SVM-based Nonparametric Discriminant Analysis, An Application to Face Detection

Rik Fransens[1], Jan De Prins[1], Luc Van Gool[1,2]
[1]ESAT-PSI, University of Leuven, Belgium
[2]Computer Vision Group (BIWI), ETH Zuerich, Switzerland
{rik.fransens, jan.deprins}@esat.kuleuven.ac.be, vangool@vision.ee.ethz.ch

Abstract

Detecting the dominant normal directions to the decision surface is an established technique for feature selection in high dimensional classification problems. Several approaches have been proposed to render this strategy more amenable to practice, but they still show a number of important shortcomings from a pragmatic point of view. This paper introduces a novel such approach, which combines the normal directions idea with Support Vector Machine classifiers. The two make a natural and powerful match, as SVs are located nearby, and fully describe the decision surfaces. The approach can be included elegantly into the training of performant classifiers from extensive datasets. The potential is corroborated by experiments, both on synthetic and real data, the latter on a face detection experiment. In this experiment we demonstrate how our approach can lead to a significant reduction of CPU-time, with neglectable loss of classification performance.

1. Introduction

In linear discriminant analysis (LDA) we are interested in those linear features which reduce the dimensionality and simultaneously preserve class seperability. Suppose we are dealing with D-dimensional data $\mathbf{x} = [x_1\, x_2\, \ldots\, x_D]^T$ which are drawn from a set of c classes with labels ω_i and respective probability density functions $p(\mathbf{x}|\omega_i)$ and priors $P(\omega_i)$, i=1,...,c. We are now looking for $d < D$ features represented by the linearly independent column-vectors $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_d$ which transform the datum $\mathbf{x}$ to a new feature vector $\mathbf{y} = [\boldsymbol{\phi}_1^T\mathbf{x}\ \boldsymbol{\phi}_2^T\mathbf{x}\ \ldots\ \boldsymbol{\phi}_d^T\mathbf{x}]^T = [y_1\, y_2\, \ldots\, y_d]^T$. This transformation can be expressed by

$$\mathbf{y} = \boldsymbol{\Phi}^T\mathbf{x}\,, \tag{1}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1\, \boldsymbol{\phi}_2\, \ldots\, \boldsymbol{\phi}_d]$ is a $(D \times d)$-rectangular matrix.

To quantify the notion of class seperability, we have to make a connection with classification theory. According to Bayes' decision rule (MAP-rule), an observation $\mathbf{x}$ is assigned to the class for which the a-posteriori probability $p(\omega_i|\mathbf{x})$ is maximal. The resulting classification error is called the Bayes-error, and for the 2-class problem it is given by

$$\varepsilon_X = P(\omega_2)\int_{R_1} p(\mathbf{x}|\omega_2)d\mathbf{x} + P(\omega_1)\int_{R_2} p(\mathbf{x}|\omega_1)d\mathbf{x}\,, \tag{2}$$

in which R_1 and R_2 are the regions where the MAP-rule decides for ω_1 and ω_2, respectively. No other decision rule can yield a smaller probability of error. The dimension reduction achieved by (1) necessarily results in loss of information. Therefore, the best we can hope for is that the error of classification in Y-space does not exceed the error in the original X-space. More specifically, we can evaluate the *discriminant effectiveness* of the transformation $\boldsymbol{\Phi}$ by the ratio $\eta = \varepsilon_Y/\varepsilon_X$. This ratio is larger or equal to 1 and expresses the increase of the probability-of-error of the best possible classifier after transforming the data to a lower dimensional subspace. The *intrinsic discriminant dimension* [6] is the lowest possible dimension of a subspace wherein the same classification accuracy can be obtained as in the original space, i.e. with associated $\eta = 1$. The objectives of LDA are now twofold: based on a sample of the underlying distributions, (*i*) predict the intrinsic discriminant dimension of the problem, and (*ii*) extract a minimal set of features with maximal discriminant effectiveness.

In this paper, we propose a new algorithm for feature extraction based on the decision boundary. The algorithm provides an estimate of the intrinsic discriminant dimension and extracts the necessary feature vectors. It starts off from ideas laid out by Fukunaga *et al.* [4] [5] and further developed by Lee *et al.* [6]. Lee *et al.* introduce the *decision boundary feature matrix* $\boldsymbol{\Sigma}_{DBFM}$ which allows to compute the intrinsic discriminant dimensionality and the associated optimal linear transformation. However, the currently available algorithms for estimating $\boldsymbol{\Sigma}_{DBFM}$ cannot be applied to large scale learning problems (e.g. appearence based object detection) which often involve hundreds-of-thousands of training examples. We propose that Support Vector Machines (SVMs) offer a natural framework to estimate $\boldsymbol{\Sigma}_{DBFM}$. Furthermore, by means of the bootstrap learning procedure introduced in [9], an arbitrary amount

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

of training data can be processed. Note that our approach should not be confused with the so-called *kernel discriminant analysis*, which incorporates kernel tricks into classical LDA to derive a non-linear extension of the algorithm and which returns non-linear discrimative features.

The paper is organized as follows: First, we give an overview of parametric and nonparametric LDA. We discuss some pitfalls of parametric LDA and describe how these can be mended by introducing a non-parametric counter version of the algorithm. We show how SVMs provide for an elegant implementation of these ideas. In the next section, we compare our algorithm with the original Fukunaga algorithm. Both methods are extensively tested for a whole range of parameter settings and we describe an evaluation framework based on the effectiveness ratio η. In the fourth section, we demonstrate the procedure for the particular case of face detection. We end the paper with some general conclusions and a description of future work.

