Pattern Anal Applic (2005) 8: 227–237
DOI 10.1007/s10044-005-0002-x

THEORETICAL ADVANCES

David Masip · Ludmila I. Kuncheva · Jordi Vitrià

An ensemble-based method for linear feature extraction for two-class problems

Received: 30 October 2004 / Accepted: 15 February 2005 / Published online: 27 September 2005

Abstract In this paper we propose three variants of a linear feature extraction technique based on Adaboost for two-class classification problems. Unlike other feature extraction techniques, we do not make any assumptions about the distribution of the data. At each boosting step we select from a pool of linear projections the one that minimizes the weighted error. We propose three different variants of the feature extraction algorithm, depending on the way the pool of individual projections is constructed. Using nine real and two artificial data sets of different original dimensionality and sample size we compare the performance of the three proposed techniques with three classical techniques for linear feature extraction: Fisher linear discriminant analysis (FLD), Nonparametric discriminant analysis (NDA) and a recently proposed feature extraction method for heteroscedastic data based on the Chernoff criterion. Our results show that for data sets of relatively low-original dimensionality FLD appears to be both the most accurate and the most economical feature extraction method (giving just one-dimension in the case of two classes). The techniques based on Adaboost fare better than the classical techniques for data sets of large original dimensionality.

1 Introduction

Usually, selected or extracted features are used to construct diverse base classifiers to be used as an ensemble. We propose to reverse the problem; that is, we will use the ensemble to extract features.

D. Masip (✉) · J. Vitrià
Computer Vision Center, Department Informàtica,
Universitat Autònoma de Barcelona, Edifici O Bellaterra,
08193 Barcelona, Spain
E-mail: davidm@cvc.uab.es

L. I. Kuncheva
School of Informatics, University of Wales, Bangor Dean Street,
Bangor, Gwynedd LL57 1UT, United Kingdom
E-mail: l.i.kuncheva@bangor.ac.uk

Given a data set, feature extraction methods transform the original feature space of dimensionality D into a new space of dimensionality M, (usually $M < D$). The transformation used, $\mathscr{T} : \Re^D \rightarrow \Re^M$ can be linear, which has given rise to a plethora of linear feature extraction methods, or nonlinear. For the linear group of feature extraction methods, the vector $\mathbf{y}$ in the new space is obtained from the vector $\mathbf{x}$ in the original space as $\mathbf{y} = \mathbf{W}\mathbf{x}$ where $\mathbf{W}$ (of size $M \times D$) is called the transformation or the projection matrix. The choice of $\mathbf{W}$ depends on the criterion being optimized. One of the best known linear feature extraction techniques is principal component analysis (PCA) tries to "explain" as much variability in the data as possible within the M projections. Independent component analysis (ICA) [1] and projection pursuit [2] are linear feature extraction methods which optimize slightly different criteria and do not impose any assumption on the data distribution. Other unsupervised methods have been developed for special cases, for example the non negative matrix factorization (NMF) algorithm which suited to positive sparse data sets [3]. When class labels of the data points are available, $\mathbf{W}$ is chosen so as to maximize discrimination between the classes (supervised methods). The most widely known method in this group is Fisher linear discriminant analysis (FLD) [4]. Many modifications of the original algorithm have been proposed depending on the criterion to maximize [5, 6, 7].

Nonlinear feature extraction methods have also been developed. Roweis and Lawrence [8] described a nonlinear method, called locally linear embedding, which preserves the local neighborhood of data samples during dimensionality reduction. Tenenbaum et al. [9] introduced the Isomap algorithm, based on preserving the geodesic distances between points from the training set. Both these methods belong in the unsupervised category.

It has been shown many times in the past that unsupervised feature extraction methods often fail to extract projections with good discriminatory information. On the other hand, the abundance of supervised linear feature extraction methods is an indication that there are still open problems. In this paper we propose a linear feature extraction methodology based on the

Adaboost algorithm. A feature selection method using Adaboost is proposed in [10]. Authors use decision stumps as base classifiers so that each tree in the ensemble consists of a root and two leaves. The split at the root is done on a single feature, different for each classifier. At each step the selected feature is removed from the set of candidates for the next classifier. This method has been applied for microarray data where there is a large number of features and a small number of samples. Athitsos et al. [11], introduced the BoostMap multidimensional embedding as a combination of simple one-dimensional embeddings that preserve information of proximity structure. They transform the simple embeddings in classifier problems, and apply Adaboost for finding the best combination of the embeddings on the training data. Sirlantzis et al. [12] suggested a fusion scheme performing a two-stage classification where first an $n-$tuple based classifier is used to extract intermediate features that are the input to the second stage classifiers. Also, Brown et al. [13] applied an ensemble of neural networks to extract features for the K-nearest neighbor classifier.

In a way, all ensemble methods can be viewed as "feature extractors". We can regard each classifier in the ensemble as a feature extractor, taking the original features as its inputs and producing a class label as the extracted feature. The combination of these extracted features can be perceived as the classifier in the new feature space [14].

Assume that we use linear classifiers as the base classifiers in the ensemble to solve a two-class problem. Instead of thresholding the output of each classifier and taking the class label to be the output, we use the value computed as the linear combination to be our extracted feature. While any ensemble method can be applied for this feature extraction, here we chose Adaboost which has been declared to be "the best off-the-shelf" ensemble method [15].

This paper is organized as follows. In the next section we give a brief review of three classical feature extraction techniques used for comparison in this study. Section 3 introduces the three proposed variants of what we call boosted discriminant projections. Section 4 contains the experimental results and Sect. 5 concludes our study.

2 Feature extraction and discrimination

2.1 Principal component analysis (PCA)

One of the most widely used algorithms for feature extraction is principal component analysis [16]. To calculate the M principal components for a $D-$dimensional data set, $\mathbf{X}$, we follow the standard procedure shown in Fig. 1 using the covariance matrix Σ of $\mathbf{X}$ as the criterion matrix M. Denote by $\mathbf{x}=[x_1, ..., x_D]^T$ a feature vector in the original $D-$dimensional space and $\mathbf{y}=[y_1, ..., y_D]^T$ the vector with the extracted features. The jth extracted feature will be:

$$y_j = a_{1j}x_1 + \cdots + a_{Dj}x_D \tag{1}$$

where $\mathbf{a}_j=[a_{1j}, ..., a_{Dj}]^T$ is the eigenvector corresponding to the jth largest eigenvalue of Σ.

Note that we can normalize $\mathbf{X}$ beforehand by using

$$x'_i = \frac{x_i - m_i}{s_i}, \quad i = 1, \ldots, D \tag{2}$$

where m_i is the sample mean and s_i is the sample standard deviation for feature i. Such normalization is recommended if the original features have different measurement units and there may be variance due to this discrepancy which need not be preserved.

The goal of PCA is to find orthogonal linear projections that account for the maximum amount of data variance. Thus D-dimensional data can be reconstructed using M dimensions ($M < D$). However, the features that minimize the reconstruction error are not necessarily the features most suitable for classification [17]. If the class labels of the training sample are taken into account, other linear projections can yield a better classification accuracy even though the reconstruction error is not minimized.

