisks, we can map the data to a higher-dimensional space where linear hyperdisks constructed in the mapped space become separable by using the kernel trick. Extension of the QCQP algorithm to the nonlinear case is easy. Note that objective function of (7) can be written in terms of the dot products of samples, which allows the use of the kernel trick, i.e., replacing the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ with the kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$.

Now let $\boldsymbol{\Phi}_+ = [\boldsymbol{\phi}(\mathbf{x}_1^+), ..., \boldsymbol{\phi}(\mathbf{x}_{n_+}^+)]$ and $\boldsymbol{\Phi}_- = [\boldsymbol{\phi}(\mathbf{x}_1^-), ..., \boldsymbol{\phi}(\mathbf{x}_{n_-}^-)]$ be matrices whose columns are the mapped samples belonging to the positive and negative classes, respectively. In the nonlinear case, hypersphere center of any class (consider positive class for example) is also expressed in terms of the mapped samples, i.e., $\boldsymbol{\phi}(\mathbf{s}_+) = \boldsymbol{\Phi}_s^+ \boldsymbol{\beta}_+$, where $\boldsymbol{\Phi}_s^+ = [\boldsymbol{\phi}(\mathbf{x}_1^+), ..., \boldsymbol{\phi}(\mathbf{x}_{l_+}^+)]$ is the matrix whose columns are

[1]available at *http://cvxr.com/cvx/*
[2]available at *http://www.mosek.com/*

the mapped samples associated to the nonzero coefficients returned by the hypersphere algorithm and the $\boldsymbol{\beta}_+$ is the vector of nonzero coefficients. Note that $l_+ \leq n_+$. Then, optimization problem becomes

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & \boldsymbol{\alpha}^\top \mathbf{G} \boldsymbol{\alpha} \\ \text{s.t.} \ & \boldsymbol{\alpha}_+^\top \mathbf{e}_+ = 1, \ \boldsymbol{\alpha}_-^\top \mathbf{e}_- = 1, \\ & \boldsymbol{\alpha}_+^\top \mathbf{K}_+ \boldsymbol{\alpha}_+ - 2\boldsymbol{\alpha}_+^\top \mathbf{K}_s^+ \boldsymbol{\beta}_+ + \boldsymbol{\beta}_+^\top \mathbf{K}_{\mathbf{s}_+} \boldsymbol{\beta}_+ \leq r_+^2, \\ & \boldsymbol{\alpha}_-^\top \mathbf{K}_- \boldsymbol{\alpha}_- - 2\boldsymbol{\alpha}_-^\top \mathbf{K}_s^- \boldsymbol{\beta}_- + \boldsymbol{\beta}_-^\top \mathbf{K}_{\mathbf{s}_-} \boldsymbol{\beta}_- \leq r_-^2, \end{aligned} \tag{10}$$

where $\mathbf{G} = (\mathbf{y}\mathbf{y}^\top) \circ \left([\boldsymbol{\Phi}_+ \ \boldsymbol{\Phi}_-]^\top [\boldsymbol{\Phi}_+ \ \boldsymbol{\Phi}_-]\right) = (\mathbf{y}\mathbf{y}^\top) \circ \mathbf{K}$, $\mathbf{K}_+ = \boldsymbol{\Phi}_+^\top \boldsymbol{\Phi}_+$, $\mathbf{K}_- = \boldsymbol{\Phi}_-^\top \boldsymbol{\Phi}_-$, $\mathbf{K}_s^+ = \boldsymbol{\Phi}_+^\top \boldsymbol{\Phi}_s^+$, $\mathbf{K}_s^- = \boldsymbol{\Phi}_-^\top \boldsymbol{\Phi}_s^-$, $\mathbf{K}_{\mathbf{s}_+} = (\boldsymbol{\Phi}_s^+)^\top \boldsymbol{\Phi}_s^+$, and $\mathbf{K}_{\mathbf{s}_-} = (\boldsymbol{\Phi}_s^-)^\top \boldsymbol{\Phi}_s^-$. Note that all kernel matrices $\mathbf{K}$, $\mathbf{K}_+$, $\mathbf{K}_-$, $\mathbf{K}_s^+$, $\mathbf{K}_s^-$, $\mathbf{K}_{\mathbf{s}_+}$, and $\mathbf{K}_{\mathbf{s}_-}$ can be easily computed by using kernel functions $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$. Given the solution $\boldsymbol{\alpha}$, the normal of the separating hyperplane is $\mathbf{w} = (1/2) \sum_{i=1}^n \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$. Bias b can be computed using (9). A new sample $\mathbf{x}_{test}$ is classified by using

$$f(\mathbf{x}_{test}) \ = \ sign(\frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_{test}) + b). \tag{11}$$

