Subclass Discriminant Analysis*

Manli Zhu and Aleix M. Martinez
Dept. Electrical and Computer Engineering
The Ohio State University, Columbus, OH 43210
{ zhum, aleix }@ece.osu.edu

Abstract

Over the years, many Discriminant Analysis (DA) algorithms have been proposed for the study of high-dimensional data in a large variety of problems. Each of these algorithms is tuned to a specific type of data distribution (that which best models the problem at hand). Unfortunately, in most problems the form of each class pdf is a priori unknown, and the selection of the DA algorithm that best fits our data is done over trial-and-error. Ideally, one would like to have a single formulation which can be used for most distribution types. This can be achieved by approximating the underlying distribution of each class with a mixture of Gaussians. In this approach, the major problem to be addressed is that of determining the optimal number of Gaussians per class; i.e., the number of subclasses. In this paper, two criteria able to find the most convenient division of each class into a set of subclasses are derived. Extensive experimental results are shown using five databases. Comparisons are given against Linear Discriminant Analysis (LDA), Direct LDA (DLDA), Heteroscedastic LDA (HLDA), Nonparametric DA (NDA) and Kernel-based LDA (K-LDA). We show that our method is always the best or comparable to the best.

Index terms: feature extraction, discriminant analysis, pattern recognition, classification, eigenvalue decomposition, stability criterion, mixture of Gaussians

1 Introduction

Feature extraction via Discriminant Analysis (DA) is one of the most sought after approaches in applications in computer vision [9, 24, 7, 3, 19]. The main advantage of these algorithms is that they can automatically extract a low-dimensional feature representation where the data can be easily separated according to their class labels [8, 23, 9, 22]. Unfortunately, to date, these techniques have only found success in a limited number of applications. Usually, one technique will work on some problems, while distinct algorithms will be preferred in others. And, there still exist problems for which no DA method will successfully find an adequate representation of the data [22, 20].

*IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 8, pp. 1274-1286, 2006.

The reason why each algorithm is only applicable to a limited number of applications is mainly due to the assumptions embedded in each of the methods. For example, the well-known Linear Discriminant Analysis (LDA) algorithm assumes the sample vectors of each class are generated from underlying multivariate Normal distributions of common covariance matrix but different means (i.e., homoscedastic data) [8, 23, 9]. This assumption has restricted the use of LDA considerably. Over the years, authors have defined several extensions to the basic formulation of LDA [9, 11, 10, 2, 16]. One such method is Nonparametric Discriminant Analysis (NDA) where the Normal assumption is relaxed in one of the two metrics [9]. However, NDA still assumes the data in each class can be grouped in a single (non-disjoined) cluster. Hence, if the data of a class is divided into two or more clusters, NDA will not generally work. Another alternative is to use a weighted version of LDA, such as the approximate Pairwise Accuracy Criterion (aPAC) [16] or Penalized DA (PDA) [11]. Here, weights are introduced in the definition of the metrics to reduce (or penalized) the role of the least stable samples (or features). Such weights allow us to slightly deviate from the Normal assumption, by making the metrics of DA a bit more flexible. These methods generally outperform LDA in practise, because the data is not usually Gaussian. However, these algorithms also assume each class is represented by a single cluster and, therefore, none of the methods defined in this paragraph can solve the problem posed by non-linearly separable classes. Fig. 1 shows an example. In this example, none of the methods described above would be able to find that one dimensional feature vector which minimizes the Bayes error.

To solve this, we could use non-linear methods. Flexible Discriminant Analysis (FDA) [10] attempts to include the idea of non-linear fitting in DA. To do that, FDA reformulates the DA problem as a regression one and, then, uses a non-linear function (e.g., such as a polynomial of order larger than two) to fit the data. Alternatively, we could search for a non-linear function that maps the data into a space of higher dimensionality where the classes are linearly separable. This is the well-known idea behind the use of kernels. Generalized Discriminant Analysis (GDA) [2] is one such method. In GDA a kernel is first used to embed the data into a space where the data is (hopefully) linearly separable and, then, LDA is used to search for those features that best divide the classes. Unfortunately, three main problems prevent the efficient use of such techniques. The first one is that of finding the appropriate kernel for each particular problem (i.e., for each set of data distributions). The second problem posed by non-linear approaches (such as the kernel trick and FDA) is that one generally requires of a very large number of samples to successfully apply those algorithms. And, third, non-linear methods usually have an associated high computational cost, both in training and testing.

Our goal is to propose an algorithm that can adapt to the data at hand. This means we need to find a way to describe a large number of data distributions, regardless of whether these correspond to compact sets or not. One way to do this, is to approximate the underlying distribution of each class as a mixture of Gaussians. Such mixtures are flexible enough to represent a large number of those distributions typically encountered in real applications [21, 13]. For example, in our Fig. 1, this means we will need one Gaussian to successfully describe the distribution of the first class, and two Gaussians to represent the data of the second. Once the data distribution of each class has been approximated using a mixture of Gaussians, it is easy to use the following generalized eigenvalue decomposition equation to find those discriminant vectors that best (linearly) classify the data,

$$\Sigma_B \mathbf{V} = \Sigma_X \mathbf{V} \Lambda, \tag{1}$$

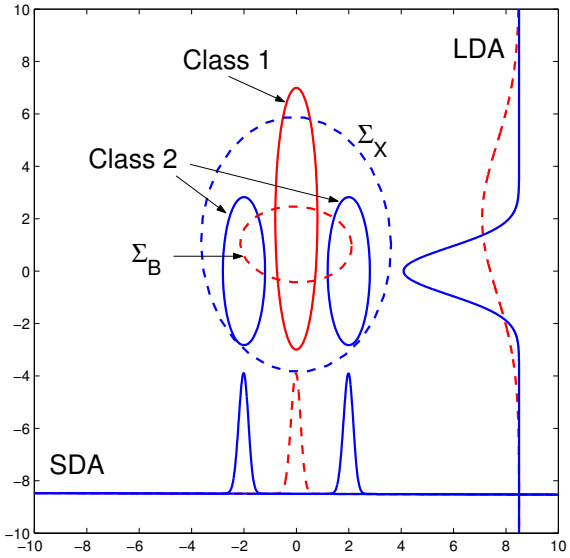


Figure 1: *In this example, the first class is represented by a single Gaussian distribution, while the second is represented by two separated Gaussians. Linear DA algorithms will not be able to successfully reduce the dimensionality of the original feature space to one, because the second class corresponds to two disjoint distributions. One can solve this problem by dividing the second class into two subclasses. Here* Σ_X *and* Σ_B *represent the sample covariance matrix and the between-subclass scatter matrix (see text).*

where Σ_B is the *between-subclass* scatter matrix, Σ_X is the covariance matrix of the data, $\mathbf{V}$ is a matrix whose columns correspond to the discriminant vectors, and Λ is a diagonal matrix of corresponding eigenvalues. Since Σ_B measures the scatter of the subclass means, we will refer to this method as *Subclass Discriminant Analysis* (SDA).

It is important to note that the difficulty in Eq. (1) is not given by the way we compute the discriminant vectors. *The real challenge (and our goal) is to find that division of the classes into a set of subclasses so that the classification in the reduced space of* $\mathbf{V}_q$ (where $\mathbf{V}_q$ is a $p \times q$ matrix which maps the original space of p dimensions to one of q, for some $q \leq p$) *is maximized*.

This paper defines simple criteria that can be used to determine those subclass divisions that optimize classification when used in a linear discriminant setting such as that defined in Eq. (1). In particular, we present two criteria. Our first solution uses the leave-one-out test to find the optimal division. Here, the samples left out are used to determine that subdivision which works best. Although this is a reasonably good approach, it comes with an associated high computational cost. To solve this problem, we will introduce a second method which takes advantage of the fact that Eq. (1) is not guaranteed to work when the smallest angle between the i^{th} eigenvector given by the metric to be maximized and the first i eigenvectors given by the metric to be minimized is close to zero [20]. We will show how we can use this known fact to define a fast, easy to use criterion.

Note that our approach (defined in the preceding paragraphs) differs from previous algorithms where the EM (Expectation-Maximization) algorithm [5] is first used to estimate the true underlying distribution of each class, before using LDA [26, 12, 19]. Our goal is to optimize classification, not to recover the true underlying (but unknown) distribution of the data. Moreover, the methods presented in [26, 12, 19] can only be applied when the number of samples is very large. Otherwise,

one cannot efficiently use the EM algorithm to fit a mixture of Gaussian distributions.

In Section 2, we show that SDA can successfully resolve all the problems addressed by the other methods reported in the literature thus far. In this section, we will also justify our definition of Σ_B. Section 3 defines the two criteria used to determine the optimal number of subclasses. Experimental results are in Section 4. We conclude in Section 5.