Alternative approaches for linear feature extraction have been proposed based upon different criteria and assumptions made on the training data. Below we briefly explain the classic FLD, and an evolution of this algorithm introduced by Fukunaga et al. [5], called nonparametric discriminant analysis (NDA). We also explain a recent technique which utilizes the Chernoff criterion to extend the FLD [6].

2.2 Fisher linear discriminant analysis (FLD)

Here we look for a transformation matrix $\mathbf{W}$ which maximizes

$$\mathcal{J} = tr((\mathbf{W}\mathbf{S}_W^{-1}\mathbf{W}^T)(\mathbf{W}\mathbf{S}_B\mathbf{W}^T)) \tag{3}$$

Here $\mathbf{S}_B$ and $\mathbf{S}_W$ are the between-class and the within-class scatter matrix respectively, calculated as explained below. This problem has an analytical solution [17]. $\mathbf{W}$ is constructed using as its rows the M eigenvectors corresponding to the largest M eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$.

1. Calculate the criterion matrix $\mathcal{M}$.
2. Find the eigenvalues and the corresponding eigenvectors of $\mathcal{M}$
3. Find the M largest eigenvalues of $\mathcal{M}$ and take the corresponding eigenvectors to be the coefficients for the extracted features[a].

[a] We assume that M is less than or equal to the number of non-zero eigenvalues of $\mathcal{M}$

Fig. 1 A general procedure for linear feature extraction

The feature extraction goes through the steps in Fig. 1 with $\mathscr{M} = \mathbf{S}_W^{-1}\mathbf{S}_B$.

The most common approach for calculating the within- and between-class scatter matrices makes use of only up to second order statistics of the data. This was proposed in the classic paper by Fisher [4] and the technique is referred to as FLD analysis. In FLD the within class scatter matrix is computed as the weighted sum of the class-conditional sample covariance matrices. If equal priors are assumed for the classes C_k, $k=1$, ..., K, then

$$\mathbf{S}_W = \frac{1}{K}\sum_{k=1}^{K}\mathbf{S}_k \tag{4}$$

where $\mathbf{S}_k$ is the class-conditional covariance matrix for C_k, estimated from the data. The between class-scatter matrix is defined as,

$$\mathbf{S}_B = \frac{1}{K}\sum_{k=1}^{K}(\mathbf{m}_k - \mathbf{m}_0)(\mathbf{m}_k - \mathbf{m}_0)^T \tag{5}$$

where $\mathbf{m}_k$ is the class-conditional sample mean and $\mathbf{m}_0$ is the unconditional (global) sample mean.

The following two limitations of FLD have to be noted. First, the rank of $\mathbf{S}_B$ is $K-1$, so the number of extracted features can be, at most, one less than the number of classes (one dimension for the two-class problem). Second, the scatter matrices are calculated assuming Gaussian classes. The solution provided by FLD is blind beyond second-order statistics, so this method may be inaccurate for complex classification structures.

2.3 Nonparametric discriminant analysis (NDA)

Fukunaga and Mantock [5] propose a nonparametric discriminant analysis method as an attempt to overcome the two limitations of FLD noted above. In NDA the between-class scatter matrix $\mathbf{S}_B$ is calculated without the assumption of Gaussian classes. This scatter matrix is generally full rank, thus loosening the bound on the extracted feature dimensionality. Below, we briefly expose this technique, extensively detailed in [17].

In NDA, the between-class scatter matrix is obtained as an average of N local covariance matrices, one for each point in the data set. This is done as follows. Let $\mathbf{x}$ be a data point in $\mathbf{X}$ with class label C_j. Denote by $x^{\text{different}}$ the subset of the k nearest neighbors of $\mathbf{x}$ among the data points in $\mathbf{X}$ with class labels different from C_j. We calculate the "local" between-class matrix for $\mathbf{x}$ as

$$\Delta_B^{\mathbf{x}} = \frac{1}{k-1}\sum_{\mathbf{z}\in x^{\text{different}}}(\mathbf{z}-\mathbf{x})(\mathbf{z}-\mathbf{x})^T \tag{6}$$

The estimate of the between-class scatter matrix $\mathbf{S}_B$ is found as the average of the local matrices

$$\mathbf{S}_B = \frac{1}{N}\sum_{\mathbf{z}\in X}\Delta_B^{\mathbf{z}} \tag{7}$$

We use $k=1$ in this study, hence $x^{\text{different}}$ contains only one element, $\mathbf{z}_{\mathbf{x}}^{\text{different}}$, and

$$\mathbf{S}_B = \frac{1}{N}\sum_{\mathbf{x}\in X}(\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{\text{different}})(\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{\text{different}})^T. \tag{8}$$

The feature extraction goes through the steps in Fig. 1 with $\mathscr{M} = \mathbf{S}_W^{-1}\mathbf{S}_B$ with the new $\mathbf{S}_B$. Bressan and Vitrià [18] introduced a non parametric form of the within-class scatter matrix $\mathbf{S}_W$, which is expected to provide features which work well with the nearest neighbor classifier. They propose to use

$$\mathbf{S}_W = \frac{1}{N}\sum_{\mathbf{z}\in X}\Delta_W^{\mathbf{z}} \tag{9}$$

where $\Delta_W^{\mathbf{x}}$ is calculated from the set of k nearest neighbors of $\mathbf{x}$ from the same class label, C_j, x^{same}

$$\Delta_W^{\mathbf{x}} = \frac{1}{k-1}\sum_{\mathbf{z}\in x^{\text{same}}}(\mathbf{z}-\mathbf{x})(\mathbf{z}-\mathbf{x})^T \tag{10}$$

For $k=1$,

$$\mathbf{S}_W = \frac{1}{N}\sum_{\mathbf{x}\in X}(\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{\text{same}})(\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{\text{same}})^T. \tag{11}$$

In this paper we use the local approximations of both $\mathbf{S}_B$ and $\mathbf{S}_W$, as in [18].

2.4 Heteroscedastic LDA (Chernoff)

Recently, Loog and Duin [6] extended the LDA criterion for the case of Gaussian classes with different covariance matrices (heteroscedastic data). They replace the criterion (3) with the Chernoff criterion which is expected to be superior to (3) for heteroscedastic data. Their experiments showed encouraging results.

