A STUDY OF THE GENERALIZED EIGENVALUE DECOMPOSITION IN DISCRIMINANT ANALYSIS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Manli Zhu, M.S.

* * * * *

The Ohio State University

2006

Dissertation Committee:

Aleix M. Martinez, Adviser
Andrea Serrani
Yuan F. Zheng

Approved by

Adviser
Graduate Program in Electrical Engineering
ABSTRACT

The well-known Linear Discriminant Analysis (LDA) approach to feature extraction in classification problems is typically formulated using a generalized eigenvalue decomposition, $S_1 V = S_2 V \Lambda$, where $S_1$ and $S_2$ are two symmetric, positive-semidefinite matrices defining the measure to be maximized and that to be minimized. Most of the LDA algorithms developed to date are based on tuning one of these two matrices to solve a specific problem. However, the search for a set of metrics that can be applied to a large number of problems has met difficulty. In this thesis, we take the view that most of these problems are caused by the use of the generalized eigenvalue decomposition equation described above. Further, we argue that many of these problems can be solved by studying and modifying this basic equation. At the core of this thesis lays a new factorization of $S_2^{-1} S_1$ that can be used to resolve several of the problems of LDA.

Three novel algorithms are derived, each based on our proposed factorization. In the first algorithm, we define a criterion to prune noisy bases in LDA. This is possible thanks to the flexibility of our factorization, which allows the suppression of a set of vectors of any metric. The second algorithm is called Subclass Discriminant Analysis (SDA). SDA can be applied to a large variety of distribution types because it approximates the underlying distribution of each class with a mixture of Gaussians. The most convenient number of Gaussians can be readily selected thanks to our proposed factorization. The third algorithm is aimed to address the over-fitting issue in LDA. A direct application of this algorithm is
tumor classification, where the ratio of samples versus features is very small. The main idea of the proposed algorithm is to take advantage of the information embedded in the testing samples – changing the role of the testing data from passive to active samples. In all three cases, extensive experimental results are provided using a large variety of data-sets. Comparative studies are given against the most advanced and standard methods available.
ACKNOWLEDGMENTS

It is a pleasure to thank the many people who made this thesis possible.

First, I would like to thank my advisor and friend, Prof. Aleix M. Martinez, for many insightful conversations during the development of the ideas in this thesis. I could not imagine having a better advisor and mentor for my PhD. Through the four-years study, his encouragement and guidance never stopped accompanying me.

Thanks to my labmates – Liya Ding, Onur Hamsici, Hongjun Jia and Jeff Fortuna for the stimulating discussions, and for providing the fun environment in which we learn and grow. I am grateful to all my good friends - that I met in America and in China. Wherever you are, without your moral support, I could not go through any difficult moments and achieve the goal.

Finally, I am forever indebted to my parents, my sister and my husband for their understanding, endless patience and encouragement when it was most required. It has been hard to spend this time far from each other, but distance strengthened our connection even more. To them I dedicate this thesis.
VITA

March 12, 1976 ...................... Born - Jinan, P.R.China

1999 ................................. B.S. Electrical Engineering,
Zhejiang University,
P.R.China

2002 ................................. M.S. Electrical Engineering,
Zhejiang University,
P.R.China

2002-present ........................ Graduate Research Associate,
Dept. of Electrical and Computer Engi-
neering, The Ohio State University,
Columbus, OH.

PUBLICATIONS

Research Publications


M. Zhu and A.M. Martinez, “Pruning Noisy Bases in Linear Discriminant Analysis”. submitted to IEEE Trans. on Neural Networks.

**Instructional Publications**


**FIELDS OF STUDY**

Major Field: Electrical Engineering

Studies in:

- Topic 1 Statistical Pattern Recognition
- Topic 2 Computer Vision
- Topic 3 Bioinformatics
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Vita</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background on Pattern Recognition</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Feature Extraction in Pattern Recognition</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Literature Review</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Contributions and Outlines of This Thesis</td>
<td>13</td>
</tr>
<tr>
<td>2. Pruning Noisy Bases in Linear Discriminant Analysis</td>
<td>16</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Factorization of $\Sigma_X^{-1}S_B$</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Correlation-based Criterion</td>
<td>27</td>
</tr>
<tr>
<td>2.4 Experimental Results</td>
<td>31</td>
</tr>
<tr>
<td>2.5 Conclusions</td>
<td>36</td>
</tr>
<tr>
<td>3. Subclass Discriminant Analysis</td>
<td>38</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>39</td>
</tr>
<tr>
<td>3.2 The Metrics of Discriminant Analysis</td>
<td>43</td>
</tr>
</tbody>
</table>
### 3. Flexibility of SDA
- 3.4 Optimal Subclass Divisions
  - 3.4.1 Leave-one-out-test criterion
  - 3.4.2 Stability criterion
  - 3.4.3 Dividing classes into subclasses

### 3.5 Experimental Results
- 3.5.1 UCI datasets
- 3.5.2 Object categorization
- 3.5.3 Face recognition
- 3.5.4 Comparison of the results on different clustering methods

### 3.6 Conclusions

### 4. Tumor Classification Based on Gene Expression Data
- 4.1 Introduction
- 4.2 Biological Background
- 4.3 Literature Review
- 4.4 Problem Statement
- 4.5 Current Algorithms
  - 4.5.1 \( k \) Nearest Neighbor (\( k \)-NN)
  - 4.5.2 Maximum Likelihood
  - 4.5.3 Weighted Voting
  - 4.5.4 Fisher Discriminant Analysis
  - 4.5.5 Support Vector Machines
- 4.6 Discriminant Power-Based Solution
- 4.7 Experimental Result
  - 4.7.1 Data Sets
  - 4.7.2 Experiments
- 4.8 Conclusion

### 5. Conclusion

### Appendices:
- A. Notation
- B. Bayes Rule and Fisher’s Criterion
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td>Proof of Result 2</td>
<td>110</td>
</tr>
<tr>
<td>C.2</td>
<td>Proof of Theorem 3</td>
<td>111</td>
</tr>
<tr>
<td>C.3</td>
<td>Proof of Result 4</td>
<td>112</td>
</tr>
<tr>
<td>C.4</td>
<td>Proof of Result 5</td>
<td>113</td>
</tr>
<tr>
<td>C.5</td>
<td>Proof of Lemma 6</td>
<td>115</td>
</tr>
<tr>
<td>C.6</td>
<td>Proof of Lemma 7</td>
<td>116</td>
</tr>
<tr>
<td>C.7</td>
<td>Proof of Lemma 9</td>
<td>116</td>
</tr>
<tr>
<td>C.8</td>
<td>Proof of Lemma 10</td>
<td>116</td>
</tr>
<tr>
<td>C.9</td>
<td>Proof of Result 14</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>118</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Statistical comparison. In this table, $eig_r$ means we use the eigenvectors that keep $r%$ of the total variance, and $J(u)_r$ indicates that all those eigenvectors of $\Sigma_X$ that keep $r%$ of the total discriminant power given by $J(u)$ are used. In these experiments a bootstrap approach is used to generate average and standard deviations. Bolded results indicate statistical significance.</td>
</tr>
<tr>
<td>3.1</td>
<td>Shown here are the classification results (in percentage) on the UCI datasets 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 parentheses correspond to the standard deviation from such means.</td>
</tr>
<tr>
<td>3.2</td>
<td>Number of subclass divisions per class obtained by each of the two implementations of SDA; i.e., $H_0/C$.</td>
</tr>
<tr>
<td>3.3</td>
<td>Classification results when clustering the data in each class with the K-means and the valley-seeking.</td>
</tr>
<tr>
<td>3.4</td>
<td>Successful classification rates as given by the VS (Valley-Stability) implementation of SDA.</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison results of different classifiers for gene-based tumor classification. The value shown in the parentheses in the column of $k$-NN is the number of neighbors determined by cross-validation.</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>(a) In this two-dimensional example, Eq. (2.3) generates the correct solution, while Eq. (2.5) does not. In (b-c), we project two Gaussians embedded in a high dimensional space onto the direction found by Eq. (2.3), shown in (b) and Eq. (2.5) shown in (c).</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>In (a) the data has large variance about the z axis. In (b) this variance is small. In neither case z provides any discriminant information.</td>
<td>28</td>
</tr>
<tr>
<td>2.3</td>
<td>(a) Plotted here are the eigenvalues of the sample covariance matrix of the ETH-80 database in decreasing order. (b) Shows the correlation values of each of the eigenvectors of $\Sigma_X$ (ordered by eigenvalue) with those of $S_B$. Dashed and dotted lines represent the cutoff points with 70% and 90% of the total variance. (c)-(d) the same for the images of AR-face database.</td>
<td>30</td>
</tr>
<tr>
<td>2.4</td>
<td>The confidence interval of the exponential function $g_y = \lambda e^{-\lambda y}$ with probability 0.9, where $\lambda = f_1 = \frac{I_1}{P_B} = 0.1428$.</td>
<td>32</td>
</tr>
<tr>
<td>2.5</td>
<td>In (a-c) we have ordered the eigenvectors of the covariance matrix $\Sigma_X$ obtained from the ETH, AR and SPDM databases, from most to least correlated to $\text{tan}(S_B)$. In (d-f) We show the recognition rates when the $k$ first eigenvectors, as ordered in (a-c), are used. Here, $k$ is the horizontal axis and the dotted vertical line is the result given by our algorithm. (g-i) Shows the recognition rate obtained when $k$ eigenvectors are kept and these are ordered in decreasing order of eigenvectors. (j-l) Does the same, but when the eigenvectors are order from most to least discriminant as given by $J(a_j)$. The dots in (g-l) specify the successful classification rate when different percentages of the variance (or discriminant power) are retained (see text).</td>
<td>35</td>
</tr>
</tbody>
</table>
3.1 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 $\Sigma_X$ and $\Sigma_B$ represent the sample covariance matrix and the between-subclass scatter matrix (see text).

3.2 (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 $S_B$ 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.

3.3 (a) In this example, the first basis vector obtained with Eq. (3.1) does not minimize the classification error. Note that $u_1$ (which is orthogonal to $v_1$ would be the optimal and desirable solution). Close analysis of this figure shows that the angle between the first eigenvector of $\Sigma_B$ and $\Sigma_X$ is zero. This means that it is not possible to maximize the metric given by $\Sigma_B$ and minimize that of $\Sigma_X$ simultaneously. (b) A classical example where most linear equations perform as expected. Note that in this case the angle between the first eigenvectors of $\Sigma_B$ and $\Sigma_X$ is large (i.e., $90^\circ$). This is formally computed by Eq. (3.10).

3.4 In (a) the results of Eq. (3.1) are incorrect. In (b) the results are correct.

3.5 (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.

3.6 (a) This figure shows an example image for each of the eight categories in the ETH-80 database [40]. In (b) we show a sample image for each of the ten different cows in this database.

3.7 (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.
3.8 (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.

3.9 Facial images from AR-face database with different expressions and different occlusions.

4.1 An example of a microarray showing the expression levels of thousands of genes from normal and abnormal tissue.

4.2 Plotted here is an illustration of the over-fitting problem. The $x$-axis represents the experiment number. In each experiment, the training samples are perfectly separated (samples from class 1 and classes 2 are projected onto two distinct points – shown as $\star$ and $\Box$), while the testing sample generally lies in the middle and cannot be classified correctly.

4.3 The procedure of classifying $x$ using the discriminant power defined in Eq. (2.8).

4.4 (a) Visualizes the data generated in $\mathbb{R}^{10}$ using the 3 dimensions with largest variance. In (b)-(e), we show the distribution of the data when projected onto the direction of $v$ found by Fisher-Rao’s criterion. In (b) and (d), the testing sample is assumed to be drawn from class 1, and in (c) and (e) the testing sample is labelled class 2. When $n > p + 1$, the $DP$ is a good measure of separability. When $n \leq p + 1$, $d$ is the appropriate measure.

4.5 The flow chart of the DP-based algorithm.
CHAPTER 1

INTRODUCTION

1.1 Background on Pattern Recognition

It seems such an easy task when people identify a person, figure out a handwritten character or recognize different kinds of sounds. However, we do not know how humans achieve this; i.e., how they recognize patterns. It is thus challenging to teach machines to do the same thing. Pattern recognition is the study of how computers can distinguish patterns of interest from each other. This technique has been widely used in a variety of engineering and scientific disciplines such as psychology, biology, computer vision, artificial intelligence and remote sensing. Well-known applications are automatic speech recognition, medical diagnosis, DNA sequence analysis, face identification and verification, and so on.

There are four main approaches to pattern recognition. These are: 1) template matching, 2) syntactic or structural matching, 3) neural network, and 4) statistical classification [34]. A brief introduction to those approaches is as follows.

In template matching, the pattern to be recognized is matched against a predefined template while taking into account all possible rotation, translation and scaling.
In syntactic pattern recognition, each pattern is viewed as being composed of subpatterns, which may be built from even simpler subpatterns or primitives. The patterns and primitives are analogous to sentences and alphabets, respectively. Sentences are generated by simple alphabets according to a pre-specified grammar. Similarly, patterns are described by primitives based on predefined rules.

Neural Networks are graphs with usually numerous nodes interconnected to other nodes by weighted directed edges. Here, the nodes represent artificial neurons and the edges the connections between the outputs and inputs. Theoretically, a neural network is capable of learning complex nonlinear input-output relationships.

In statistical pattern recognition, each pattern is represented by a set of $p$ features and is hence viewed as a point vector in a $p$-dimensional space. Statistical concepts and algorithms are used to build the decision boundary or model the distribution of the data.

Depending on the criterion used to adjust the structure or parameters of our classifier, approaches in Statistical Pattern Recognition are further separated into two categories: generative and discriminative [35]. Generative methods emphasize the learning of the conditional probability distribution of the samples and their class labels. It is called generative since one can artificially generate more samples once the probability density function (pdf) has been estimated. For examples, Gaussian Mixture Models are one of the most generally used generative methods and the EM (Expectation-Maximization) algorithm is a well-known algorithm to determine the parameters of such model. It estimates the priors and parameters of the Gaussian Mixture by maximizing the likelihood of the data given an expression of the pdf. Discriminative methods focus on the difference between classes, and extract the discriminative information between samples from distinct classes. There exist several discriminative methods, such as Linear Discriminant Analysis (LDA) and Support
Vector Machine (SVM). LDA searches for the linear solution (a vector in a two dimensional space or a hyperplane in a high dimensional space) that separates the two classes by maximizing their class separability. SVM seeks the decision boundary between two classes such that the margin between the samples and decision boundary is largest. My research work focuses on discriminative methods.