2. Parametric vs. Nonparametric LDA

2.1. Parametric LDA

The most popular way to find the best linear features makes use of so-called *scatter matrices*. The within-class scatter matrix specifies the spread of the samples around their respective class expected vectors, and is defined as

$$\begin{aligned} \mathbf{\Sigma}_w &= \sum_{i=1}^{c} P(\omega_i) E[(\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)^T] \\ &= \sum_{i=1}^{c} P(\omega_i)\mathbf{\Sigma}_i \ , \end{aligned} \tag{3}$$

where $\boldsymbol{\mu}_i$ and $\mathbf{\Sigma}_i$ are the expected vector and covariance matrix of the ith class, respectively. The between-class scatter matrix, on the other hand, specifies the spread of the individual class means around the overall mixture mean $\boldsymbol{\mu}_0$, and is defined as

$$\mathbf{\Sigma}_b = \sum_{i}^{c} P(\omega_i)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_0)^T \ , \tag{4}$$

One possible way to achieve class separability of the transformed data $\mathbf{y}$ is to demand that the class means in Y-space are well separated while simultaneously the spread of the data around their respective class means is small, i.e. the classes remain compact. To this end we can define a criterion function that attains its maximal value when both demands are met. A typical criterion is the following:

$$J = tr(\widehat{\mathbf{\Sigma}}_{wY}^{-1}\widehat{\mathbf{\Sigma}}_{bY}) \ , \tag{5}$$

where the subscript Y indicates that the sample scatter matrices are computed in Y-space and $tr(.)$ represents the trace operation. We can now express this function in X-space as follows:

$$J = tr((\mathbf{\Phi}^T\widehat{\mathbf{\Sigma}}_{wX}\mathbf{\Phi})^{-1}(\mathbf{\Phi}^T\widehat{\mathbf{\Sigma}}_{bX}\mathbf{\Phi})) \ . \tag{6}$$

It is easy to show that J is maximized by using the first d eigenvectors of $\widehat{\mathbf{\Sigma}}_{wX}^{-1}\widehat{\mathbf{\Sigma}}_{bX}$, corresponding to the highest d eigenvalues, as the columns of $\mathbf{\Phi}$. This result is the same as obtained by classical Fisher LDA.

One of the reasons of the popularity of this approach is the fact that the optimum is analytically defined. There are, however, several problems associated with the procedure. First of all, because the sample between-class scatter matrix $\widehat{\mathbf{\Sigma}}_{bX}$ is derived from only c class-means, its rank and therefore also the rank of $\widehat{\mathbf{\Sigma}}_{wX}^{-1}\widehat{\mathbf{\Sigma}}_{bX}$ is maximally $c-1$. This implies that at most $c-1$ eigenvalues of $\widehat{\mathbf{\Sigma}}_{wX}^{-1}\widehat{\mathbf{\Sigma}}_{bX}$ are non-zero and the discriminant subspace computed in this way will have at most $c-1$ dimensions. Second, if a class has a mean vector very different from the mean vectors of the other classes, that class dominates $\mathbf{\Sigma}_{bX}$, which results in ineffective feature extraction [6].

2.2. Nonparametric LDA

The forementioned problems can be overcome by introducing a non-parametric between-class scatter matrix, which measures between-class scatter on a local basis in the neighborhood of the decision boundary. We first briefly review some basic properties of discriminant subspaces, for a more complete description and proofs we refer to [6].

Property 1 *If a vector is parallel to the tangent hyper plane to the optimal decision boundary S at every point on S, this vector is discriminantly redundant. In any other case, the vector is discriminantly informative.*

This property can be explained as follows: if a vector $\boldsymbol{\phi}$ is parallel to the tangent hyper plane to S at every point of S, then moving any point $\mathbf{x}$ in the direction of $\boldsymbol{\phi}$ can never move the point across the boundary, i.e. can never influence the classification outcome. Hence $\boldsymbol{\phi}$ is discriminantly redundant. It is easy to see that the discriminant effectiveness of a vector is proportional to the area of the optimal decision boundary that has a normal pointing in the same direction. The decision boundary feature matrix is now defined as

$$\mathbf{\Sigma}_{DBFM} = \frac{1}{C}\int_S \mathbf{N}(\mathbf{x})\mathbf{N}(\mathbf{x})^T p(\mathbf{x}) d\mathbf{x} \ , \tag{7}$$

where $\mathbf{N}(\mathbf{x})$ is the unit normal vector to the optimal decision boundary S at point $\mathbf{x}$, $p(\mathbf{x})$ is the data density, $C = \int_S p(\mathbf{x})d\mathbf{x}$ and all integrations are performed over the boundary. This matrix can be seen as the covariance structure of the boundary surface normal vectors where the density on S is defined by $p(\mathbf{x})/\int_S p(\mathbf{x})d\mathbf{x}$. Notice that the decision boundary feature matrix characterises the between-class scatter in the neighborhood of the decision boundary. The matrix has some interesting properties:

Property 2 *The rank r of the decision boundary feature matrix equals the intrinsic discriminant dimension of the classification problem.*

Property 3 *The eigenvectors of the decision boundary feature matrix corresponding to the r non-zero eigenvalues are the necessary feature vectors for the optimal transformation, i.e. the transformation with effectiveness ratio $\eta = 1$.*

Unfortunately, in general it is not possible to compute $\mathbf{\Sigma}_{DBFM}$. First of all, it relies on the optimal decision boundary for which accurate estimations of the a posteriori class probabilities are required. These are hard to obtain, especially when the dimensionality of the problem increases. Second, we need to integrate the density $p(\mathbf{x})$ over the optimal decision surface. For a D-dimensional classification problem this surface is a $(D-1)$-dimensional manifold which is only implicitly described. Except for very simple and low-dimensional problems, integration is impossible. Nonparametric LDA relies on sample-based approximations of the decision boundary feature matrix.