Let $\mathbf{m}_1$ and $\mathbf{m}_2$ be the estimated means of the two classes, p_1 and p_2 be the estimated prior probabilities ($p_2 = 1 - p_1$), and $\mathbf{S}_1$ and $\mathbf{S}_2$ be the respective covariance matrices. For simplicity of notation, denote by $\mathbf{S}$ the within-class covariance matrix $\mathbf{S}_W$ calculated as $\mathbf{S} = p_1\mathbf{S}_1 + p_2\mathbf{S}_2$. The criterion matrix is again $\mathscr{M} = \mathbf{S}_W^{-1}\mathbf{S}_B$ where $\mathbf{S}_B$ is calculated as

$$\mathbf{S}_B = p_1p_2\mathbf{S}^{\frac{1}{2}}\Big(\mathbf{S}^{-\frac{1}{2}}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{S}^{-\frac{1}{2}} - \frac{1}{p_2}\log\Big(\mathbf{S}^{-\frac{1}{2}}\mathbf{S}_1\mathbf{S}^{-\frac{1}{2}}\Big) - \frac{1}{p_1}\log\Big(\mathbf{S}^{-\frac{1}{2}}\mathbf{S}_2\mathbf{S}^{-\frac{1}{2}}\Big)\Big)\mathbf{S}^{\frac{1}{2}} \tag{12}$$

Here a function f (logarithm or power) of a matrix $\mathbf{A}$ is calculated in the following way. Let $\mathbf{VDV}^{-1}$ be the eigenvalue decomposition of $\mathbf{A}$, i.e., $\mathbf{V}$ is the matrix of eigenvectors and $\mathbf{D}$ is a diagonal matrix with the eigenvalues on the leading diagonal. The function is applied to the eigenvalues and the results are placed at the

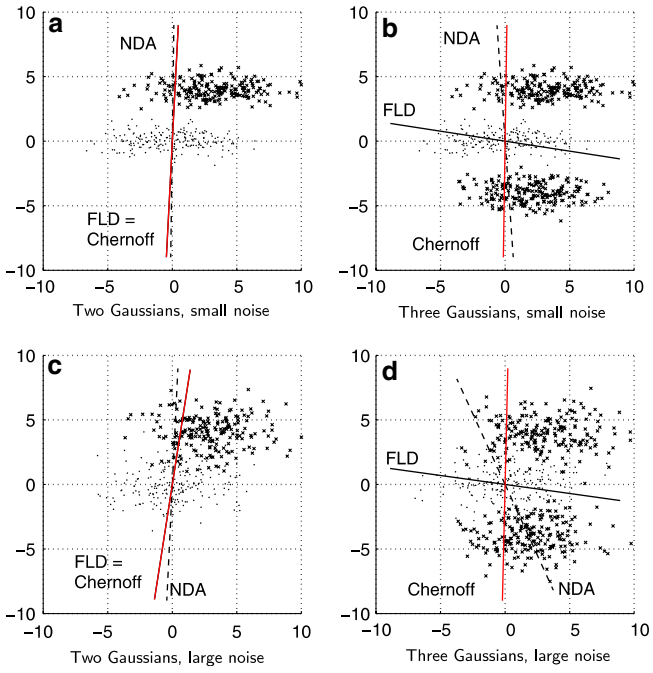

Fig. 2 Examples of FLD, NDA and Chernoff for Gaussian (**a** and **c**) and non-Gaussian (**b** and **d**) classes for two levels of noise on the y-axis

leading diagonal of a diagonal matrix, denoted $f(\mathbf{D})$. Then $f(\mathbf{A}) = \mathbf{V} f(\mathbf{D}) \mathbf{V}^{-1}$.

For equal covariance matrices $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}$, the Chernoff matrix $\mathbf{S}_B$ in (12) is the same as the matrix $\mathbf{S}_B$ of FLD.

Figure 2 shows four two-class problems and the projections obtained through FLD, NDA and Chernoff. In subfigures (a) and (c) both classes have Gaussian distribution, with the same covariance matrices. The Chernoff and FLD projections overlap, and NDA also yields a similar projection. On the other hand, in (b) and (d) only one of the two classes is Gaussian. The bimodality of one of the classes displaces the class mean introducing errors in the estimate of the parametric version of $\mathbf{S}_B$ in FLD. NDA and Chernoff are not affected by this for small noise levels. Chernoff seems to tolerate noise better than NDA. Subplot (d) shows that the projection selected by NDA is affected by noise on the y-axis while the Chernoff projection is not.

3 Boosted discriminant projections

As explained in Sect. 2, linear discriminant analysis techniques tend to maximize criteria under some assumptions on the data samples. Our proposed feature extraction technique makes no assumptions on the statistical distribution of the data. We propose to find the projection matrix incrementally, using a modified Adaboost algorithm.[1] The original Adaboost algorithm [19, 20] is based on incrementally building a set of classifiers that are combined to yield a more powerful decision rule. At each boosting step a new classifier is generated, and the training samples are reweighted according to the classification results. The weights are used to generate the next step classifier. Two different implementation approaches have been followed in the literature: reweighting (using classifiers that can directly accommodate weights) and resampling (sampling from the training set for each classifier according to the distribution defined by the weights [15]).

In our proposal at each boosting step we generate projection vectors (candidate linear features) and select the best one to be the extracted feature for this step. To evaluate a candidate feature, we build a classifier on it. The values of the feature are calculated for the training data and an optimal threshold is found minimizing the number of misclassified training samples. The weighted error is calculated using the weights for the data points at the present Adaboost step. The output of the algorithm is a projection matrix $\mathbf{W}$.

Depending on the generation and the selection of the projections at each step, different methods can be derived from the general idea. In this section three variants of the algorithm are proposed.

3.1 Random boosted discriminant projections (RBDP)

Maybe the simplest way of generating one-dimensional projections from the data is to randomly select pairs of points, one point from each class, and take the vector between the two as the candidate projection. Using a large enough set of candidate projections generated in this way, we can use Adaboost weights to select the projection with the smallest weighted error. The general algorithm is shown in Fig. 3. The random generation explained above is implemented as 3(a). The parameters of the algorithm which need to be picked in advance are: M, the number of classifiers in the ensemble (desired number of projections); and P, the size of the "projection pool", i.e., how many candidate-projections are generated at each step of Adaboost.

The calculation of the optimal threshold for projection p in 3(b)(ii) follows the steps below. Let $\mathbf{p} = [p_1, p_2, \ldots, p_D]^T$ be the coefficient vector for projection p. First, caclulate $y_i = \mathbf{p}^T \mathbf{x}_i$, $i = 1, \ldots, N$. Second, sort the y'_is. Third, calculate the classification error for all the values of the threshold in the middle between every two consecutive y'_is. Fourth, choose and retain the threshold with the minimum error.

Note that there is no need for calculating the final hypothesis in this Adaboost version because the information needed at the end is only the projection matrix $\mathbf{W}$.