Generally speaking, generative methods and discriminative methods emphasize on different aspects of the class information: the former looks for the information within every class, while the latter works on the information which is most different between classes. However, those two methods cannot always be separated in practice. For example, some discriminative methods are generated on the assumption that the underlying distribution of the data is Gaussian and the parameters of the distribution are estimated through generative methods. Therefore, the combination of these two methods is quite common.

To successfully classify each sample, we need to find an appropriate representation of the data. Our representation should (ideally) guarantee that the samples belonging to different classes form a compact set in the feature space. There is a large variety of features one can use to achieve this. For instance, if the pattern to be classified is a kind of diseases, the feature set can be chosen to be the patient’s age, gender, body temperature and blood pressure. If the pattern is an image, one can use the grey-level of each pixel in that image as features. Once the data is represented appropriately, a sophisticated classifier or decision rule constructed from the training data can be used for classification.

1.2 Feature Extraction in Pattern Recognition

With the development of new sensors, it has become much easier to gain access to high-resolution data. Hence, the original data we get from those data acquisition devices is
usually in a high dimensional space. Intuitively, the larger the number of features, the more precise the representation of the data is, and the better the classification can be. Therefore, one would expect to improve the classification performance by adding more features to the representation. It is well-known that the probability of misclassification of a classifier does not increase when the number of features increases, provided that the class-conditional densities are completely known (or equivalently, the number of training samples is arbitrarily large and representative of the underlying distribution) [34]. However, in practice, since the underlying distribution of the data is unknown and has to be estimated from a limited number of training samples, the added features will actually degrade the performance of our classifier. This phenomenon is known as the curse of dimensionality. This can be stated as follows: with a large number of features, a small number of samples can only form a sparse space and, hence, the estimated parameters of the underlying distribution and the classification may not be reliable. This means that, in practice, we cannot arbitrarily increase the number of features when the parameters of the class-conditional density are estimated from a limited number of training samples. Although it is difficult to derive an exact relationship between the number of samples, the dimensionality of the feature space, and the classification accuracy, it is generally accepted that the number of training sample \( n \) should be comparable to \( 10 \times C \times p \); where \( C \) is the number of classes and \( p \) is the dimensionality of the feature space. Since it is hard to collect enough samples for large feature spaces, a dimensionality reduction is usually necessary. Another advantage of reducing the dimensionality is that small number of features may also help us build a classifier with a smaller computational and memory cost.

In general terms, we say that there are two ways to reduce the dimensionality: feature selection and feature extraction. Feature selection refers to the algorithms that choose the
“best” subset of features to represent the data. **Feature extraction** – the one that my work focuses on – determines an appropriate subspace of dimensionality $m$ from the original feature space of dimensionality $p$ ($m < p$), by either linear or nonlinear transformations according to some predefined criterion. After feature selection, the new features usually keep their physical meanings, but after feature extraction, the new features may not be so easily associated to physical meanings.

Numerous feature extraction methods have been developed within the pattern recognition community. **How to choose the one that is best for the problem at hand is a critical issue.** Our work concerns this problem.

On the one hand, distinct feature extraction methods are constructed by optimizing different criteria. For example, Principle Component Analysis (PCA) seeks the projection that best approximates the original data in the low dimensional space in the sense of minimum mean-square error (MSE). Linear Discriminant Analysis (LDA) looks for the projection matrix that maximizes the ratio of the between-class scatter to the within-class scatter, and when the data are homoscedastic (i.e., all classes share the same covariances but have different means), the minimum classification error is obtained according to the Bayes Rule (we will come back to this point in detail later). Independent Component Analysis (ICA) maximizes the degree of statistical independence among output variables using contrast functions such as the Kullback-Leibler divergence, negentropy $^1$ and cumulants [33]. Therefore, the selection of one feature extraction algorithm over the rest depends on the criterion one wants to optimize.

On the other hand, current feature extraction approaches are optimal under different assumptions. The “Ugly Ducking Theorem” states that in the absence of assumptions, there

---

$^1$Negentropy defines a measure of non-gaussianity, which is always non-negative and is zero for a Gaussian variable.
is no “best” feature representation [18]. No general algorithm exists which can always obtain superior performance compared to those of others. The selection of each feature extraction method is data driven. For example, if the whole data corresponds to a Gaussian distribution, PCA (Principle Component Analysis) is expected to obtain a good representation and is optimal according to MSE (mean square error). When the data in each class also corresponds to a single Gaussian and all classes share a common covariance matrix, LDA (Linear Discriminant Analysis) obtains optimal results. For the data of a high dimensional space whose distribution cannot be visualized on a computer screen, we will never know which method is best unless we try them all. It is one of our goals to construct a new feature extraction algorithm which is flexible enough to adapt itself to a large number of distributions.

Among all the feature extraction algorithm, Linear Discriminant Analysis (LDA) is one of the most popular and powerful techniques. The popularity of LDA is accounted for its efficiency and simplicity. LDA attempts to minimize the Bayes error by selecting those feature vectors which maximize the Fisher’s criterion $\frac{|v^T S_B v|}{|v^T S_W v|}$. And these feature vectors can be easily obtained by means of the generalized eigenvalue decomposition on $S_B$ with respect to $S_W$ (i.e., $S_B V = S_W V \Lambda$), where $S_B$ and $S_W$ represent the between-class scatter matrix and within-class scatter matrix. They are defined as follows,

$$S_B = \sum_{i=1}^{C} p_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{1.1}$$

$$S_W = \sum_{i=1}^{C} p_i \Sigma_i,$$

where $n$ is the number of samples, $\mu$ is the sample mean, and $\mu_i$, $\Sigma_i$ and $p_i$ are the mean, covariance matrix and the prior of the $i^{th}$ class. Refer to Appendix A for details of our
notation. Because the covariance matrix of the data $\Sigma_X$ is equal to $S_B + S_W$, minimizing the metric given by $S_W$ is equivalent to minimizing that of $\Sigma_X$. The rank of $\Sigma_X$ is generally larger than $S_W$. Hence, $\Sigma_X$ is an appropriate alternative for $S_W$. The discriminant vectors can be obtained by $S_B V = \Sigma_X V \Sigma$, where

$$\Sigma_X = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T.$$ 

### 1.3 Literature Review

LDA has received considerable attention since it was first introduced by Fisher for the two-class problem [22] and later extended by Rao for the multiple-class case [60]. LDA was originally developed as a linear classifier. In the classification setting, an object is assumed to be a member of one class and an error is incurred if it is assigned to a different class. If all the conditional probability density functions (pdf) and class priors were known, the Bayes rule would achieve, of course, the minimal classification error among all possible classifiers [18, 23, 25]. The Bayes rule is most often applied to the Normal distributions (i.e., each class corresponds to a multivariate Normal pdf). In this case, the Bayes rule is referred to as Quadratic Discriminant Analysis (QDA), since it separates the disjoint region of the feature space corresponding to each class using a quadratic boundary. An important special case occurs when all the class covariance matrices are identical. This is referred to as LDA because the quadratic terms are cancelled, resulting in a simple linear decision boundary. When the assumption on the data does not hold, LDA may not obtain the best solution (since Fisher’s criterion is identical to Bayes rule only when the classes are multivariate Gaussians with equal covariance; see Appendix B). However, LDA provides a mapping to a space of reduced dimensionality in which discrimination is generally easier.
LDA has been used in many applications. For instance, LDA has been applied to face recognition [4, 48, 47], handwritten numeral recognition [39], ECG (electrocardiogram) signals classification [24], automatic accent classification [57] etc. Many generalizations and modifications have been proposed to improve its performance. The following are some notable examples.

Nonparametric Discriminant Analysis (NDA) [25] uses a nonparametric between-class scatter matrix instead of the one defined in Eq. (1.1). This is useful when the data distribution is significantly different from Normal. Because the use of a parametric form cannot be expected to indicate which features should be extracted to preserve complex structures needed for classification. Here, the between-class scatter is measured on a local basis, using the $k$-nearest neighbor ($k$NN) technique. Since those samples that are far away from the boundary may distort the information of the boundary structure, NDA applies some weight function to penalize them. And since this nonparametric between-class scatter matrix is generally of full rank, NDA can extract more than $C - 1$ directions.

By decomposing the $k$-class Fisher criterion into a sum of $\frac{1}{2}k(k-1)$ two-class criteria, Loog et al. [43] find that LDA preserves the distances of already well-separated classes, causing a large overlap of those classes that are close to each other in the original space. Therefore, LDA is suboptimal with respect to the classification error. They attempt to solve this problem by weighting the contribution of individual class pairs to the overall criterion. This weighting is derived such that their new criterion approximates the Bayes accuracy for pairs of classes.

Friedman [23] analyzes the estimation of the covariance matrix in small-sample, high-dimensional settings. Roughly speaking, the problem is poor-posed if the dimensionality is comparable to the number of observations and ill-posed if the dimensionality exceeds the
sample size. In these cases the estimate of the covariance matrix can be highly unstable, giving rise to high unexpected variances. In his paper, Friedman introduces a regularization term which attempts to improve our estimate by biasing it away from the sample-based value toward the value that is deemed to be more “physically plausible” [11]. This regularization is characterized by two parameters. One controls the degree of shrinkage of each individual sample class covariance matrix toward the pooled estimate. The other controls the shrinkage toward a multiple of the identity matrix. This regularization reduces the variance associated with the sample-based estimate at the expense of increased bias. This bias variance trade-off is regulated by the above two parameters. However, a good pair of parameters is not likely to be known in advance. Friedman suggests to use cross-validation or bootstrapping methods to estimate these from training samples.

Hastie et al. [32] look at LDA from the multivariate linear regression point of view. They define a function that assigns scores to the classes such that the transformed class labels are optimally predicted by linear regression on predictors. They showed that “LDA can be performed by a sequence of linear regression steps, followed by classification to the closest centroid in the space of fits.” The real power of this method is its generalization. One can replace the linear regression fit by far more flexible, nonparametric fits to achieve a more flexible classifier than LDA. For example, if a degree-two polynomial regressor is used instead of a linear one, the decision boundary will be a quadratic surface. They call this method Flexible Discriminant Analysis (FDA).

Penalized Discriminant Analysis (PDA) [29] introduces a penalized metric in LDA by replacing the within-class scatter matrix $S_W$ by a regularized version $S_W + \Omega$. The penalization matrix $\Omega$ is determined such that the coefficients of the discriminant vectors are smoother and interpretable. In the case with large number of correlated feature variables,
for example, in speech recognition and handwritten character recognition, this regularized version of LDA has been shown to improve the classification performance on testing data significantly.

Mixture Discriminant Analysis (MDA) [31] fits a mixture of Gaussians to each class to facilitate learning in the non-Normal or multi-modal settings. The subclasses are assumed to have the same covariance matrix. The EM algorithm is used to determine the maximum-likelihood estimates for the parameters of these mixtures.

Jing et al. [37] propose another improved LDA approach. This approach includes LDA modification at three different points: 1) improving the selection of discriminant vector by choosing vectors with larger Fisher discriminant values, 2) improving the statistical uncorrelations of discriminant vectors by combining LDA and UODV [36] (Uncorrelated Optimal Discriminant Vector) which optimizes Fisher’s criterion with the constrain: \( \mathbf{v}_i \mathbf{\Sigma}_X \mathbf{v}_j = 0 \), for \( i \neq j \), and 3) improving the selection of principle components by choosing those principle components with Fisher discriminant values larger than a pre-specified threshold.

Cooke [9] proposes two variations on Fisher’s linear discriminant in the two-class setting. The first variation is called 1D parameter search, which is developed from Anderson [2] and Cooke [10]. They showed that the best normal vector to the decision hyperplane separating two classes will be given by \( (\mathbf{S}_{W_1} + \gamma \mathbf{S}_{W_2})^{-1}(\mu_2 - \mu_1) \), regardless of the distribution type, when only the first and second moments of the data distribution are known, where \( \mathbf{S}_{W_1} \) and \( \mathbf{S}_{W_2} \) are the within-class scatter matrices of class 1 and class 2 respectively. Therefore, varying \( \gamma \) produces a one dimensional set of directions which can be used to search for the optimal solution. The second variation on LDA is called recursive Fisher, which uses the idea of support vectors in SVM (Support Vector Machines)[6, 71, 12]. The
decision boundary is first found using all samples. Then $S$ percentage of the samples that are not correctly classified by the decision boundary from each class are used to refine the boundary. They are called support vectors. After several iterations, the decision boundary is adjusted more precisely.

In [45], Lotilikar et al. argue that the optimal criterion based on scatter matrices in LDA is not directly related to classification accuracy. The classes which are closer together are more likely to have confusion and should be more heavily weighted (i.e., the weight should be inversely proportional to the distance between class pairs). They propose the fractional-step dimensionality reduction method called Fractional-LDA (F-LDA) to address this problem. The idea is to reduce the dimensionality from $n$ to $m$ ($m \leq n$) in small fractional steps, which allows for the relevant distance to be more correctly weighted. Specifically, in each fractional step, the data of the last dimension is squeezed. If there exists a pair of classes which are separable only in this direction, in the next step, they will be highly weighted (since the distance between them is squeezed). That means this direction will not be the last one any longer and will not be deleted. Therefore, the fractional steps attempts to minimize the number classes that overlap in the low dimensional subspace.

Another limitation of LDA is that it merely tries to separate class means as much as possible but does not take into account the discriminatory information present in the difference of the covariance. Hence, it is incapable of dealing with heteroscedastic data, i.e., the data from distinct classes have unequal covariance matrices [51]. Loog et al. [44] address this problem of LDA using the so-called Chernoff criterion which is based on the well-known Chernoff distance [25, 18]. This measure of affinity of two densities considers mean differences as well as covariance differences.
Over the last few years, Support Vector Machines (SVM) have received considerable attention. One reason for the success of SVM is given by their use of the kernel trick, which provides an efficient way to compute nonlinear features of the input space. Baudat et al. [3] propose a Generalized Discriminant Analysis (GDA) algorithm, which can generalize discriminant analysis to deal with nonlinear problems using kernel function operators. Lu [46] and Liu [42] further improve the performance of GDA applied to face recognition. The main idea of GDA is to map the input space into a higher-dimensional space using nonlinear functions, where the classes are linearly separable. Then, LDA is used to extract a linear projection matrix in this now high-dimensional space. This is equivalent to map the original input space into a feature space in which variables are nonlinearly related to the input space. Kernel functions are used to avoid the expensive operation that explicitly projects samples to a high-dimensional space – the dot product of two samples in the high-dimensional space is computed by means of kernel functions of the corresponding two inputs in the original space. Using different kinds of kernels, we can obtain a wide class of nonlinearities.