In the approach of Fukunaga *et al.* [4], the normal vectors on the boundary are approximated by the directions of the lines that connect points from one class to points in the other class. More specifically, every point from one class is connected to the average of its k nearest neighbors in the other class. A non-parametric scatter matrix $\mathbf{S}_{bk}$ is formulated as

$$\begin{aligned}\mathbf{S}_{bk} &= \frac{1}{N}\sum_{i=1}^{N_1} w_i(\mathbf{x}_i - M_2^k(\mathbf{x}_i))(\mathbf{x}_i - M_2^k(\mathbf{x}_i))^T \\ &\quad + \frac{1}{N}\sum_{i=1}^{N_2} w_i(\mathbf{x}_i - M_1^k(\mathbf{x}_i))(\mathbf{x}_i - M_1^k(\mathbf{x}_i))^T ,\end{aligned} \tag{8}$$

where N_1 and N_2 are the number of samples of both classes, N=N_1+N_2 is the total number of samples, $M_j^k(\mathbf{x}_i)$ is the average of the k nearest neighbors in class ω_j to a point $\mathbf{x}_i$ and w_i is a weight. The weighting function is defined as

$$w_i = \frac{\min\{\|\mathbf{x}_i - \mathbf{x}_{kNN}^{(1)}\|^\alpha, \|\mathbf{x}_i - \mathbf{x}_{kNN}^{(2)}\|^\alpha\}}{\|\mathbf{x}_i - \mathbf{x}_{kNN}^{(1)}\|^\alpha + \|\mathbf{x}_i - \mathbf{x}_{kNN}^{(2)}\|^\alpha} , \tag{9}$$

where $\mathbf{x}_{kNN}^{(j)}$ is the *k*-th nearest neighbor of the point $\mathbf{x}_i$ in class ω_j, $\|.\|$ is the Euclidean norm and α is an integral power. The purpose of this weight is to de-emphasize the contribution of samples that lie far from the classification boundary. For a sample far from the boundary, the distance to the *k*-th NN in its own class is much smaller than the distance to the *k*-th NN in the other class, resulting in a small value for w_i. Note that the algorithm has a combinatorial nature, because it needs to compute the distance between all pairs of points of both classes. Furthermore, it needs to store the connecting vector between every element of either class and the average of its k NNs in the other class. For a typical pattern recognition problem in computer vision (e.g. face detection), the 'non-object' class is the rest of the world and the learning machine needs to process hundreds-of-thousands negative examples to represent it well. Applying the Fukunaga algorithm in this context is not possible due to the size of the problem.

In the approach of Lee *et al.* [6] a quadratic decision surface, based on the class-covariance matrices, is computed. Next, this surface is intersected with all lines that connect a point from one class to a point from the other class. Finally, the decision boundary normal vectors are computed as the normalized gradient of the decision surface evaluated at the intersection points.

2.3. SVM-based Nonparametric LDA

In this paragraph we explain how SVMs can be used to estimate the decision boundary feature matrix. We restrict ourselves to the 2-class problem, an extension to the multi-class case is given at the end of this paragraph.

Suppose we have a set of training data $(\mathbf{x}_1, y_1)$, ... , $(\mathbf{x}_N, y_N)$ where $\mathbf{x}_i$ is a D-dimensional vector with class label $y_i \in \{-1, +1\}$. Suppose both classes can be separated by a hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$. From all planes that separate the classes, the SVM algorithm chooses the one with maximal margin [1]. The classification function is given by

$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N_{sv}} \lambda_i y_i \mathbf{x_i}\mathbf{x} + b) , \tag{10}$$

where the summation is performed over the support vectors $\mathbf{x}_i$. These vectors are a subset of the training data and are called 'Support Vectors' (SVs) because they completely determine the decision plane; if they were the only data available, exactly the same solution would have been obtained. An interesting characteristic of the SVs is that they lie geometrically nearby the decision plane. If the data are not linearly separable, they are mapped from input space $\mathbb{R}^D$ to a high dimensional feature space $\mathcal{F}$ where linear separation is possible. Next, a separating hyper plane is computed in $\mathcal{F}$. The decision function becomes

$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N_{sv}} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + b) , \tag{11}$$

where $K(.)$ is a kernel function. Several kernels are possible, including radial basis functions, polynomial and sigmoid kernels. The choice of the kernel and kernel parameters (e.g. the degree of the polynomial kernel) has to be made by the user, and the optimal choices are problem dependent. Furthermore, in the optimisation procedure underlying the SVM algorithm, an error weighting constant C is introduced, which assigns a cost to the misclassification of certain examples. The value of C must also be set by the user.

From (11) we can see that the decision boundary corresponds to the level surface $s(\mathbf{x}) = 0$ of the function

$$s(\mathbf{x}) = \sum_{i=1}^{N_{sv}} \lambda_i y_i K(\mathbf{x_i}, \mathbf{x}) + b . \tag{12}$$

Since the SVs are geometrically nearby the decision surface, the gradient of $s(\mathbf{x})$, evaluated at the positions of the SVs, provides a good approximation of the normal direction of the decision boundary nearby the SVs. This property was used in [2] to define the most important pixels (MIPS) in a face-detection algorithm. We can now approximate the decision boundary feature matrix as follows:

$$\mathbf{S}_b = \frac{1}{C}\mathbf{G}\mathbf{G}^T \ , \tag{13}$$

$$\text{with} \quad \begin{array}{l} \mathbf{G} = [\nabla s(\mathbf{x}_1)\, \nabla s(\mathbf{x}_2) \ldots \nabla s(\mathbf{x}_{N_{sv}})] \\ C = \sum_{i=1}^{N_{sv}} \|\nabla s(\mathbf{x}_i)\|^2 \ . \end{array}$$

The first d eigenvectors of $\mathbf{S}_b$, corresponding to the highest d eigenvalues, are used as the columns of the transformation matrix $\mathbf{\Phi}$.