2 The Metrics of Discriminant Analysis

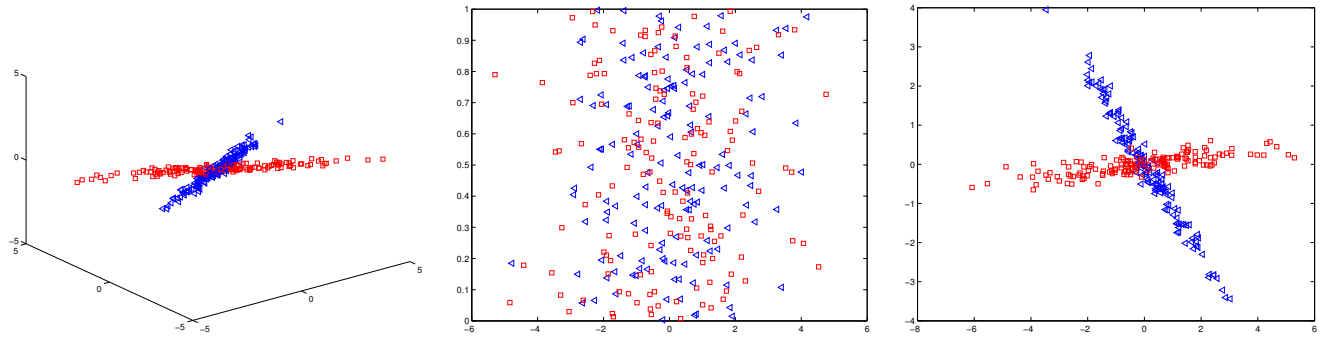
Most DA algorithms defined thus far are based on Fisher-Rao's criterion [8, 23], which is given by

$$\frac{|\mathbf{v}^T \mathbf{A}\, \mathbf{v}|}{|\mathbf{v}^T \mathbf{B}\, \mathbf{v}|}, \tag{2}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$, are assumed to be symmetric and positive-definite, so that they define a metric. It is well-known that LDA uses the between- and within-class scatter matrices, $\mathbf{A} = \mathbf{S}_B$ and $\mathbf{B} = \mathbf{S}_W$ resp., in Eq. (2); where $\mathbf{S}_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$, C is the number of classes, μ_i the sample mean of class i, μ the global mean (including the samples of all classes), $\mathbf{S}_W = \frac{1}{n}\sum_{i=1}^{C}\sum_{j=1}^{n_i}(\mathbf{x}_{ij}-\mu_i)(\mathbf{x}_{ij}-\mu_i)^T$, $\mathbf{x}_{ij}$ is the j^{th} sample of class i, and n_i the number of samples in that class. In the section above, we have introduced several extensions to the classical LDA algorithm as defined by Fisher and Rao. Such modifications usually re-define one of these two metrics, $\mathbf{A}$ or $\mathbf{B}$. For example, NDA uses the following non-parametric version of the between-class scatter matrix for $\mathbf{A}$, $\mathcal{S}_B = \frac{1}{n}\sum_{i=1}^{C}\sum_{j=1}^{n_i}\sum_{\substack{l=1\\l\neq i}}^{C}\alpha_{ij}^{l}\left(x_{ij}-\mathcal{M}_{ij}^{l}\right)\left(x_{ij}-\mathcal{M}_{ij}^{l}\right)^T$; where $\mathcal{M}_{ij}^{l}$ is the mean of the k nearest samples to $\mathbf{x}_{ij}$ belonging to class $l(l \neq i)$ and α_{ij}^{l} is any scale factor which prevents the results to be affected by isolated samples located far from the boundary. Other DA algorithms also rework the definition of the between-class scatter matrix. aPAC is such an algorithm where the new scatter matrix is a weighted version of $\mathbf{S}_B$; i.e., $\sum_{i=1}^{C-1}\sum_{j=i+1}^{C}\omega\left(d_{ij}\right)\mathbf{S}_{ij}$, where $\mathbf{S}_{ij} = (\mu_i - \mu_j)(\mu_i - \mu_j)^T$, d_{ij} is the Mahalanobis distance between classes i and j, and $\omega(\cdot)$ is a weight function which makes the contribution of each class pair be equal to the classification accuracy (one minus the Bayes error) between these two classes up to an additive constant. Other methods modify the definition of $\mathbf{B}$. For instance, PDA includes a penalizing matrix Ω in the definition of the within-class scatter matrix, $\mathbf{B} = \mathbf{S}_W + \Omega$. Here, Ω penalizes those eigenvectors of Eq. (2) that are noisy (e.g., by assigning weights that are proportional to the second derivative over the components in each eigenvector).

Each of these methods solves a specific problem where LDA is known to fail. And, therefore, each method will be best when applied to a specific problem. Earlier, we mentioned how SDA can be used to efficiently solve all the problems addressed by each of the methods just defined. For example, SDA can represent each class as a mixture of Gaussians to solve the problem targeted by NDA. And, as illustrated in Fig. 1, SDA can also be used to classify non-linearly separable classes if one divides each class into a set of disjoint clusters.

Another problem that was not discussed above but is common to most DA methods, is given by the deficiency in the rank of $\mathbf{A}$. For example, the $rank(\mathbf{S_B}) \leq C-1$ and, therefore, LDA, PDA, aPAC and other DA methods can only extract $C-1$ features from the original feature space. Shall the classes be linearly separable in the original space, $C-1$ features would guarantee



(a) (b) (c)

Figure 2: *(a) Shows a two-class example where the data is embedded in a three-dimensional space. A two-dimensional representation suffices to optimally discriminate between the samples of the two classes. (b) LDA cannot find such a two-dimensional solution, because the rank of* $\mathbf{S}_B$ *is one. Here, the horizontal axis corresponds to the 1-dimensional space found by LDA. For easy view, the data has also been uniformly distributed about the vertical axis. (c) SDA can recover this two-dimensional discriminant subspace by dividing each of the classes into two subclasses.*

to be sufficient to discriminate the C classes (assuming that the data distributions are correctly represented by our metrics $\mathbf{A}$ and $\mathbf{B}$) [9]. In practice, however, the data is rarely linearly separable and the $C-1$ features obtained are consequently not the optimal ones. To address this problem, some researchers have defined alternative metrics that can be used to extract more than $C-1$ features. This can then be used to define a subspace of larger dimensionality where the data is separable.

An example of this is shown in Fig. 2(a). In this example, LDA cannot recover the two-dimensional space necessary to successfully discriminate both classes, because $rank(\mathbf{S}_B) = 1$; Fig. 2(b). This problem can be solved by defining a metric for $\mathbf{A}$ that not only considers the difference between class means but also between their covariances. Heteroscedastic LDA [17] (HLDA) is such a method, where the authors use the Chernoff distance to estimate class similarity based on both means and covariances; i.e., $\mathbf{A}$ is redefined as

$$\mathbf{S}_C = \sum_{i=1}^{C-1}\sum_{j=i+1}^{C}\left[\Sigma_{ij}^{-1/2}(\mu_i-\mu_j)(\mu_i-\mu_j)^T\Sigma_{ij}^{-1/2} + 4\left(\log\Sigma_{ij} - \frac{1}{2}\log\Sigma_i - \frac{1}{2}\log\Sigma_j\right)\right], \quad (3)$$

where Σ_i is the covariance matrix of the samples in class i, Σ_{ij} the average between Σ_i and Σ_j, and we have assumed equal priors. This algorithm can now be used to find the two-dimensional subspace where the classes are linearly separable.

Note, however, that by dividing the data of each class into a set of subclasses, we can also define scatter matrices whose ranks are (in general) larger than $C-1$. For instance, following the most traditional definition of a scatter matrix in LDA [8, 23, 9], one could define the between-subclass scatter matrix as [31],

$$\widehat{\Sigma}_B = \sum_{i=1}^{C}\sum_{j=1}^{H_i} p_{ij}(\mu_{ij}-\mu)(\mu_{ij}-\mu)^T, \quad (4)$$

where H_i is the number of subclass divisions in class i, and p_{ij} and μ_{ij} are the prior and mean of the j^{th} subclass in class i. Such a definition is consistent with the theory of DA. Indeed, further

analysis shows that Eq. (4) works similarly to LDA in that it simultaneously attempts to maximize the distance between the class means and between the subclass means in the same class. This is formally summarized in the following result.