Another problem with Fisher Linear Discriminant Analysis is that where the number of samples is comparable to or less than the dimensionality, the within-class scatter matrix $S_W$ is singular. As a result, LDA cannot be applied. Many researchers have attempted to solve this problem. One approach to solve this problem is to project the feature vectors onto a lower dimensional space spanned by the first few eigenvectors of PCA. In this projected space, the within-class scatter matrix is not singular [4]. However, it has been augured in the literature [76, 8] that the initial dimensionality reduction step may result in a loss of important discriminant information. Chen et al. [8] argue that the null space of $S_W$ contains the most discriminant information and present an algorithm using this null space. But
their method is not adequate, since it does not use any information outside the null space, which Jing et al. [37] have shown to contain discriminant information. Yu’s direct LDA (DLDA) approach [76] is also proposed to solve this singularity problem. In this case, $S_W$ is projected onto the range space of $S_B$ and then those eigenvectors of this projected matrix corresponding to smaller eigenvalues are kept. However, there is a contradiction between their theory and their experiments. The theory requires discarding those largest eigenvalues of the projected matrix. Yet, in their experiments, the recognition results improve when more eigenvectors are used.

1.4 Contributions and Outlines of This Thesis

As already mentioned, LDA has received considerable attention since it was first introduced by Fisher for the two-class problem [22] and latter extended by Rao for the multiple-class case [60]. Many generalizations and modifications have been proposed to improve its performance. Some notable examples were introduced in section 1.3. However, little work has looked at the generalized eigen-based linear equation $S_1V = S_2V\Lambda$, which has been extensively used to get the solution of LDA. In this thesis, $S_2^{-1}S_1$ is factorized as a combination of all eigenvectors and eigenvalues of $S_1$ and $S_2$. This factorization explicitly explains how the relationship between $S_1$ and $S_2$ influences the performance of LDA, providing a deep understanding on this “simple” equation.

Three new feature extraction and classification algorithms are developed from this factorization. They are listed as follows:

- A fundamental drawback of LDA is that it cannot be efficiently applied whenever the matrix $S_2$ is singular or when some of the smallest variances in $S_2$ are due to noise. In Chapter 2, a novel correlation-based criterion is proposed to prune those
noisy bases not correlated to class discrimination. The stability of this approach is demonstrated theoretically and practically using a large variety of databases.

- 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 by 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 Chapter 3, two criteria able to find the most convenient division of each class into a set of subclasses are derived. The resulting discriminant analysis algorithm is Subclass Discriminant Analysis (SDA).

- Precise classification of tumors is essential for the successful treatment and early detection of cancer. The emergence of modern experimental technology, such as DNA microarray facilitates the research in cancer classification. DNA microarray offers scientist the ability to monitor the expression patterns of thousands of genes simultaneously, allowing them to study how these genes function, how they act under different conditions. And therefore, lead to a more complete understanding of the molecular variation in addition to morphologic changes. Hence, a finer and more precise diagnosis of cancer is expected. DNA microarray is characterized by the large number of features (thousands of genes) and smaller number samples (rarely
over a hundred). This property makes most traditional classifiers not applicable. In Chapter 4, a new classifier is proposed based on the factorization of $S_2^{-1}S_1$. This algorithm not only takes care of the large number of features, but also relieves the influence of over-fitting by including the information of testing sample in the building of the classifier.

Conclusion and future work are then summarized in Chapter 5.
CHAPTER 2

PRUNING NOISY BASES IN LINEAR DISCRIMINANT ANALYSIS

The success of Linear Discriminant Analysis (LDA) methods is due in part to the simplicity of their formulation, which reduces to a simultaneous diagonalization of two symmetric matrices $S_1$ and $S_2$ (i.e., $S_1V = S_2V\Lambda$). However, a fundamental drawback of this approach is that it cannot be efficiently applied wherever the matrix $S_2$ is singular or when some of the smallest variances in $S_2$ are due to noise. In this chapter, we present a factorization of $S_2^{-1}S_1$ and a correlation-based criterion that can be readily employed to solve these problems. The stability of the proposed approach is demonstrated thoroughly using a large variety of databases.

2.1 Introduction

Simultaneous diagonalization of two matrices is a powerful tool applicable to a large variety of problems in computer vision and pattern recognition. This is usually achieved by means of an eigenvalue decomposition of $S_2^{-1}S_1$, where $S_1$ and $S_2$ are two symmetric matrices and $S_2$ is positive-semidefinite [25, 50]. Discriminant analysis algorithms [25, 38, 50, 80, 22, 60] employ this to find those bases that maximize the metric defined by $S_1$ and minimize the metric given by $S_2$. Arguably, the most known algorithm using this basic approach is Fisher-Rao’s Linear Discriminant Analysis (LDA) [22, 60]. In this case, $S_1$ is
defined as the sample between-class scatter matrix (i.e., the covariance of the class means) $S_B$. A first option for $S_2$ is the within-class scatter matrix $S_W$. Because the covariance matrix of the data $\Sigma_X = S_B + S_W$, minimizing $\Sigma_X$ is equivalent to the minimization of $S_W$. The rank of $\Sigma_X$ is usually higher than that of $S_W$, so $\Sigma_X$ is an appropriate alternative (refer to Chapter 1.2 for the definitions of $S_B$, $S_W$ and $\Sigma_X$).

The discriminant vectors are obtained through the following eigenvalue decomposition

$$\Sigma_X^{-1}S_BV = V\Lambda,$$  \hspace{1cm} (2.1)

where $V = \{v_1, \cdots, v_p\}$ is a matrix whose columns describe the discriminant feature vectors and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_p)$ is the diagonal matrix of associate discriminant variances, with $\lambda_1 \geq \cdots \geq \lambda_p$. Note that whenever $p > \text{rank}(S_B)$, the last $p - p_B$ variances will be zero, where $p_B = \text{rank}(S_B)$ and $p$ is the dimensionality of the feature space. The eigenvectors corresponding to the largest eigenvalues maximize the fisher-Rao’s criterion:

$$\frac{v^T S_B v}{v^T \Sigma_X v}.$$  \hspace{1cm} (2.2)

It is clear from Eq. (2.2) that the resulting discriminant feature vectors $V$ will be dominated by the eigenvectors of $\Sigma_X$ associated to small eigenvalues. Because for those vector $v$, $v^T \Sigma_X v \to 0$, Eq. (2.2) goes to infinity if $v^T S_B v \neq 0$. This works well when the eigenvectors associated to small variances describe the dimensions where the discriminant information is. Unfortunately, many of these small variances correspond to noisy features. Here, by noisy features we mean those dimensions that do not contribute to discrimination. This can either be caused by noise in the data or by the features uncorrelated with the classification parameter (e.g., illumination variations in object recognition).
Discerning which bases are due to noise and which to the true discrimination of the data – that is to say, the ones to prune and the ones to keep – is a fundamental problem in discriminant analysis.

In attempting to solve this problem, researchers have previously looked at a related issue – that given by the singularity of $\Sigma_X$. That is, when $\Sigma_X$ cannot be inverted, Eq. (2.1) can be preceded by a Principle Component Analysis (PCA) step where the original dimensionality of the data is reduced to one of $k$-dimensions, i.e., $\mathbb{R}^p \rightarrow \mathbb{R}^k$. Here, $k$ is chosen to guarantee that the projection of $\Sigma_X$ onto $\mathbb{R}^k$ is no longer singular, i.e., $p \geq \text{rank}(\Sigma_X) \geq k$ [4, 63, 21].

A well-known solution for selecting a value for $k$ examines the plot of the eigenvalues (conveniently ordered from largest to smallest) and then searches for the “elbow” where the values fall sharply. However, there exist too many applications where the eigenvalue plot drifts without any obvious cutting point. When this is the case, the first $k$ PCs that account for

$$r = 100 \frac{\sum_{j=1}^{k} \lambda_{X_j}}{\sum_{j=1}^{p} \lambda_{X_j}}$$

per cent of the total variance can be used, where $\lambda_{X_1} \geq \cdots \geq \lambda_{X_j} \geq \cdots \geq \lambda_{X_{pX}}$ are the eigenvalues of $\Sigma_X$ and the corresponding eigenvectors are $\{u_1, \cdots, u_{pX}\}$. Although, $r$ must be specified by the user, in [38] it is suggested that a value of $r$ between 70% and 90% preserves most of the information needed for representing Gaussian-like distributions. But determining the most convenient value is problem-specific.

An alternative to the method just described, is to define a criterion that accounts for the discriminant information on each basis vector $u_j$. Most notably, [15, 38] define the
discriminatory power of a vector $u_j$ as

$$J(u_j) = \frac{u_j^T S_B u_j}{\lambda_{X_j}}.$$ 

In this case, the numerator measures the agreement between the eigenvectors of $\Sigma_X$ and those of $S_B$, while the denominator is used to account for the variances.

However, it is unclear how many PCs would be needed to make this criterion useful in practice. Since the estimate of small eigenvalues are biased downwards, and the estimate of large eigenvalues are biased upwards, the reciprocal of eigenvalues can lead to an exaggerated influence of the low-variance PCs in the estimate of the discriminative power [38]. In addition, because $J(u_j)$ is consistent with Fisher’s criterion, it seems redundant to choose PCs according to $J(u_j)$ before applying $J(V)$.

The major problem with the solutions summarized in the preceding paragraphs, is that we do not know which of the eigenvectors of $\Sigma_X$ are associated to noise, because the eigenvalue does not distinguish between noisy features and bases describing the true discriminant information.

This means we need to define a criterion that is unbiased by the values of the variances associated to each of the bases of $\Sigma_X$, which is indeed the method we will propose in this chapter. In section 2.2, we first present a factorization of $\Sigma_X^{-1} S_B$, the relationship between $\Sigma_X$ and $S_B$ are explicitly illustrated in this factorization. Following that, a correlation-based criterion is derived for pruning noisy bases in section 2.3. Experimental results and conclusion are in Section 2.4 and 2.5 respectively.

### 2.2 Factorization of $\Sigma_X^{-1} S_B$

To see why it is useful to factorize $\Sigma_X^{-1} S_B$, we will start with the spectral decomposition of Principle Component Analysis (PCA). PCA is a useful feature extraction algorithm for
data description but not for discrimination, since it does not take class labels into account (i.e., PCA is unsupervised). The space spanned by the first component vectors will not necessarily be the best for discrimination. However, the insufficiency of PCA for discrimination does not decrease its popularity as a tool for data description. It is well-known that PCA can be considered as a spectral-decomposition of the (sample) covariance matrix of the data [38]

\[ \Sigma_X U = U \Lambda_X. \]

This can be represented as

\[ \Sigma_X = \lambda_{X_1} u_1 u_1^T + \lambda_{X_2} u_2 u_2^T + \cdots + \lambda_{X_p} u_p u_p^T, \]

where \( \lambda_{X_1} \geq \lambda_{X_2} \geq \cdots \geq \lambda_{X_p} \) are the eigenvalues of \( \Sigma_X \) and \( U = \{u_1, u_2, \cdots, u_p\} \) are the corresponding eigenvectors. The covariance matrix \( \Sigma_X \) is decomposed into contributions \( \lambda_{X_k} u_k u_k^T \) from each principle component (PC). Geometrically, PCA can be thought of as a rotation of the axes of the original coordinate system to a new set of orthonormal basis which are ordered in terms of the amount of variance of the original data they account for.

Different kinds of analysis on the data can be done depending on which PCs are used to reconstruct the space. For example, it is clear that the covariance matrix can be reconstructed exactly from the \( r \) PCs corresponding to all non-zero eigenvalues, where \( r = p_X = \text{rank}(\Sigma_X) \). The first \( m \) (\( m \leq r \)) PCs construct the \( m \)-dimension subspace which keeps the most variance of the data compared to all the other \( m \)-dimensional subspaces. The last several PCs have small variance and may help to detect near-constant linear relationship between variables [38].
Similar to the “spectral decomposition” of PCA, we show that $\Sigma_{X}^{-1}S_{B}$ can also be decomposed as a combination of eigenvalues and eigenvectors of $\Sigma_{X}$ and $S_{B}$ [50].

**Theorem 1.** The basis vectors of Eq. (2.1) are equivalent to those of

$$
\sum_{i=1}^{p_{B}} \sum_{j=1}^{p_{X}} \frac{\lambda_{B_{i}}}{\lambda_{X_{j}}} u_{j}^{T}w_{i}w_{i}^{T}V = V\Lambda
$$

(2.3)

where $\lambda_{B_{i}}$ and $w_{i}$ are eigenvalues and eigenvectors of $S_{B}$, $\lambda_{X_{j}}$ and $u_{j}$ are eigenvalues and eigenvectors of $S_{X}$, $\text{rank}(S_{B}) = p_{B}$ and $\text{rank}(\Sigma_{X}) = p_{X}$.

**Proof.** Eq. (2.1) can be reworked as follows

$$
\sum_{i=1}^{p_{B}} \lambda_{B_{i}} w_{i}^{T}V = U\Lambda_{X}U^{T}V\Lambda
$$

$$
\left(\begin{array}{c}
\left(u_{1}^{T}\right) \\
\vdots \\
\left(u_{p}^{T}\right)
\end{array}\right)\lambda_{B_{1}}w_{1}^{T} + \cdots + \left(\begin{array}{c}
\left(u_{1}^{T}\right) \\
\vdots \\
\left(u_{p}^{T}\right)
\end{array}\right)\lambda_{B_{p}}w_{p}^{T}V = \Lambda_{X}U^{T}V\Lambda_{U}
$$

If $\Sigma_{X}$ is full ranked, $p = p_{X}$ (where $p \times p$ is the size of $\Sigma_{X}$), we can multiply both sides by $U\Lambda_{U}^{-1}$,

$$
\left(\begin{array}{c}
\frac{1}{\lambda_{u_{1}}}u_{1} \\
\ldots \\
\frac{1}{\lambda_{u_{p}}}u_{p}
\end{array}\right)\left(\begin{array}{c}
\sum_{i=1}^{p_{B}} \lambda_{B_{i}}\langle u_{1}, w_{i} \rangle w_{i}^{T} \\
\vdots \\
\sum_{i=1}^{p_{B}} \lambda_{B_{i}}\langle u_{p}, w_{i} \rangle w_{i}^{T}
\end{array}\right)V = V\Lambda
$$

$$
\sum_{i=1}^{p_{B}} \sum_{j=1}^{p_{X}} \frac{\lambda_{B_{i}}}{\lambda_{X_{j}}} \langle u_{j}, w_{i} \rangle u_{j}w_{i}^{T}V = V\Lambda.
$$

(2.4)
When $\Sigma_X$ is not full ranked, i.e., $p_X < p$, \[ \sum_{j=1}^{p_X} \frac{\lambda_{Bj}}{\lambda_{Xj}} u_j^T w_i u_j = \sum_{j=1}^{p_X} \frac{\lambda_{Bj}}{\lambda_{Xj}} u_j^T w_i u_j. \]

This is easy to show, since all those $u_j$ corresponding to a dimension of zero variance are orthogonal to all $w_i$, and, therefore, $u_j^T w_i = 0$.