The next question is how we can choose an appropriate value for the dimension of the subspace. The sum of all eigenvalues, which equals the trace of $\mathbf{S}_b$, is referred to as the *total scatter*. The eigenvalue corresponding to a particular eigenvector can be interpreted as the amount of scatter explained by this vector, and is a measure for the importance of the feature. If the intrinsic discriminant dimension is smaller than the original dimension D, the rank of S_b should be smaller than D. However, in practice the last eigenvalues will be small but not exactly zero. In this case, we can choose the number of features such that the fraction of explained scatter relative to the total scatter exceeds a certain threshold, say 99%. So, we can choose for d the minimal value for which the following inequality holds:

$$\frac{\sum_{i=1}^{d} v_i}{\sum_{i=1}^{D} v_i} \geq 0.99 \ , \tag{14}$$

where the eigenvalues v_i are ordered from high to low.

This SVM-based approach has several desirable features:

- SVMs have a proven ability to separate very high dimensional data.
- Using the unnormalised gradient-vectors like in (13) is a natural weighting scheme: at locations where both classes are nearby, the gradient of $s(\mathbf{x})$ is high and as a result these regions are well represented in the approximation of $\mathbf{\Sigma}_{DBFM}$. Therefore, in the resulting discriminant subspace these regions will be relatively well preserved. Conversely, relatively less attention is paid to regions where both classes are well separated.
- Contrary to the previously described methods, we do not need to incorporate all training-examples in the approximation. Only the SVs are needed, because they fully specify the computed decision boundary.

The procedure can be easily extended to the multi-class case. If we have c classes, the one-against-the-rest method [1] is used to construct c classifiers. Here, the *i*-th classifier separates the *i*-th class from all the others. Suppose we have a total of N training examples, of which N_i belong to the *i*-th class. An approximation for $\mathbf{S}_b$ is then given by

$$\mathbf{S}_b = \sum_{i=1}^{c} \left(\frac{N_i}{N}\right)\left(\frac{1}{C_i}\mathbf{G}_i\mathbf{G}_i^T\right) \ , \tag{15}$$

where C_i and $\mathbf{G}_i$ are based on the *i*-th classifier and defined as in (13), and the fraction N_i/N is an estimate for the class prior $P(\omega_i)$. Including the class priors places more weight on the representation of the boundaries of the dominant class(es).

3. Experiments with Synthetic Data

3.1. Experimental Setup

In this section we compare the SVM-based approach with the Fukunaga-algorithm, described in section 2.2, for the 2-class case. We use the discriminant effectiveness $\eta = \varepsilon_Y/\varepsilon_X$ as a means to measure and compare the performance of both algorithms. In the experiments, we define a probability density function $p(\mathbf{x}|\omega_i)$ for each class from which we sample a (labeled) data set. Based on these samples, we compute the transformation $\mathbf{y} = \mathbf{\Phi}^T\mathbf{x}$ and we estimate the Bayes error, both in X- and Y-space, like in Eq. (2). This is done by numerical integration of the original densities $p(\mathbf{x}|\omega_i)$ and transformed densities $p(\mathbf{y}|\omega_i)$ over the MAP-defined regions R_1 and R_2. Obviously, the results will be different for every sample. Therefore, the procedure is repeated a number of times and we report the average $\bar{\eta}$.

In our experiments, we make use of mixture-of-Gaussian densities. For class ω_i, the pdf is defined as

$$p(\mathbf{x}|\omega_i) = \sum_{j=1}^{L_i} \pi_{ij} G(\mathbf{x}; \boldsymbol{\mu}_{ij}, \mathbf{\Sigma}_{ij}) \ , \tag{16}$$

where $G(.)$ is the Gaussian density function, L_i is the number of mixture components, and π_{ij}, $\boldsymbol{\mu}_{ij}$ and $\mathbf{\Sigma}_{ij}$ are the mixture proportion, the expected vector and the covariance matrix of the j-th component of the mixture density, respectively. The Gaussian density has the interesting property that it remains Gaussian under linear transformations. More specifically, given the transformation $\mathbf{y} = \mathbf{\Phi}^T\mathbf{x}$ where $\mathbf{\Phi}$ is a $(D \times d)$-rectangular matrix $(d \leq D)$ with linearly independent columns, the pdf of the transformed data becomes

$$p(\mathbf{y}|\omega_i) = \sum_{j=1}^{L_i} \pi_{ij} G(\mathbf{y};\ \mathbf{\Phi}^T\boldsymbol{\mu}_{ij}\ ,\ \mathbf{\Phi}^T\mathbf{\Sigma}_{ij}\mathbf{\Phi}) \ . \tag{17}$$

Prior to analysis, the samples are whitened. In this process, the sample within-class scatter matrix $\widehat{\mathbf{\Sigma}}_w$ is trans-

formed to the identity matrix, by applying the linear transformation $\mathbf{z} = \Lambda^{-1/2}\Psi^T\mathbf{x}$. Here $\Psi = [\psi_1 \dots \psi_D]$ is a square matrix with the eigenvectors of $\widehat{\mathbf{\Sigma}}_w$ as columns, and $\Lambda^{-1/2}$ is a matrix whose diagonal elements are the inverse square roots of the eigenvalues of $\widehat{\mathbf{\Sigma}}_w$. This normalisation step assures that all individual features in the original space get a similar metric. Note that after the extraction of the discriminant features ϕ_i, this transformation needs to be accounted for. So, the full transformation from the original X-space to the discriminant subspace becomes $\mathbf{y} = \Phi^T\Lambda^{-1/2}\Psi^T\mathbf{x}$. The Gaussian mixture densities in (16) have to be transformed likewise. We can now present the experimental procedure in its entirety:

- Define the densities $p(\tilde{\mathbf{x}}|\omega_1)$ and $p(\tilde{\mathbf{x}}|\omega_2)$ for $\tilde{\mathbf{x}} \in \mathbb{R}^2$. Compute $\varepsilon_{\tilde{X}}$ by numerical integration of (2) over a regular grid. In our experiments we used a grid of 200 by 200.