Result 1. *The covariance matrix of the subclass means $\widehat{\Sigma}_B$ can be decomposed as $\widehat{\Sigma}_B = \mathbf{S}_B + \mathbf{Q}$ with $\mathbf{Q} = \sum_{i=1}^{C} \frac{n_i}{n}\mathbf{Q}_i$, and where $\mathbf{Q}_i$ measure the distance between the means of the subclasses in the same class only; i.e., $\mathbf{Q}_i = \sum_{j=1}^{H_i} \frac{n_{ij}}{n_i}(\mu_{ij} - \mu_i)(\mu_{ij} - \mu_i)^T$, and n_{ij} is the number of samples in the j^{th} subclass of class i.*

This result (the proof of which is in Appendix A) shows that when we use Eq. (4) in Eq. (1), we will find those basis vectors that maximize the distance between both, the class means and the means of the subclasses that correspond to the same class. Shall we have the correct division of classes into their optimal number of subclasses, this definition will work well. To see this, note that the classes only need to be divided into subclasses when these also correspond to distinct clusters in the reduced space. An example of this was illustrated in Fig. 1. In practise, however, the data may sometimes be over-segmented into too many subclasses. This may be due to complex nonlinear distributions, outliers or other factors. When this happens, it is convenient to define a Σ_B that favors the separability of those subclasses that correspond to different classes. Formally,

$$\Sigma_B = \sum_{i=1}^{C-1}\sum_{j=1}^{H_i}\sum_{k=i+1}^{C}\sum_{l=1}^{H_k} p_{ij}\, p_{kl}\, (\mu_{ij} - \mu_{kl})(\mu_{ij} - \mu_{kl})^T, \tag{5}$$

where $p_{ij} = \frac{n_{ij}}{n}$ is the prior of the j^{th} subclass of class i.

The advantage of this new definition of between-subclass scatter is that it emphasizes the role of class separability over that of intra-subclass scatter. However, it is important to note that the information of the latter is not lost. Knowing the distances from a distribution f_1 to two others, f_2 and f_3, also provides information on the distance between f_2 and f_3. Therefore, although indirectly, Eq. (5) also measures the distance between the subclasses in the same class. This means that if some subclasses (belonging to the same class) need to be separated, these will (because the required information has not been lost). However, since Eq. (5) favors class separability, only those subclasses that are essential to boost the classification result will actually be separated in the reduced space.

We also note that Eq. (5) is equal to $\mathbf{S}_B$ when $H_i = 1$ for all $i = \{1, \ldots, C\}$. And, since it can be readily shown that the $rank(\Sigma_B) \leq \min(H - 1, p)$ (where $H = \sum_{i=1}^{C} H_i$ is the total number of subclass divisions and p is the dimensionality of the range of the covariance matrix), SDA can also solve the problem posed by the deficiency of the rank of $\mathbf{S}_B$. An example of this is shown in Fig. 2. In this example, the range of Σ_X is three, $p = 3$, the number of classes is two, $C = 2$, and the number of samples $n = 100$. When using SDA we can vary the value of H from a low of 2 to a high of 100, which produces a set of solutions with $1 \leq rank(\Sigma_B) \leq 3$. One such solution, for $H = 4$ is shown in Fig. 2(c), in which case the result is optimal. Therefore, as stated earlier, the goal is now reduced to finding a way to automatically select the optimal number of subclass divisions.

3 Optimal Subclass Divisions

The most convenient criterion to use for the selection of the optimal number of subclasses is the leave-one-out-test (LOOT). In this case, we use all but one sample for training ($n-1$ samples) and then test whether our results generalize to the sample left out. Although convenient, this solution is computationally expensive. In this section, we will show that we can also use a recent result which shows that the basis vectors obtained by any generalized eigenvalue decomposition equation are not guaranteed to be correct when the smallest angle between the i^{th} eigenvector given by the metric to be maximized (i.e., $\mathbf{A}$) and the first i eigenvectors given by the metric to be minimized ($\mathbf{B}$) is close to zero. We will conclude this section with a discussion on how to cluster the data into the set of possible subclass divisions we want to analyze with our criteria.

3.1 Leave-one-out-test criterion

For each possible value of H (i.e., number of subclasses), we work as follows. We first divide the data into H subclasses and then calculate the projection matrix $\mathbf{V}_q$ from Eq. (1) using all but the i^{th} sample of the training data, $\mathbf{X}_i = \{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n\}$. The sample left-out, $\mathbf{x}_i$, is used for testing. This means we will need to repeat this procedure n times – one for each of the samples we can leave out.

Now, we let $r_{i,H} = 1$ if we correctly classify $\mathbf{x}_i$ when we use $\mathbf{X}_i$ for training. Otherwise, we let $r_{i,H} = 0$. The recognition rate for a fixed value of H is then given by

$$R_H = \frac{1}{n}\sum_{i=1}^{n} r_{i,H}.$$

The optimal value of H, H_o, is

$$H_o = \underset{H}{argmax}\ R_H. \tag{6}$$

This allows us to compute the optimal Σ_B with H_o. The resulting projection matrix $\mathbf{V}_q^*$ is a $p \times q$ matrix whose columns are the eigenvectors associated to the q largest eigenvalues of Eq. (1) when $H = H_o$. This is summarized in Algorithm 1.

Since H_o is chosen according to how well our classifier generalizes to new samples, this leave-one-out-test criterion generally gives an appropriate value for H.

3.2 Stability criterion

Our second approach is based on a stability criterion which we have recently introduced in [20]. In particular, we showed that the success of any linear method, such as that given in Eq. (1), does not depend on whether the data is homoscedastic or not. We have demonstrated that the results of the linear methods based on Eq. (2) are not guaranteed to be correct when the smallest angle between the i^{th} eigenvector of $\mathbf{A}$ and the first i eigenvectors of $\mathbf{B}$ is close to zero. We will now show how we can use this result to find the optimal value of H. This will lead to the design of a low cost algorithm that will facilitate the use of SDA in a larger number of applications.

To understand the reason why an algorithm that is based on a generalized eigenvalue decomposition equation (such as Eq. (1)) may not produce the most desirable result, we need to go back to the Fisher-Rao criterion given in Eq. (2). Note that in this equation, we want to simultaneously maximize the measure given

Algorithm 1 SDA_{loot}

Initialization: $p_H = 0, \forall H$.
for $i = 1$ to n **do**
  Generate the training set $\mathbf{X}_i = \{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n\}$ using the NN clustering defined in Algorithm 3.
  Calculate Σ_X using $\mathbf{X}_i$.
  for $H = C$ to $t \cdot C$ † **do**
    Calculate Σ_B using Eq. (5).
    Perform the eigenvalue decomposition $\Sigma_X^{-1}\Sigma_B\mathbf{V} = \mathbf{V}\Lambda_X$.
    Project the data onto the subspaces of $\mathbf{V}_q$; i.e., $\mathbf{Y}_i = \mathbf{V}_q^T\mathbf{X}_i$.
    if the sample $\mathbf{x}_i$ is classified correctly **then**
      p_H++
    end if
  end for
end for
$H_o = \underset{H}{argmax}\ p_H$.
Calculate Σ_B and Σ_X using all our training data and H_o.
The final projection matrix $\mathbf{V}_q^*$ is given by the first q columns of $\mathbf{V}$, where $\Sigma_X^{-1}\Sigma_B\mathbf{V} = \mathbf{V}\Lambda_X$.

† The value of t can be specified by the user or be set so as to guarantee that the minimum number of samples per subclass is sufficiently large.

by the scatter matrix $\mathbf{A}$ (which in our case is Σ_B) and minimize that computed by $\mathbf{B}$ (in our case, $\mathbf{B} = \Sigma_X$). Unfortunately, there may be some cases where this cannot be accomplished in parallel. This problem is illustrated in Fig. 3. In this example, $\mathbf{u}_i$ ($\mathbf{w}_i$ resp.) is the eigenvector of Σ_X (Σ_B resp.) associated to the i^{th} largest eigenvalue. In Fig. 3(a), we show an example where Eq. (1) fails. This is so, because, in this particular case, it is not possible to maximize Σ_B and minimize Σ_X simultaneously. As seen in the figure, one would want to select $\mathbf{u}_2$ as a solution to minimize Σ_X but $\mathbf{w}_1$ to maximize Σ_B. Since it is impossible to solve this, Eq. (1) will choose that direction $\mathbf{v}_1$ associated to the largest relative variance as given by Eq. (2). In our Fig. 3(a), this means that we will select $\mathbf{u}_2$ as our solution, which leads to an incorrect classification.