The above result is very useful for computing the subspace defined by the first several basis vectors of Eq. (2.1), since it does not require the calculation of the inverse of $\Sigma_X$. This is achieved by means of the factorization of Eq. (2.1) derived in Eq. (2.3). This property is especially useful when $\Sigma_X$ is singular because it only involves non-zero eigenvalues of $\Sigma_X$. The motivation is that the subspace spanned by basis vectors $\{w_i\}$ of $S_B$ is a subspace of the one spanned by $\Sigma_X$, since $S_B$ represents the scatter of the class means, and $\Sigma_X$ represents the scatter of all the data. Hence, the eigenvectors of $\Sigma_X$ which correspond to zero eigenvalues are, in general, orthogonal to the space spanned by $\{w_i\}$ (i.e., $u_j^T w_i = 0$, when $\lambda_{Xj} = 0$, where $1 \leq i \leq p_B$).

A large number of methods have been proposed as solutions when $\Sigma_X$ is singular. For example, the PCA-LDA strategy first projects the data onto the range space of $\Sigma_X$ such that Eq. (2.1) is applicable. This technique has received criticism [76, 8] because the discarded null space may contain significant discriminant information. However, our discussion above shows that this is not true. Because $\text{NULL}(\Sigma_X) = \text{NULL}(S_B) \cap \text{NULL}(S_W)$, where $\text{NULL}(\cdot)$ denotes the null space, the discarded null space of $S_X$ is also the null space of $S_B$, which is orthogonal to all non-zero eigenvectors of $S_B$. Hence, there is no between-class discriminant information lost.

It is important to note that Eq. (2.3) is equivalent to the classical PCA-LDA algorithm iff all the eigenvectors associated to non-zero eigenvalues are used [79]. This can be summarized in the following result.
**Result 2.** The projection matrix $V$ found by Eq. (2.3) and the projection matrix $U_s \tilde{V}$ given by

$$(U_s^T S_B U_s) \tilde{V} = (U_s^T \Sigma_X U_s) \tilde{V} \Lambda,$$

are identical; where $U_s = \{u_j|\lambda_{Xj} \neq 0\}$.

See Appendix C.1 for proof.

Another well-known algorithm is called direct LDA (DLDA). It first projects the data onto the space spanned by $\{w_i\}$, which usually has much lower dimension than the original space, then compute the basis vectors of $\Sigma_X$ in this reduced space and choose the ones with small eigenvalues as the solution, then map this result back to the original space. Mathematically, we do the following steps:

$$S_B W = W \Lambda_B$$

$$Z = W \Lambda_B^{-1/2}$$

$$Y = Z^T \Sigma_X Z$$

$$YV_Y = V_Y \Lambda_Y$$

$$V = \Lambda_Y^{-1/2} V_Y^T Z^T.$$
Figure 2.1: (a) In this two-dimensional example, Eq. (2.3) generates the correct solution, while Eq. (2.5) does not. In (b-c), we project two Gaussians embedded in a high dimensional space onto the direction found by Eq. (2.3), shown in (b) and Eq. (2.5) shown in (c).
In Fig. 2.1, the two examples show that our solution is superior to DLDA. In Fig. 2.1 (a) we have a simple two-dimensional example with two classes and a non-singular $\Sigma_X$. In this case, the direction found by Eq. (2.5) is not a desirable solution because it is dominated by the between-class variance, while Eq. (2.3) gives the optimal solution, which is identical to LDA in this case. Fig. 2.1 (b) and (c) show the projection of two Gaussians embedded in a high dimensional space (where $\Sigma_X$ is singular and LDA is not applicable) onto the one-dimensional space. As we can see in the figure, Eq. (2.3) also outperforms Eq. (2.5) in this case.

Eq. (2.3) can also be used wherever one does not want to compute the inverse of $\Sigma_X$. For example, this is useful for large matrices, because it saves us of extensive computations. The algorithm in Theorem 1 thus defines a robust, easy way of finding the discriminant features of the data.

Another advantage of Eq. (2.3) is that it can now be used to eliminate some of the eigenvectors of $\Sigma_X$ that are unnecessary or corrupted by noise. Following our notation, we define the reconstruction of $\Sigma_X^{-1}S_B$ in the subspace given by a set of $k$ eigenvectors of $\Sigma_X$ as

$$
(\Sigma_X^{-1}S_B)_S = \sum_{u_j \in S} \sum_{i=1}^{p_B} \frac{\lambda_{B_i}}{\lambda_{X_j}} u_j^T u_i w_i^T,
$$

where $S$ is a set of $k$ elements drawn from $\{u_1, \cdots, u_{p_X}\}$ according to criterion $I$, without repetition.

Note that Eq. (2.6) defines a general form for the use of a subset of bases of $\Sigma_X$. Depending on the criterion $I$ used for such a selection, the final set of discriminant feature vectors $V$ will vary; where now

$$
(\Sigma_X^{-1}S_B)_S V = \Lambda.
$$
The goal thus reduces to finding the criterion $I$ that prunes those noisy bases of $\Sigma_X$ but keeps those that contribute to the discrimination of the class distributions.

To see why Eq. (2.7) is equivalent to the pruning of noisy bases, we need to validate the reconstruction of $\Sigma_X^{-1}S_B$ by Eq. (2.6) and justify its physical meaning. Firstly, we will introduce the concept of *discriminant power*.

The objective of Eq. (2.2) is to maximize $\frac{v^T S_B v}{v^T \Sigma_X v}$. For the generalized eigenvalue decomposition as defined in Eq. (2.1),

$$
\lambda_i = \frac{v_i^T S_B v_i}{v_i^T \Sigma_X v_i},
$$

the eigenvalue reflects the ratio of between-class variance to the within-class variance in the direction $v_i$, i.e., how well the data is separated in the direction $v_i$. Thus we refer to $\sum_{i=1}^{PB} \lambda_i$ as the *discriminant power* of LDA.

It is well-known that $\sum_{i=1}^{PB} \lambda_i = tr(\Sigma_X^{-1}S_B)$ (where $tr(A)$ is the trace of $A$). In the following theorem, we show that the *discriminant power* can also be expressed by the combination of eigenvalues and eigenvectors of $S_B$ and $\Sigma_X$.

**Theorem 3.** The $tr(\Sigma_X^{-1}S_B)$ is equal to

$$
\sum_{i=1}^{PB} \sum_{j=1}^{PX} \frac{\lambda_{Bi}}{\lambda_{Xj}} (u^T_j w_i)^2,
$$

where $\{\lambda_{Bi}\}_{i=1}^{PB}$ and $\{\lambda_{Xj}\}_{i=1}^{PX}$ are eigenvalues of $S_B W = W \Lambda_B$ and $\Sigma_X U = U \Lambda_X$, $W = \{w_1, \ldots, w_{PB}\}$ and $U = \{u_1, \ldots, u_{PX}\}$ are the corresponding eigenvectors.

See Appendix C.2 for proof.

Therefore, the discriminant power of Eq. (2.6) is the summation of discriminant power of each PCs selected by our rule $I$. 

26
The advantage of Eq. (2.6) is that it allows us to eliminate those noisy bases from the definition of discriminability in a single equation. This way, one can calculate the true discriminability of the class distribution, without being influence by the noisy components.

The problem to be solved next is that of selecting a criterion $I$ that distinguishes between the noisy and discriminant bases.

### 2.3 Correlation-based Criterion

The decomposition shown in Eq. (2.3) will now prove instrumental for the understanding of the role the eigenvectors of each matrix in discriminant analysis. In Eq. (2.3), $u_j^T w_i$ acts as a “switch,” controlling which $u_j$’s are correlated to the $w_i$’s. To be able to play a role in the selection of discriminant vectors $V$, $u_j$ has to be correlated to the $\text{ran}(S_B)$, i.e., $u_j^T w_i \neq 0$ for at least one $i$ in $\{1, \ldots, p_B\}$, regardless of the value of $\lambda_{X_j}$, where $\text{ran}(S_B)$ is the range space of $S_B$. This is illustrated in Fig. 2.2. In (a) the two distributions have a large variance about the $z$ axis. In (b) this variance is close to zero. In both cases, however, the eigenvector aligned with $z$ is not correlated to the $\text{ran}(B)$ (which in this case is the $x$-$y$-plane). This means that the eigenvector aligned with $z$ can be eliminated from consecutive computations because this does not (and cannot) carry any discriminant information.

In the examples shown in Fig. 2.2, $S$ will thus be constructed with those eigenvectors that define the $x$-$y$-plane only.

The above result can be easily formulated in the form of a correlation-based measure as

$$I_j = \sum_{i=1}^{p_B} (u_j^T w_i)^2, \quad 1 \leq j \leq p_X. \quad (2.9)$$
Figure 2.2: In (a) the data has large variance about the $z$ axis. In (b) this variance is small. In neither case $z$ provides any discriminant information.

To illustrate the use of this measure, we will employ the data of the ETH-80 database [40]. The ETH-80 database contains images of eight object categories. Each category includes images of ten objects, each photographed from 41 distinct orientations. For each of the images, we obtain the silhouette of the object (i.e., the contour that separates the object from the background) and compute its centroid. Then, we calculate the length of equally separated lines that connect the centroid with the silhouette. In our experiments we obtain a total of 300 distances, generating a feature space of 300 dimensions. This allows us to calculate the between-class scatter matrix (to represent the metric given by $S_B$) and the sample covariance matrix of the data (as $\Sigma_X$). The eigenvectors of $\Sigma_X$ can now be sorted in decreasing order of eigenvalue. This is shown in Fig. 2.3(a), where we only show the eigenvalue plot for the first 60 eigenvectors for clarity. In this plot, we also show dashed and dotted lines to represent the cut at 70% and 90% of the total variance. Maintaining this order of eigenvectors, we calculate the correlation of each eigenvector $u_j$ to $\text{ran}(S_B)$ using Eq. (2.9). These values are plotted in Fig. 2.3(b), where the lines defining the 70% and 90% of the total variance are still shown. Keeping the same ordering of
eigenvectors in (a) and (b) allows us to see that some of the eigenvectors associated to small eigenvalues are very correlated to the $\text{ran}(S_B)$. Therefore, eliminating all the eigenvectors with associated small variance is not a good idea, because these should contribute to the outcome $V$. Furthermore, it is important to note that the other eigenvectors $u_j$ that should most worry us are those that are only slightly correlated to $\text{ran}(S_B)$ but have an associated small eigenvalue. While these eigenvectors should not play a role in defining $V$, some will, because their variance is sufficiently small to have an effect in (2.6). In summary, we should keep the eigenvectors $u_j$ that are most correlated to $\text{ran}(S_B)$ and prune the rest. Hence, it is only natural to sort the eigenvectors $u_j$ from most to least correlated with $\text{ran}(S_B)$.

One could argue that keeping those eigenvectors that account for 90% of the total variance would solve some of the problems defined above. To show that this is not so, we will now use our set of face images. The AR-face database consists of frontal-view face images of 100 people. Here, we will use the first 13 images taken during a first session, including images with distinct facial composites, occlusions and illuminations. We first cropped the face part of these images (without including hair or background) and resized 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 aligned all faces to have a common position for the eyes and mouth. We then calculate the between-class scatter matrix (to represent the metric given by $S_B$) and the covariance matrix of the data (as $\Sigma_X$). The resulting eigenvectors of $\Sigma_X$ are finally sorted in describing order of eigenvalue. This is shown in Fig. 2.3(c). Again, we show dashed and dotted lines to represent the cut at 70% and 90% of the total variance. In Fig. 2.3(d) we show the correlation value of the eigenvectors $u_j$ with those of $S_B$. In this case, we see that even if one included all the $a_j$ that account for 90% of the total variance, many useful eigenvectors would still be erroneously eliminated.
Figure 2.3: (a) Plotted here are the eigenvalues of the sample covariance matrix of the ETH-80 database in decreasing order. (b) Shows the correlation values of each of the eigenvectors of $\Sigma_X$ (ordered by eigenvalue) with those of $S_B$. Dashed and dotted lines represent the cutoff points with 70% and 90% of the total variance. (c)-(d) the same for the images of AR-face database.

The last problem that needs to be addressed within our framework is that of defining a simple, efficient way of selecting the cutoff point which specifies the bases to be pruned. This we can do with the help of a monotonically decreasing density function that approximates the correlation values, because this can (in turn) be employed to specify a confidence interval. This can be easily done, because the sum of the correlations over all eigenvectors of $\Sigma_X$ is equal to $p_B$. This means that

$$\frac{1}{p_B} \sum_{j=1}^{p_X} I_j = \frac{1}{p_B} \sum_{j=1}^{p_X} \sum_{i=1}^{p_B} (u_j^T w_i)^2 = 1,$$

which facilitates the definition of the following normalized correlation value

$$f_j = \frac{1}{p_B} I_j = \frac{1}{p_B} \sum_{i=1}^{p_B} (u_j^T w_i)^2.$$  \hspace{1cm} (2.10)

Noting that when the number of classes is large (i.e., $p_B$ is large) the curve defined by the set $\{f_1, \ldots, f_{p_X}\}$ is less steeped (because $\text{ran}(S_B)$ is a subspace of high dimensionality) and that when $p_B$ is small the curve goes to zero much faster (because $\text{ran}(S_B)$ represents
a subspace of low dimensionality), allows us to approximate this curve with an exponential function of the form

\[ g_y = \lambda e^{-\lambda y}, \]

where \( \lambda = f_1 \). In Fig. 2.3(c) we show how \( g_y \) can approximate the shape of the original curve defined by \( \{f_1, \ldots, f_{p_X}\} \), which corresponds to the eigenvectors of \( \Sigma_X \) computed earlier using the ETH-80 database. As seen in the figure, this approximation can now be used to calculate the confidence interval \( h \) of \( g_y \) as

\[ \int_0^m \lambda e^{-\lambda y} dy = h. \]

This result is important because it can be used to find the most adequate number \( k \) of eigenvectors of \( \Sigma_X \) as \( -e^{-\lambda y}|_0^m = h \), which means

\[ 1 - e^{-\lambda m} = h \]

\[ m = -\ln(1 - h)/\lambda \]

\[ k = \min(\lfloor m \rfloor, p_X). \tag{2.11} \]

In our algorithm, we will always select the \( k \) providing a 90% confidence interval. In the example shown in Fig. 2.4 \( k = 16 \).