3.2 Local boosted discriminant projections (LBDP)

Another approach to feature extraction using Adaboost is to build a set of projections in a deterministic way. We implement 3(a) in Fig. 3 by the following steps:

[1] We assume that the reader is familiar with Adaboost although the feature extraction should be reproducible from the Fig. 3

Fig. 3 A generic learning algorithm for boosted discriminant projections

1. Given are the matrix $\mathbf{X}$ containing data samples $\mathbf{x}_i$, and the vector $\mathbf{y}$ with the corresponding labels $y_i \in \{-1, 1\}$ $(i = 1 \ldots N)$
2. Initialize a set of weights: $W_i(1) = \frac{1}{N}$.
3. For $t = 1 \ldots M$:
   (a) Generate P projections from the original space to an $1-$dimensional subspace.
   (b) For $p = 1 \ldots P$
      i. Project the training data into the $1-$dimensional space using projection p.
      ii. Learn the threshold that best separates the samples into two classes, thereby constructing hypothesis $h_{t,p}$ $(h_{t,p}(\mathbf{x}_i) \in \{-1, 1\}, \forall \mathbf{x}_i \in \mathbf{X})$. Denote by $l_{t,p}(\mathbf{x}_i)$ the loss incurred in labeling $\mathbf{x}_i$ by $h_{t,p}$. The loss is $l_{t,p}(\mathbf{x}_i) = 1$ if a misclassification occurs and $l_{t,p}(\mathbf{x}_i) = 0$, otherwise.
      iii. Compute the weighted error for the projection as:

$$Err_p = \sum_{i=1}^{N} W_i(t) l_{t,p}(\mathbf{x}_i). \tag{13}$$

   (c) Find the projection, m, with the minimum error, i.e., $Err_m = \min_{p=1}^{P} Err_p$. Classify the training set using m.
   (d) Calculate the weight for classifier t, β_t, as:

$$\beta_t = \frac{Err_m}{1 - Err_m} \tag{14}$$

   (e) Update the data weights:

$$W_i(t+1) = W_i(t) \beta_t^{(1 - l_{t,m}(\mathbf{x}_i))}, \quad i = 1, \ldots, N. \tag{15}$$

   (f) Normalize the weights so that $W(t+1)$ is a distribution.

$$W_i(t+1) \leftarrow \frac{W_i(t+1)}{\sum_j W_j(t+1)} \tag{16}$$

   (g) Store m as the t-th projection in the projection matrix $\mathbf{W}$.
4. Output the projection matrix $\mathbf{W}$, built using the vectors selected at each boosting step as columns.

1. For each point $\mathbf{x}_i$ find the nearest neighbor from the same class, $\mathbf{z}^{\text{same}}$, and the nearest neighbor from the opposite class, $\mathbf{z}^{\text{different}}$.
2. The points $\mathbf{x}_i$, $\mathbf{z}^{\text{same}}$ and $\mathbf{z}^{\text{different}}$ define a plane, γ, in the initial space of dimensionality D. We propose that the linear projection that we are looking for lies in γ. The transformation matrix $\mathbf{A}_{2\times D}$ is found using $\mathbf{x}_i$, $\mathbf{z}^{\text{same}}$ and $\mathbf{z}^{\text{different}}$. We construct vectors $\mathbf{v}$ and $\mathbf{w}$, $\mathbf{v}$,$\mathbf{w} \in \gamma$, as

$$\mathbf{v} = [v_1, v_2]^T = \mathbf{x}_i - \mathbf{z}^{\text{same}}$$
$$\mathbf{w} = [w_1, w_2]^T = \mathbf{x}_i - \mathbf{z}^{\text{different}}. \tag{17}$$

These vectors can be perceived as local descriptors for the within and between class distances. To illustrate the calculations, consider the following example for $D = 3$. Let $\mathbf{x}_i = [0, 0, 0]^T$, $\mathbf{z}^{\text{same}} = [1,3,6]^T$, and $\mathbf{z}^{\text{different}} = [5,1,-2]^T$. The transformation matrix is

$$\mathbf{A} = \begin{bmatrix} 0.9129 & 0.1826 & -0.3651 \\ 0.1474 & 0.4423 & 0.8847 \end{bmatrix}.$$

The projections of the three points on the plane γ, as well as vectors $\mathbf{v}$ and $\mathbf{w}$ are shown in Fig. 4.

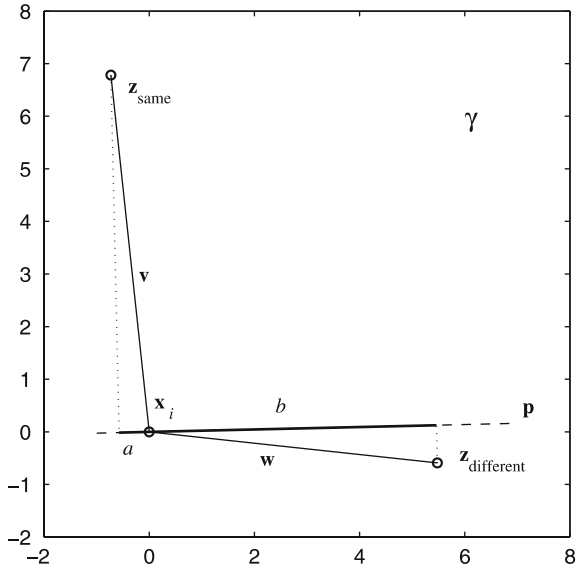

Fig. 4 Projections of $\mathbf{x}_i$, $\mathbf{z}^{\text{same}}$ and $\mathbf{z}^{\text{different}}$ on γ. Vectors $\mathbf{v}$ and $\mathbf{w}$ are shown. The *solid line* indicates the direction of the optimal projection $\mathbf{p}$

3. In the two-dimensional subspace, γ, we are looking for a direction vector $\mathbf{p}_i$ such that the projections of $\mathbf{x}_i$ and $\mathbf{z}^{\text{same}}$ on $\mathbf{p}_i$ are close to one another while the projection of $\mathbf{z}^{\text{different}}$ on $\mathbf{p}_i$ is as far away from $\mathbf{x}_i$ as possible. Assuming that $\mathbf{p}_i$ has unit length, the distance between the projections of $\mathbf{x}_i$ and $\mathbf{z}^{\text{same}}$ on $\mathbf{p}_i$ is

$$\left|\mathbf{p}_i^T\mathbf{x}_i - \mathbf{p}_i^T\mathbf{z}^{\text{same}}\right| = \left|\mathbf{p}_i^T\mathbf{v}\right|. \tag{18}$$

Therefore a possible criterion function to maximize is

$$\max\left\{\left(\mathbf{p}_i^T\mathbf{w}\right)^2 - \left(\mathbf{p}_i^T\mathbf{v}\right)^2\right\}. \tag{19}$$

The four solutions of (19) for $\mathbf{p}_i = [p_1, p_2]^T$ are

$$p_1 = \pm\sqrt{1-(p_2)^2}, \quad p_2 = \sqrt{\frac{1}{2}\left(1 \pm \frac{w_2^2 - v_2^2 - w_1^2 + v_1^2}{\sqrt{4(w_1w_2 - v_1v_2)^2 + (w_2^2 - v_2^2 - w_1^2 + v_1^2)^2}}\right)}. \tag{20}$$

For each $\mathbf{x}$ we take the solution that maximizes (19). Then we project back $\mathbf{p}_i$ in the original D-dimensional space using $\mathbf{A}^{-1}$.

Plotted in Fig. 4 is the optimal direction $\mathbf{p}_i$ (denoted $\mathbf{p}$) and the projections of $\mathbf{x}_i$, $\mathbf{z}^{\text{same}}$ and $\mathbf{z}^{\text{different}}$. The criterion requires simultaneous maximization of b and minimization of a. The direction $\mathbf{p}_i$ for the above example was found to be $[0.9997 \quad 0.0229]^T$. To transform $\mathbf{p}_i$ back to the original space, the inverse transformation is applied

$$\mathbf{A}^{-1}\mathbf{p}_i = \begin{bmatrix} 0.9396 & 0.2486 \\ 0.2329 & 0.4674 \\ -0.2731 & 0.8552 \end{bmatrix} \begin{bmatrix} 0.9997 \\ 0.0229 \end{bmatrix} = \begin{bmatrix} 0.9451 \\ 0.2436 \\ -0.2534 \end{bmatrix}$$

Note that the pool of candidate projections for LBDP consists of N projections and can be calculated in advance, before running Adaboost.