C. Extension to the Multi-Class Classification Problems

To use the proposed methods in multi-class classification problems, we can use most of the strategies adopted for extending binary SVM classifiers to the multi-class case. We used the most popular two strategies in our experiments: one-against-one (OAO) and one-against-rest (OAR). For a c-class classification problem, the OAR strategy trains c binary classifiers, in which each classifier separates one class from the remaining $c-1$ classes. All classifiers are needed to be trained on the entire training set, and the class label of a test sample is determined according to the highest output of the classifiers in the ensemble. On the other hand, the OAO strategy constructs all possible $c(c-1)/2$ binary classifiers out of c classes. The decision of the ensemble is decided by max wins algorithm: each OAO classifier casts one vote for its preferred class, and the final decision is the class with the most votes. In addition to these, one can also use Directed Acyclic Graphs (DAGs) [21] or Binary Decision Trees [22] for multi-class classification.

III. Experiments

We tested[3] the proposed method, Large Margin Classifier based on HyperDisks (LMC-HD), on a number of datasets and compared them to the SVM classifier and large margin classifier based on affine hulls (LMC-AH). Both OAR and OAO approaches are used for multi-class classification problems and we report the results of whichever yields the best.

A. Experiments with Linear Large Margin Classifiers

Here we test large margin classifiers on high-dimensional linearly separable datasets.

[3]For software see *http://www2.ogu.edu.tr/∼mlcv/softwares.html*.

Fig. 2. Aligned images of one subject from the AR face database.

1) AR Face Database: The AR Face data set [23] contains 26 frontal images with different facial expressions, illumination conditions and occlusions for each of 126 subjects, recorded in two 13-image sessions spaced by 14 days. For this experiment, we randomly selected 20 male and 20 female subjects. The images were down-scaled (from 768×576), aligned so that centers of the two eyes fell at fixed coordinates, then cropped to size 105×78. Some cropped images are shown in Fig. 2. Raw pixel values were used as features. The design parameters are set based on grid search using random partitions of datasets into training and test set. For training we randomly selected $n = 5, 10, 15, 20, 25$ samples for each individual, keeping the remaining $26 - n$ for testing. This process was repeated 15 times, with the final classification rates being obtained by averaging the 15 results. The results are plotted in Fig. 3.

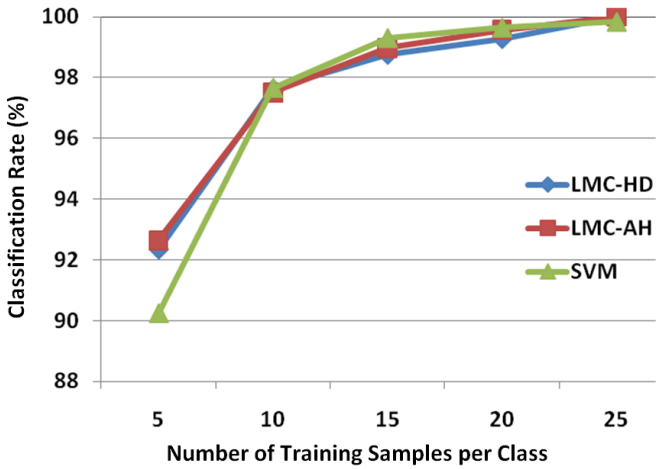

Fig. 3. Classification rates as a function of different number of samples per class on the AR Face Database.

When 5 samples per class are used, large margin classifiers based on affine hulls and hyperdisks respectively yield 92.64% and 92.34% classification accuracy, and they significantly outperform SVM, which yields 90.24% accuracy. As the number

Fig. 4. Selected 40 objects from the Coil100 database.

of samples per class is increased, all methods begin to yield similar classification accuracies. These results show that affine hull and hyperdisks are better models for representing classes in high-dimensional spaces with limited number of samples.