### 2.4 Experimental Results

In our first set of experiments we used three different datasets. Two of these are the ETH-80 database and the AR-face database [48] previously introduced. The other set is the Sitting Posture Distribution Maps (SPDM) set of [66].

In the ETH test, we selected the 41 images of one randomly chosen object to represent the testing set (i.e., a total of 328 images) and used the rest of the images for training.
\[
\int_0^m \lambda e^{-\lambda y} dy = 0.9
\]

\[
k = \left\lfloor \frac{m}{\lambda} \right\rfloor = 16
\]

Figure 2.4: The confidence interval of the exponential function \(g_y = \lambda e^{-\lambda y}\) with probability 0.9, where \(\lambda = f_1 = \frac{1}{\mu} = 0.1428\).

Fig. 2.5(a) we show the correlation plot of the eigenvectors of the sample covariance matrix with those of the between-class scatter matrix together with their approximation \(g_y\). The vertical axis in Fig. 2.5(d) shows the recognition rate obtained when one keeps the first \(k\) eigenvectors shown in (a). The dotted line in (a) and (d) represents the cutoff at the 90% confidence interval.

In Fig. 2.5(g) we show the results obtained when one selects the \(k\) eigenvectors that keep \(r\)% of the total variance. Here, we have included a set of large dots to indicate the recognition rates obtained when \(r = \{90, 91, \ldots, 99\}\). Similarly, the dots in Fig. 2.5(j) specify the recognition rates when \(r\)% of the total discriminant information defined by \(J(a_j)\) is kept, where now \(r = \{50, 60, 70, 80, 90, 91, \ldots, 99\}\). We note that in both cases, the recognition rate vary considerably depending on the value of \(r\). In (g) the results vary from a high of 75.9% to a low of 68.29%, which are achieved when \(r = 96\) and \(r = 90\), respectively. In (j) the recognition rates vary from 75% to 51.8%, and the maximum is achieved when \(r = 60\). We will now show that the values of \(r\) that maximizes (or
minimizes) the recognition rates obtained using these two other methods, are very different when the database changes. However, our method will remain stable.

To see this, we will now use the images of the AR-face database. The AR-face set consists of frontal-view face images of 100 people. Here, we will use the first 13 images taken during a first session, including images with distinct facial composites, occlusions and illuminations, for training. The other 13 images (taken during a second session) are used for testing. Again, the sample covariance matrix and the between-class scatter matrix are used as metrics for $\Sigma_X$ and $S_B$. Fig. 2.5(b) plots the correlation values $f_j$ and their approximation $g_y$. Fig. 2.5(e) plots the corresponding recognition rates. Our result is specified by the dotted line.

As we did above, we now compare the results obtained using our method to those calculated when $r\%$ of the total variance or the total discriminant information is kept. This is shown in Figs. 2.5(h) and (k). Note that the maximum recognition rate in (h) is obtained when $r = 99$ and in (k) when $r = 80$. These are different to the optimal values of $r$ reported in (g) and (j). Nonetheless, we see that the proposed approach provides a stable way of selecting the best bases of $\Sigma_X$ – only pruning those bases that are either noisy (reduce the recognition rate) or uninformative (do not provide a statistically significant increment of the recognition rate).

In the SPDM database, the sample feature vectors were collected using a chair equipped with a pressure sensor sheet. Feature vectors correspond to 1,280 pressure (force) values given by equally distributed points on the seating-pan of the chair. Here, the goal is to classify each feature vector into one of the 10 possible sitting postures [66]. The SPDM database includes five samples for each of the postures for a total of 50 people. Of the five samples available from each person, three are randomly selected as training samples.
and two for testing. This divides the data to 1,500 samples for training and 1,000 for testing. Correlation values $f_j$ and their approximation $g_y$ are shown in Fig. 2.5(c). The corresponding recognition rates are in Fig. 2.5(f), where the dotted line marks our solution. In this case, the results obtained with the other two methods are unable to select a $r$ which archives a good performance. This is shown in Figs. 2.5(i) and (l).

A statistical analysis of the results obtained using the ETH and SPDM databases is shown in the first two rows of Table 2.1. In this case, a bootstrap strategy is used. In the ETH set, the 41 images of a randomly selected object are used for testing and the rest for training. This is repeated 100 times. In the SPDM set 100 samples of each posture are randomly selected for training and the rest are used for testing. This is also repeated 100 times. Average and standard deviations are shown in the table.

In these results, we can see that the proposed algorithm consistently outperforms the others. This is because, in the ETH-80 and SPDM datasets, there is a considerable amount of noisy bases that make the classification problem challenging. When the number of noisy features is reduced, this difference should become smaller. To test this, we used two additional databases from the UCI repository [5]. The first is the “Multi-feature Digit Dataset” (MDD) set, which includes 200 handwritten samples of each of the ten digits (0 to 9), and we used a representation ($MDD$-kar) based on 47 of the Zernike moments of the image [5]. The number of noisy bases in a 47-dimensional space is expected to be much lower than those of the ETH and SPDM sets. The bootstrap approach is employed to produce our results, with a training set containing 50% of the samples (randomly selected) and the testing set containing the remaining 50%. The operation is repeated 100 times and average and standard deviations are reported. As one can see in Table 2.1, the difference in performance is now reduce to a minimum.
Figure 2.5: In (a-c) we have ordered the eigenvectors of the covariance matrix $\Sigma_X$ obtained from the ETH, AR and SPDM databases, from most to least correlated to $\mathbf{r}_\mathbf{m}(\mathbf{S}_B)$. In (d-f) We show the recognition rates when the $k$ first eigenvectors, as ordered in (a-c), are used. Here, $k$ is the horizontal axis and the dotted vertical line is the result given by our algorithm. (g-i) Shows the recognition rate obtained when $k$ eigenvectors are kept and these are ordered in decreasing order of eigenvectors. (j-l) Does the same, but when the eigenvectors are order from most to least discriminant as given by $J(a_j)$. The dots in (g-l) specify the successful classification rate when different percentages of the variance (or discriminant power) are retained (see text).
Table 2.1: Statistical comparison. In this table, \( \text{eig}_r \) means we use the eigenvectors that keep \( r\% \) of the total variance, and \( J(\mathbf{u})_r \) indicates that all those eigenvectors of \( \Sigma_X \) that keep \( r\% \) of the total discriminant power given by \( J(\mathbf{u}) \) are used. In these experiments a bootstrap approach is used to generate average and standard deviations. Bolded results indicate statistical significance.

To further illustrate this point we used the Ionosphere set of [5], where the dimensionality of the data is even smaller. In this case, 34 radar attributes are employed to describe signals bouncing off the ionosphere. Our task is to determine whether the incoming signal is good (i.e., shows evidence of some type of structure) or not. Again half of the 351 samples are used for training and half for testing. This is repeated 100 times and average and standard deviations are shown in Table 2.1. As the dimensionality becomes more manageable (34 dimensions in this final test), the number of noisy bases decreases. As a consequence, and as predicted, Fisher-Rao’s LDA and the other methods perform better, making it less necessary to use pruning methods.

### 2.5 Conclusions

The classical formulation of LDA, based on the simultaneous diagonalization of two symmetric matrices (i.e., \( \Sigma^{-1}_X S_B V = V \Lambda \)), is known to be very sensitive to noisy bases in \( \Sigma_X \). In this chapter, we have presented a simple correlation-based criterion that can be efficiently used to prune those noisy bases without eliminating those associated to the true
discriminant power of the underlying structure of the data. Experimental results using five different databases have been used to illustrate the robustness of the proposed approach.
CHAPTER 3

SUBCLASS DISCRIMINANT ANALYSIS

Over the years, many Discriminant Analysis (DA) algorithms have been proposed for the study of high-dimensional data in a large variety of problems. As shown earlier, 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 chapter, 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.
3.1 Introduction

Feature extraction via Discriminant Analysis (DA) is one of the most sought after approaches in applications in computer vision [25, 63, 21, 4, 49]. 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 [22, 61, 25, 51]. 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 [51, 50].

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) [22, 61, 25]. This assumption has restricted the use of LDA considerably. Over the years, authors have defined several extensions to the basic formulation of LDA [25, 29, 32, 3, 43]. One such method is Non-parametric Discriminant Analysis (NDA) where the Normal assumption is relaxed in one of the two metrics [25]. 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) [43] or Penalized DA (PDA) [29]. 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
Figure 3.1: 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 $\Sigma_X$ and $\Sigma_B$ represent the sample covariance matrix and the between-subclass scatter matrix (see text).

slightly deviate from the Normal assumption, by making the metrics of DA a bit more flexible. These methods generally outperform LDA in practice, 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. 3.1 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) [32] 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) [3] 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 [52, 34]. For example, in our Fig. 3.1, 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,

$$\Sigma_B V = \Sigma_X V \Lambda,$$

(3.1)
where $\Sigma_B$ is the *between-subclass* scatter matrix, $\Sigma_X$ is the covariance matrix of the data, $V$ is a matrix whose columns correspond to the discriminant vectors, and $\Lambda$ is a diagonal matrix of corresponding eigenvalues. Since $\Sigma_B$ 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. (3.1) 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 $V_q$ (where $V_q$ 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 chapter 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. (3.1). 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. (3.1) 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 [50]. 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 [14] is first used to estimate the true underlying distribution of each class, before using LDA [75, 31, 49]. Our goal is to optimize classification, not to recover the true underlying (but unknown) distribution of the data. Moreover, the methods presented in [75, 31, 49] 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 3.2, 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 $\Sigma_B$. Section 3.4 defines the two criteria used to determine the optimal number of subclasses. Experimental results are in Section 3.5. We conclude in Section 3.6.

### 3.2 The Metrics of Discriminant Analysis

Most DA algorithms defined thus far are based on Fisher-Rao’s criterion [22, 61], which is given by

$$\frac{|v^T S_1 v|}{|v^T S_2 v|},$$

(3.2)

where the matrices $S_1$ and $S_2$, 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, $S_1 = S_B$ and $S_2 = S_W$ resp., in Eq. (3.2); where $S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$, $C$ is the number of classes, $\mu_i$ the sample mean of class $i$, $\mu$ the global mean (including the samples of all classes), $S_W = \frac{1}{n} \sum_{i=1}^{C} \sum_{j=1}^{n_i} (x_{ij} - \mu_i)(x_{ij} - \mu_i)^T$, $x_{ij}$ 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, $S_1$ or $S_2$. For example, NDA uses the following non-parametric version of the between-class scatter matrix for $S_1$, $S_B = \frac{1}{n} \sum_{i=1}^{C} \sum_{j=1}^{n_i} \sum_{l=1}^{C} \alpha^l_{ij} (x_{ij} - M^l_{ij})(x_{ij} - M^l_{ij})^T$; where $M^l_{ij}$ is the mean of the $k$ nearest samples to $x_{ij}$ belonging to class $l (l \neq i)$ and $\alpha^l_{ij}$ 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 $S_B$; i.e., $\sum_{i=1}^{C-1} \sum_{j=i+1}^{C} \omega(d_{ij}) S_{ij}$, where $S_{ij} = (\mu_i - \mu_j)(\mu_i - \mu_j)^T$, $d_{ij}$ 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 $S_2$. For instance, PDA includes a penalizing matrix $\Omega$ in the definition of the within-class scatter matrix, $S_2 = S_W + \Omega$. Here, $\Omega$ penalizes those eigenvectors of Eq. (3.2) 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. 3.1, 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 $S_1$. For example, the $\text{rank}(S_B) \leq C - 1$ 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 to be sufficient to discriminate the $C$ classes (assuming that the data distributions are correctly represented by our metrics $S_1$ and $S_2$) [25]. 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
Figure 3.2: (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 $S_B$ 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.

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. 3.2(a). In this example, LDA cannot recover the two-dimensional space necessary to successfully discriminate both classes, because $\text{rank}(S_B) = 1$; Fig. 3.2(b). This problem can be solved by defining a metric for $S_1$ that not only considers the difference between class means but also between their covariances. Heteroscedastic LDA [44] (HLDA) is such a method, where the authors use the Chernoff
distance to estimate class similarity based on both means and covariances; i.e., \( S_1 \) is redefined as

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

(3.3)

where \( \Sigma_i \) is the covariance matrix of the samples in class \( i \), \( \Sigma_{ij} \) the average between \( \Sigma_i \) and \( \Sigma_j \), 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 [22, 61, 25], one could define the between-subclass scatter matrix as [78],

\[
\hat{\Sigma}_B = \sum_{i=1}^{C} \sum_{j=1}^{H_i} p_{ij} (\mu_{ij} - \mu)(\mu_{ij} - \mu)^T,
\]

(3.4)

where \( H_i \) is the number of subclass divisions in class \( i \), and \( p_{ij} \) and \( \mu_{ij} \) 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. (3.4) 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 4.** The covariance matrix of the subclass means \( \hat{\Sigma}_B \) can be decomposed as \( \hat{\Sigma}_B = S_B + Q \) with \( Q = \sum_{i=1}^{C} \frac{n_i}{n} Q_i \), and where \( Q_i \) measure the distance between the means of the subclasses in the same class only; i.e., \( Q_i = \sum_{j=1}^{H_i} \frac{n_{ij}}{n_i} (\mu_{ij} - \mu_i)(\mu_{ij} - \mu_i)^T \), 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 C.3) shows that when we use Eq. (3.4) in Eq. (3.1), 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. 3.1. 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 $\Sigma_B$ that favors the separability of those subclasses that correspond to different classes. Formally,

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

(3.5)

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

However, it is important to note that the information of the intra-subclass scatter 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. (3.5) 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. (3.5) favors class separability, only those subclasses that are essential to boost the classification result will actually be separated in the reduced space.

**Result 5.** Compared to $\hat{\Sigma}_B$, the new definition of between-subclass scatter $\Sigma_B$ emphasizes the role of class separability over that of intra-subclass scatter.

See Appendix C.4 for the proof.
3.3 Flexibility of SDA

To see the flexibility of SDA, in this section, we list the relationship between SDA and some well-known algorithms.

1. SDA versus LDA: We note that Eq. (3.5) is equal to $S_B$ when $H_i = 1$ for all $i = \{1, \ldots, C\}$. And, in general, it can be readily shown that the $\text{rank}(\Sigma_B) \leq \min(H - 1, p)$, 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, and, hence, SDA can solve the problem posed by the deficiency of the rank of $S_B$. An example of this is shown in Fig. 3.2. In this example, the range of $\Sigma_X$ 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 $1 \leq \text{rank}(\Sigma_B) \leq 3$. One such solution, for $H = 4$ is shown in Fig. 3.2(c), in which case the result is optimal.