3.3 Boosted fisher projections (BFP)

The third variant that we propose is to use more than two points to compute the projection at each boosting step. The P projections in 3(a) are generated as follows:

1. Sample K points from each class according to the distribution defined by the Adaboost weights at the current step.
2. Perform FLD on the selected samples to obtain the projection that best separates the data.

We expect BFP to be more robust than RBDP because more points are involved in the calculation of each projection. Thus, the projections in BFP are expected to be more accurate but more similar to one another compared to these in RBDP.

We note that contrary to the previous feature extraction methods explained in Sect. 2, we do not start with a criterion function to optimize but use heuristics which have proven to work for classifiers. It is not straightforward to find an explicit expression of the three criteria behind the three variants, especially when there is a random component involved.

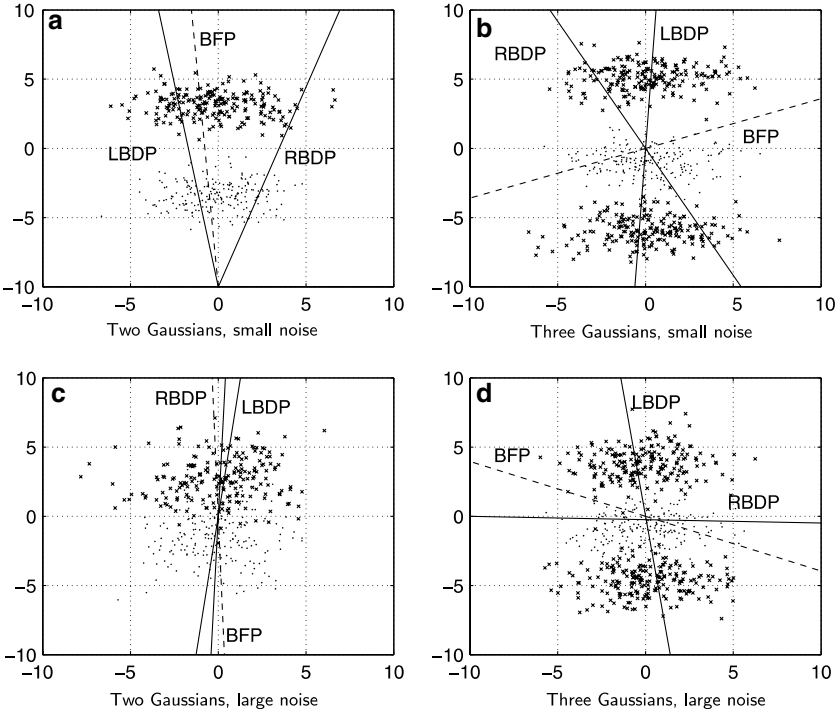

Fig. 5 Examples of RBDP, LBDP and BFP for Gaussian (**a** and **c**) and non-Gaussian (**b** and **d**) classes for two levels of noise on the y-axis

Table 1 The 11 data sets used in the experiments. We show the database name, dimensionality D, the number of features preserved after the PCA was performed, the total number of samples available (after removing the samples with missing values), and the Sparseness of the data (number of data points/dimensionality). The two last data sets are separated to indicate that they have considerably larger dimensionality

Database	Label	Features	Features PCA	Samples	Sparseness
BUPA liver disorder	(a)	6	6	345	57.50
Wisconsin diagnostic breast cancer	(b)	30	7	569	81.28
8-D banana shaped data	(c)	8	8	500	62.50
Wisconsin breast cancer	(d)	9	9	666	74.00
Cleveland heart disease	(e)	13	13	297	22.84
German database	(f)	24	24	1,000	41.67
Ionosphere database	(g)	34	33	351	10.63
Sonar signals database	(h)	60	59	208	3.52
SPECTF heart	(i)	44	44	349	7.93
500-D gaussian	(j)	500	449	500	1.11
AR face database	(k)	2,964	453	500	1.10

Figure 5 shows the same two-class problems as in Fig. 2 using the RBDP, LBDP and BFP methods. First, as can be expected, the projections obtained by RBDP are almost arbitrary as seen in the four plots. On the other hand, the BFP algorithm extracts a single feature using the classic FLD. If the number of points from each class participating in the calculation, K is chosen so that all points are used, BFP is exactly equivalent to FLD. In our example $K=100$ which explains both the similarities and differences between BFP and FLD. Finally, LBDP finds a good projection in all four cases because it picks empirically the best projection out of N candidates. These findings are not unexpected as the strength of the proposed methods is supposed to come from applying Adaboost for constructing a collection of projections.

4 Experiments

This section compares the performance of the proposed feature extraction techniques with the three methods from Sect. 2. Other comparative studies of classic discriminant analysis techniques can be found in [7]. In our experimental study we compare FLD, the NDA, the discriminant analysis using the Chernoff criterion (Chernoff), and the three techniques proposed here are the RBDP, the LBDP, and the BFP.

We have tested the six methods with 11 data sets (see Table 1), eight of them taken from the UCI machine learning repository [21]. We also have generated two synthetic data sets in a similar way as in [22]:

- *8-D banana shape*. In the space of the first two features, the two classes are uniformly distributed along two concentric arcs with radii $r_1=0.125$ and $r_2=0.375$, respectively. Gaussian noise with unit variance is added to each class. The remaining eight features have Gaussian distribution with mean 0 and variance 0.1.
- *500-D Gaussian data*: This set consists of two Gaussian classes. The covariance matrices for the classes are the identity matrices. The mean for the first class is $[0,0,...,0]^T$, and the mean for the second class is $[0.1,0,1,...,0.1]^T$. The last data set is extracted from the AR face database [23], and the goal is to solve a gender recognition problem. We have taken 500 examples from male and female face images, each image represented as a 2,964-dimensional vector.

Our comparative study uses four different classifiers in the space of the extracted features[2]

- Linear classifier, which assumes normal distribution of the classes and equal covariance matrices. To avoid computational problems due to the appearance of nearly singular covariance matrices (when the number of training examples is smaller than the data dimensionality), we have set the regularization parameter to 0.5.
- Quadratic classifier, also assuming normal distributions but with arbitrary covariance matrices. The regularization parameter has been set to 0.5.
- The one-nearest neighbor classifier.
- Support vector machines classifier (SVM), using radial basis functions (with parameters $\gamma=1$, and cost $c=1$).

4.1 Experimental protocol

For each data set we performed one hundred times the experiment described below:

1. First the data set is randomly split into training data and testing data, using 90% of the samples for training and the rest for testing.
2. Following [6], we transform the data using PCA. We compute a PCA projection matrix using the training samples. We select the eigenvectors corresponding to eigenvalues larger than 10^{-7}. The training and testing

[2] Functions from the PRTOOLS 3.1.7 toolbox [24] have been used for classifiers 1–3. For the SVM classifier we used the OSU SVM Classifier Matlab toolbox 3.00 that can be downloaded from http://www.ece.osu.edu/~maj/osu_svm/.

