Hyperdisk Based Large Margin Classifier

Hakan Cevikalp[a], Bill Triggs[b]

[a]*Electrical and Electronics Engineering Department of Eskisehir Osmangazi University, Meselik 26480 Eskisehir, Turkey*
[b]*Laboratoire Jean Kuntzmann, Grenoble, France*

Abstract

We introduce a large margin linear binary classification framework that approximates each class with a hyperdisk – the intersection of the affine support and the bounding hypersphere of its training samples in feature space – and then finds the linear classifier that maximizes the margin separating the two hyperdisks. We contrast this with Support Vector Machines (SVMs), which find the maximum-margin separator of the pointwise convex hulls of the training samples, arguing that replacing convex hulls with looser convex class models such as hyperdisks provides safer margin estimates that improve the accuracy on some problems. Both the hyperdisks and their separators are found by solving simple quadratic programs. The method is extended to nonlinear feature spaces using the kernel trick, and multi-class problems are dealt with by combining binary classifiers in the same ways as for SVMs. Experiments on a range of data sets show that the method compares favorably with other popular large margin classifiers.

Keywords: Large margin classifier, classification, convex approximation, hyperdisk, kernel method, Support Vector Machine.

1. Introduction

Large margin classifiers are successful in many fields including computer vision, text analysis, biometrics and bioinformatics [10, 18, 7, 11]. The prototypical method of this kind, the Support Vector Machine (SVM) [11], finds a linear hyperplane in feature space that maximizes the "margin" – the Euclidean distance between the hyperplane and the closest training samples of each class. This can be formulated as a quadratic program and solved efficiently by various methods [36, 20, 15, 47]. The solution is sparse in the sense that once the closest points have been found it depends only on them. Owing to the fact that they try to ensure the widest possible margin for error, the resulting classifiers often have very good practical performance, especially in cases where the classes are in fact linearly separable. If not, the separation can often be improved by making the classifier nonlinear using the kernel trick.

However SVM's are not perfect. Roughly speaking, the setting for which they were designed is one in which the classes can be modeled as non-overlapping convex clouds in a (possibly kernelized) feature space, which can thus be separated using affine hyperplanes. If the clouds overlap significantly, statistically based classifiers are likely to be more appropriate than geometric ones such as SVM. But even if there is no overlap, training SVMs from samples is problematic in that it can seriously underestimate the true extent of the classes involved. SVM is equivalent to approximating each class with the convex hull of its training samples (the tightest possible convex approximation), then finding the best linear separator of the resulting convex hulls [3, 12]. Unfortunately, in high-dimensional spaces, the convex hull of any sub-exponential number of training samples from a convex set typically has a volume that is exponentially smaller than that of the parent set. For example, this is the case for the simplex spanning any set of $d+1$ points sampled from an ellipsoid or box in d dimensions: the overwhelming majority of the set lies outside the given simplex, and a new sample from it will almost surely lie well outside the simplex in some direction. Similarly, for Gaussians, the convex hull of any probable placement of a sub-exponential number of samples contains exponentially little of the probability mass.

To limit the effects of this endemic underestimation of class boundaries, it seems useful to develop maximum margin classifiers that are based on somewhat looser convex approximations. Here we focus on a particular example of this: classifiers based on the *mini-*

Email addresses: hakan.cevikalp@gmail.com (Hakan Cevikalp), Bill.Triggs@imag.fr (Bill Triggs)

mal bounding hyperdisks of the classes. By minimal bounding hyperdisk we mean the disk-shaped set that is produced by intersecting the affine hull of the training samples (the smallest affine subspace containing them) and their bounding hyperball (the smallest-volume hypersphere containing them). Fig. 1 illustrates the hoped-for gains. We will argue that hyperdisks have properties that make them particularly well-suited to approximating classes in high dimensional feature spaces: they are relatively simple models that are easy to estimate and that have compact parametrizations; they provide comparatively tight bounds on the extents of the classes that remain close to the spirit of SVM; distance computations are efficient so the final classifiers are easy to find; and being Euclidean constructs it is straightforward to kernelize them.

Other simple convex class models that could be used include affine hulls, bounding hyperspheres, and bounding hyper-ellipsoids. The maximum margin affine hull case is studied in [10]. Despite the looseness of its class approximations, it turns out to give surprisingly good performance in many high-dimensional problems. It is also a special case of the Least Squares [43] and Proximal [16] SVM's (*i.e.* of linear least squares regression of the $y = \pm 1$ class labels from the feature vectors), in which the classes lie in disjoint affine hulls and the weight regularization is deactivated so that y is predicted perfectly on the training set. Bounding hyperspheres are used in the one-class SVM [45] and its variants. They are useful models but again their approximations are quite loose. Bounding hyper-ellipsoids are potentially tighter models than bounding hyperdisks, but they have many more parameters owing to the need to represent the full covariance matrix. The Minimax Probability Machine [25] and its variants [13] are alternative approaches that use covariance estimates to find maximum margin linear separators. Although such models are very powerful in low dimensions, in high dimensional problems they typically lead to overfitting owing to the number of free parameters. Here we will focus on hyperdisks because we believe that they offer a good compromise between complexity and tightness of approximation.

A related stream of work studies *nearest neighbour* classifiers based on distances to convex class models [50, 24, 9, 17]. For example, distance to affine hull models have been proposed for isolated word and hand written digit classification [17, 19, 24]. The distance to hyperdisk model is studied in [8]. Distance to hypersphere based models have been constructed by [27] (using the distance for probability estimation) and [21] (using SVDD projection for distance calculation). Distance-based methods can also be localized – by constructing the model on the fly from the k nearest neighbours of the test sample, *c.f.* [50] in the affine/convex case – and kernelized, *c.f.* [9]. In general, the relative ranking found for the different distance-based models reflects that advocated here for the corresponding margin based ones [8]: hypersphere models tend to be too loose, giving poor performance; affine hull models give good results in some cases but not all; the intersection of affine hulls and hyperspheres, hyperdisks, gives uniformly good results; and convex hulls give good results as well, but are often outperformed by hyperdisks.

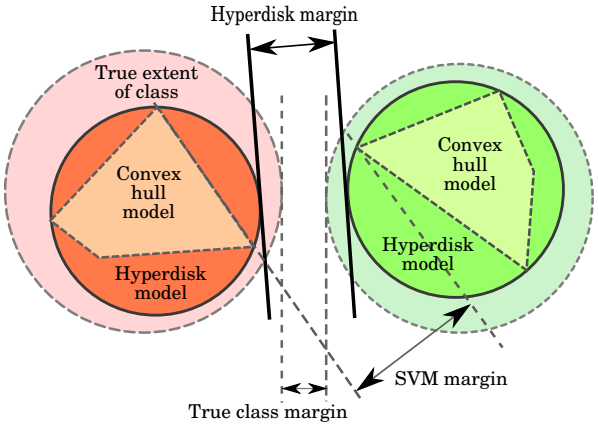

Figure 1: In high dimensions, the convex hull of the training samples is typically a significant underestimate of the true extent of the class. This often causes the corresponding maximum margin classifier (SVM) to overestimate the margin and misestimate the separation direction. Hyperdisk classifiers adopt looser class models, so they sometimes provide better estimates of the true separators and margins.