2) Coil100 Object Database: The Coil100 dataset[4] includes 72 views each of 100 different objects taken on a turntable at orientations spaced at 5 degree intervals. We chose 40 objects randomly for the experiments, and these objects are shown in Fig. 4. We used the raw grayscale pixels of the down-sampled 32×32 images as input features, without applying any further visual preprocessing. For training we randomly selected $n = 10, 20, ..., 60$ images of each object, keeping the remaining $72 - n$ for testing. The results are given in Fig. 5. As can be seen in the figure, all classification methods yielded same classification accuracies for this particular database.

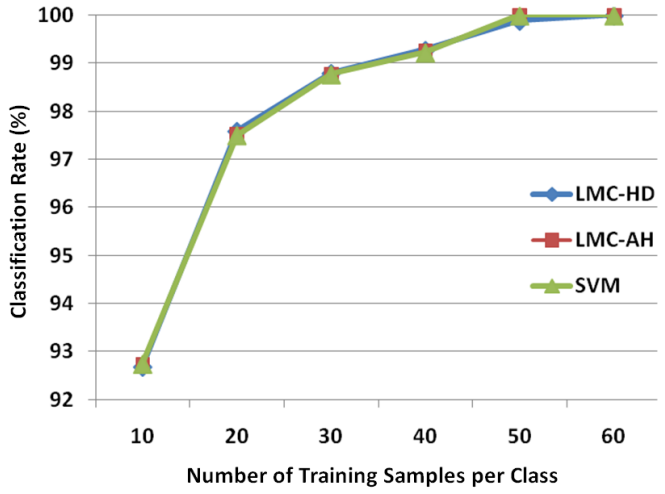

Fig. 5. Classification rates for different number of samples per class on the Coil Database.

B. Experiments with Nonlinear Large Margin Classifiers

In this group of experiments, we tested the kernelized versions of the methods on eight lower-dimensional datasets from the UCI repository[5]: Ionosphere, Iris, Image Segmentation (IS), Letter Recognition (LR), Multiple Features (MF) - pixel averages, Pima Indian Diabetes (PID), Wine, and Wisconsin Diagnostic Breast Cancer (WDBC). The key parameters of UCI Repository datasets are summarized in Table I. We used the Gaussian kernels for all datasets. All design parameters are set based on grid search using random partitions of datasets.

[4]available at *www1.cs.columbia.edu/CAVE/ software/softlib/coil-100.php*
[5]available at *at http://archive.ics.uci.edu/ml/*

TABLE I
LOW-DIMENSIONAL DATABASES SELECTED FROM UCI REPOSITORY

Databases	Number of Classes	Data Set Size	Dimensionality
Ionosphere	2	351	34
Iris	3	150	4
IS	7	2310	19
LR	26	20000	16
MF	10	2000	256
PID	2	768	8
Wine	3	178	13
WDBC	2	569	30

TABLE II
CLASSIFICATION RATES (%) ON THE UCI DATASETS.

UCI	LMC-HD	LMC-AH	SVM
Ionosphere	**94.01**±3.1	93.73±3.4	92.87±3.2
Iris	**96.67**±2.3	94.67±2.9	95.33±3.8
IS	**97.23**±0.3	95.28±0.7	97.10±0.4
LR	**99.99**±0.02	**99.99**±0.02	99.64±0.12
MF	**98.30**±0.5	**98.30**±0.5	98.00±0.4
PID	**99.87**±0.3	**99.87**±0.3	**99.87**±0.3
Wine	**98.82**±1.6	**98.82**±1.6	98.20±1.6
WDBC	97.01±0.5	96.00±0.8	**97.36**±0.9

Classification accuracies obtained by 5-fold cross-validation for UCI databases are given in Table II. Among all tested methods, the proposed hyperdisk based large margin classifier achieves the best results except for the WDBC database. For Ionosphere, LR, MF, and Wine databases, both affine hull and hyperdisk based classifiers achieve better results than SVM. On the other hand, for IS and Iris databases SVM achieves better results than affine hull based classifier, yet the hyperdisk classifier yields even higher results than SVM. There is only one case where the hyperdisk classifier is outperformed by SVM. Overall these results show that the hyperdisk model captures the best aspects of affine hulls and convex hulls. Thus, the corresponding classifier using hyperdisks either achieves the best classification accuracy or comparable results to the other convex class model classifiers using affine or convex hulls as demonstrated in Table II.