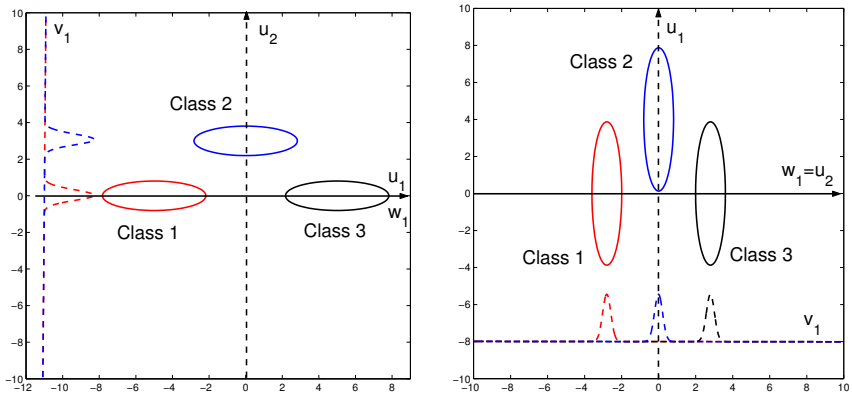
Fortunately, this problem can be easily detected because when this happens the angle between the first eigenvectors of Σ_B and Σ_X is small (e.g., in Fig. 3(a) it is zero). This can be formally computed as [20],

$$K = \sum_{i=1}^{m}\sum_{j=1}^{i}\left(\mathbf{u}_j^T\mathbf{w}_i\right)^2, \tag{7}$$

where $m < rank(\Sigma_B)$. As argued above and as shown in Fig. 3, one wants the value of K defined above to be as small as possible. For example, in Fig. 3(b) we illustrate a case where the cosine of the angle between $\mathbf{w}_1$ and $\mathbf{u}_1$ is zero and, therefore, the result $\mathbf{v}_1$ is correct. This is so, because now the first eigenvector of Σ_B and the last of Σ_X agree; i.e., $\mathbf{w}_1 = \mathbf{u}_2$. That is to say, one would always want the ideal case, where the two metrics (that to be maximized and that to be minimized) agree. To achieve this, we can define a method that divides each class into that number of subclasses which minimizes Eq. (7).

To efficiently use this criterion, we will need to compute the value of Eq. (7) for each of the possible values of H and then select the minimum. This means we will first calculate

$$K_H = \sum_{i=1}^{m}\sum_{j=1}^{i}\left(\mathbf{u}_j^T\mathbf{w}_{H,i}\right)^2, \tag{8}$$



(a) (b)

Figure 3: *(a) In this example, the first basis vector obtained with Eq. (1) does not minimize the classification error. Note that* $\mathbf{u}_1$ *(which is orthogonal to* $\mathbf{v}_1$ *would be the optimal and desirable solution). Close analysis of this figure shows that the angle between the first eigenvector of* Σ_B *and* Σ_X *is zero. This means that it is not possible to maximize the metric given by* Σ_B *and minimize that of* Σ_X *simultaneously. (b) A classical example where most linear equations perform as expected. Note that in this case the angle between the first eigenvectors of* Σ_B *and* Σ_X *is large (i.e.,* 90^o*). This is formally computed by Eq. (7).*

where $\mathbf{w}_{H,i}$ is the i^{th} eigenvector of Σ_B when the data is divided by a total of H subclasses. The problem of selecting H now reduces to finding that H_o which minimizes Eq. (8).

It is important to note that the larger the value of H, the larger the number of eigenvectors in Σ_B (or, equivalently, the larger m can be). Unfortunately, when this happens, the number of summations in Eq. (8) increases. This means that K_H will generally grow with m. This calls for a normalization step.

From elementary geometry we know that

$$\sum_{j=1}^{p} \left(\mathbf{u}_j^T \mathbf{w}_{H,i}\right)^2 = \left(\cos^2\theta_{i,1} + \cos^2\theta_{i,2} + \cdots + \cos^2\theta_{i,p}\right) = 1,$$

where $\theta_{i,j}$ is the angle between $\mathbf{u}_j$ and $\mathbf{w}_{H,i}$ and $p = rank(\Sigma_X)$. In other words, the inner summation in (8) is always guaranteed to be bounded by zero and one, $\sum_{j=1}^{i} \left(\mathbf{u}_j^T \mathbf{w}_{H,i}\right)^2 \in [0,1]$.

The above argument shows that Eq. (8) depends on the value of m. The larger m is, the larger Eq. (8) can be. To prevent our algorithm to be biased toward small values of H, we need to normalize Eq. (8) with $1/m$. Our optimal number of classes is therefore given by

$$H_o = \arg\min_H \frac{1}{m} K_H. \tag{9}$$

This second criterion is summarized in Algorithm 2. We see that the complexity of this algorithm is of the order of $n^2p + lp^3$, while that of our LOOT-based SDA had a complexity larger than $n^3p + tnp^3$ (recall that n is the number of sample and p is the dimensionality of the range of Σ_X). Note that regardless of the values of t and l, the complexity of SDA using the stability criterion is much smaller than that of the leave-one-out test. This complexity can still be further constrained by the clustering algorithm used to separate the classes into subclasses. This is to be defined next.

Algorithm 2 $SDA_{\texttt{stability}}$

Sort the training samples using the NN clustering defined in Algorithm 3.
Calculate Σ_X.
Calculate $\Sigma_X U = U\Lambda_X$.
for $H = C$ to $l \cdot C$ [†] **do**
  Calculate Σ_B.
  Compute $\Sigma_B W = W\Lambda_B$.
  Calculate K_H using Eq. (8).
end for
$H_o = \underset{H}{argmin}\ K_H$
Calculate Σ_B with $H = H_o$.
The final projection matrix $\mathbf{V}_q^*$ is given by the first q columns of $\mathbf{V}$, where $\Sigma_X^{-1}\Sigma_B V = V\Lambda_X$.

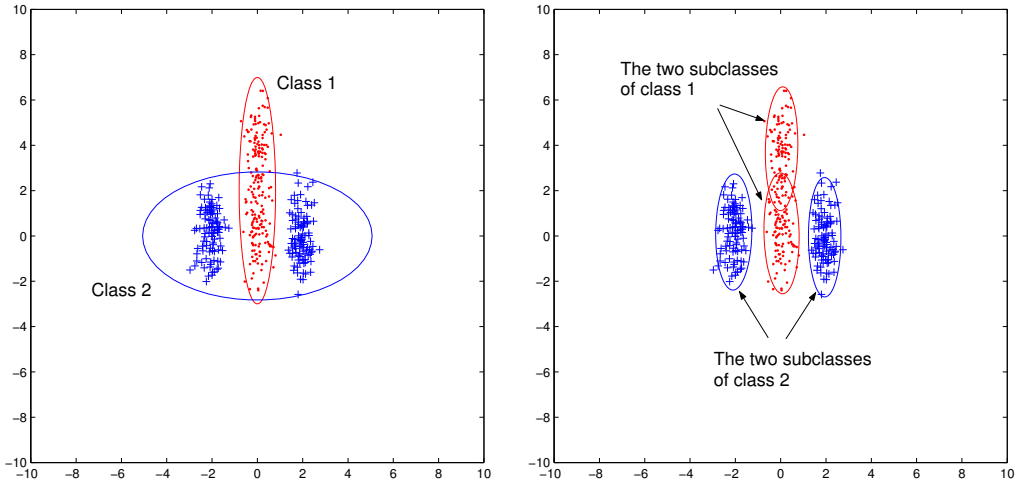
[†] Again, the value of l can be specified by the user or be set so as to guarantee that the minimum number of samples per subclass is sufficiently large.

3.3 Dividing classes into subclasses

Thus far, we have defined two criteria that can be used to determine the optimal number of subclass divisions. We now turn to another related problem – that of deciding how to divide each class into a set of possible subclasses. For computational reasons, it is quite obvious that we cannot test every possible division of n samples into H subclasses for every possible value of H. However, when these subclasses are assumed to be Gaussian, their samples will cluster together. This is easy to prove from the known fact that Gaussian distributions are convex.

This means we can use a clustering approach to divide the samples in each class into a set of subclasses (i.e., clusters). And although the number of clustering methods available is very large, we want one that can efficiently work when the number of samples in each class is either large or small. This means that the K-means algorithm or the estimation of a Gaussian mixture are not adequate, because such methods do not work properly when the number of samples is small. When this happens, one is usually left to use non-parametric methods such as those based on the Nearest Neighbor (NN) rule [6]. In this section, we define a NN-based algorithm to be used as clustering method in SDA. We will nonetheless compare our results to those obtained with K-means and Gaussian mixtures. We will see that the classification results obtained when one uses our NN-clustering algorithm are always superior or equivalent to the ones obtained when we use the K-means or the mixture of Gaussians instead.

The reader may have already noticed that our clustering method is non-parametric, whereas the other two algorithms mentioned above (K-means and Gaussian mixture) are parametric. Hence, we will also compare our results to those obtained with the non-parametric clustering (valley-seeking) algorithm of Koontz and Fukunaga [9]. In this case, we will show that the classification results obtained with our method and that of Koontz and Fukunaga are equivalent. However, our method presents two important advantages. 1^{st}*)* While in the clustering approach of Koontz and Fukunaga one needs to optimized a parameter (which controls the size of the local area where computations are carried out), our method is parameter-free. 2^{nd}*)* The algorithm of Koontz and Fukunaga is iterative in nature and, therefore, associated with a large computational cost and is not guaranteed to converge.