2. SDA versus PCA: We now explore the relationship between the eigenvectors of Eq. (3.1) and those of PCA under some specific conditions. All the proofs are in Appendix C.

**Lemma 6.** If we have an infinite number of samples, $n = \infty$, and the data of each class is distributed according to a common Gaussian distribution, then there is always a partition of the data into a set of subclasses for which $\Sigma_B = \alpha I$, where $I$ is the identity matrix.

**Lemma 7.** When $\Sigma_B = \alpha I$, Eq. (3.1) selects the eigenvectors associated to the smallest eigenvalues of $\Sigma_X$.  

48
Lemma 6 and Lemma 7 can be used to prove the following.

**Proposition 8.** If $n = \infty$ and the data of each class corresponds to a common Gaussian distribution, we can always find a partition of the data for which the eigenvectors associated to the smallest eigenvalues of Eq. (3.1) are proportional to those of PCA.

This result shows that in order to obtain equivalent eigenvectors to those of PCA, we need to select those eigenvectors of Eq. (3.1) associated to the smallest eigenvalues.

Next, we will show the relationship between those eigenvectors of Eq. (3.1) and those of PCA when another assumption on the data holds.

**Lemma 9.** If $\Sigma_X$ and $\Sigma_B$ are full rank matrices, and have identical eigenvectors but $\Sigma_X \neq \Sigma_B$ (i.e., different eigenvalues), then the eigenvectors of PCA, $U$, are related to those of Eq. (3.1), $V$, by $V = UM$, where $M$ is the eigenvector matrix of

$$\left( \Lambda^{-1}_X \Lambda_B \right) M = MA,$$

(3.6)

$\Lambda_B$ is the eigenvalue matrix of $\Sigma_B U = U \Lambda_B$, and $\Lambda_X$ is the eigenvalue matrix of the PCA equation, $\Sigma_X U = U \Lambda_X$.

**Lemma 10.** Let $\Lambda_B = \text{diag}(\lambda_{B_1}, \lambda_{B_2}, \ldots, \lambda_{B_p})$, and $\Lambda_X = \text{diag}(\lambda_{X_1}, \lambda_{X_2}, \ldots, \lambda_{X_p})$ in Eq. (3.6), then:

- if $\frac{\lambda_{B_1}}{\lambda_{X_1}} \geq \frac{\lambda_{B_2}}{\lambda_{X_2}} \geq \ldots \geq \frac{\lambda_{B_p}}{\lambda_{X_p}} \Rightarrow V = \{u_1, \ldots, u_p\}$.
- if $\frac{\lambda_{B_1}}{\lambda_{X_1}} \leq \frac{\lambda_{B_2}}{\lambda_{X_2}} \leq \ldots \leq \frac{\lambda_{B_p}}{\lambda_{X_p}} \Rightarrow V = \{u_p, \ldots, u_1\}$.

We can now use Lemmas 9 and 10 to prove the following.
Proposition 11. If $\Sigma_X$ and $\Sigma_B$ are full rank matrices, and have identical eigenvectors but $\Sigma_X \neq \Sigma_B$ (i.e., different eigenvalues), then the $i^{th}$ eigenvector of Eq. (3.1) is the $j^{th}$ eigenvector of PCA if $\frac{\lambda_B}{\lambda_X}$ is the $i^{th}$ largest among $\left\{ \frac{\lambda_B}{\lambda_X^1}, \ldots, \frac{\lambda_B}{\lambda_X^{H-1}} \right\}$.

3. SDA versus Nearest Neighbor rule: Nearest Neighbor (NN) rule is a simple and efficient idea that can be used as a classifier. After the dimensionality reduction, we usually use NN for classification.

When each sample corresponds to one subclass, it is easy to show that Eq. (3.1) is equivalent to $IV = V\Lambda$ since $\Sigma_B = \Sigma_X$. That means that $V = I$, and SDA does not reduce the dimensionality at all. Here, classification after SDA is equal to using NN in the original feature space.

Thus far, we have formulated SDA and shown its flexibility, as stated earlier, the goal is now reduced to finding a way to automatically select the optimal number of subclass divisions.

3.4 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., $S_1$) and the first $i$ eigenvectors given by the metric to be minimized ($S_2$) 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.

### 3.4.1 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 $V_q$ from Eq. (3.1) using all but the $i^{th}$ sample of the training data, $X_i = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. The sample left-out, $x_i$, 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 $x_i$ when we use $X_i$ 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

$$H_o = \underset{H}{\text{argmax}} \ R_H. \quad (3.7)$$

This allows us to compute the optimal $\Sigma_B$ with $H_o$. The resulting projection matrix $V_q^*$ is a $p \times q$ matrix whose columns are the eigenvectors associated to the $q$ largest eigenvalues of Eq. (3.1) 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 $H$.

### 3.4.2 Stability criterion

Our second approach is based on a stability criterion which we have recently introduced in [50]. In particular, we showed that the success of any linear method, such as that given
Algorithm 1 $SDA_{loot}$

Initialization: $R_H = 0, \forall H$.

for $i = 1$ to $n$ do
  Generate the training set $X_i = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$ using the NN clustering defined in Algorithm 3.
  Calculate $\Sigma_X$ using $X_i$.
  for $H = C$ to $t \cdot C$ do
    Calculate $\Sigma_B$ using Eq. (3.5).
    Perform the eigenvalue decomposition $\Sigma^{-1}_X \Sigma_B V = V \Lambda_X$.
    Project the data onto the subspaces of $V_q$; i.e., $Y_i = V_q^T X_i$.
    if the sample $x_i$ is classified correctly then
      $R_H +=$
    end if
  end for
end for

$H_o = \arg\max_H R_H$.

Calculate $\Sigma_B$ and $\Sigma_X$ using all our training data and $H_o$.

The final projection matrix $V_q^*$ is given by the first $q$ columns of $V$, where $\Sigma^{-1}_X \Sigma_B V = V \Lambda_X$.

$^\dagger$ 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.
in Eq. (3.1), does not depend on whether the data is homoscedastic or not. We have demonstrated that the results of the linear methods based on Eq. (3.2) are not guaranteed to be correct when the smallest angle between the $i^{th}$ eigenvector of $S_1$ and the first $i$ eigenvectors of $S_2$ 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. (3.1)) may not produce the most desirable result, we need to go back to the Fisher-Rao criterion given in Eq. (3.2) and analyze its factorization.

In Eq. (2.3), we have shown the factorization of $\Sigma_X^{-1}S_B$. Similarly, $\Sigma_X^{-1}S_B$ can be decomposed as the combination of eigenvectors and eigenvalues of $\Sigma_B$ and $\Sigma_X$ as following:

$$\Sigma_X^{-1}S_B = \sum_{i=1}^{H-1} \sum_{j=1}^{p_X} \frac{\lambda_B_i}{\lambda_X_j} u_j^T w_i u_i^T w_j T.$$ (3.8)

This decomposition has a simple geometric interpretation. Based on Eq. (3.8), Eq. (3.1) weights each pair of vector $(u_j, w_i)$ according to their agreement $(u_j^T w_i)$ and variance $\lambda_B_i/\lambda_X_j$. Those pairs with similar direction will have higher weight $(u_j^T w_i)$, than those that differ. Then, for those pairs that agree, the goal is to maximize the ratio $\lambda_B_i/\lambda_X_j$.

Eq. (3.8) also helps us better understand how the relationship of the two scatter matrices (the relationship between corresponding eigenvectors and eigenvalues) influences the result of Eq. (3.1). One special case is when for every $w_i$, $(1 \leq i \leq H - 1)$, there exits $u_{k_i}$, $(1 \leq k_i \leq p)$, such that $w_i$ and $u_{k_i}$ are in the same direction, i.e.,

$$\langle u_j, w_i \rangle = \begin{cases} 1, & \text{for } j = k_i \\ 0, & \text{for } j \neq k_i. \end{cases}$$

Since $u_j$ and $w_i$ are vectors with unit norm, $(u_j^T w_i)$ is equal to $\cos \theta_{i,j}$, where $\theta_{i,j}$ is the angle between those two vectors. $(u_j^T w_i) = 1$ when $\theta_{i,j} = 0$, and 0 when $\theta_{i,j} = \pi/2$. 

53
then,
\[
\frac{\Sigma_B}{\Sigma_X} = \sum_{i=1}^{H-1} \frac{\lambda_{B_i}}{\lambda_{X_k}} w_i w_i^T = \sum_{i=1}^{H-1} \frac{\lambda_{B_i}}{\lambda_{X_k}} u_{k_i} u_{k_i}^T
\] (3.9)

**Proposition 12.** If \( \Sigma_X \) is full rank and has \( H - 1 \) eigenvectors which are equal to those of \( \Sigma_B \) (i.e. \( u_{k_i} = w_i \), where \( 1 \leq i \leq H - 1 \) and \( 1 \leq k_i \leq p \)), then the \( i \)th eigenvector of Eq. (3.1) is \( u_j \) if \( \frac{\lambda_{B_j}}{\lambda_{X_k}} \) is the \( i \)th largest among \( \left\{ \frac{\lambda_{B_1}}{\lambda_{X_k}}, \ldots, \frac{\lambda_{B_{H-1}}}{\lambda_{X_k}} \right\} \).

Especially, if \( k_i = i \), \( 1 \leq i \leq H - 1 \), i.e., \( w_1 = u_1, w_2 = u_2, \ldots, w_{H-1} = u_{H-1} \), then, the eigenvectors of Eq. (3.1) are equal to the first \( H - 1 \) eigenvectors of \( \Sigma_X \) (the first \( H - 1 \) eigenvectors of PCA), and the ordering depends on the ratio \( \frac{\lambda_{B_i}}{\lambda_{X_k}} \). For example:

- if \( \frac{\lambda_{B_1}}{\lambda_{X_1}} \geq \frac{\lambda_{B_2}}{\lambda_{X_2}} \geq \cdots \geq \frac{\lambda_{B_{H-1}}}{\lambda_{X_{H-1}}} \), the eigenvectors of Eq. (3.1) are \( \{ u_1, u_2, \ldots, u_{H-1} \} \).
- if \( \frac{\lambda_{B_1}}{\lambda_{X_1}} \leq \frac{\lambda_{B_2}}{\lambda_{X_2}} \leq \cdots \leq \frac{\lambda_{B_{H-1}}}{\lambda_{X_{H-1}}} \), the eigenvectors of Eq. (3.1) are \( \{ u_{H-1}, \ldots, u_2, u_1 \} \).

**Corollary 13.** Assume \( \Sigma_X \) is full rank and its first \( H - 1 \) eigenvectors are identical to those of \( \Sigma_B \) but with different eigenvalues, then the \( i \)th eigenvector of Eq. (3.1) is the \( j \)th eigenvector of PCA if \( \frac{\lambda_{B_j}}{\lambda_{X_j}} \) is the \( i \)th largest among \( \left\{ \frac{\lambda_{B_1}}{\lambda_{X_1}}, \ldots, \frac{\lambda_{B_{H-1}}}{\lambda_{X_{H-1}}} \right\} \).

This is a generalized case of Proposition 11, because there, we require that \( \operatorname{rank}(\Sigma_B) = rank(\Sigma_X) = p \), which only happens when the number of subclasses is larger or equal to the dimensionality of the original feature space. While here, only \( rank(\Sigma_X) = p \) is required.

Proposition 12 tells us that the ratio of the eigenvalues determines the solution of Eq. (3.1). However, the objective of Eq. (3.1) is to maximize \( \frac{v^T \Sigma_B v}{v^T \Sigma_X v} \). Do they contradict to one other? The answer is no, because Theorem 3 tells us that \( \sum_{i=1}^{H-1} \lambda_i = \sum_{i=1}^{H-1} \frac{v_i^T \Sigma_B v_i}{v_i^T \Sigma_X v_i} = \sum_{i=1}^{H-1} \sum_{j=1}^{p} \frac{\lambda_{B_j}}{\lambda_{X_j}} (u_j^T w_i)^2 \). It is noted that when there are \( H - 1 \) eigenvectors of \( \Sigma_X \) in the same direction with those of \( \Sigma_B \), \( tr(\Sigma_X^{-1} \Sigma_B) = \sum_{i=1}^{H-1} \frac{\lambda_{B_i}}{\lambda_{X_i}} \), which is equivalent to Proposition 12.
Summarizing the above discussion, in the Fisher-Rao’s criterion, we want to simultaneously maximize the measure given by the scatter matrix $S_1$ (which in our case is $\Sigma_B$) and minimize that computed by $S_2$ (in our case, $S_2 = \Sigma_X$). When the $i^{th}$ eigenvector of $\Sigma_B$ and the $j^{th}$ eigenvector of $\Sigma_X$ (for any $j \leq i$) are the same, the result obtained using Eq. (3.1) will depend on the ratio between the eigenvalues of $\Sigma_B$ and $\Sigma_X$. In this case, there is no way of knowing which option is best for classification that give by $\Sigma_B$ or that given by $\Sigma_X$.

This problem is illustrated in Fig. 3.3. In this example, $u_i$ ($w_i$ resp.) is the eigenvector of $\Sigma_X$ ($\Sigma_B$ resp.) associated to the $i^{th}$ largest eigenvalue. In Fig. 3.3(a), we show an example where Eq. (3.1) fails. This is so, because the first eigenvector of $\Sigma_B$ and $\Sigma_X$ are the same (i.e., $w_1 = u_1$) and therefore, when we use Eq. (3.1) to reduce the dimensionality of our feature space to one, we cannot select a vector that has large variance according to $\Sigma_B$ and small variance according to $\Sigma_X$. As we can see in the figure, $\Sigma_B$ would like to select $w_1$ as a solution, whereas $\Sigma_X$ would discourage the use of $w_1$. In our example, $\Sigma_X$ has larger variance along $w_1$ than $\Sigma_B$ does and, therefore, Eq. (3.1) will select the vector orthogonal to $u_1$ (i.e., $u_2$) as the solution. This is obviously an undesirable solution.

We refer to this problem as the existence of a conflict between $\Sigma_X$ and $\Sigma_B$. In Fig. 3.3(b), there is no conflict ($w_1 = u_2$, i.e., $v_1$ simultaneously maximizes $\Sigma_B$ and minimizes $\Sigma_X$), $v_1$ is thus the optimal solution.