Although nearest convex model methods perform well in many applications, their philosophy is somewhat different from the margin-based one advocated here. Each class is modeled independently, making it easy to update the model to include new samples, but the classifiers are not precompiled so distance-to-convex-model computations are required for each test example at run time. Moreover, distance-based classifiers typically generate piecewise polynomial inter-class separators, so they may be more sensitive to noise than the corresponding large margin approaches which use linear separators. For example, the nearest-hyperdisk rule implicitly separates classes by piecewise quartic hypersurfaces, and a nearest convex hull one by piecewise quadratic ones.

The rest of the paper is organized as follows. Section 2 introduces the proposed method, giving two ap-

proaches for the important task of finding the maximum margin separator of two hyperdisks. Section 3 discusses the issue of sparsity. Section 4 presents our experimental results and Section 5 concludes the paper. An early version of the work presented here appeared in [6].

2. Method

2.1. Motivation and Problem Setting

Consider a binary classification problem with training data $\{\mathbf{x}_i, y_i\}$, $i = 1, ..., n$, $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \mathbb{R}^d$. As mentioned above, SVM can be viewed as a method that first approximates each class with the convex hull of its training samples, then finds the hyperplane that maximizes the separation (margin) between the two hulls [3]. The convex hull consists of all points that can be expressed as convex combinations of the training samples (*i.e.* linear combinations with nonnegative coefficients summing to 1). If $\{\mathbf{x}_{ci}\}_{i=1,...,n_c}$ are the samples for class c, the convex hull is

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

Given the hulls, their separating hyperplane can be determined by finding a pair of points, one in each hull, that minimizes the distance between the hulls and taking the orthogonal bisector of the line segment joining them [3, 12].

The convex hull is the smallest convex set containing the samples. Our method is based on a somewhat looser convex approximation, the minimal bounding hyperdisk, which is the intersection of the samples' affine hull and their minimal bounding hypersphere in the input space. Equivalently, it is their minimum bounding hypersphere within the subspace spanned by their affine hull. The affine hull – the smallest affine subspace containing the samples – is found by relaxing the positivity constraint in (1):

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

Although affine hulls are unbounded and hence necessarily rather loose models of the classes, large margin classifiers based on separating them work surprisingly well in many high-dimensional problems with limited numbers of samples – often better than SVMs [50, 10, 7]. This is one indication that convex hull based methods may be too tight to be realistic. Nevertheless, it seems useful to tighten the affine hull approximation if we can do so without adding too many parameters to the model. The hyperdisk is our way of doing this. By restricting the model to be the bounding hypersphere of the training samples within the affine hull, we provide better localization of the class within the affine hull without losing the simplicity and stability of the affine approach. The resulting model may both underestimate and overestimate the true extent of the (convex) class – see Fig. 2 – but it often does so less severely than the convex hull, especially in high dimensions. Hypersphere models are useful for outlier detection [45]. They have also been proposed as nearest-convex-model type classifiers [21, 27, 34, 53], but our previous studies show that in such applications they are outperformed by other convex models such as convex hulls, affine hulls and hyperdisks [8]. This is why we use hyperdisks not simply hyperspheres here.

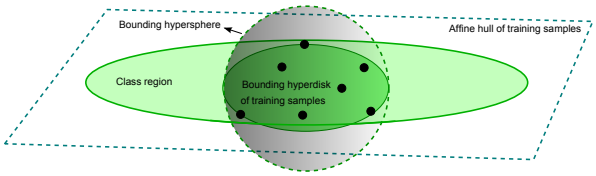

Figure 2: Hyperdisk model of a class is the intersection of affine hull and bounding hypersphere of class samples.

The minimal bounding hyperdisk of a class can be written as

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

where $\mathbf{s}_c$ is the center of the hyperdisk and r_c is its radius. These parameters can be found robustly by solving the quadratic program [45]

$$\begin{aligned} \min_{\mathbf{s}_c, r_c, \xi_i} \quad & \left(r_c^2 + \gamma \textstyle\sum_i \xi_i\right) \\ \text{s.t.} \quad & \|\mathbf{x}_{ci} - \mathbf{s}_c\|^2 \leq r_c^2 + \xi_i, \;\; \xi_i \geq 0, \;\; i = 1, ..., n_c \end{aligned} \quad (4)$$

or its dual

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

where $\langle - \rangle$ denotes the inner product between samples. Here, the α_i are Lagrange multipliers for the center $\mathbf{s}_c = \sum_i \alpha_i \mathbf{x}_{ci}$ and $\gamma \in [1/n_c, 1]$ is a ceiling parameter that can be set to a finite value to eliminate overdistant points as outliers. The corresponding radius is $r_c = \|\mathbf{x}_{ci} - \mathbf{s}_c\|$ for any $\mathbf{x}_{ci}$ for which $0 < \alpha_i < \gamma$. In d dimensions, at most $d + 1$ of the α_i are typically nonzero.

To find the maximum margin separator between two (non-intersecting) hyperdisks, we find a closest pair of

points between them (one in each disk) and take the orthogonal bisector of the line segment joining them. If the points are $\mathbf{x}_+$ and $\mathbf{x}_-$, the bisector is $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ where

$$\mathbf{w} = \frac{\mathbf{x}_+ - \mathbf{x}_-}{\|\mathbf{x}_+ - \mathbf{x}_-\|} \tag{6}$$

$$b = -\langle\, \mathbf{w},\, (\mathbf{x}_+ + \mathbf{x}_-)/2 \,\rangle\,. \tag{7}$$

We propose two methods for finding such points. The first [7] adopts a linear parametrization and reduces the problem to 2D Newton root-finding, the second formulates it as a quadratically constrained quadratic convex program.

2.2. Closest Points Using Newton Root Finding

We begin by rewriting the affine hull of each class c as $\{\mathbf{x} = \mathbf{U}_c\mathbf{v}_c + \boldsymbol{\mu}_c \,|\, \mathbf{v}_c \in \mathbb{R}^l\}$, where $\boldsymbol{\mu}_c = \frac{1}{n_c}\sum_i \mathbf{x}_{ci}$ is the mean of the samples (or any other reference point in the hull) and $\mathbf{U}_c$ is an orthonormal basis for the directions spanned by the affine subspace. The vector $\mathbf{v}_c$ contains the reduced coordinates of the point within the subspace, expressed with respect to the basis $\mathbf{U}_c$. For example, if we estimate the hull using orthogonal least squares, the U-matrix of the thin Singular Value Decomposition (SVD) of $[\mathbf{x}_{c1} - \boldsymbol{\mu}_c, ..., \mathbf{x}_{cn_c} - \boldsymbol{\mu}_c]$ can be used as $\mathbf{U}_c$, where 'thin' means that we take only the columns of U corresponding to "significantly non-zero" singular values λ_k. l is the number of these singular values. Discarding the near-zero singular values corresponds to disregarding directions that appear to be predominantly "noise". If the data may be polluted with outliers, the hull can be estimated with a robust procedure such as the L1 norm methods of [14, 22], but a $\mathbf{U}_c, \boldsymbol{\mu}_c$ parametrization still exists.