- Extend the dimensionality of the input space by complementing the feature vectors of each class with $D-2$ i.i.d. features. To that end, we choose a Gaussian distribution with zero mean and unit variance. This transforms each component of the mixture densities from $G(\tilde{\mathbf{x}}; \tilde{\boldsymbol{\mu}}_{ij}, \tilde{\mathbf{\Sigma}}_{ij})$ to $G(\mathbf{x}; \boldsymbol{\mu}_{ij}, \mathbf{\Sigma}_{ij})$ with

$$\boldsymbol{\mu}_{ij} = [\tilde{\boldsymbol{\mu}}_{ij}^T \; 0 \dots 0]^T \quad \text{and} \quad \mathbf{\Sigma}_{ij} = \begin{pmatrix} \tilde{\mathbf{\Sigma}}_{ij} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix},$$

  where the $\mathbf{0}$'s are zero matrices and $\mathbf{1}$ is a diagonal matrix with unit diagonal elements. Because the added features are identically distributed for both classes, they are discriminantly redundant. Therefore, the Bayes-error in $\mathbb{R}^D$ is the same as in $\mathbb{R}^2$, i.e. $\varepsilon_X = \varepsilon_{\tilde{X}}$. In our experiments we added 18 extra dimensions rendering X-space equal to $\mathbb{R}^{20}$.

- Sample both density functions and compute and apply the whitening transformation. We assumed equal priors and sampled 100 data points from each class.

- Compute the transformation matrix $\Phi = [\phi_1 \; \phi_2]$. Because the intrinsic dimensionallity of the problem is maximally 2, this transformation should suffice to retrieve all relevant information.

- Transform the density functions $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$ to Y-space. Compute ε_Y by numerical integration over a regular grid, and output $\eta = \varepsilon_Y/\varepsilon_X$.

3.2. Results and Discussion

We defined 3 problems, hereafter referred to as problem A, B and C. The parameters of the mixture-of-Gaussian pdf's are given in table 1 and the contour plots of the densities are shown in figure 1. Problem A is the most simple case: both classes consist of one Gaussian, and the intrinsic discriminant dimension is 1. For problems B and C the intrinsic discriminant dimension is 2.

	class 1			class 2		
	π_{1j}	$\boldsymbol{\mu}_{1j}$	$\mathbf{\Sigma}_{1j}$	π_{2j}	$\boldsymbol{\mu}_{2j}$	$\mathbf{\Sigma}_{2j}$
problem A	1	$[5\ 5]^T$	$\mathbf{\Sigma}$	1	$[6\ 5]^T$	$\mathbf{\Sigma}$
problem B	1/3	$[4\ 5]^T$	$\mathbf{\Sigma}$	1/3	$[6\ 5]^T$	$\mathbf{\Sigma}'$
	1/3	$[5\ 4]^T$	$\mathbf{\Sigma}'$	1/3	$[7\ 6]^T$	$\mathbf{\Sigma}$
	1/3	$[5\ 6]^T$	$\mathbf{\Sigma}'$	1/3	$[6\ 7]^T$	$\mathbf{\Sigma}'$
problem C	1	$[5\ 5]^T$	$\mathbf{\Sigma}''$	1/4	$[4\ 5]^T$	$\mathbf{\Sigma}$
				1/4	$[5\ 6]^T$	$\mathbf{\Sigma}'$
				1/4	$[6\ 5]^T$	$\mathbf{\Sigma}$
				1/4	$[5\ 4]^T$	$\mathbf{\Sigma}'$

$$\mathbf{\Sigma} = \begin{pmatrix} \frac{1}{9} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \quad \mathbf{\Sigma}' = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{9} \end{pmatrix} \quad \mathbf{\Sigma}'' = \begin{pmatrix} \frac{1}{9} & 0 \\ 0 & \frac{1}{9} \end{pmatrix}$$

Table 1: *The mixture-of-Gaussian parameters of both classes for problems A, B and C.*

In the SVM-based approach, we used a polynomial kernel of degree p which is defined as follows:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\mathbf{y} + 1)^p \; . \tag{18}$$

Note that for $p = 1$, this results in a linear classifier which tries to separate the data with a hyperplane. All SVMs were trained with SVM^{light} [3]. In the experiments, we computed η for several combinations of parameters. The polynomial degree p was varied from 1 to 7, while the cost factor C was set to 1, 10, 100, 1000 and 10000. For the Fukunaga algorithm, we experimented with different values for the number of nearest neighbours k, and the power in the weighting function α. Every experiment was repeated 100 times, and we report the average $\bar{\eta}$ together with a 95%-confidence interval. The results of both algorithms are shown in tables 2 and 3.

From table 2 we can see that the Fukunaga algorithm is quite robust with respect to its parameters, i.e. the differences between the solutions for different combinations of k and α are relatively small. In fact, the value of α seems to exert no influence on the solutions at all. For problems A and B, the best solutions ($\bar{\eta}$ = 1.29 and 2.44, respectively) are obtained for k = 1. This is rather counterintuitive because we would expect a more robust estimate of the normal direction for $k > 1$. For problem C, on the other hand, the best solution ($\bar{\eta}$ = 1.91) is obtained for k = 3, i.e. the optimal parameter settings seem to be problem dependent.