Moreover, it is important to note that the results of Eq. (3.1) may still be correct when a conflict exists. This is illustrated in Fig. 3.4. In this figure, we apply LDA to two three-class examples. The first is the same as that shown in Fig. 3.3(a), which did not produce the correct results. For comparison, we now show a second example in Fig. 3.4(b) where we also have $u_1 = w_1$ (same as above). Yet, the result obtained by Eq. (3.1) in this second example is correct. This is the reason why we refer to the cases with conflict as examples.
where Eq. (3.1) is not stable, because, in such cases, it is not know a priori whether the results given by Eq. (3.1) will be correct or not. Ideally, we would only want to use Eq. (3.1) where we know it to be stable. But, in practise, we may decide to proceed with caution where the algorithm is know to be unstable.

The results presented above have the following interpretation. When the agreeing pair of vectors \((u_j, w_i)\) correspond to a \(u_j\) with associated small eigenvalue and a \(w_i\) with associated large variance, the result of Eq. (3.1) is optimal according to Bayes. However, when the \((u_j, w_i)\) pair agreeing correspond to a \(w_i\) with smaller eigenvalue than that of \(u_j\), the results of Eq. (3.1) are not guaranteed to be optimal because we cannot maximize \(|v^T \Sigma_B v|\) and minimize \(|v^T \Sigma_X v|\) simultaneously.

Fortunately, this problem can be easily detected because when this happens the angle between the first eigenvectors of \(\Sigma_B\) and \(\Sigma_X\) is small (e.g., in Fig. 3.3(a) it is zero). This can be formally computed as,

\[
K = \sum_{i=1}^{m} \sum_{j=1}^{i} (u_j^T w_i)^2 ,
\]

where \(m < \text{rank}(\Sigma_B)\). As argued above and as shown in Fig. 3.3, one wants the value of \(K\) defined above to be as small as possible. For example, in Fig. 3.3(b) we illustrate a case where the cosine of the angle between \(w_1\) and \(u_1\) is zero and, therefore, the result \(v_1\) is correct. This is so, because now the first eigenvector of \(\Sigma_B\) and the last of \(\Sigma_X\) agree; i.e., \(w_1 = u_2\). 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. (3.10).

To efficiently use this criterion, we will need to compute the value of Eq. (3.10) for each of the possible values of \(H\) and then select the minimum. This means we will first
Figure 3.3: (a) In this example, the first basis vector obtained with Eq. (3.1) does not minimize the classification error. Note that $u_1$ (which is orthogonal to $v_1$ would be the optimal and desirable solution). Close analysis of this figure shows that the angle between the first eigenvector of $\Sigma_B$ and $\Sigma_X$ is zero. This means that it is not possible to maximize the metric given by $\Sigma_B$ and minimize that of $\Sigma_X$ simultaneously. (b) A classical example where most linear equations perform as expected. Note that in this case the angle between the first eigenvectors of $\Sigma_B$ and $\Sigma_X$ is large (i.e., $90^\circ$). This is formally computed by Eq. (3.10).

Figure 3.4: In (a) the results of Eq. (3.1) are incorrect. In (b) the results are correct.
calculate

\[ K_H = \sum_{i=1}^{m} \sum_{j=1}^{i} (u_j^T w_{H,i})^2, \]  

(3.11)

where \( w_{H,i} \) is the \( i \)th eigenvector of \( \Sigma_B \) 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. (3.11).

It is important to note that the larger the value of \( H \), the larger the number of eigenvectors in \( \Sigma_B \) (or, equivalently, the larger \( m \) can be). Unfortunately, when this happens, the number of summations in Eq. (3.11) increases. This means that \( K_H \) will generally grow with \( m \). This calls for a normalization step.

From elementary geometry we know that

\[ \sum_{j=1}^{p} (u_j^T w_{H,i})^2 = (\cos^2 \theta_{i,1} + \cos^2 \theta_{i,2} + \cdots + \cos^2 \theta_{i,p}) = 1, \]

where \( \theta_{i,j} \) is the angle between \( u_j \) and \( w_{H,i} \) and \( p = rank(\Sigma_X) \). In other words, the inner summation in (3.11) is always guaranteed to be bounded by zero and one, \( \sum_{j=1}^{i} (u_j^T w_{H,i})^2 \in [0, 1] \).

The above argument shows that Eq. (3.11) depends on the value of \( m \). The larger \( m \) is, the larger Eq. (3.11) can be. To prevent our algorithm to be biased toward small values of \( H \), we need to normalize Eq. (3.11) with \( 1/m \). Our optimal number of classes is therefore given by

\[ H_o = \arg \min_H \frac{1}{m} K_H. \]  

(3.12)

This second criterion is summarized in Algorithm 2.

We now check the complexity of Algorithm 1 and Algorithm 2. The complexity of an algorithm is the summation of the complexity of each step. Therefore, the complexity of Algorithm 1 covers the calculation on \( n\{NN Clustering + \Sigma_X + t[\Sigma_B + (\Sigma_X^{-1}\Sigma_B V = V \Lambda) + \).
\( V_q^T X_i + \text{NN Classification} \}, \) which is equal to \( n \{ n^2 O(p) + nO(p^2) + t[H^2O(p^2) + O(p^3) + nHO(p) + nO(p)] \}. \) The dominant terms are \( n^3 O(p) + tnO(p^3). \) Hence, the complexity of Algorithm 1 is \( n^3 O(p) + \) \(tnO(p^3). \) Similarly, the complexity of Algorithm 2 covers the calculation on \( \{ \text{NN clustering} + \Sigma X + (\Sigma X U = U \Lambda_X) + l[\Sigma B + (\Sigma B W = W \Lambda_B) + k_H] \}, \) which is equal to \( n^2 O(p) + nO(p^2) + O(p^3) + l[H^2O(p^2) + O(p^3) + \frac{H^2}{2}O(p)] \}. \) The complexity of Algorithm 2 is approximated by the dominant term \( n^2 O(p) + lO(p^3) \) (recall that \( n \) is the number of sample and \( p \) is the dimensionality of the range of \( \Sigma X \)). 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\(_{stability}\)**

1. Sort the training samples using the NN clustering defined in Algorithm 3.
2. Calculate \( \Sigma X \).
3. Calculate \( \Sigma X U = U \Lambda_X \).
4. for \( H = C \) to \( l \cdot C \) \( \dagger \) do
   1. Calculate \( \Sigma B \).
   2. Compute \( \Sigma B W = W \Lambda_B \).
   3. Calculate \( K_H \) using Eq. (3.11).
5. end for
6. \( H_o = \arg\min_k K_H \)
7. Calculate \( \Sigma B \) with \( H = H_o \).
8. The final projection matrix \( V_q^* \) is given by the first \( q \) columns of \( V \), where \( \Sigma X^{-1} \Sigma B V = V \Lambda_X \).

\( \dagger \) 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.
3.4.3 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 [16]. 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 [25]. In this case,
Figure 3.5: (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.

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. 1st) 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. 2nd) The algorithm of Koontz and Fukunaga is iterative in nature and, therefore, associated with a large computational cost and is not guaranteed to converge.

**NN clustering**

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

We can now readily use this sorted set \( \{ x_{i1}, \ldots, x_{in_i} \} \) 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 \( \{ x_{i1}, \ldots, x_{i[n_i/2]} \} \) and \( \{ x_{i[(n_i/2)+1]}, \ldots, x_{in_i} \} \). 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. 3.5 shows an example. In Fig. 3.5(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. 3.5(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 \( \Sigma_B \) in Section 2. Our definition ensures that between sub-class scatter is only used when needed.

### 3.5 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.
Algorithm 3 NN clustering

for $i = 1$ to $C$ do
  Let $\hat{X}_i = \{\hat{x}_{i1}, \ldots, \hat{x}_{in_i}\}$ be the samples in class $i$.
  Construct the matrix $D = \{d_{ij}\}$ 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 $[s, b] = \arg\max_{j,k} d_{jk}$.

for $g = 1$ to $\lfloor n_i/2 \rfloor$ † do
  Let $x_{ig+1} = \hat{x}_{im}$, where $m = \arg\min_{j \neq s} d_{sj}$.
  $d_{sm} = \infty$.
  Let $x_{i-g} = \hat{x}_{ik}$, where $k = \arg\min_{j \neq b} d_{bj}$.
  $d_{bk} = \infty$.
end for

Divide the resulting sorted set $\{x_{i1}, \ldots, x_{in_i}\}$ 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 $x_{i,\lfloor n_i/2\rfloor+1}$.

available at [5]. Moreover, we will show how our algorithm applies to two particular prob-
lems in computer vision: a) categorization of images of objects, and b) face recognition.

We will compare our results to LDA [61, 22], Direct LDA (DLDA) [76], Nonparametric
DA (NDA) [25], Heteroscedastic LDA (HLDA) [44], Kernel LDA (K-LDA) [3, 74] and
Complete Kernel Fisher Discriminant analysis (CKFD) [73]. In K-LDA, we have used a
second degree polynomial kernel, $k(x, y) = (x, y)^2$, which we will refer to as the KP-LDA
method, and a Gaussian kernel, $k(x, y) = \exp\left(\frac{\|x-y\|^2}{\sigma}\right)$, that we named KG-LDA. Note
that in KG-LDA we need to specify the value of $\sigma$. This value will be optimized for each of
the datasets by means of the cross-validation procedure. In contrast, the polynomial kernel
used in CKFD is $k(x, y) = (x^T y + 1)^a$ 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 $\min\{C + 40, p - 1\}$.

3.5.1 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 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 3.1. 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.
<table>
<thead>
<tr>
<th>Database \ Method</th>
<th>LDA</th>
<th>NDA</th>
<th>HLDA</th>
<th>DLDA</th>
<th>KP-LDA</th>
<th>KG-LDA</th>
<th>P-CKFD</th>
<th>G-CKFD</th>
<th>$SDA_{stability}$</th>
<th>$SDA_{loot}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDBC</td>
<td>0.94</td>
<td>0.73 (0.08)</td>
<td>0.95 (0.009)</td>
<td>0.87</td>
<td>0.80</td>
<td>0.95</td>
<td>0.94</td>
<td>0.90</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>LSD</td>
<td>0.84</td>
<td>0.48 (0.11)</td>
<td>0.88 (0.011)</td>
<td>0.86</td>
<td>0.85</td>
<td>0.91</td>
<td>0.85</td>
<td>0.88</td>
<td>0.88</td>
<td>0.87</td>
</tr>
<tr>
<td>MDD-pix</td>
<td>0.93</td>
<td>0.84 (0.13)</td>
<td>0.82 (0.064)</td>
<td>0.95</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>MDD-fou</td>
<td>0.81</td>
<td>0.70 (0.10)</td>
<td>0.83 (0.008)</td>
<td>0.80</td>
<td>0.82</td>
<td>0.85</td>
<td>0.85</td>
<td>0.77</td>
<td>0.83</td>
<td>0.81</td>
</tr>
<tr>
<td>MDD-fac</td>
<td>0.97</td>
<td>0.79 (0.20)</td>
<td>0.96 (0.006)</td>
<td>0.91</td>
<td>0.96</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>MDD-kar</td>
<td>0.96</td>
<td>0.90 (0.08)</td>
<td>0.97 (0.006)</td>
<td>0.96</td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>MDD-zer</td>
<td>0.79</td>
<td>0.69 (0.08)</td>
<td>0.79 (0.004)</td>
<td>0.79</td>
<td>0.81</td>
<td>0.83</td>
<td>0.82</td>
<td>0.83</td>
<td>0.79</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Table 3.1: Shown here are the classification results (in percentage) on the UCI datasets 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 parentheses correspond to the standard deviation from such means.
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 Table 3.1.

The final test shown in Table 3.1 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 3.1, 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,
Table 3.2: Number of subclass divisions per class obtained by each of the two implementations of SDA; i.e., $H_o/C$.

and a fusion coefficient which task is to combine the results obtained in the range and null spaces of $S_W$. 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 3.2 we show the number of 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 3.1. 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.

### 3.5.2 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 [40]. Examples of these are shown in Fig. 3.6(a). Each category includes the images of ten objects (e.g., ten different cows – as shown Fig. 3.6(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. 3.7(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. 3.7(c). In our experiments we obtain a total of 300 distances, which generates a feature space of 300 dimensions.

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. 3.8(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 $SDA_{foot}$ 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. 3.8(b). In this case, the average value of $H_o$ was 80 when we used $SDA_{stability}$ and 27 when using $SDA_{foot}$.

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
Figure 3.8: (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.
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.

### 3.5.3 Face recognition

In our last experiment, we used the AR-face database [48]. 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. 3.9(a)) of 100 randomly selected individuals for training; i.e., 1,300 training images. The 13 images of the second session (shown in Fig. 3.9(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. 3.8(c). In this test both methods resulted in $H_o = 100$.

### 3.5.4 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 3.4.3) 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 $SDA_{K\text{-means}}$ and $SDA_{valley}$. The classification accuracy for each of these implementations of SDA, are shown in Table 3.5.4. 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 3.5.4 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. (3.12). The valley-seeking algorithm works differently. For a fixed window size, the value of $H_0$ 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 3.5.4. We see that this new algorithm results in similar classification rates 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 [69, 77].

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 can not 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., $H_o = 100$. 

74
<table>
<thead>
<tr>
<th>Method</th>
<th>ARface</th>
<th>ETH-80 (feature)</th>
<th>ETH-80 (appearance)</th>
<th>WDBC</th>
<th>LSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SDA_{NN}$</td>
<td>0.782</td>
<td>0.749</td>
<td>0.731</td>
<td>0.944</td>
<td>0.881</td>
</tr>
<tr>
<td>$SDA_{K\text{-means}}$</td>
<td>0.776 (0)</td>
<td>0.754 (0.030)</td>
<td>0.789 (0.071)</td>
<td>0.942 (0.007)</td>
<td>0.884 (0.007)</td>
</tr>
<tr>
<td>$SDA_{valley}$</td>
<td>0.666 (0.150)</td>
<td>0.753 (0.005)</td>
<td>0.701 (0.025)</td>
<td>0.956 (0.005)</td>
<td>0.876 (0.004)</td>
</tr>
<tr>
<td>Method</td>
<td>MDD-pix</td>
<td>MDD-fou</td>
<td>MDD-fac</td>
<td>MDD-kar</td>
<td>KDD-zer</td>
</tr>
<tr>
<td>$SDA_{NN}$</td>
<td>0.957</td>
<td>0.834</td>
<td>0.957</td>
<td>0.966</td>
<td>0.793</td>
</tr>
<tr>
<td>$SDA_{K\text{-means}}$</td>
<td>0.956 (0.004)</td>
<td>0.828 (0.007)</td>
<td>0.935 (0.029)</td>
<td>0.965 (0.006)</td>
<td>0.7947 (0.005)</td>
</tr>
<tr>
<td>$SDA_{valley}$</td>
<td>0.936 (0.011)</td>
<td>0.811 (0.011)</td>
<td>0.971 (0.006)</td>
<td>0.962 (0.005)</td>
<td>0.799 (0.009)</td>
</tr>
</tbody>
</table>

Table 3.3: Classification results when clustering the data in each class with the K-means and the valley-seeking.
### 3.6 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 3.2, we have shown how such subclass division 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. (3.5).