Given the center and the radius of the bounding hypersphere of each class from (5), we find the closest point pair $\mathbf{v}_+, \mathbf{v}_-$ between the disks by minimizing the inter-point distance

$$\min_{\mathbf{v}_+, \mathbf{v}_-} \|(\mathbf{U}_+\,\mathbf{v}_+ + \mathbf{s}_+) - (\mathbf{U}_-\,\mathbf{v}_- + \mathbf{s}_-)\|^2 \tag{8}$$

subject to the within-disk constraints $\|\mathbf{v}_+\|^2 \le r_+^2$ and $\|\mathbf{v}_-\|^2 \le r_-^2$. Introducing Lagrange multipliers $\lambda_+ - 1$ and $\lambda_- - 1$ for the two inequality constraints, this reduces to solving the linear system

$$\begin{pmatrix} \lambda_+\,\mathbf{I} & -\mathbf{U}_+^\top\mathbf{U}_- \\ -\mathbf{U}_-^\top\mathbf{U}_+ & \lambda_-\,\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} = \begin{pmatrix} -\mathbf{U}_+^\top \\ \mathbf{U}_-^\top \end{pmatrix} (\mathbf{s}_+ - \mathbf{s}_-) \tag{9}$$

subject to $\lambda_+ \ge 1$, $\lambda_- \ge 1$. (We have added one to the Lagrange multipliers to account for the $\mathbf{U}_\pm^\top\,\mathbf{U}_\pm = \mathbf{I}$ terms on the diagonal of the matrix). We can find $(\lambda_+, \lambda_-) \ge 1$ such that $\|\mathbf{v}_\pm\|^2 \le r_\pm^2$ efficiently with a 2D Newton root-finding process analogous to the 1D one used by eigenvalue finders. First, note that by changing coordinates to the principal angle basis between the supporting affine subspaces of the hyperdisks, and hence diagonalizing the off-diagonal blocks of the matrix, the linear system can be reduced to a set of decoupled 2×2 subsystems. The SVD of $\mathbf{U}_+^\top\,\mathbf{U}_-$ gives the necessary linear bases, with the singular values being the cosines of the corresponding principal angles. (In cases where there is no principal angle or the angle is 90 degrees, the equations separate further to sets of two 1×1 ones). For any given (λ_+, λ_-), we can solve these equations componentwise in closed form to find $(\mathbf{v}_+, \mathbf{v}_-)$. To find (λ_+, λ_-), we therefore run a Newton root finding iteration on the 2D equation $(\|\mathbf{v}_+\|^2, \|\mathbf{v}_-\|^2) = (r_+^2, r_-^2)$ with $(\mathbf{v}_+, \mathbf{v}_-)$ as functions of (λ_+, λ_-). This holds for $(\lambda_+, \lambda_-) > 1$. If either λ_+ or λ_- becomes 1 (indicating that the solution is interior to the corresponding disk so that the norm constraint on $\mathbf{v}$ is inactive), we use the Newton method to solve the corresponding 1D equation for the remaining variable. In either case the process converges rapidly and reliably, usually within 3-5 iterations. MATLAB code for this is available from the authors. Given the optimal $\mathbf{v}_\pm$, we reconstruct the corresponding points $\mathbf{x}_\pm = \mathbf{U}_\pm\mathbf{v}_\pm + \mathbf{s}_\pm$ to find the separating hyperplane.

2.2.1. Kernelization

The above method can be kernelized simply by introducing explicit orthogonal coordinates in a suitable kernel feature space and running the Euclidean affine hull, hyperdisk and separator finding algorithms in these coordinates. The feature space must include the training samples of both classes as the separator depends on both. Test samples can be classified by mapping them into this space, which allows the equivalent kernelized classification rule to be obtained in the input space.

The construction is similar to KPCA [39]. Let $\boldsymbol{\phi}(\cdot)$ be the implicit feature space embedding and $k(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \right\rangle$ be the corresponding kernel function. Suppose that we want to project points $\mathbf{x}$ onto the affine hull of a given set of samples $\{\mathbf{x}_i\}_{i=1,...,n}$. Let $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), ..., \boldsymbol{\phi}(\mathbf{x}_n)]$ be their implicit feature space embedding matrix, $\mathbf{K} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} = [k(\mathbf{x}_i, \mathbf{x}_j)]$ be their $n \times n$ kernel matrix and $\mathbf{k}_\mathbf{x} = \boldsymbol{\Phi}^\top\boldsymbol{\phi}(\mathbf{x}) = [k(\mathbf{x}_i, \mathbf{x})]$ be the $n \times 1$ kernel vector of $\mathbf{x}$ against the samples. The feature space mean of the samples is $\boldsymbol{\mu} = \frac{1}{n}\boldsymbol{\Phi}\,\mathbf{1}_n$ (where $\mathbf{1}_n$ is an n-vector of 1's). So the matrix of centered sample features is $[\boldsymbol{\phi}(\mathbf{x}_1) - \boldsymbol{\mu}, ..., \boldsymbol{\phi}(\mathbf{x}_n) - \boldsymbol{\mu}] = \boldsymbol{\Phi}\,\boldsymbol{\Pi}$, where $\boldsymbol{\Pi} = \mathbf{I} - \frac{1}{n}\mathbf{1}_n\,\mathbf{1}_n^\top$ is the orthogonal projection in sample space that implements subtraction of the mean on $\boldsymbol{\Phi}$. If $\boldsymbol{\phi}$ were an explicit embedding, the thin SVD $\mathbf{UDV}^\top$ of $\boldsymbol{\Phi}\,\boldsymbol{\Pi}$ would

yield an orthogonal basis $\mathbf{U} = \mathbf{\Phi}\,\mathbf{\Pi}\,\mathbf{V}\,\mathbf{D}^{-1} = \mathbf{\Phi}\,\mathbf{A}^\top$ for the affine subspace, where $\mathbf{A} = \mathbf{D}^{-1}\mathbf{V}^\top\mathbf{\Pi}$. ($\mathbf{D}$ is invertible because we use a thin SVD containing only the significantly non-zero singular values). Although we can not calculate this SVD explicitly, we can get the same result by taking the corresponding thin eigendecomposition $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ of the centered kernel matrix $\widetilde{\mathbf{K}} = (\mathbf{\Phi}\,\mathbf{\Pi})^\top(\mathbf{\Phi}\,\mathbf{\Pi}) = \mathbf{\Pi}\,\mathbf{K}\,\mathbf{\Pi}$ and defining $\mathbf{D} = \mathbf{\Lambda}^{1/2}$, i.e., $\mathbf{A} = \mathbf{\Lambda}^{-1/2}\,\mathbf{V}^\top\mathbf{\Pi}$. The projection of a new sample $\mathbf{x}$ onto $\mathbf{U}$ coordinates in the affine hull is simply $\mathbf{U}^\top\boldsymbol{\phi}(\mathbf{x}) = \mathbf{A}\,\mathbf{k}_\mathbf{x}$, so the kernelized classification rule is $\mathbf{w}^\top\mathbf{A}\,\mathbf{k}_\mathbf{x} + b > 0$. It is also possible to work in terms of linear spans rather than affine ones, explicitly subtracting the mean after reduction to $\mathbf{U}$ coordinates. The same formulae apply with $\mathbf{\Pi}$ omitted.