IV. SUMMARY AND CONCLUSION

We investigated the idea of basing large margin classifiers on hyperdisks of classes as an alternative to the affine and convex hull classifiers. Given two hyperdisk models, their corresponding large margin classifier is easily determined by finding a closest pair of points on these two models and bisecting the displacement between them. To this end, we formulated the problem as a convex quadratically constrained quadratic optimization problem. Extension to the nonlinear case is realized by using the kernel trick.

Hyperdisk is a model between an affine and convex hull, and it captures the best aspects of these. More precisely, since an affine hull is restricted to lie in a bounding hypersphere in an hyperdisk model, hyperdisks provide better localization of class samples compared to affine hulls. At the same time, hyperdisks are looser than convex hulls, and this enables us to approximate classes more accurately in high-dimensinal spaces. Experimental results verify these facts. Hyperdisk classifier always produced the best classification results or comparable results to the other convex class model classifiers achieving the best result. There is not a single case where the hyperdisk classifier is significantly outperformed by the other convex class model classifiers, whereas it significantly outperforms others on some databases. However, these improvements come with a price. Training time of the hyperdisk classifier is slow compared to other large margin classifiers since it requires running a quadratic programming algorithm (for finding hypersphere parameters) followed by QCQP algorithm. Another limitation is related to the real-time efficiency (testing time). Hyperdisk classifier does not return a sparse solution as in affine hull classifier, thus its real-time efficiency is slow compared to the SVM classifier. However, this limitation can be overcome by running a reduced set algorithm [24,25] that enables us to derive a sparse solution. As a future work, we are considering to revise the QCQP algorithm such that it returns sparse solutions.

ACKNOWLEDGMENT

This work is supported by the Young Scientists Award Programme (TÜBA-GEBİP/2010-11) of the Turkish Academy of Sciences.

REFERENCES

[1] H. Cevikalp, B. Triggs, H. S. Yavuz, Y. Kucuk, M. Kucuk, and A. Barkana. Large margin classifiers based on affine hulls. *Neurocomputing*, 73:3160–3168, 2010.

[2] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[3] H. Cevikalp and B. Triggs. Large margin classifiers based on convex class models. In *International Conference on Computer Vision Workshops*, 2009.

[4] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[5] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, Cambridge, 1999.

[6] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, Cambridge, 1999.

[7] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training svm. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[8] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[9] K. P. Bennett and E. J. Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, 2000.

[10] D. J. Crisp and C. J. Burges. A geometric interpretation of ν-svm classifiers. In *Neural Information Processing Systems*, 1999.

[11] H. Cevikalp, B. Triggs, and R. Polikar. Nearest hyperdisk methods for high-dimensional classification. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 120–127, 2008.

[12] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. In *NIPS*, 2001.

[13] J. Laaksonen. *Subspace classifiers in recognition of handwritten digits*. PhD thesis, Helsinki University of Technology, 1997.

[14] H. Cevikalp, D. Larlus, M. Neamtu, B. Triggs, and F. Jurie. Manifold based local classifiers: linear and nonlinear approaches. *Journal of Signal Processing Systems*, 61:61–73, 2010.

[15] M. B. Gulmezoglu, V. Dzhafarov, and A. Barkana. The common vector approach and its relation to principal component analysis. *IEEE Trans. Speech Audio Proc.*, 9:655–662, 2001.

[16] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.

[17] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[18] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2003.

[19] C.-M. Tang and J.-B. Jian. A sequential quadratically constrained quadratic programming method with an augmented lagrangian line search function. *Journal of Computational and Applied Mathematics*, 220:527–547, 2008.

[20] H. Tuy and N. T. Hoai-Phuong. A robust algorithm for quadratic optimization under quadratic constraints. *Journal of Global Optimization*, 37:557–569, 2007.

[21] J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.

[22] V. Vural and J. G. Dy. A hierarchical method for multi-class support vector machines. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 105, New York, NY, USA, 2004. ACM.

[23] A. M. Martinez and R. Benavente. The AR face database. Technical report, Computer Vision Center, Barcelona, Spain, 1998.

[24] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Müller, G. Ratsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10:1000–1017, 1999.

[25] S. Mika, B. Scölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Ratsch. Kernel pca and de-noising in feature spaces. In *Neural Information Processing Systems (NIPS)*, 1998.