From table 3 we see that the SVM-based approach is sensitive with respect to its parameters. This is logical because the shape of the decision boundary is directly related to the kernel parameter p and the error weighting constant C. For example, in problem C, the best solution is obtained for p = 7 and C = 10000. Obviously, no good solution can be

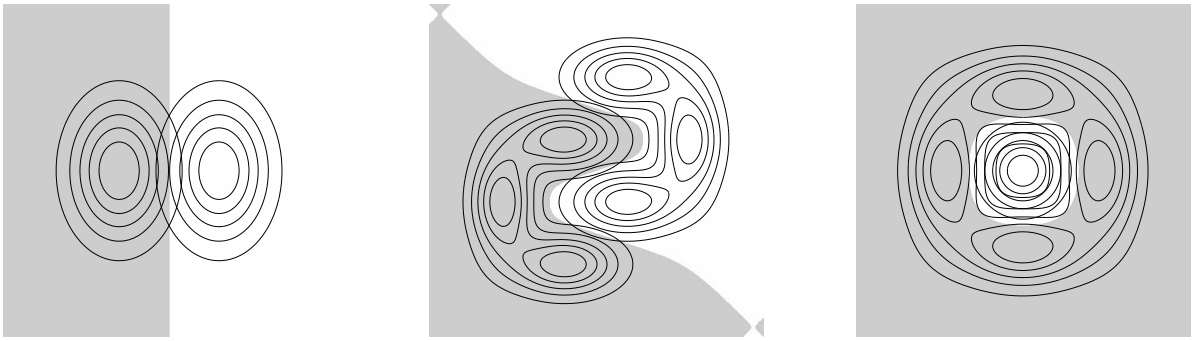

Figure 1: *Contour plots of the probability density functions for problem A (left), B (middle) and C (right). The MAP-defined regions R_1 and R_2 are shown with different shades of gray, and the optimal decision boundary is the transition between these regions. The Bayes-errors for problems A, B and C are 6.7%, 4.4% and 10.0%, respectively.*

k	α	A	B	C
1	1	1.29 ± 0.02	2.44 ± 0.09	2.08 ± 0.17
	2	1.29 ± 0.02	2.44 ± 0.09	2.08 ± 0.17
	3	1.29 ± 0.02	2.44 ± 0.09	2.09 ± 0.17
2	1	1.31 ± 0.02	2.57 ± 0.07	1.94 ± 0.14
	2	1.31 ± 0.02	2.56 ± 0.07	1.94 ± 0.13
	3	1.31 ± 0.02	2.56 ± 0.07	1.94 ± 0.13
3	1	1.31 ± 0.03	2.73 ± 0.05	1.92 ± 0.13
	2	1.31 ± 0.03	2.72 ± 0.05	1.91 ± 0.13
	3	1.31 ± 0.02	2.71 ± 0.05	1.91 ± 0.12
4	1	1.31 ± 0.02	2.82 ± 0.04	1.97 ± 0.13
	2	1.31 ± 0.02	2.82 ± 0.04	1.96 ± 0.13
	3	1.31 ± 0.02	2.82 ± 0.04	1.96 ± 0.13
5	1	1.31 ± 0.03	2.87 ± 0.03	1.98 ± 0.13
	2	1.31 ± 0.03	2.87 ± 0.03	1.97 ± 0.12
	3	1.31 ± 0.03	2.87 ± 0.03	1.97 ± 0.12
6	1	1.31 ± 0.03	2.89 ± 0.03	2.00 ± 0.12
	2	1.31 ± 0.03	2.89 ± 0.03	1.99 ± 0.12
	3	1.31 ± 0.03	2.89 ± 0.03	1.99 ± 0.12

Table 2: *Results of the Fukunaga algorithm for problems A, B and C, and for different combinations of k and α.*

expected for $p = 1$, since the optimal decision boundary is highly non-linear. Remarkably, for problem B, the best solution ($\bar{\eta} = 2.80$) is obtained for $p = 1$ and $C = 1$, i.e. from a linear classifier. Apparently, the SVM classifier was not able to discover the non-linearity in the data. Looking at the results of the non-linear kernels for problem B, we can also see that the higher the value of C, i.e. the less wrongly classified points we allow in the solution, the worse the results get. This means that if we allow the decision surface to bend, this also introduces overfitting and the net outcome, at least for problem B, is negative.

If we compare the best results of the Fukunaga algorithm with the best results of the SVM-based approach, the outcome is ambiguous. For problem A, both methods perform roughly equally well, with a slight advantage for the SVM-based approach (1.26 *vs* 1.29). For problem B, the Fukunaga algorithm performs significantly better (2.44 *vs* 2.80) but still the result is not very good. For problem C, the SVM-based approach performs best (1.76 *vs* 1.91).

4. An Application to Face Detection

4.1. Discriminant Subspace Computation

The goal of face detection is to determine whether or not there are any faces in the image, and if present, return the image location and extent of the face [7]. SVMs, which are an example of the so-called *appearence based methods*, try to separate the face class from the non-face class based an a set of training examples. However, the non-face class is not well defined and it is virtually impossible to obtain a representative sample set from it.

In [9], a bootstrap procedure was proposed to overcome this problem. The procedure starts off with a sample set $\mathcal{F}$ of facial images and an initial set $\mathcal{N}$ of non-face images. Next, a SVM is trained with these data, and the SVs originating from $\mathcal{N}$ are stored in a new set $\mathcal{N}_{SV}$. The classifier is then applied to an image database which contains no faces and all detections, which are false positives, are stored in the set $\mathcal{N}_{FP}$. Finally, $\mathcal{N}_{FP}$ and $\mathcal{N}_{SV}$ are combined to form the new non-faces training set $\mathcal{N}$ and a new SVM is trained with these data. The procedure is repeated untill a preset number of iterations is reached, or the fraction of false positives drops below a certain threshold. By incorporating $\mathcal{N}_{SV}$ of the previous iteration in every next training, we assure that the classifier doesn't unlearn all previously seen non-face examples, i.e. these SVs serve as a kind of memory. The net result is that at every iteration the decision boundary fences off the face examples more tightly. In this way, millions of non-face examples are processed.