Table 2 Maximum number of different projections, M, for the six compared feature extraction techniques (D is the initial dimensionality of the original data, N_1 and N_2 are the sample sizes for the two classes ($N = N_1 + N_2$) and K is the number of samples from each class used for a projection construction in BFP)

Method	FLD	NDA	Chernoff	RBDP	LBDP	BFP
Maximum M	1	D	D	$N_1\ N_2$	N	$\binom{N_1}{K}\binom{N_2}{K}$

vectors are projected using the same PCA projection matrix.

3. Next we find a projection matrix using each of the six feature extraction algorithms.
4. For each feature extraction algorithm we train the four classifiers (linear, quadratic, nearest neighbor and SVM) on the space of extracted features. We store the accuracies on the training and the testing sets for each possible dimension up to $M = 40$. For example, FLD allows for one feature only whereas in RBDP there can be $N_1\ N_2$ different projections, where N_1 and N_2 are the number of samples from classes C_1 and C_2, respectively ($N = N_1 + N_2$). Table 2 shows the maximum number of different projections that each of the six compared techniques can extract. The results from the experiments are shown in Tables 3, 4, 5, 6. These are computed as follows: for each feature extraction algorithm and each classifier, we find the number of extracted features M^{tr} for which the training error is minimum. Then using this dimensionality we take its corresponding error on the testing results. The final error rates and optimal dimensionalities shown in the tables are the means of the one hundred runs.

4.2 Experimental results

Tables 3, 4, 5, 6 show the mean classification error (MCE) for each database, and the mean optimal dimensionality in the brackets. Also we have marked in bold and with an '*' the method that achieves the minimum MCE for each database. We have also computed the 95% confidence interval for each MCE value, and have marked in bold the MCE of the methods whose confidence intervals overlap with the interval for the best method.[3]

Since the values of MCE are not directly comparable across the data sets and the classifiers, in order to have a measure of overall performance, we calculated the ranks for the six compared methods. Each row in Tables 3–6 is arranged in ascending order of MCE and ranks are assigned to the methods. The method with the smallest MCE obtains rank 1 (best) and the method with the largest MCE obtains rank 6 (worst) for the particular data set and classifier. For example, the first row in Table 3 corresponds to the BUPA liver disorder data classified in the space of extracted features by the linear discriminant classifier. FLD gets rank 1 (smallest MCE = 0.317), LBDP gets rank 2, etc., and NDA gets rank 6. The ranks for each method were then averaged across the 11 data sets and are shown at the bottom of the respective table.

In general, for the four classifiers, we observed that as dimensionality of the data increases, the ensemble based methods perform better than the other methods. Among the three ensemble algorithms, BFP achieved best overall result for all four classifiers.

As expected, for the nearest neighbor classifier, NDA is either the best or not significantly different from the best method in 8 of the 11 databases. In fact, NDA is specially designed to achieve low errors using NN. However, the performance of NDA is considerably worse for the linear and quadratic classifiers.

Two tendencies can be observed from Tables 3–6: when the dimensionality of the data is high, the methods based on Adaboost perform better in almost all databases and classifiers. However, when original data is low dimensional, FLD often achieves the best result, despite using only a single dimension.

The MCEs are in generally lower using the SVM classifier. The Chernoff technique ranks much better with SVM than with any of the other three classifiers.

We found that the higher the dimensionality is, the clearer becomes the advantage of the ensemble feature extraction. This is demonstrated in data sets (j) and (k). The BFP is significantly better than all the other methods except for RBDP with the linear classifier (Table 3). The overall ranks suggest that BFP is the most successful one among the examined feature extraction techniques. We conjecture that the random component combined with the weighting mechanism of Adaboost are responsible for the good performance of the ensemble feature extraction methods in high-dimensional spaces.

The second best method is FLD which shows that for many cases the simple classical methods might be the best solution. Disappointingly, Chernoff was the worst of the methods compared here. While it is optimal for heteroscedastic data, other data distributions appear to be a challenge for this method. In the reference where Chernoff method was advocated [6], regularization via special parameter was not considered. The preprocessing through PCA ensures that the covariance matrices are not singular and thus no further regularization has been suggested. Here we used the Matlab implementation of Chernoff mapping found in PRTOOLS 3.1.7. This implementation does not provide for a regularization parameter either.

[3] Full information about the standard deviations and the calculated confidence intervals can be found at http://www.cvc.uab.es/~davidm/experiments.htm

Table 3 MCE results with the linear classifier

DB	FLD	NDA	Chernoff	RBDP	LBDP	BFP
(a)	**0.317*** (1.0)	0.355 (4.8)	**0.349** (5.1)	**0.343** (4.5)	**0.341** (4.8)	**0.334** (3.5)
(b)	**0.052*** (1.0)	0.093 (3.7)	0.189 (6.2)	0.117 (4.7)	0.114 (3.2)	**0.059** (3.4)
(c)	**0.154** (1.0)	**0.156** (3.9)	0.512 (2.9)	**0.152** (1.3)	**0.151*** (5.1)	**0.162** (4.2)
(d)	**0.041** (1.0)	**0.044** (3.6)	0.061 (6.0)	**0.040** (5.5)	**0.040** (4.6)	**0.040*** (4.0)
(e)	**0.167*** (1.0)	0.239 (5.6)	0.386 (5.2)	0.293 (5.5)	0.292 (6.3)	**0.173** (6.2)
(f)	**0.131** (1.0)	**0.136** (19.4)	0.291 (2.0)	**0.141** (21.0)	**0.141** (19.7)	**0.129*** (19.9)
(g)	**0.235*** (1.0)	0.277 (18.5)	0.312 (1.0)	0.294 (5.5)	0.294 (13.7)	0.272 (2.8)
(h)	**0.251** (1.0)	**0.260** (30.8)	0.474 (1.3)	**0.244** (23.9)	**0.237*** (29.0)	0.292 (9.2)
(i)	0.234 (1.0)	0.240 (35.1)	0.302 (4.5)	**0.201*** (16.3)	**0.206** (24.3)	**0.225** (23.3)
(j)	0.491 (1.0)	0.402 (36.0)	0.542 (6.0)	**0.297** (37.8)	0.306 (36.8)	**0.278*** (30.7)
(k)	0.448 (1.0)	0.085 (24.3)	0.518 (5.9)	0.097 (28.8)	0.091 (28.5)	**0.025*** (6.4)
Rank	2.73	3.82	5.91	3.36	2.91	2.27