(a) (b)

Figure 4: *(a) Shown here are the two original Gaussian distributions of the data as given by the mean and covariance matrix of the samples in each of the two classes. (b) Each class is now described using two Gaussian distributions. These have been estimated using the NN-based approach summarized in Algorithm.3.*

NN clustering

Our clustering algorithm (Nearest Neighbor-based) first sorts the vectors in class i so that the set $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$ is constructed as follows. $\mathbf{x}_{i1}$ and $\mathbf{x}_{in_i}$ are the two most distant feature vectors of class i; i.e., the Euclidean distance between these two vectors is the largest, $argmax_{jk}\|\mathbf{x}_{ij} - \mathbf{x}_{ik}\|_2$, where $\|\mathbf{x}\|_2$ is the norm-2 length of $\mathbf{x}$. $\mathbf{x}_{i2}$ is the closest feature vector to $\mathbf{x}_{i1}$; and $\mathbf{x}_{i(n_c-1)}$ is the closest to $\mathbf{x}_{in_c}$. In general, $\mathbf{x}_{ij}$ is the $j-1^{th}$ feature vector closest to $\mathbf{x}_{i1}$, and $\mathbf{x}_{i(n_c-j)}$ is the j^{th} closest to $\mathbf{x}_{in_c}$.

We can now readily use this sorted set $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$ to divide the data in each class into H_i subclasses. For example, we can divide the data into two equally balanced (in the sense of having the same number of samples) clusters ($H_i = 2$) by simply partitioning this set into two parts $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{i\lfloor n_i/2 \rfloor}\}$ and $\{\mathbf{x}_{i(\lfloor n_i/2 \rfloor+1)}, \ldots, \mathbf{x}_{in_i}\}$. Or, generally, we can divide each class into h (equally balanced) subclasses; i.e., $H_i = h$, $\forall i$. This is suitable for those cases where *i)* the underlying distribution of each of the classes is not Gaussian, but can be represented as a combination of two or more Gaussians, or *ii)* the classes are not separable, but the subclasses are.

Fig. 4 shows an example. In Fig. 4(a) we show the original clustering given by the labels associated to each of the samples. Then, after sorting the data with the algorithm just described (and summarized in Algorithm 3) and dividing the data in each class into two clusters (i.e., Gaussians), we obtain the result shown in Fig. 4(b).

In our method, we have assumed that every class is divided by the same number of subclasses h at each given time. Although this may seem simplistic, it is based on the fact that we can use two Gaussian distributions to represent the data generated by a single Gaussian (but not vice versa). Moreover, recall that we already took care of the problems caused by over-segmentation when we defined Σ_B in Section 2. Our definition ensures that between sub-class scatter is only used when needed.

Algorithm 3 NN clustering

for $i = 1$ to C **do**
  Let $\hat{X}_i = \{\hat{x}_{i1}, \cdots, \hat{x}_{in_i}\}$ be the samples in class i.
  Construct the matrix $\mathbf{D} = \{d_{ij}\}$ where d_{ij} is the Euclidean distance between samples i and j; i.e., $d_{jk} = \|\hat{x}_{ij} - \hat{x}_{ik}\|_2$.
  Let $x_{i1} = \hat{x}_{is}$ and $x_{in_i} = \hat{x}_{ib}$, where $[s, b] = \underset{j,k}{argmax}\ d_{jk}$.
  for $g = 1$ to $\lfloor n_i/2 \rfloor$ † **do**
    Let $x_{ig+1} = \hat{x}_{im}$, where $m = \underset{j \neq s}{argmin}\ d_{sj}$.
    $d_{sm} = \infty$.
    Let $x_{in_i-g} = \hat{x}_{ik}$, where $k = \underset{j \neq b}{argmin}\ d_{bj}$.
    $d_{bk} = \infty$.
  end for
  Divide the resulting sorted set $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$ into H_i subclasses. In doing this, try to keep the same number of samples in each subclass.
end for

†For the case where $mod(n_i/2) \neq 0$, there will be a sample left to be sorted. This will obviously correspond to $x_{i,\lfloor n_i/2 \rfloor + 1}$.

4 Experimental Results

DA algorithms have been applied to a large variety of problems – especially in computer vision. In this section, we will first show results on three datasets of the UCI repository available at [1]. Moreover, we will show how our algorithm applies to two particular problems in computer vision: *a)* categorization of images of objects, and *b)* face recognition. We will compare our results to LDA [23, 8], Direct LDA (DLDA) [28], Nonparametric DA (NDA) [9], Heteroscedastic LDA (HLDA) [17], Kernel LDA (K-LDA) [2, 27] and Complete Kernel Fisher Discriminant analysis (CKFD) [25]. In K-LDA, we have used a second degree polynomial kernel, $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})^2$, which we will refer to as the KP-LDA method, and a Gaussian kernel, $k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma}\right)$, that we named KG-LDA. Note that in KG-LDA we need to specify the value of σ. This value will be optimized for each of the datasets by means of the cross-validation procedure. The polynomial kernel used in CKFD is $k(\mathbf{x}, \mathbf{y}) = \left(\mathbf{x}^T\mathbf{y} + 1\right)^a$ where a is now estimated through cross-validation. We refer to this method as P-CKFD. Similarly, the Gaussian kernel defined above is used to construct G-CKFD.

We also note that both, NDA and HLDA, have a free parameter to be specified by the user. In NDA, this is the number of K nearest neighbors. In HLDA, it is the number of dimensions of the reduced space. Since the results vary depending on the value of such parameters, we have decided to calculate the results of NDA and HLDA for a typical range of values and then report on the mean and standard deviation of these. For the NDA case, this means we will vary K from a low of 5 to maximum of $n' - 1$, where $n' = \min_i n_i$ and n_i is the number of samples in class i. In HLDA, we vary the number of dimensions from a low of $C - 1$ (which is the value used by LDA) to the $\min\{C + 40, p - 1\}$.

4.1 UCI datasets

The first of the databases used was the "Wisconsin Diagnostic Breast Cancer" (WDBC) set. In this database, we have thirty parameters defining the shape and texture of the nucleus of the cells to be analyzed. The task

Table 1: Results on the UCI datasets.

Database\Method	LDA	NDA	HLDA	DLDA	KP-LDA	KG-LDA	P-CKFD	G-CKFD	$SDA_{stability}$	SDA_{loot}
WDBC	0.94	0.73 (0.08)	0.95 (0.009)	0.87	0.80	0.95	0.94	0.90	0.94	0.94
LSD	0.84	0.48 (0.11)	0.88 (0.011)	0.86	0.85	0.91	0.85	0.88	0.88	0.87
MDD-pix	0.93	0.84 (0.13)	0.82 (0.064)	0.95	0.97	0.98	0.98	0.98	0.96	0.96
MDD-fou	0.81	0.70 (0.10)	0.83 (0.008)	0.80	0.82	0.85	0.85	0.77	0.83	0.81
MDD-fac	0.97	0.79 (0.20)	0.96 (0.006)	0.91	0.96	0.97	0.98	0.98	0.96	0.96
MDD-kar	0.96	0.90 (0.08)	0.97 (0.006)	0.96	0.99	0.99	0.98	0.98	0.97	0.97
MDD-zer	0.79	0.69 (0.08)	0.79 (0.004)	0.79	0.81	0.83	0.82	0.83	0.79	0.81

Shown here are the classification results (in percentage) obtained using the nearest neighbor rule in the reduced spaces given by each of the methods. Results for NDA and HLDA are obtained by averaging over a large set of possible parameters. Values shown in prentices correspond to the standard deviation from such means.

is to discriminate between benign and malignant cells. The 569 samples available are randomly divided into a training set 285 samples and a testing set of 284 testing instances. This guarantees that the sample-to-dimension ratio is appropriate. The discriminant methods mentioned above are first used to find a low-dimensional representation of the data and, then, the nearest neighbor algorithm is used to classify each of the testing samples according to the class label of the closest training vector. These results are summarized in Table 1. The results shown for NDA and HLDA correspond to the average and standard deviation (which is in parentheses) over the range of values of their parameters as described above.

The second dataset used from the UCI database was the "Landset Satellite Data" (LSD). In this case, the data is already divided into a training set of 4435 samples and a testing set of 2000 instances. Each sample is a feature vector of 36 dimensions which define the spectral information of the sensors located in a satellite. Here, the goal is to classify the data into six different land types. The results are also summarized in Table1.