In our experiments, we used templates of size 40×30 (rows $\times$ columns). We collected 800 face examples from the NIST Special Database 18. All examples were alligned

C	p	A	B	C
1	1	1.26 ± 0.02	2.81 ± 0.02	3.20 ± 0.20
	2	1.28 ± 0.02	2.84 ± 0.02	3.14 ± 0.18
	3	1.28 ± 0.02	2.84 ± 0.02	3.14 ± 0.18
	4	1.28 ± 0.02	2.83 ± 0.02	3.14 ± 0.18
	5	1.28 ± 0.02	2.83 ± 0.02	3.14 ± 0.18
	6	1.29 ± 0.02	2.85 ± 0.02	3.13 ± 0.18
	7	1.28 ± 0.02	2.88 ± 0.03	3.13 ± 0.18
10	1	1.26 ± 0.02	2.80 ± 0.02	3.18 ± 0.19
	2	1.28 ± 0.02	2.85 ± 0.02	3.14 ± 0.18
	3	1.29 ± 0.02	2.90 ± 0.03	3.14 ± 0.18
	4	1.29 ± 0.02	2.93 ± 0.03	3.13 ± 0.18
	5	1.30 ± 0.02	2.97 ± 0.03	3.13 ± 0.18
	6	1.31 ± 0.02	2.99 ± 0.03	3.13 ± 0.17
	7	1.32 ± 0.02	3.00 ± 0.03	3.11 ± 0.17
100	1	1.30 ± 0.02	2.92 ± 0.03	3.20 ± 0.19
	2	1.30 ± 0.02	3.00 ± 0.03	3.13 ± 0.17
	3	1.31 ± 0.02	3.01 ± 0.03	3.10 ± 0.17
	4	1.32 ± 0.02	3.03 ± 0.04	2.93 ± 0.18
	5	1.33 ± 0.02	3.06 ± 0.04	2.54 ± 0.18
	6	1.34 ± 0.02	3.07 ± 0.04	2.22 ± 0.16
	7	1.36 ± 0.02	3.08 ± 0.04	2.04 ± 0.13
1000	1	1.33 ± 0.02	3.04 ± 0.04	3.31 ± 0.18
	2	1.34 ± 0.02	3.07 ± 0.04	2.09 ± 0.14
	3	1.36 ± 0.02	3.08 ± 0.04	1.99 ± 0.11
	4	1.38 ± 0.03	3.11 ± 0.04	1.99 ± 0.11
	5	1.41 ± 0.03	3.12 ± 0.04	1.98 ± 0.11
	6	1.42 ± 0.03	3.12 ± 0.04	1.97 ± 0.11
	7	1.44 ± 0.03	3.15 ± 0.05	1.93 ± 0.11
10000	1	1.39 ± 0.03	3.09 ± 0.04	3.35 ± 0.17
	2	1.43 ± 0.03	3.13 ± 0.04	1.99 ± 0.11
	3	1.46 ± 0.03	3.13 ± 0.04	1.98 ± 0.11
	4	1.48 ± 0.03	3.16 ± 0.05	1.93 ± 0.11
	5	1.48 ± 0.03	3.20 ± 0.05	1.81 ± 0.11
	6	1.48 ± 0.03	3.27 ± 0.06	1.77 ± 0.10
	7	1.29 ± 0.03	3.33 ± 0.06	1.76 ± 0.09

Table 3: *Results of the SVM based approach for problems A, B and C, and for different values of C and p.*

with respect to the coordinates of the eyes and mouth and rescaled to the required size. This set was virtually extended by applying small scale, translation and rotation perturbations and the final training set consists of 16,695 examples. For the collection of non-faces we used 1000 manually selected images from the Corell® database. Prior to classification, all image windows are first normalised and then masked. The normalisation (substraction of the mean and division by the standard deviation) establishes robustness against global lighting conditions. The masking sets all pixels outside a central elliptic region in the image window to zero. All SVMs were trained with SVM^{light}, and we used a polynomial kernel of degree 5 and set the error weighting constant C to 100, which proved to be optimal in cross validation experiments on an independent validation set.

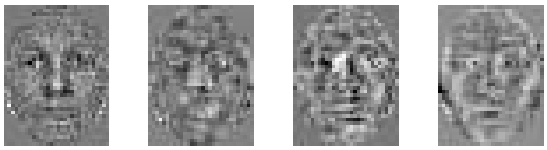

Figure 2: *The first 4 discriminant features. The homogenous areas at the corners of the templates result from the masking of the image windows.*

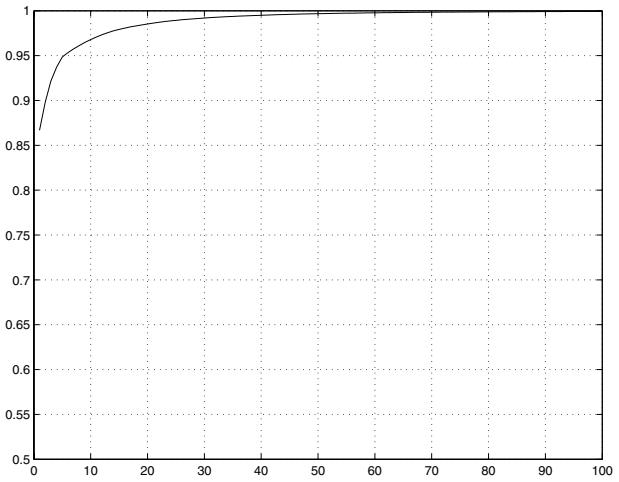

Figure 3: *Cumulative plot of eigenvalues. The first feature explains 87% of the total scatter. To account for 99% of the total scatter, the discriminant subspace needs to be 30-dimensional. This plot suggests that a good classification performance can be obtained on relatively few dimensions.*

The discriminant features are the eigenvectors of (13). The 4 most important features, i.e. with highest associated eigenvalues, are shown in figure 2. Note that the most important feature more or less zeroes out the contribution of all pixels, except for the ones in the eyes, nose and mouth region. Figure 3 shows a portion of the cumulative plot of the eigenvalues. From this figure we can see that the SVM discriminates between faces and non-faces on relatively few dimensions. Remarkably, the first feature explains almost 90% of the total scatter, which indicates that large parts of the decision boundary are planar.