2.3. Closest Points Using Quadratically Constrained Quadratic Programming

We now give an alternative approach to finding the closest points on the hyperdisks, by solving a quadratically constrained quadratic program (QCQP). Let $\mathbf{X}_+$ and $\mathbf{X}_-$ denote matrices whose columns are respectively the positive and negative training samples. As before, we first compute the hypersphere centers and radii for both classes. Then, finding the closest points $\mathbf{x}_\pm$ on the two hyperdisks can be written as the following optimization problem, where $\boldsymbol{\alpha}_\pm$ are vectors of expansion weights for $\mathbf{x}_\pm = \mathbf{X}_\pm\,\boldsymbol{\alpha}_\pm$

$$\begin{aligned}\min_{\boldsymbol{\alpha}_+,\boldsymbol{\alpha}_-}\ & \|\mathbf{X}_+\boldsymbol{\alpha}_+ - \mathbf{X}_-\boldsymbol{\alpha}_-\|^2 \\ \text{s.t.}\ & \textstyle\sum_{i=1}^{n_+}\alpha_{+i} = 1,\ \sum_{j=1}^{n_-}\alpha_{-j} = 1, \\ & \|\mathbf{X}_+\boldsymbol{\alpha}_+ - \mathbf{s}_+\|^2 \le r_+^2,\ \|\mathbf{X}_-\boldsymbol{\alpha}_- - \mathbf{s}_-\|^2 \le r_-^2.\end{aligned} \tag{10}$$

Given the optimal $\boldsymbol{\alpha}_\pm$ and hence $\mathbf{x}_\pm$, the separating hyperplane can be found as before.

Letting $\boldsymbol{\alpha} \equiv \binom{\boldsymbol{\alpha}_+}{\boldsymbol{\alpha}_-}$, $\mathbf{X} = [\mathbf{X}_+,\ -\mathbf{X}_-]$ and $\mathbf{e}_\pm$ be column vectors of ones of the appropriate dimensions, the problem can be rewritten as

$$\begin{aligned}\min_{\boldsymbol{\alpha}}\ & \boldsymbol{\alpha}^\top\mathbf{G}\boldsymbol{\alpha} \\ \text{s.t.}\ & \boldsymbol{\alpha}_+^\top\mathbf{e}_+ = 1,\ \boldsymbol{\alpha}_-^\top\mathbf{e}_- = 1, \\ & \boldsymbol{\alpha}_+^\top\mathbf{G}_+\boldsymbol{\alpha}_+ - 2\boldsymbol{\alpha}_+^\top\mathbf{X}_+^\top\mathbf{s}_+ + \mathbf{s}_+^\top\mathbf{s}_+ \le r_+^2, \\ & \boldsymbol{\alpha}_-^\top\mathbf{G}_-\boldsymbol{\alpha}_- - 2\boldsymbol{\alpha}_-^\top\mathbf{X}_-^\top\mathbf{s}_- + \mathbf{s}_-^\top\mathbf{s}_- \le r_-^2,\end{aligned} \tag{11}$$

where $\mathbf{G} = \mathbf{X}^\top\mathbf{X}$, $\mathbf{G}_+ = \mathbf{X}_+^\top\mathbf{X}_+$ and $\mathbf{G}_- = \mathbf{X}_-^\top\mathbf{X}_-$. This QPQP is convex as both the Hessian $\mathbf{G}$ and the constraint Hessians $\mathbf{G}_+$ and $\mathbf{G}_-$ are positive semidefinite.

QCQP's can be transformed into Semidefinite Programs (SDPs), *i.e.* optimization problems over the intersection of a cone of positive semi-definite matrices with an affine set [49]. The CVX solver[1] uses this approach but in our simulations with synthetic data, it sometimes failed to find a solution or returned an incorrect one. We therefore preferred to use the MOSEK solver[2], which always succeeded in finding the correct solution in our simulations. MOSEK transforms the QCQP into a Second-Order Cone Program (SOCP). These can be solved in polynomial time by interior points methods, which is more efficient than solving an SDP [1]. Recently, more efficient algorithms have been introduced for solving QCQP problems [44, 48].

If the hyperdisks overlap owing to outliers, two approaches are possible. Firstly, the ceiling parameters γ in (5) can be set to values less than 1 to find more compact hyperspheres that exclude the outliers. If this does not suffice, upper and lower bounds can be enforced on the coefficients $\boldsymbol{\alpha}$ in (11) to constrain the positions of the points $\mathbf{x}_\pm$, as in the soft SVM and the convex hull reduction method of [3]. In our experiments, setting $\gamma < 1$ sufficed to remove the overlap without bounds on the $\boldsymbol{\alpha}$.

2.3.1. Kernelization

The QCQP algorithm is already expressed in terms of dot products so it is straightforward to kernelize it. Given the kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \left\langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\right\rangle$, let $\mathbf{\Phi}_+ = [\boldsymbol{\phi}(\mathbf{x}_1^+), ..., \boldsymbol{\phi}(\mathbf{x}_{n_+}^+)]$ and $\mathbf{\Phi}_- = [\boldsymbol{\phi}(\mathbf{x}_1^-), ..., \boldsymbol{\phi}(\mathbf{x}_{n_-}^-)]$ be the implicit matrices of feature vectors of the positive and negative training sets, and let the corresponding explicit kernel matrices be $\mathbf{K}_+ = \mathbf{\Phi}_+^\top\mathbf{\Phi}_+$, $\mathbf{K}_- = \mathbf{\Phi}_-^\top\mathbf{\Phi}_-$ and $\mathbf{K}_{+-} = \mathbf{\Phi}_+^\top\mathbf{\Phi}_-$. Define $\mathbf{K} = \begin{pmatrix}\mathbf{K}_+ & -\mathbf{K}_{+-} \\ -\mathbf{K}_{+-}^\top & \mathbf{K}_-\end{pmatrix}$. Let the hypersphere centres in feature space be $\mathbf{K}_+\boldsymbol{\beta}_+$ and $\mathbf{K}_-\boldsymbol{\beta}_-$. Then the kernelized QCQP becomes

$$\begin{aligned}\min_{\boldsymbol{\alpha}}\ & \boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.}\ & \boldsymbol{\alpha}_+^\top\mathbf{e}_+ = 1,\ \boldsymbol{\alpha}_-^\top\mathbf{e}_- = 1, \\ & \boldsymbol{\alpha}_+^\top\mathbf{K}_+\boldsymbol{\alpha}_+ - 2\boldsymbol{\alpha}_+^\top\mathbf{K}_+\boldsymbol{\beta}_+ + \boldsymbol{\beta}_+^\top\mathbf{K}_+\boldsymbol{\beta}_+ \le r_+^2, \\ & \boldsymbol{\alpha}_-^\top\mathbf{K}_-\boldsymbol{\alpha}_- - 2\boldsymbol{\alpha}_-^\top\mathbf{K}_-\boldsymbol{\beta}_- + \boldsymbol{\beta}_-^\top\mathbf{K}_-\boldsymbol{\beta}_- \le r_-^2.\end{aligned} \tag{12}$$

Note that both $\boldsymbol{\beta}_+$ and $\boldsymbol{\beta}_-$ may be sparse as the centres depend only on the training points that lie exactly on the sphere. This can be used to reduce the amount of computation required. Given the weights $\boldsymbol{\alpha}$, the classifier is $\boldsymbol{\alpha}_+^\top\mathbf{K}_{+\mathbf{x}} - \boldsymbol{\alpha}_-^\top\mathbf{K}_{-\mathbf{x}} + b > 0$ where $b = -(\boldsymbol{\alpha}_+^\top\mathbf{K}_+\boldsymbol{\alpha}_+ + \boldsymbol{\alpha}_-^\top\mathbf{K}_-\boldsymbol{\alpha}_-)/2$ and $\mathbf{K}_{+\mathbf{x}}$ and $\mathbf{K}_{-\mathbf{x}}$ are the kernel vectors of the example being classified against the positive and negative training sets.