Table 4 MCE results with the quadratic classifier

DB	FLD	NDA	Chernoff	RBDP	LBDP	BFP
(a)	**0.363*** (1.0)	0.427 (3.5)	**0.385** (3.1)	0.399 (1.6)	0.399 (1.5)	**0.364** (2.0)
(b)	**0.055*** (1.0)	0.099 (2.5)	0.158 (4.8)	0.074 (2.6)	0.068 (1.9)	**0.057** (2.2)
(c)	**0.154** (1.0)	**0.156** (3.7)	0.503 (3.1)	**0.149*** (1.5)	**0.149** (5.3)	**0.164** (4.0)
(d)	**0.030** (1.0)	**0.033** (2.0)	0.047 (3.2)	**0.029*** (1.1)	**0.031** (2.5)	**0.031** (2.6)
(e)	**0.168*** (1.0)	0.230 (11.4)	0.378 (5.3)	0.315 (5.0)	0.313 (6.5)	**0.181** (8.2)
(f)	0.130 (1.0)	0.091 (15.2)	0.263 (4.6)	0.084 (15.7)	**0.061*** (23.1)	**0.068** (7.8)
(g)	**0.234*** (1.0)	0.284 (21.5)	0.328 (2.3)	0.299 (8.1)	0.291 (11.4)	0.272 (3.5)
(h)	0.251 (1.0)	0.210 (30.9)	0.505 (5.2)	**0.204** (22.0)	**0.172*** (32.3)	0.300 (7.7)
(i)	0.254 (1.0)	**0.220** (31.4)	**0.214** (8.3)	0.265 (2.0)	0.291 (3.0)	**0.214*** (9.3)
(j)	0.491 (1.0)	0.480 (25.1)	0.506 (17.2)	0.314 (38.8)	0.362 (38.5)	**0.278*** (27.0)
(k)	0.448 (1.0)	0.049 (30.7)	0.428 (18.0)	0.092 (31.5)	0.083 (27.6)	**0.025*** (5.7)
Rank	3.00	3.82	5.27	3.36	3.09	2.45

Table 5 MCE results with the nearest neighbor classifier

DB	FLD	NDA	Chernoff	RBDP	LBDP	BFP
(a)	0.414 (1.0)	**0.370*** (4.0)	0.442 (1.0)	**0.384** (3.2)	**0.397** (2.0)	**0.372** (4.3)
(b)	0.082 (1.0)	**0.045*** (5.9)	0.392 (1.0)	0.091 (4.0)	0.081 (1.7)	0.070 (4.9)
(c)	0.192 (1.0)	**0.023*** (4.2)	0.498 (1.9)	**0.030** (5.2)	**0.029** (4.2)	0.152 (5.7)
(d)	**0.041*** (1.0)	**0.045** (4.5)	0.074 (1.0)	**0.041** (3.7)	**0.048** (1.5)	**0.044** (4.1)
(e)	**0.229*** (1.0)	0.285 (7.7)	0.452 (1.7)	0.417 (3.2)	0.397 (1.2)	**0.245** (4.7)
(f)	0.170 (1.0)	**0.123** (11.5)	0.385 (7.9)	0.116 (11.4)	**0.125** (11.6)	**0.105*** (6.1)
(g)	**0.314*** (1.0)	**0.315** (12.2)	0.399 (2.9)	0.394 (7.4)	0.386 (1.5)	**0.314** (14.3)
(h)	0.267 (1.0)	**0.175** (18.9)	0.483 (5.2)	**0.180** (15.3)	**0.161*** (23.8)	0.365 (4.5)
(i)	0.143 (1.0)	**0.130** (21.0)	0.185 (4.8)	**0.114** (8.3)	**0.108*** (3.9)	0.134 (8.8)
(j)	0.419 (1.0)	0.469 (4.8)	0.517 (27.2)	0.399 (25.5)	0.451 (26.1)	**0.289*** (34.9)
(k)	0.033 (1.0)	0.040 (9.3)	0.431 (16.5)	0.057 (28.2)	0.040 (30.4)	**0.022*** (6.8)
Rank	3.27	2.64	6.00	3.36	3.27	2.45

Table 6 MCE results with the support vector machines classifier

DB	FLD	NDA	Chernoff	RBDP	LBDP	BFP
(a)	0.324 (1.0)	**0.273** (3.1)	0.344 (2.2)	0.383 (1.4)	0.340 (1.8)	**0.271*** (3.2)
(b)	0.056 (1.0)	**0.041** (3.7)	0.068 (3.9)	0.370 (1.0)	0.088 (1.0)	**0.038*** (2.9)
(c)	0.152 (1.0)	**0.024** (2.3)	0.208 (7.7)	0.132 (1.5)	**0.020*** (2.3)	0.106 (4.7)
(d)	**0.028** (1.0)	**0.027** (1.3)	**0.034** (2.6)	0.042 (1.0)	**0.028** (1.7)	**0.025*** (1.8)
(e)	**0.165** (1.0)	0.221 (2.6)	0.231 (8.1)	0.448 (1.4)	0.354 (1.5)	**0.145*** (2.5)
(f)	**0.225*** (1.0)	0.272 (3.2)	0.256 (15.1)	0.294 (1.9)	0.278 (2.7)	0.248 (3.1)
(g)	0.130 (1.0)	0.097 (4.2)	**0.049*** (17.3)	0.088 (3.7)	**0.054** (9.0)	0.082 (2.9)
(h)	0.268 (1.0)	0.226 (2.1)	0.470 (1.0)	**0.105** (12.3)	**0.101*** (15.0)	0.329 (1.5)
(i)	0.248 (1.0)	**0.057** (5.5)	**0.057** (4.6)	**0.058** (1.4)	**0.059** (2.9)	**0.055*** (4.9)
(j)	0.505 (1.0)	0.435 (5.7)	0.423 (7.4)	0.446 (17.5)	**0.377*** (23.1)	0.442 (31.6)
(k)	0.142 (1.0)	0.139 (23.2)	0.122 (12.3)	**0.058*** (17.8)	**0.064** (33.0)	0.114 (12.7)
Rank	4.82	3.09	3.36	3.27	3.73	2.73