The final test shown in Table 1 was obtained using the "Multi-feature Digit Dataset" (MDD). In MDD, the task is to classify each of the images of handwritten digits into one of the 10 possible classes. These digits can be represented using a variety of feature sets. The first one, referred to as "MDD-fou," represents the digits using 76 of the Fourier coefficients of the image. The second, "MDD-fac," uses a set of 216 profile correlations to describe each of the images. The third, "MDD-kar," is based on a PCA representation of the image pixels. The fourth, "MDD-pix," uses the averages of a window of two by three pixels scanned over all the possible image locations, which generates feature vectors of 240 dimensions. And, fifth, Zernike moments are used to obtained a feature representation of 47 dimensions – referred to as "MDD-zer."

We see that the results produced by the two criteria for SDA defined in this paper are equivalent. Furthermore, these are among the best. In particular, KG-LDA is probably the most comparable to SDA. However, the computational time required to train and test the KG-LDA algorithm on the UCI datasets defined above, was about one order of magnitude (or more) over that of $SDA_{stability}$.

In Table 1, we have also shown the results obtained with CKFD (with a polynomial and a Gaussian kernel). The reason for showing this results was that this method uses cross-validation to optimize the three parameters of the algorithm. These parameters are: that kernel variable, the dimensionality of the reduced space generated by the algorithm, and a fusion coefficient which task is to combine the results obtained in the range and null spaces of $\mathbf{S}_W$. Again, we see that both implementations of SDA are generally equivalent to those of CKFD. However, the computational cost of CKFD is over an order of magnitude larger than the complexity of $SDA_{stability}$.

Our implementations of SDA automatically divide the data into a set of subclasses. Note that each algorithm, $SDA_{stability}$ and SDA_{loot}, may divide the data differently. In Table 2 we show the number of

Table 2: Subclasses.

Method	WDBC	LSD	MDD-pix	MDD-fou	MDD-fac	MDD-kar	KDD-zer
$SDA_{stability}$	8	4	6	4	8	5	1
SDA_{loot}	4	10	6	2	2	9	8

Number of subclass divisions per class obtained by each of the two implementations of SDA; i.e., H_o/C.

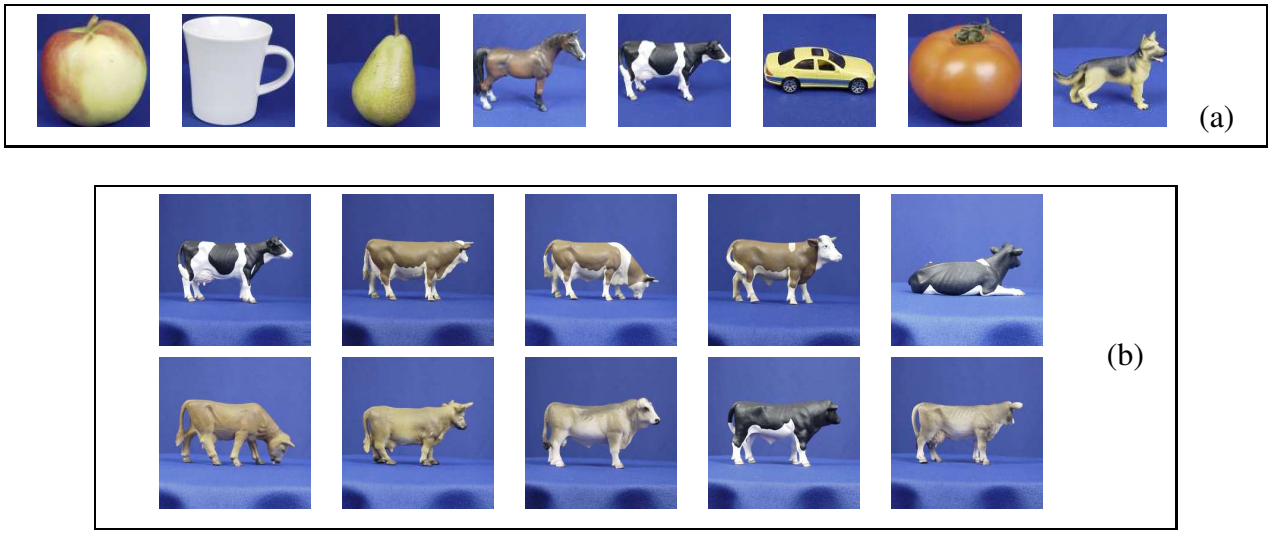


Figure 5: *(a) This figure shows an example image for each of the eight categories in the ETH-80 database [14]. In (b) we show a sample image for each of the ten different cows in this database.*

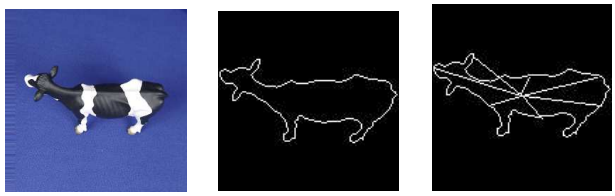
subclass divisions per class automatically obtained by each of the two criteria defined in this paper. This table shows the value of H_o/C. Although each criterion may result on slightly distinct subclass divisions, the classification accuracy does not vary too much as shown in Table 1. Recall that our goal was not to find that division of the data which best approximates the true underlying distribution of each class. Rather, we want to find that division which *best classifies the data in the reduced space*. We may hence conclude that each criterion is able to find a division (albeit a different one) that generates good results.

4.2 Object categorization

A classical problem in computer vision is to classify a set of objects into a group of known categories (e.g., apples, cars, etc.). In our experiments, we have used the ETH-80 database which contains images of the following categories: apples, pears, cars, cows, horses, dogs, tomatoes, and cups [14]. Examples of these are shown in Fig. 5(a). Each category includes the images of ten objects (e.g., ten different cows – as shown Fig. 5(b)) photographed at a total of 41 orientations. This gives us a total of 410 images per category.

Feature-based classification

In this approach, we first obtain the silhouette of the object (i.e., the contour that separates the object from the background), Fig. 6(b). Then, we compute the centroid of this silhouette and, finally, calculate the length of equally separated lines that connect the centroid with the silhouette; see Fig. 6(c). In our experiments we obtain a total of 300 distances, which generates a feature space of 300 dimensions.



(a) (b) (c)

Figure 6: *(a) A cow seen from one of the 41 different orientations. (b) The silhouette of this cow. (c) Each line measures the distance from the centroid to several points in the silhouette.*

To test the classification accuracy of each method, we use the leave-one-object out test. This means that we use 79 of the 80 objects for training and one for testing. Since there are obviously 80 ways in which we can leave one object out, we repeat the test eighty times and computed the average.

These results are summarized in Fig. 7(a). Since we used the leave-one-object out for testing, the value of H_o varied each time. For the $SDA_{stability}$ algorithm, the average value of H_o was 79, which means we would have an average of about 42 samples per subclass. For the SDA_{loot} this average was 25.

Appearance-based classification

For comparison purposes, we have also considered the classification of the objects in the ETH-80 database using a simple (pixel-) appearance-based approach. Here, we first crop the sub-image corresponding to the smallest rectangle that circumscribes the silhouette of the object. Since distinct images will generate croppings of different sizes, we resize each of the resulting sub-images to a standard size of 25 by 30 pixels; i.e., a 750-dimensional feature space. Because we have a total of $3,239$ samples for training, the samples-to-feature ratio is still adequate. Again, we use the leave-one-object out test to obtain our results, which are summarized in Fig. 7(b). In this case, the average value of H_o was 80 when we used $SDA_{stability}$ and 27 when using SDA_{loot}.

In the two experiments shown above, using the ETH-80 database, we see that SDA always yield the highest recognition rate. Furthermore, SDA gives very similar results for both the feature- and appearance-based methods. Previous results reporting the superiority of one method over the other might have been due to the DA approach used, rather than the discriminant information contained in each feature-space.

4.3 Face recognition

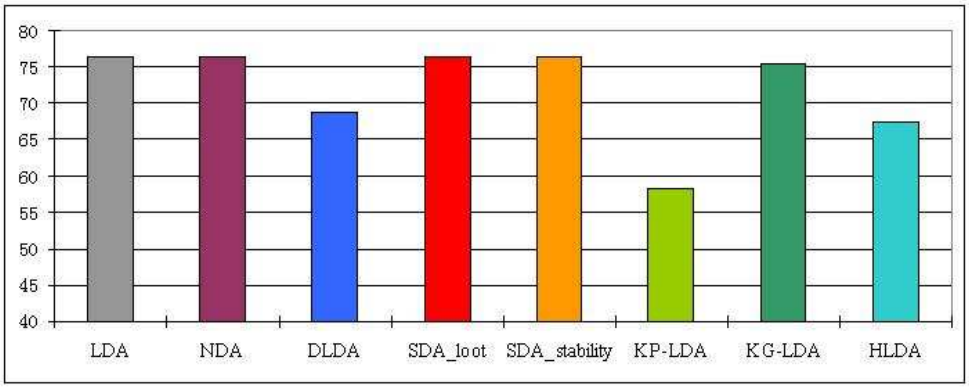
In our last experiment, we used the AR-face database [18]. The AR-face database consists of frontal-view face images of over 100 people. Each person was photographed under different lighting conditions and distinct facial expressions, and some images have partial occlusions (sunglasses or scarf). A total of 13 images were taken in each session for a total of two sessions; i.e., 26 images per person. These sessions were separated by an interval of two weeks.