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

2.4. Extension to Multi-Class Problems

The hyperdisk method can be extended to multi-class problems using any of the common strategies developed for extending binary SVMs. We tested the popular one-against-one (OAO) and one-against-rest (OAR) approaches. For a c-class problem, OAR trains c binary classifiers, each separating one class from the remaining $c-1$, and classifies test examples to the class whose classifier has the highest output. There are only c classifiers, but each needs to be trained on the entire training set. In contrast, OAO constructs all possible $c(c-1)/2$ binary pairwise classifiers for the c classes and classifies examples to the class that wins the most pairwise decisions. Other strategies such as Directed Acyclic Graphs [37] and Binary Decision Trees [52, 5] can also be used.

3. Complexity and Sparsity

Although sparsity is not our focus here, like SVMs, hyperdisk classifiers have intrinsic sparsity properties. Suppose that two classes are contained in a d-dimensional affine subspace of (possibly kernelized) feature space. It is well known that the corresponding SVM can be specified in terms of at most $d+1$ support points (training examples with nonzero weights). In fact, if sparse expansion is the goal, *any* $d+1$ points that span the subspace are sufficient to express *any* classifier based on Euclidean geometry (SVM, hyperdisk, *etc*.): by using the points to define an orthonormal basis for the subspace as in section II.B and expressing the training and test points in terms of this, the classifier(s) can be constructed and used in explicit Euclidean coordinates. Hence, hyperdisk margin classifiers have the same ($d+1$ basis point) intrinsic complexity bound as the corresponding SVM. However unlike SVMs, they essentially always meet this bound, so in practice they are typically somewhat more complex than the corresponding SVM.

Similarly, if we count irreducible numbers of support points (ones on which the solution depends) rather than basis points, $d+1$ support points are sufficient to specify a d-dimensional hyperdisk: the hypersphere is defined by $d+1$ or fewer support points, these are affinely independent, and if necessary additional points can be included to make up the $d+1$ needed to specify the affine support. Hence, the two class hyperdisk margin classifier depends on at most $2d+2$ support points – and on just $d+1$ in the common case where the two affine supports are non-intersecting, *i.e*. of dimension d_1, d_2 with $d = d_1 + d_2 + 1$. In multiclass problems, the hyperdisk margin approach can even be sparser than SVM because every hyperdisk classifier involving a class reuses the same set of support points for that class, whereas each SVM classifier typically uses a different set of support points.

Note that while SVM support points always lie close to the separating hyperplane, this is not necessarily the case for support points on the rims of hyperdisks: they are always boundary points of the class, but not necessarily the most difficult ones to classify against the given other class. This is the price payed for the extra regularization provided by modeling class extents rather than pure discrimination. Similarly, the above bounds are for affine hulls estimated from the minimum possible numbers of support points. If we use more global strategies such as affine least squares fitting (as we do in the experiments below), the number of participating points increases in return for improved noise averaging in the affine estimate.

If greater sparsity is needed, several strategies are possible. Firstly, increasing the threshold that the affine hull eigenvalues must exceed in order to be considered significantly nonzero typically decreases the dimensionality of the affine hulls and hence the number of basis points needed. Secondly, most of the available methods [35, 40, 33, 4] for simplifying kernel SVM and KPCA classifiers (*i.e*. approximating feature space expansions $\mathbf{\Phi}\boldsymbol{\alpha}$ with more compact ones) can be applied to hyperdisk classifiers. Below we test two reduced set methods of this kind: the simple iterative subspace estimation algorithm of [40, 33], and the quadratic programming approach of [40]. (These methods only approximate the hyperplane normal / kernel expansion $\mathbf{w} = \mathbf{\Phi}\boldsymbol{\alpha}$: to complete the classifiers we also re-estimate their offsets b to minimize the error rates on their training sets).

4. Experiments

We provide illustrative tests of maximum margin classifiers[3] based on separating hyperdisks (LMC-HD), affine hulls (LMC-AH) and convex hulls (SVM) on a number of datasets. For multi-class problems we tested both the one-against-rest (OAR) and one-against-one (OAO) approaches, reporting whichever yielded the best results. By default we used the QCQP algorithm to find hyperdisk separators.

4.1. Linear (Non-kernelized) Classifiers

4.1.1. Honda/UCSD Dataset

This video-based face recognition dataset [28] contains 59 300–500 frame video sequences of 20 individ-

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

Figure 3: Honda/UCSD dataset: some detected face images from videos of two subjects.

Figure 4: Aligned images of one subject from the AR Face dataset.

Table 1: Classification Rates (%) for linear classifiers on the Honda/UCSD dataset

Method	Clean	Noisy Training	Noisy Test	Noisy Training+Test
LMC-HD	**97.4**	**97.4**	**92.3**	**89.7**
LMC-AH	**97.4**	**97.4**	**92.3**	87.2
SVM	94.9	92.3	**92.3**	82.1

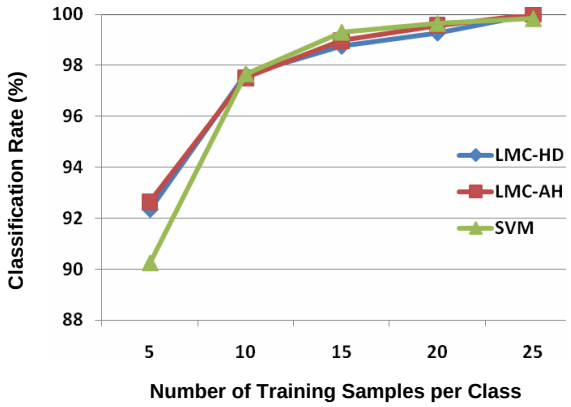

Figure 5: Classification rate (%) versus the number of training samples per class on the AR Face dataset.

uals. One video per person is designated for training, with the remaining 39 for testing. We use a multi-image face recognition setting where each probe is a set of face images taken from a single video and the most similar set in the training data must be determined. To find the faces we ran the OpenCV Viola-Jones face detector [51] on each video, adding all of the detected face regions to the set after resizing them to 40×40 pixel gray scale images and applying histogram equalization. Some examples of detections are shown in Fig. 3.

For this experiment we use linear classifiers, with the Newton root finding algorithm for the hyperdisk distance computations. Affine hull dimensions were set to retain enough eigenvalues to account for 98% of the total energy in the eigen-decomposition. To quantify the robustness to outliers, we give classification rates for both clean and noisy training and test sets. The noisy examples were created by augmenting each training and/or test example with two random images from each other class ($2(c-1)$ in all).

The results are given in Table I. The hyperdisk method has the best or best equal accuracy in all of the tests, but the affine hull method is a close second. The SVM has the worst accuracy. All of the methods are relatively robust to noise in the training samples, but significantly more sensitive to noise in the test examples.

4.1.2. AR Face Dataset

The AR Face dataset [32] provides frontal views of 126 subjects, with two groups of 13 images per subject recorded in two sessions spaced by 14 days. The 13 images contain different facial expressions, illumination conditions and occlusions. For this experiment we randomly selected 20 male and 20 female subjects, aligning, downscaling and cropping the images to 105×78 pixels so that the centres of the two eyes fall at fixed coordinates. Some examples of the results are shown in Fig. 4. Raw pixel values were used as features. For training we randomly selected $n = 5, 10, 15, 20, 25$ images of each individual, keeping the remaining $26 - n$ for testing. This process was repeated 15 times, with the final classification rates being averages over the 15 rounds.