4.2. Classification in Discriminant Subspaces

If we want to separate faces from non-faces in the discriminant subspace, any classifier could be used. However, if we use SVMs we can kick-start the training procedure by projecting the SVs into the subspace and by using these projections as initial training data in the first iteration of the bootstrap procedure. In this way, all information about the decision boundary is immediately incorporated in the new

training.

We trained several SVM classifiers in different d-dimensional subspaces, with $d \in \{5, 10, 20, 40\}$. For all classifiers, we used a polynomial kernel of degree 5 and set C to 100. The resulting classifiers were evaluated on the CMU Frontal Face Test Set [8] which contains 130 images with 507 faces. The performance was measured with ROC curves, with as free parameter the constant b in (11). Note that no form of post-processing (e.g. grouping of detections) was performed, i.e. the ROC curves represent raw classification outcome. The results are shown in figure 4. This figure clearly shows that, compared to the classifier in the original 1200 dimensions, a similar performance can be reached in only 40 dimensions. This confirms the results shown in figure 3.

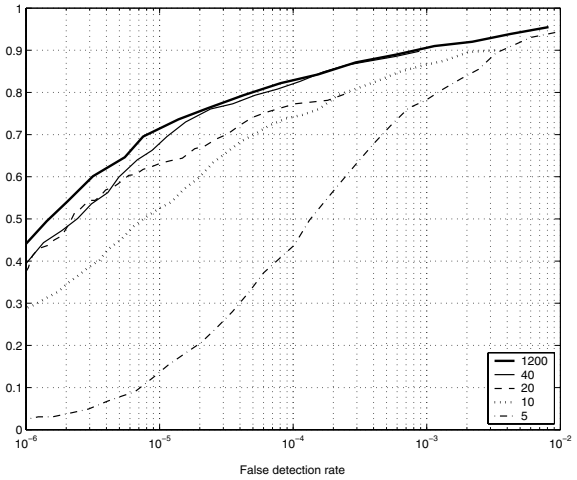

Figure 4: *ROC curves for classifiers in the original dimension and several low dimensional subspaces.*

If prior to classification the dimensionality of the data is reduced from D to d, a substantial reduction of CPU-time can be achieved, as long as $d \ll N_{SV}$. For a classification in D dimensions, the number of operations is $2DN_{SV} + N_K N_{SV} + 2N_{SV}$, where N_K is the number of operations needed to evaluate the kernel function on the inner product of a SV with the input vector (see Eq.(11)). If we first perform a reduction to d dimensions, the total number of operations becomes $2dD + 2dN'_{SV} + N_K N'_{SV} + 2N'_{SV}$, where the first term results from the subspace projection. For our problem, we typically end up with about 1000 SVs, in the original dimension as well as in the reduced dimensions. Given D=1200, this means that for d=50, a 10-fold speed-up can be obtained.

5. Conclusions and Future Work

In this paper, we described a novel approach for feature selection in high dimensional classification problems. The algorithm estimates the decision boundary feature matrix, based on the gradient of the SVM decision function evaluated at the SVs. We introduced a testing methodology based on the discriminant effectiveness ratio. This allows an objective comparison of LDA algorithms because it does not refer to any particular classifier on the induced subspace. The experiments on the synthetic data show that the SVM-based approach has a performance comparable to the Fukunaga algorithm but, contrary to the latter, it can process arbitrary amounts of training data. In a face detection experiment, we demonstrated that it is possible to reduce the problem dimension to a fraction of the original with negligible loss of classification performance. Currently, we are investigating the potential of other kernels than the polynomial ones. There are strong indications that RBF-kernels can still improve the results, e.g. in [10] it was shown that these kernels perform consistently well on 20 public domain benchmark datasets. We are also employing the proposed technique in the construction of a generic real-time object detection system, which is based on a cascade of classifiers operating on successively higher dimensional subspaces.

Acknowledgments We acknowledge support by the European IST project CogViSys.

References

[1] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *DMKD*, Vol.2(2), pp. 121-167, 1998.

[2] T. Evgeniou, C. Papageorgiou, M. Pontil, T. Poggio, "Image Representations for Object Detection using Kernel Classifiers", *ACCV*, 2000.

[3] T. Joachims, "Making large-Scale SVM Learning Practical", *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges and A. Smola (ed.), MIT-Press, 1999.

[4] K. Fukunaga, J. M. Mantock, "Nonparametric Discriminant Analysis", *PAMI*, Vol.5(6), pp. 671-678, 1983.

[5] K. Fukunaga, *Introduction to Statistical Pattern Recognition (Second Edition)*, Academic Press Inc., 1990.

[6] C. Lee, D. A. Landgrebe, "Feature Extraction Based on Decision Boundaries", *PAMI*, Vol.15(4), pp. 388-400, 1993.

[7] M. H. Yang, D. Kriegman, N. Ahuja, "Detecting Faces in Images: A Survey", *PAMI*, Vol.24(1), pp. 34-58, 2002.

[8] H. Rowley, S. Baluja, T. Kanade, "Neural Network-Based Face Detection", *ICCV*, pp. 203-208, 1996.

[9] E. Osanu, R. Freund, F. Girosi, "Training SVMs: An Application to Face Detection", *CVPR*, pp. 130-136, 1997.

[10] T. Van Gestel, J. Suykens, "Benchmarking Least Squares SVM classifiers", *internal report, ESAT SISTA KUL*, 2000.