We used the first 13 images corresponding to the first session (shown in Fig. 8(a)) of 100 randomly selected individuals for training; i.e., $1,300$ training images. The 13 images of the second session (shown in Fig. 8(b)) were used for testing.

In this case, we first crop the face part of the image (without including hair or background) and then resize all images to a standard image size of 29 by 21 pixels. This yields a 609-dimensional feature space. During the cropping process we also align all faces to have a common position for the eyes and mouth (since this is known to improve accuracy). The results are shown in Fig. 7(c). In this test both methods resulted in $H_o = 100$.

(a)

(b)



(c)

Figure 7: *(a) Results of the categorization problem using the feature-based approach. (b) Results of the categorization problem using the appearance-based approach. (c) Face recognition results using an appearance-based approach.*

Figure 8: *Facial images from AR-face database with different expressions and different occlusions.*

4.4 Comparison of the results on different clustering methods

As mentioned earlier, there are many clustering algorithms that could have been used to divide the data in each class into a set of subclasses. We now show that the classification results obtained using our NN-based approach (defined in Section 3.3) are comparable to those obtained with the K-means and the valley-seeking algorithms. To show this, we have rerun all our experiments described above but now using the K-means and the valley-seeking algorithms to divide the data into subclasses. These algorithms are refereed to as $SDA_{K-means}$ and SDA_{valley}. The classification accuracy for each of these implementations of SDA, are shown in Table 3. Note that while our NN-based clustering approach always generated the same partition of the data, this is not the case for the K-means and valley-seeking algorithms. On the one hand, K-means is an iterative approach that requires an initial guess (which is usually given at random). Different initializations result in distinct partitions. On the other hand, the valley-seeking algorithm requires that we manually choose a window size where local computations are to be conducted. Different window sizes will produce distinct clustering results. To make a fair comparison, we have randomly initialized the K-means ten times and then computed the average and standard deviation (which is shown in parentheses in the table). Similarly, the valley-seeking algorithm was run for several window sizes – average and standard deviation are shown. Finally, the results labelled SDA_{NN} in Table 3 are those obtained with our NN-clustering method and the stability criterion derived earlier. The results of the $SDA_{K-means}$ method were also obtained using our stability criterion. This means, we tried several values of H for clustering and then select the optimal according to Eq. (9). The valley-seeking algorithm works differently. For a fixed window size, the value of H_o is automatically given by the clustering method. Therefore, there is no need for the optimization step. Nonetheless, one may want to optimize the window size, because different sizes will result in very different partitions. We can do this by means of the LOOT or stability criteria defined in this paper. For example, we could define a new algorithm which uses our stability criterion to optimize the window size of the valley-seeking method – we referred to this as SDA_{VS} of Valley-Stability (VS). The results obtained with this new approach are summarized in Table 4. We see that this new algorithm results in similar classification rates

Table 3: $SDA_{K-means}$ and SDA_{valley}

Method	ARface	ETH-80 (feature)	ETH-80 (appearance)	WDBC	LSD
SDA_{NN}	0.782	0.749	0.731	0.944	0.881
$SDA_{K-means}$	0.776 (0)	0.754 (0.030)	0.789 (0.071)	0.942 (0.007)	0.884 (0.007)
SDA_{valley}	0.666 (0.150)	0.753 (0.005)	0.701 (0.025)	0.956 (0.005)	0.876 (0.004)
Method	MDD-pix	MDD-fou	MDD-fac	MDD-kar	KDD-zer
SDA_{NN}	0.957	0.834	0.957	0.966	0.793
$SDA_{K-means}$	0.956 (0.004)	0.828 (0.007)	0.935 (0.029)	0.965 (0.006)	0.7947 (0.005)
SDA_{valley}	0.936 (0.011)	0.811 (0.011)	0.971 (0.006)	0.962 (0.005)	0.799 (0.009)

Classification results when clustering the data in each class with the K-means and the valley-seeking.

Table 4: SDA_{VS}

Method	ARface	ETH-80 (feature)	ETH-80 (appearance)	WDBC	LSD
SDA_{VS}	0.757	0.759	0.733	0.947	0.88
method	MDD-pix	MDD-fou	MDD-fac	MDD-kar	KDD-zer
SDA_{VS}	0.927	0.82	0.968	0.959	0.793

Successful classification rates as given by the VS (Valley-Stability) implementation of SDA.

to those of our previous methods – SDA_{loot} and $SDA_{stability}$. This illustrates how the results described in this paper can be extended or used to define novel algorithms. For example, we could use our results to efficiently find the optimal division in the methods defined in [4, 29].

We would now like to get back to the possibility of estimating the underlying distribution of the data using a mixture of Gaussians. Unfortunately, and as already anticipated, we were unable to obtain such results because the EM algorithm requires to invert every sample covariance. In the databases used above, it is only possible to calculate such inverses in the WDBC set. In this case, the result obtained using a mixture of Gaussians (estimated with the EM algorithm) and our stability criterion was 95% which is comparable to those reported above. This method would however be restricted to applications where the number of samples per class is much larger than the dimensionality of the data.

As a final note, we should mention that the algorithms presented in this paper assume there is a sufficiently large number of samples per class. Note that when the number of classes is small, these cannot be reliably divided into subclasses. In this case, SDA will not generally divide the data into subclasses and, hence, will be equivalent to LDA. To show this, we have run $SDA_{stability}$ using the images of the first session in the AR face database. In this case, however, we only used four randomly chosen images for training. The remaining nine were subsequently used for testing. This process was repeated five times. In all cases $SDA_{stability}$ was equivalent to LDA, i.e., $H_o = 100$.

5 Conclusions

Over the years, many discriminant analysis algorithms have been proposed, each targeted to a specific type of data distribution. Since the type of distribution is usually unknown in practice, this meant that we were left

to test each of the existing methods before choosing that which worked best on our data. This was however very time consuming and would not guarantee we find that algorithm that best adapts to our problem. To remedy this, we have proposed a new method whose goal is to adapt to a large variety of data distributions. Rather than working with complex nonlinear methods (which require a large number of training samples to work properly and usually have a large computational cost), we have shown how we can solve this by dividing each class into a set of subclasses. In Section 2, we have shown how such subclass divisions can be used to adapt to different types of class distributions and how this allows us to find the most adequate subspace representation. This lead us to define a new between-subclass scatter matrix given in Eq. (5).

The main problem that needed to be addressed next, was to define an easy-to-use criterion that could provide a good estimate for the number of subclasses needed to represent each class pdf. In Section 3, we defined two such criteria. Our first criterion, is based on the idea of the leave-one-sample-out estimate. Here, the best partition is estimated using a cross-validation test on the training set. Although this was shown to be a reasonable approach, it came to an associated high computational cost. To solve this, we introduced a second criterion which takes advantage of the fact that any generalized eigenvalue decomposition equation (such as that used by our method) is not guaranteed to work when the smallest angle between the i^{th} eigenvector given by the metric to be maximized and the first i eigenvectors given by the metric to be minimized is close to zero.

We have shown experimental results on several datasets from the UCI repository (each with features of very distinct sorts) and on two computer vision applications (object categorization and face recognition). Our results show that the two implementations of SDA yield the highest classification rates.

Appendix A

Proof of Result 1. We can rework Eq. (4) as follows,

$$\widehat{\Sigma}_B = \sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_{ij}-\mu_i+\mu_i-\mu)(\mu_{ij}-\mu_i+\mu_i-\mu)^T =$$
$$\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}[(\mu_{ij}-\mu_i)(\mu_{ij}-\mu_i)^T+(\mu_i-\mu)(\mu_i-\mu)^T+2(\mu_{ij}-\mu_i)(\mu_i-\mu)^T] =$$
$$\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_{ij}-\mu_i)(\mu_{ij}-\mu_i)^T+\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_i-\mu)(\mu_i-\mu)^T+2\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_{ij}-\mu_i)(\mu_i-\mu)^T. \quad (10)$$

The first summing term can be further simplified as,

$$\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_{ij}-\mu_i)(\mu_{ij}-\mu_i)^T=\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}\frac{n_i}{n_i}(\mu_{ij}-\mu_i)(\mu_{ij}-\mu_i)^T=$$
$$\sum_{i=1}^{C}\frac{n_i}{n}\sum_{j=1}^{H_i}\frac{n_{ij}}{n_i}(\mu_{ij}-\mu_i)(\mu_{ij}-\mu_i)^T=\sum_{i=1}^{C}\frac{n_i}{n}\mathbf{Q}_i=\mathbf{Q},$$

where $\mathbf{Q}_i=\sum_{j=1}^{H_i}\frac{n_{ij}}{n_i}(\mu_{ij}-\mu_i)(\mu_{ij}-\mu_i)^T$ is the between-subclass scatter matrix of class i, and $\mathbf{Q}$ is the average of $\mathbf{Q}_i$ for $1\leq i\leq C$.