The results are shown in Fig. 5. When 5 samples per class are used, the affine hull and hyperdisk classifiers yield respectively 92.6% and 92.3% accuracy, sig-

Table 2: Low-dimensional datasets from the UCI Repository

Dataset	# Classes	# Examples	Dimension
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

nificantly (with 95% confidence) outperforming SVM, which yields 90.2%. With 10 or more samples per class, the performance rapidly saturates and all of the methods provide similar classification rates.

4.2. Experiments with Kernelized Classifiers

We tested the kernelized versions of the methods on the Volatile Organic Compound identification (VOC), Caltech-4 visual object recognition, and United States Postal Service (USPS) handwritten digit datasets and on eight lower-dimensional datasets from the UCI repository[4]: 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 sizes of the UCI problems are summarized in Table II. Gaussian kernels, $k(\mathbf{x}, \mathbf{y}) = exp(-\|\mathbf{x} - \mathbf{y}\|^2/(2\sigma^2)$, were used in all cases. The parameters were set using random training/test partitions for the VOC, Caltech-4 and UCI datasets and the designated validation set for USPS.

4.2.1. Volatile Organic Compound Dataset

The Volatile Organic Compound (VOC) dataset[5] comes from the real-world application of determining the identity of volatile organic compounds detected by an array of six quartz crystal microbalance sensors. There are 384 samples and five VOCs of interest: ethanol, octane, toluene, xylene and trichloroethylene. The small size of the training set and the fact that the identification needs to be made independently of the VOC concentration, on which the individual sensor signals are strongly dependent, make the problem more challenging.

The results are given in Table III. The Gaussian kernel width is set to $\sigma = 0.4$. The asterisks indicate

Table 3: Classification Rates (%) on the VOC Dataset

Method	Classification Rate
LMC-HD	**95.3** ± 1.5
LMC-AH	92.1* ± 3.6
SVM	93.5* ± 1.7

Table 4: Classification Rates (%) on the Caltech-4 dataset.

Method	Classification Rate
LMC-HD	**76.9** ± 4.3
LMC-AH	73.0* ± 4.3
SVM	74.2 ± 4.8

performance differences that are statistically significant at 95% level between the given method and the corresponding best result in bold. Based on the results, the hyperdisk classifier is the best, followed by the SVM. The SVM and affine hull methods both have lower performance than the hyperdisk one at 95% significance.

4.2.2. Caltech-4 Dataset

Caltech-4[6] is a visual object recognition dataset containing 2876 images from four categories: airplanes, cars, faces and motorcycles – *c.f.* Fig. 6. The largest class, airplanes, has 1074 samples while the smallest one, faces, has 450. The images have significant background clutter and intra-class and scale variability, so we use a "bag of features" representation that does not require geometric alignment of the targets. SIFT [29] descriptors (local histograms of gradient orientation on image patches extracted at multiscale Difference of Gaussian interest points) are vector quantized against a codebook of 1000 visual words obtained by clustering training descriptors using k-means, and the resulting codes are histogrammed over the image to make a 1000-D descriptor vector. We use 5-fold cross-validation over a random partition of the images of each class to evaluate the performance, averaging over all five choices of 4 fold for training and the remaining 1 for testing.

The results are summarized in Table IV. The best Gaussian kernel parameter is found as $\sigma = 2$. The hyperdisk method gives the best results, followed by SVM. The affine hull method gives the worst accuracy, and the proposed hyperdisk method significantly outperforms the affine hull method at 95% confidence level.

[4] *http://archive.ics.uci.edu/ml*

[5] *http://users.rowan.edu/~polikar/RESEARCH/vocdb.html*

[6] *http://www.vision.caltech.edu/html-files/archive.html*

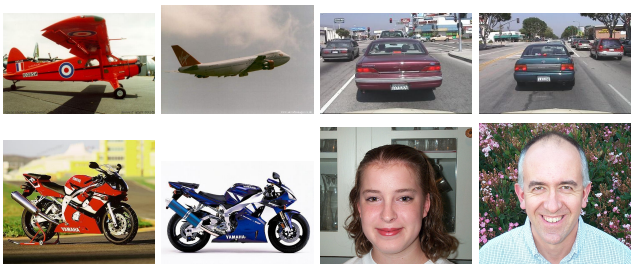

Figure 6: Some examples of images from the Caltech-4 dataset.

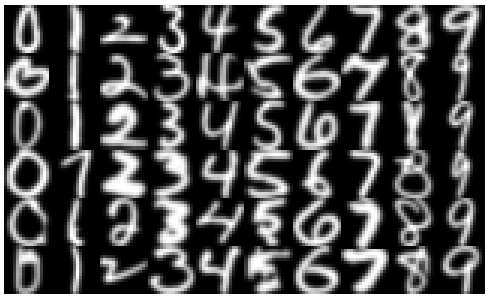

Figure 7: Some examples of images from the USPS dataset.

Table 5: Classification Rates (%) on the UCI Datasets.

Dataset	LMC-HD	LMC-AH	SVM
Ionosphere, $\sigma = 0.90$	**94.0** ± 3.1	93.7 ± 3.4	92.9 ± 3.2
Iris, $\sigma = 2.00$	**96.7** ± 2.3	94.7* ± 2.9	95.3 ± 3.8
IS, $\sigma = 0.35$	**97.2** ± 0.3	95.3* ± 0.7	97.1 ± 0.4
LR, $\sigma = 3.00$	**97.5** ± 0.4	**97.5** ± 0.4	96.5* ± 0.3
MF, $\sigma = 5.00$	**98.3** ± 0.5	**98.3** ± 0.5	98.0* ± 0.4
PID, $\sigma = 40.00$	**78.4** ± 1.3	77.5 ± 1.9	78.1 ± 1.7
Wine, $\sigma = 2.00$	**98.8** ± 1.6	**98.8** ± 1.6	98.2 ± 1.6
WDBC, $\sigma = 5.00$	97.0 ± 0.5	96.0* ± 0.8	**97.4** ± 0.9

Table 6: Classification Rates (%) on the USPS dataset.

Method	Classification Rate
LMC-HD	**95.7**
LMC-AH	**95.7**
SVM	95.4

4.2.3. *UCI Repository Datasets*

The results obtained on the eight UCI datasets and the chosen Gaussian kernel parameters are summarized in Table V. The differences in accuracy are modest, but the hyperdisk method is still the best classifier tested on four of the sets, and the best equal with the affine one on three more. The SVM outperforms the hyperdisk classifier on just one dataset, WDBC, while it outperforms the affine one on four and is outperformed by the affine one on four. For the results marked with asterisks – two sets for SVM and three for the affine method – the hyperdisk method outperforms the given classifier at at least 90% confidence level. SVM never outperforms the hyperdisk method at this level, even on WDBC.

4.2.4. *USPS Dataset*

The USPS dataset[7] contains 9298 16×16 gray-scale images of handwritten digits, with 7291 reserved for training and validation and the remaining 2007 for testing. Some samples are shown in Fig. 7. Many sophisticated methods have been applied to this dataset, but our aim here is only to evaluate the relative performance of the three basic methods that we test so we use raw gray-scale pixels as features, without any preprocessing or feature extraction. The results are therefore most directly comparable to older work such as [38] for Radial Basis Function Neural Network classifiers. The parameters of the methods were set using the designated validation set, then the methods were retrained on the combined training and validation sets and tested on the test set. The best Gaussian kernel width is found to be equal to $\sigma = 5$. Table VI summarizes the results: the hyperdisk and affine hull methods are equal best, with the SVM only slightly behind.