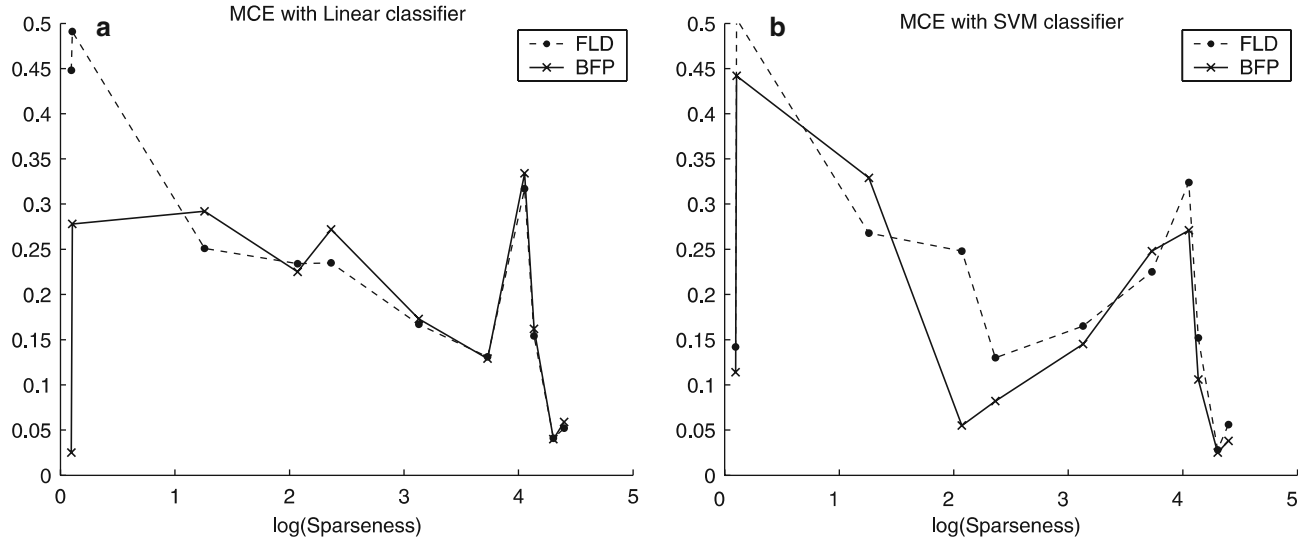

Fig. 6 Classification error versus data sparseness for the best feature extraction methods: FLD and BFP

To examine the relationship between the sparseness of the data and the feature extraction methods we plotted the mean classification error (MCE) against logarithm of the sparseness. Figure 6 shows the graphs for the two most successful feature extraction methods, Fisher's Linear Discriminant (dashed line) and the proposed variant of Adaboost feature extraction, BFP (solid line). The left subplot gives the results for the linear classifier and the right subplot gives the results with SVM.

The graphs show that sparseness is not strongly related to classification error nor is it a reliable guide as to which type of feature extraction method should be preferred. For data with large dimensionality (low value of the sparseness index), we found that the ensemble feature extraction gives lower classification error (subplot (a)). On the other hand, the features selected by BFP are more useful for SVM for data sets in the middle range of sparseness (subplot (b)). For large values of the sparseness index the results are almost identical for both classifiers. This gives us ground to propose that the simple Fisher's linear discriminant may be sufficient when the ratio data size to dimensionality is large.

BFP has two parameters to be tuned, the number of samples K, taken from each class at each step, and the number of projections P, from which we choose at each step. In all our experiments these parameters were set to $K = 100$, and $P = 1$. We believe that the proposed methods are not critically sensitive to the choice of these parameters. A sensitivity study with respect to K and P is a possible future direction.

5 Conclusions

In this paper we propose three linear feature extraction methods based on Adaboost. The main algorithm does not require making any assumption on the data distribution. At each boosting step we select from a pool of linear projections the one that minimizes the weighted error. Different algorithms can be derived from this idea depending on how the projections are selected within an Adaboost step.

Experiments were performed on 9 real and 2 artificial data sets. It seems that high-dimensional data sets are the best target for the boosted techniques, compared to the three methods based on eigenvector decomposition.

For low-dimensionality FLD often gave superior accuracy compared to the other methods, at the same time reducing the dimensionality to a single feature.

There are many possible further applications of Adaboost to feature extraction, one of which is its extension to the multiclass case. There is a variety of possibilities for such extensions. These include (but are not limited to)

- Using a straightforward extension of Adaboost such as Adaboost.M1
- Using a variant of Adaboost based on error correcting codes (ECOC) preserving the most important projections obtained for each two-class classifier.

Along with the possibilities coming from Adaboost, there are multiple choices to be made arising from the specifics of the projection generation in 3(a) in Fig. 3. For example, in LBDP, we need to select one point from the same class as $\mathbf{x}_i$, and one point from the "opposite" class. When there are more than one opposite classes, different paths can be followed:

- use the closest point from any other opposite class,
- take the mean of the nearest neighbors of the opposite classes, or
- just randomly select a class and take the closest vector from this class. As the range of options is large and there is no obvious guide as to what is best, we felt

that expanding the study towards multiple classes was beyond this study.

Different choices of individual one-dimensional projections at each step can lead to different unexplored variants of the method with different accuracy. An interesting open question is developing a guideline towards a more systematic choice of projection generation at step 3(a) in Fig. 3.

Acknowledgements This work is supported by MCYT grant TIC2003-00654, and FP2000-4960 Ministerio de Ciencia y Tecnologia, Spain.

References

1. Hyvarinen A, Karhunen J, Oja E (2001) Independent component analysis. John Wiley and Sons
2. Friedman JH (1987) Explanatory projection pursuit. J Am Statistical Assoc 82:249–266
3. Lee DD, Seung HS (1999) Learning the parts of objects with nonnegative matrix factorization. Nature 401:788–791
4. Fisher R (1936) The use of multiple measurements in taxonomic problems. Ann Eugenics 7:179–188
5. Fukunaga K, Mantock J (1983) Nonparametric discriminant analysis. IEEE T Pattern 5(6):671–678
6. Loog M, Duin RPW (2004) Linear dimensionality reduction via a heteroscedastic extension of lda: The Chernoff criterion. IEEE T Pattern Anal 26(6):732–739
7. McLachlan GJ (2004) Discriminant analysis and statistical pattern recognition. John Wiley and Sons, Inc, New York
8. RST, SLK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326
9. Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2323
10. Long PM, Vega VB (2003) Boosting and microarray data. Machine Learning 52:31–44
11. Athitsos V, Alon J, Sclaroff S, Kollios G (2004) Boostmap: a method for efficient approximate similarity rankings. In: CVPR (2), 2004, pp 268–275
12. Sirlantzis K, Hoque S, Fairhurst MC (2002) Trainable multiple classifier schemes for handwritten character recognition. In: Multiple classifier systems, 2002, pp 169–178
13. Brown G, Yao X, Wyatt J, Wersing H, Sendhoff B (2002) Exploiting ensemble diversity for automatic feature extraction. In: Proc. of the 9th international conference on neural information processing (ICONIP'02), 2002, pp 1786–1790
14. Kuncheva LI (2004) Combining pattern classifiers. John Wiley and Sons
15. Breiman L (1998) Arcing classifiers. Ann Stat 26(3):801–849
16. Kirby M, Sirovich L (1990) Application of the Karhunen-Loeve procedure for the characterization of human faces. IEEE T Pattern Anal 12(1):103–108
17. Fukunaga K (1990) Introduction to statistical pattern recognition, 2nd edn. Academic Press, Boston
18. Bressan M, Vitria J (2003) Nonparametric discriminant analysis and nearest neighbor classification. Pattern Recogn Lett 24(15):2743–2749
19. Schapire RE (1999) A brief introduction to boosting. In: IJCAI, 1999, pp 1401–1406
20. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: International conference on machine learning, 1996, pp 148–156
21. Blake C, Merz C (1998) UCI repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html
22. Skurichina M (2001) Stabilizing weak classifiers, Ph.D. thesis, Delft University of Technology
23. Martinez A, Benavente R (1998) The AR face database. Tech Rep 24, Computer Vision Center (June 1998)
24. Duin RPW (2004) PRTOOLS v3.17, Tech. rep., Delft University of Technology. http://www.ph.tn.tudelft.nl/~bob/PRTOOLS.html