Similarly, we can simplify the second term in Eq. (10) as

$$\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_i-\mu)(\mu_i-\mu)^T=\sum_{i=1}^{C}\frac{n_i}{n}(\mu_i-\mu)(\mu_i-\mu)^T=\mathbf{S}_B.$$

The third and final term in Eq. (10) is easily shown to be equal to zero,

$$\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_{ij}-\mu_i)(\mu_i-\mu)^T=\sum_{i=1}^{C}\sum_{j=1}^{H_i}\frac{n_{ij}}{n}(\mu_{ij}\mu_i^T-\mu_{ij}\mu^T-\mu_i\mu_i^T+\mu_i\mu^T)=$$
$$\sum_{i=1}^{C}\frac{n_i}{n}\mu_i\mu_i^T-\sum_{i=1}^{C}\frac{n_i}{n}\mu_i\mu^T-\sum_{i=1}^{C}\frac{n_i}{n}\mu_i\mu_i^T+\sum_{i=1}^{C}\frac{n_i}{n}\mu_i\mu^T=0.$$

□

Appendix B: Notation

C	Number of classes
n	Number of samples
n_i	Number of samples in class i
n_{ij}	Number of samples in the j^{th} subclass of class i
H_i	Number of subclasses in class i
$H=\sum_{i=1}^{C}H_i$	Total number of subclasses
H_o	Optimal number of subclasses
$\mathbf{x}_{ij}$	The j^{th} sample of class i, where $1\leq i\leq C$ and $1\leq j\leq n_i$
μ	The sample mean for all the data
μ_i	The sample mean for class i
μ_{ij}	The sample mean for subclass j in class i
$\mathbf{S}_B$	Between-class scatter matrix
$\mathbf{S}_W$	Within-class scatter matrix
Σ_X	Sample Covariance matrix
Σ_B	Between-subclass scatter matrix
p	Rank of Σ_X
q	Rank of Σ_B
$\mathbf{W}=\{\mathbf{w}_1,\ldots,\mathbf{w}_q\}$	eigenvectors of Σ_B
$\mathbf{U}=\{\mathbf{u}_1,\ldots,\mathbf{u}_p\}$	eigenvectors of Σ_X

Acknowledgments

We would like to thank the referees for their comments. Thanks also go to Bastian Leibe and Bernt Schiele for making the ETH-80 database available to us. This research was partially supported by NIH under grant R01 DC 005241.

References

[1] C.L. Blake and C.J. Merz, "UCI Repository of Machine Learning Databases," URL = `http://www.ics.uci.edu/~mlearn/MLRepository.html`, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[2] G. Baudat and F. Anouar, "Generalized Discriminant Analysis Using a Kernel Approach," Neural Computation, 12:2385-2404, 2000.

[3] P.N. Belhumeur, J.P. Hespanha and D.J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," IEEE Trans. on Pattern Analysis and Machine Intelligence 19(7):711-720, 1997.

[4] F. De la Torre and T. Kanade, "Oriented Discriminant Analysis," In Proc. British Machine Vision Conference, London (UK), September 2004.

[5] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum Likelihood From Incomplete Data via the EM Algorithm," Journal Royal Statistical Society 30(1):1-38, 1977.

[6] L. Devroye, L. Györfi and G. Lugosi, "A Probabilistic Theory of Pattern Recognition," Springer-Verlag 1996.

[7] K. Etemad and R. Chellapa, "Discriminant Analysis for Recognition of Human Face Images," Journal of Optical Society of American A 14(8):1724-1733, 1997.

[8] R.A. Fisher, "The Statistical Utilization of Multiple Measurements," Annals of Eugenics, 8:376-386, 1938.

[9] K. Fukunaga, "Introduction to Statistical Pattern Recognition (2nd edition)," Academic Press, 1990.

[10] T. Hastie, R. Tibshirani and A. Buja, "Flexible Discriminant Analysis by Optimal Scoring," J. Amer. Statist. Assoc. 89: 1255-1270, 1994.

[11] T. Hastie, A. Buja and R. Tibshirani, "Penalized Discriminant Analysis," Annals of Statistics 23: 73-102, 1995.

[12] T. Hastie, R. Tibshirani, "Discriminant Analysis by Gaussian Mixture," J. Royal. Statist. Soc. B. 58: 155-176, 1996.

[13] A.K. Jain, R.P.W. Duin and J. Mao, "Statistical Pattern Recognition: A Review," IEEE Trans. on Pattern Analysis and Machine Intelligence 22(1):4-37, 2000.

[14] B. Leibe and B. Schiele, "Analyzing Appearance and Contour Based Methods for Object Categorization," In Proc. IEEE Computer Vision and Pattern Recognition, Madison (WI), June 2003.

[15] W.L.G. Koontz and K. Fukunaga, "A Nonparametric Valley-seeking Technique for Cluster Analysis," IEEE Trans. on Computers, C-21: 171-178, 1972.

[16] M. Loog, R.P.W. Duin and T. Haeb-Umbach, "Multiclass Linear Dimension Reduction by Weighted Pairwise Fisher Criteria," IEEE Trans. on Pattern Analysis and Machine Intelligence 23(7): 762-766, 2001.

[17] M. Loog and R.P.W. Duin, "Linear Dimensionality Reduction via a Heteroscedastic Extension of LDA: The Chernoff Criterion," IEEE Trans. on Pattern Analysis and Machine Intelligence 26(6): 732-739, 2004.

[18] A.M. Martínez and A.C. Kak, "PCA versus LDA," IEEE Trans. Pattern Analysis and Machine Intelligence 23(2):228-233, 2001.

[19] A.M. Martínez and J. Vitrià, "Clustering in Image Space for Place Recognition and Visual Annotations for Human-robot Interaction," IEEE Trans. System Man and Cybernetics B 31(5):669-682, 2001.

[20] A.M. Martínez and M. Zhu, "Where Are Linear Feature Extraction Methods Applicable?," IEEE Trans. on Pattern Analysis and Machine Intelligence 27(12):1934-1944, 2005.

[21] G.J. McLanchlan and K.E. Basford, "Mixture Models: Inference and applications to clustering," New York: Marcel Dekker, 1988.

[22] G.J. McLanchlan, "Discriminant Analysis and Statistical Pattern Recognition," New York: Wiley, 2004.

[23] C.R. Rao, "Linear Statistical Inference and Its Applications (Second Edition)," Wiley Interscience, 2002.

[24] D.L. Swets and J.J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," IEEE Trans. on Pattern Analysis and Machine Intelligence 18(8):831-836, 1996.

[25] J. Yang, A.F. Frangi, J. Yang, D. Zhang and Z. Jin, "KPCA Plus LDA: A Complete Kernel Fisher Discriminant Framework for Feature Extraction and Recognition," IEEE Trans. on Pattern Analysis and Machine Intelligence 27(2):230-244, 2005.

[26] M. Yang, D. Kriegman and N. Ahuja, "Face Detection Using Multimodal Density Models," Computer Vision and Image Understanding 84(2):264-284, 2001.

[27] M. Yang, "Kernel Eigenfaces vs. Kernel Fisherfaces: Face recognition using kernel methods," Proc. Fifth International Conference on Automatic Face and Gesture Recognition, pp. 215-220, May 2002.

[28] H. Yu and J. Yang, "A Direct LDA Algorithm for High-dimensional Data – with applications to face recognition," Pattern Recognition 34:2067-2070, 2001.

[29] S.K. Zhou and R. Chellappa, "Multiple-Exemplar Discriminant Analysis for Face Recognition," In Proc. International Conference on Pattern Recognition, pp. 191-194, Cambridge (UK), 2004.

[30] M. Zhu, A.M. Martínez and H. Tan, "Template-based Recognition of Sitting Postures," In Proc. IEEE Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction, Madison (WI), 2003.

[31] M. Zhu and A.M. Martínez, "Optimal Subclass Discovery for Discriminant Analysis", *Proceedings of IEEE Workshop on Learning in Computer Vision and Pattern Recognition (LCVPR)*, 2004.