4.2.5. *Sparsification using Reduced Set Methods*

Finally, we test the effect of applying the reduced set methods of [40, 33] to sparsify the hyperdisk classifier and hence improve its run time. Both reduction methods gave similar results, so here we only report results for the subspace estimation one [33]. For the experiment we take the two most confused classes from the USPS dataset – the digits '4' and '6'. Together the two classes contain 1214 training samples and 326 test samples. We use kernelized classifiers with the global USPS parameter settings. The kernel has the same width for all of the methods.

For unreduced methods, the hyperdisk and affine hull approaches again come out top equal with 96.6% classification rate, with SVM second at 96.0%. Fig. 8 summarizes the results of applying the reduced set method for various levels of sparsity. The full SVM (96.0% accuracy) has 261 support vectors, while the reduced

[7] Retrieved from *ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data*.

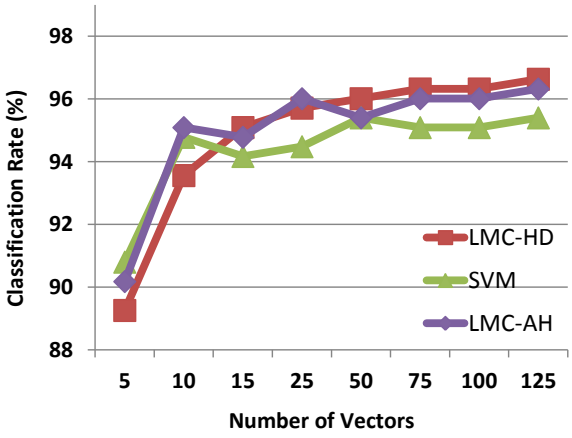

Figure 8: Classification rates (%) on the USPS two digit subset for the reduced set large margin classifiers under various levels of sparsity (numbers of support vectors).

set hyperdisk method still has unchanged performance (96.6% accuracy) with 125, and 95.7% accuracy with only 25. Hence, in this experiment the hyperdisk method still outperforms full SVM, even when it is significantly sparser than it and hence faster to run.

4.3. Statistical Comparisons of Classifiers Over All Tested Data Sets

A statistical comparison of the classification accuracies between the proposed hyperdisk method and other large margin classifiers, namely LMC-AH and SVM, is provided in Table VII. Comparison is done over all tested data sets except for the Honda/UCSD since the experimental setup for this database is different than the others (Classification accuracies of image sets are computed rather than image samples for Honda/UCSD). For the AR face database, we take the average of all different test cases as the final classification accuracy. Overall the hyperdisk method performs better than the affine hull method on 7 out of 11 data sets, and the affine hull method outperforms the hyperdisk method only on one data set. Similarly, the hyperdisk method outperforms the SVM on 10 out of 11 data sets, and SVM beats it only once. In general, the proposed hypedisk method significantly performs better than other tested large margin classifiers. We justify this by conducting a two-tailed Wilcoxon signed rank test over all experimental results as presented in Table VII. Note that the null hypothesis is rejected at a significance level $\alpha = 0.01$ when the hyperdisk method is compared to the affine hull method, and it is rejected at a significance level $\alpha = 0.05$ when the hyperdisk method is compared to SVM. These results indicate that the hyperdisk method is significantly better than the affine hull and SVM methods.

5. Summary and Conclusions

Classifiers based on the maximum margin separators of the convex hulls (SVM) and the affine hulls (LMC-AH) of the training samples of the classes have proven very successful. Here we investigated large margin classification based on an alternative convex model of the classes, the minimal bounding hyperdisk, *i.e.* the intersection of the affine hull and the minimal bounding hypersphere of the samples. We showed how to construct such hyperdisk models from samples, and we introduced two algorithms for finding the closest pair of points between two hyperdisks and hence for finding their maximum margin separating hyperplane. The first finds an orthonormal parametrization and reduces the problem to 2D Newton root finding. The second formulates the problem as a convex Quadratically Constrained Quadratic Program. The method was also kernelized to allow more flexible classifiers.

We argued that in high dimensions, the convex hull of the training samples (the class approximation used by SVM) is typically a significant underestimate of the true extent of the class. Hyperdisk models are comparatively simple and compact supersets of the convex hull model, and subsets of the affine hull one. We argued that this intermediate size allows them to counteract some of the ill effects of class underestimation in the SVM, and hence to return better classification performance in many problems. Our experiments supported this hypothesis, with the hyperdisk classifiers producing the best or best equal results overall on all of the datasets tested, often outperforming SVM to a statistically significant extent whereas the converse never occurred.

On the negative side, the hyperdisk classifiers are in practice less sparse, and hence somewhat slower to train and run, than the corresponding SVM. However, when we consider the real-time efficiency, the difference between run times of SVM and the proposed method is not so big especially in high-dimensional spaces with limited number of samples. This is due to the fact that most of the training samples become support vectors in such cases. For example, when 5 samples per class are used in AR database, SVM algorithm returns 198 support vectors out of 200 samples. Similarly, when 25 samples are used, 864 training examples become support vectors out of 1000. For Caltech-4, SVM algorithm returns 1862 support vectors for 2296 training samples. Thus,

Table 7: Pairwise classification comparison between the proposed method and related margin classifiers based on Wilcoxon signed rank test.

Classifier Pair	Hypothesis Test
LMC-HD versus LMC-AH	Null Hypothesis H_0: The LMC-HD is equivalent to LMC-AH Alternative Hypothesis H_1: The LMC-HD is significantly better than LMC-AH Wilcoxon Signed Rank Test: $R^+ = 35$, $R^- = -1$, and $T = 1$. Null hypothesis is rejected at significance level $\alpha = 0.01$.
LMC-AH versus SVM	Null Hypothesis H_0: The LMC-HD is equivalent to SVM Alternative Hypothesis H_1: The LMC-HD is significantly better than SVM Wilcoxon Signed Rank Test: $R^+ = 72$, $R^- = -6$, and $T = 6$. Null hypothesis is rejected at significance level $\alpha = 0.05$.

the run times of SVM and the proposed method are similar. But, when the dimensionality is small and the training set size is large relative to the dimension, SVM returns less support vectors, which in turn offers an advantage over the proposed method. Slower test time of the proposed method is largely an algorithmic issue: our current algorithms do not do enough to reduce the number of support points involved to the strict minimum, and we are currently working on rectifying this. However there is also an intrinsic difference: even though it actually has the same overall complexity bound as SVM in terms of intrinsic dimensionality / numbers of support vectors, the hyperdisk method generically attains this bound whereas SVM often does not. To overcome this, reduced set approximations can be used and we gave one experiment where this strategy was successful. We are currently working on ways to reformulate the QCQP method so that it returns sparser solutions.

Acknowledgments

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

References

[1] F. Alizadeh and D. Goldfarb, 2003. Second-order cone programming, Mathematical Programming, vol. 95, pp. 3-51.

[2] K. P. Bennett, 1999. Combining support vector and mathematical programming methods for classification, Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, pp. 307-326.

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

[4] C. J. C. Burges, 1996. Simplified support vector decisions, International Conference on Machine Learning.

[5] H. Cevikalp, 2010. New Clustering Algorithms for the Support Vector Machine Based Hierarchical Classification, Pattern Recognition Letters, vol. 31, pp. 1285-1291.

[6] H. Cevikalp, 2011. Large margin classifier based on hyperdisks, International Conference on Machine Learning and Applications.

[7] H. Cevikalp and B. Triggs, 2009. Large Margin Classifiers Based on Convex Class Models, International Conference on Computer Vision Workshops.

[8] H. Cevikalp, B. Triggs and R. Polikar, 2008. Nearest hyperdisk methods for high-dimensional classification, International Conference on Machine Learning.

[9] H. Cevikalp, D. Larlus, M. Neamtu, B. Triggs and F. Jurie, 2010. Manifold based local classifiers: linear and nonlinear approaches, Journal of Signal Processing Systems, vol. 61(1), pp. 61-73.

[10] H. Cevikalp, B. Triggs, H. S. Yavuz, Y. Kucuk, M. Kucuk and A. Barkana, 2010. Large margin classifiers based on affine hulls, Neurocomputing, vol. 73, pp. 3160-3168.

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

[12] D. J. Crisp and C. J. Burges, 1999. A geometric interpretation of ν-SVM classifiers, Neural Information Processing Systems.

[13] Z. Deng, F. L. Chung and S. Wang, 2007. A new minimax probability based classifier using fuzzy hyper-ellipsoid, Proc. of International Joint conference on Neural Networks.

[14] C. Ding, D. Zhou, X. He and H. Zha, 2006. R1-pca: Rotational invariant l1-norm principal component analysis for robust subspace factorization, International Conference on Machine Learning.

[15] R.-E. Fan, P.-H. Chen and C.-J. Lin, 2005. Working set selection using second order information for training SVM, Journal of Machine Learning Research, vol. 6, pp. 1889-1918.

[16] G. Fung and O. L. Mangasarian, 2001. Proximal support vector machine classifiers, Proc. of Knowledge Discovery and Data Mining.

[17] M. B. Gulmezoglu, V. Dzhafarov and A. Barkana, 2001. The common vector approach and its relation to principal component analysis, IEEE Trans. Speech Audio Proc., vol. 9, pp. 655-662.

[18] I. Guyon, J. Weston, S. Barnhill and V. Vapnik, 2002. Gene selection for cancer classification using support vector machines, Machine Learning, vol. 46, pp. 389-422.

[19] G. E. Hinton, P. Dayan, and M. Revow, 1997. Modeling the manifolds of images of handwritten digits, IEEE Trans. on Neural Networks, vol. 18, pp. 65-74.

[20] T. Joachims, 1999. Making large-scale support vector machine learning practical, Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, pp. 185-208.

[21] D. Kang, J. Park and J. C. Principe, 2010. Binary classification based on SVDD projection and nearest neighbors, International Joint Conference on Neural Networks.

[22] Q. Ke and T. Kanade, 2005. Robust L1 norm factorization in the presence of outliers and missing data by alternative convex programming, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[23] V. Kecman and I. Hadzic, 2000. Support vectors selection by linear programming, International Joint Conference on Artificial Neural Networks.

[24] J. Laaksonen, 1997. Subspace classifiers in recognition of handwritten digits, Ph. D. thesis.

[25] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya and M. I. Jordan, 2003. A robust minimax approach to classification, Journal of Machine Learning Research, vol. 3, pp. 555-582.

[26] S. Lazebnik, C. Schmid and J. Ponce, 2005. A maximum entropy framework for part-based texture and object recognition, International Conference on Computer Vision.

[27] D. Lee and J. Lee, 2006. Domain described support vector classifier for multi-classification problems, Pattern Recognition, vol. 40, pp. 41-51.

[28] K. C. Lee, J. Mo, M. H. Yang and D. Kriegman, 2003. Video-based face recognition using probabilistic appearance manifolds, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[29] D. Lowe, 2004. Distinctive image features from scale invariant keypoints, International Journal of Computer Vision, vol. 60, pp. 91-110.

[30] O. L. Mangasarian and D. R. Musicant, 2001. Lagrangian support vector machines, Journal of Machine Learning Research, vol. 1, pp. 161-177.

[31] J. S. Marron, M. Todd and J. Ahn, 2007. Distance weighted discrimination, Journal of the American Statistical Association, vol. 102, pp. 1262-1271.

[32] A. M. Martinez and R. Benavente, 1998. The AR Face Database, Technical Report, Computer Vision Center, Barcelona, Spain.

[33] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz and G. Ratsch, 1999. Kernel pca and de-noising in feature spaces, Neural Information Processing Systems (NIPS).

[34] T. Mu and A. K. Nandi, 2009. Multiclass classification based on extended support vector data description, IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 39, pp. 1206-1216.

[35] S. Narayanan, 2009. Support Vector Machine Approximation using Kernel PCA, Technical Report, EECS Department, University of California, Berkeley.

[36] J. Platt, 1999. Fast training of support vector machines using sequential minimal optimization, Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, pp. 185-208.

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

[38] B. Schölkopf, K.-K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, 1997. Comparing support vector machines with Gaussian kernels to Radial Basis Function classifiers, IEEE Transactions on Signal Processing, vol. 45, pp. 2758-2765.

[39] B. Schölkopf, A. J. Smola and K. R. Müller, 1998. Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation, vol. 10, pp. 1299-1319.

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

[41] B. Schölkopf, A. J. Smola, R. C. Williamson and P. L. Bartlett, 2000. New support vector algorithms, Neural Computation, vol. 12, pp. 1207-1245.

[42] S. Shalev-Shwartz, Y. Singer and N. Srebro, 2007. Pegasos: Primal estimated sub-gradient solver for SVM, International Conference on Machine Learning.

[43] J. A. K. Suykens and J. Vandewalle, 1999. Least squares support vector machine classifiers, Neural Processing Letters, vol. 9, pp. 293-300.

[44] C.-M. Tang and J.-B. Jian, 2008. A sequential quadratically constrained quadratic programming method with an augmented Lagrangian line search function, Journal of Computational and Applied Mathematics, vol. 220, pp. 527-547.

[45] D. M. J. Tax and R. P. W. Duin, 2004. Support vector data description, Machine Learning, vol. 54, pp. 45-66.

[46] M. E. Tipping, 2001. Sparse Bayesian learning and the relevance vector machine, Journal of Machine Learning Research, vol. 1, pp. 211-244.

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

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

[49] L. Vandenberghe and S. Boyd, 1996. Semidefinite programming, SIAM Review, vol. 38, pp. 49-95.

[50] P. Vincent and Y. Bengio, 2001. K-local hyperplane and convex distance nearest neighbor algorithms, Advances in Neural Information Processing Systems.

[51] P. Viola and M. Jones, 2004. Robust real-time face detection, International Journal of Computer Vision, vol. 57, pp. 137-154.

[52] V. Vural and J. G. Dy, 2004. A hierarchical method for multi-class support vector machines, International Conference on Machine Learning.

[53] J. Wang, P. Neskovic and L. N. Cooper, 2005. Pattern classification via single spheres, Lecture Notes in Computer Science, vol. 3735/2005, pp. 241-252.

[54] S. Zafeiriou, A. Tefas and I. Pitas, 2007. Minimum class variance support vector machines, IEEE Transactions on Image Processing, vol. 16, pp. 2551-2564.

[55] W. Zhou, L. Zhang and L. Jiao, 2002. Linear programming support vector machines, Pattern Recognition, vol. 35, pp. 2927-2936.