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 3.4, 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

<table>
<thead>
<tr>
<th>Method</th>
<th>ARface</th>
<th>ETH-80 (feature)</th>
<th>ETH-80 (appearance)</th>
<th>WDBC</th>
<th>LSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SDA_{VS}$</td>
<td>0.757</td>
<td>0.759</td>
<td>0.733</td>
<td>0.947</td>
<td>0.88</td>
</tr>
<tr>
<td>Method</td>
<td>MDD-pix</td>
<td>MDD-fou</td>
<td>MDD-fac</td>
<td>MDD-kar</td>
<td>KDD-zer</td>
</tr>
<tr>
<td>$SDA_{VS}$</td>
<td>0.927</td>
<td>0.82</td>
<td>0.968</td>
<td>0.959</td>
<td>0.793</td>
</tr>
</tbody>
</table>

Table 3.4: Successful classification rates as given by the VS (Valley-Stability) implementation of SDA.
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.
CHAPTER 4

TUMOR CLASSIFICATION BASED ON GENE EXPRESSION DATA

4.1 Introduction

Precise classification of tumor is essential for the successful prediction and treatment of cancer. Traditional diagnosis of cancer relies on microscope histology and tumor morphology. Therefore, it is hard to separate those cancers that have similar histopathologic features while varying in clinical course. The emergence of modern experimental technology, such as DNA microarray, facilitates research in cancer classification. DNA microarray offers scientist the ability to monitor the expression patterns of thousands of genes simultaneously, allowing them to study how these genes function and how they act under different conditions, leading to a more complete understanding of the molecular variation, in addition to morphologic variations among tumors. A large number of studies have used microarrays to analyze the gene expression for breast cancer, leukemia and colon tissue, demonstrating the potential power of microarray in tumor classification [27, 58, 1, 59, 62].

Gene microarrays are characterized by huge numbers of features (genes) and a limited number of samples. This leads to a common problem in pattern recognition, called overfitting. Most researchers use “marker genes” to reduce the number of features and avoid
the problem of over-fitting. But how to correctly select these marker genes is still an open
problem. In this chapter, we propose a new classification strategy, which can relieve the
over-fitting problem by taking advantage of the information embedded in the testing sam-
ple. We compare this approach with several well-known discriminant methods for tumor
classification on a variety of datasets. The experimental results show that our solution is
always the most adequate in making predictions on new samples.

4.2 Biological Background

From ancient times, it was suspected that there existed some sort of hereditary mecha-
nism that carried information from parents to child. Family members tend to exhibit similar
characteristics. Following centuries of research, the existence of a hereditary unit has been
firmly established, which we now called gene. Each gene, either by itself or in combination
with some other genes, provides a clear and unambiguous set of instruction for producing
some properties of its organism. Chemically, a gene is a specific subsequence of DNA.
A deoxyribonucleic acid (DNA) molecule has the well-known double-helix structure – the
two strands are connected by the nitrogen bases according to the base pairing rule (A pairs
with T, and C pairs with G, where A, T, C, and G, represent adenine, thymine, cytosine,
and guanine, respectively). The A-T-G-C sequence is the code required for constructing
proteins – the building block of human basic function. Therefore, knowing the sequence
and the meaning of the sequence of genes is important to understand life. Genomics is the
study developed for this research. Structural genomics aims to find the genome sequences
of each organism. However, knowing the sequence of a gene does not mean its function
is also known. The cells in an organism have almost the same genes, but not all cells ex-
press the same set of genes. Therefore, different cells have different functions. Even within
the same cell, different genes are expressed under different conditions. This is the concern of functional genomics. The huge progress made in structural genomics has promoted the study of functional genomics, and, microarrays are an extremely useful technique for studying functional genomics. It allows us to monitor the expressions of thousands of genes simultaneously – which genes are expressed, which are depressed, leading to a more complete understanding of molecular variations under different conditions. Fig. 4.1 shows an example of a microarray. Each column represents a sample and each row a gene. It is observed from this example that the gene expression of normal tissues are different from that of abnormal ones. Therefore, microarrays have been extensively applied in cancer research to study the molecular variations among tumors. This will lead to a more accurate classification of tumors, and help prevent and treat cancer more successfully.
4.3 Literature Review

There is a large number of data-sets available for cancer classification based on gene expression data. These data present multiple challenges, including a large number of gene expression values per experiment (several thousands or tens of thousands), and a relatively small number of samples (a few dozen). A well-known problem faced in classification and machine learning is called over-fitting, which arises when the number of features is large and the number of samples is small. In such a situation, one can easily find a decision boundary which separates the training samples perfectly while performing poorly on independent testing samples. This is a typical problem in microarray data because of the above mentioned property.

There are generally two strategies to avoid over-fitting. The first one is to reduce the dimensionality of the original space by choosing a subset of genes – so called “marker genes” – to represent the sample. A second solution is to increase the generalizing ability of the classifier in the original space by maximizing the decision margin, such as Support Vector Machines.

Gene selection algorithms search for a subset of genes that can discriminate tumor tissue from normal tissue. Those genes may have explicitly biological meaning or implication in the molecular mechanism of the tumorigenesis [28, 72]. The objective of gene selection is to increase the classification accuracy, decrease the computation cost of the classifier and clarify the biological interpretation of cancers. In the following, several widely used gene selection strategies are discussed.
Golub et al. [27] sort all genes according to their correlation to the class distinction. They define a measure of correlation as
\[ P(g) = \frac{\mu_1(g) - \mu_2(g)}{\sigma_1(g) + \sigma_2(g)} \]
where \( \{\mu_1(g), \sigma_1(g)\} \) and \( \{\mu_2(g), \sigma_2(g)\} \) denote the mean and standard deviation (SD) of the log of the expression level of gene \( g \). \( P \) reflects the difference between the classes relative to the SD within the class. Large values of \( |P| \) indicates high correlation between the gene expression and class distinction. The first several genes with highest correlation values are chosen for classification. In addition to the value of \( P \), one can also use \( t \)-statistics or other similar measurement such as Pearson’s correlation coefficient. Dudoit et al. [19] perform a preliminary selection of genes on the basis of the ratio of their between-group to within-groups sum of squares. For gene \( g \), the ratio is
\[
\frac{BSS(g)}{WSS(g)} = \frac{\sum_i \sum_k I(y_i = k)(\bar{x}_{kg} - \bar{x}_g)^2}{\sum_i \sum_k I(y_i = k)(x_{ig} - \bar{x}_kg)^2},
\]
where \( \bar{x}_g \) denotes the average expression level of gene \( g \) across all samples, \( \bar{x}_{kg} \) denotes the average expression level of gene \( g \) across samples belonging to class \( k \), \( x_{ig} \) denotes the expression level of gene \( g \) in the \( i^{th} \) sample, \( y_i \) denotes the class label for the \( i^{th} \) sample, and \( I(\cdot) \) is the indicator function, which is 1 if the condition in the parentheses is true.

All those methods, select genes that individually classify the training data best. They can eliminate genes that are useless for classification, but they could not generate compact gene sets because some genes are highly correlated. Another concern is that the genes that do not separate classes individually may improve the separability when combined with other genes.

In [72], Xiong et al. discuss two gene selection methods – stepwise selection and Monte Carlo. In the forward stepwise selection, they pick the two genes with highest classification

\[ A \text{ class distinction is represented by an expression pattern } G = (g_1, g_2, \cdots, g_n), \text{ where } g_i = 1 \text{ or } 0 \text{ according to whether the } i^{th} \text{ sample belongs to class 1 or class 2} \]
rate on the training data from all possible pairs of genes. Those two genes compose of the initial optimal subset of genes. The classification is implemented on each of the remaining genes together with the optimal set. The gene with the highest classification accuracy is then added to the optimal set. This procedure continues until the predetermined classification accuracy or a fixed number of $K$ genes is reached. In the Monte Carlo method, a subset of $k$ genes is randomly selected $n$ times ($k$ starts from 1). The subset with highest classification out of the $n$ subsets is chosen as the solution. $k$ is increased and this procedure is repeated until a fixed value of $K$ or the predetermined accuracy is reached. Those two methods select genes as a subset and therefore take the correlation between genes into account. However, these two methods are not exhaustive, the selected subset of genes are not necessarily optimal.

Guyon et al. [28] propose a new method of gene selection utilizing SVM based on Recursive Feature Elimination (RFE). They train a support vector machine using all the training data and get the decision function $C(x) = w \cdot x + b$, where the weight vector $w = \sum_i \alpha_i y_i x_i$ is a linear combination of the training patterns. The training patterns with non-zero weights $\alpha_i$ are support vectors. Those with weights satisfying the strict inequality $0 < \alpha_i < C$ (where $C$ is the soft margin parameter) are the margin support vectors. The bias value $b = \sum_{0<\alpha_i<C} (y_i - w \cdot x_i)$. SVM-RFE uses the weight magnitude $w$ as a ranking criterion, and eliminates those genes with small weight magnitude. However, how small these values need to be is hard to determined.

The study on cancer classification using microarray data generally uses two different strategies: unsupervised (or clustering) and supervised. Unsupervised learning finds the groupings using only the input feature variables. The objective of clustering is to group the samples with similar properties. The clustering algorithms that are widely used on the
analysis of microarray data are hierarchical clustering [20, 1], gene shave [30], k-mean clustering [67], decision tree [41] and self-organizing maps (SOM) [64, 27]. Moler et al. [55] uses a combination of unsupervised and supervised methods, where the unsupervised learning method is used to group the input vectors and the estimated classes are compared to those that would be expected based on the output label. The discrepancy provides the indication of the quality of the data used. The estimated class may correspond to sub-categories of profile with the same label or could suggest mislabelled examples. Because unsupervised algorithms do not use the information contained in the class label, they will not be considered in this dissertation.

In supervised algorithms, we are given the training data that contains vectors of gene expression as well as the corresponding class label. We use a \( p \)-by-\( n \) matrix \( X \) to represent the training data, where \( p \) is the number of genes (features) and \( n \) is the number of experiments (samples), \( X = (x_1, \ldots, x_n), x_i = (x_{i1}, \ldots, x_{ip})^T \), and \( Y = (y_1, \ldots, y_n) \) represent the class label, where \( y_i = +1 \) or \( -1 \) depends on whether \( x_i \) is a tumor tissue or not. Let \( \bar{x}_{gk} \) be the mean expression value of gene \( g \) in class \( k \) and \( \bar{x}_g \) the mean expression value of gene \( g \) across all samples. Diverse approaches are used to classify tumor tissue on the basis of gene expression data. The following are some example.

Tibshirani et al. [68] propose a method for class prediction in DNA microarray study based on an enhancement of a nearest prototype classifier. For each class \( k \), they calculate the \( t \)-statistics for gene \( g \) as
\[
d_{gk} = \frac{\bar{x}_{gk} - \bar{x}_g}{m_k \cdot s_g} ,
\]
where \( s_g^2 \) is the pooled within-class standard deviation (SD) for gene \( g \) and \( m_k = \sqrt{1/n_k - 1/n} \).

The shrunken centroid for class \( k \) is \( \bar{x}'_{gk} = \bar{x}_g + m_k s_g d_{gk}' \), where \( d_{gk}' = \text{sign}(d_{gk})(|d_{gk} - \Delta|)_+ \) is the shrunken statistic \((z_+ = z \text{ if } z > 0, \text{ and } z_+ = 0 \text{ otherwise})\). Thus, all \( d_{gk} \)
that are less than $\Delta$ in absolute value are shrunken to zeros (inactive genes) and the rest are shrunken to somewhere between zero and their original values (active genes). Then, a nearest centroid classifier based on Linear Discriminant Analysis for classification of new sample $x = (x_1, \cdots, x_p)^T$ is given by

$$C_k(x) = \sum_{g=1}^{p} \frac{(x_g - \bar{x}'_{gk})^2}{s^2_g} - 2 \cdot \log \pi_k.$$  

The shrinkage factor is used to denoise the data, and stabilize the statistics used. However, Dabney [13] points out that this shrinkage procedure makes the class centroids look more similar to each other, it is unclear why this should be expected to make it easier to distinguish the classes from each other. Nonetheless, shrinkage has been shown to produce more accurate mean estimate in noisy data [17]. They then formulate the gene selection process without shrinkage as $d'_{gk} = d_{gk} I(|d_{gk}| > \Delta_k)$, with a class-specific $\Delta_k$ instead. In both methods, the parameter $\Delta$ is allowed to vary over a wide range, and for each $\Delta$, a classifier is built and its error is estimated using cross-validation.

Geman et al. [26, 65] present the top-scoring pairs classifier (TSP) for molecular classification based on Microarray comparison. In this approach, they focus on detecting “marker” gene pairs $(i; j)$ for which there is a significant difference in the probability of $x_i < x_j$ from class 1 to class 2. Classification is then based on this collection of distinguished pairs. The quantity of interest is $p_{ij} = P(x_i < x_j|c)$, $c = 1, 2$. Let $\Delta_{ij} = |p_{ij}(1) - p_{ij}(2)|$ be the score for the gene pair $\{i; j\}$. From all possible combinations of $i \neq j$, choose the ones with top scores for classification. That is, if we observe $x_i < x_j$, then pair $(i; j)$ votes for class 1 if $p_{ij}(1) \leq p_{ij}(2)$, otherwise it votes for class 2.

Golub et al. [27] propose a weighted-voting (WV) scheme for binary classification. Each gene casts a weighted vote for one of the classes $v(g) = P(g)(x(g) - b(g))$, where the correlation ($P(g)$) between its expression level and the class distinction serves as the
weight and \( b(g) = (\mu_1(g) + \mu_2(g))/2 \). All the votes are combined to determine the winning class. This method turns out to be a minor variant of the sample maximum likelihood rule when all genes are all uncorrelated.

In addition, some well-known classifiers, such as Fisher-Rao’s Linear Discriminant Analysis [72], nearest-centered prediction rule [13, 68] and support vector machines [55, 70, 7] are widely used for tumor classification.

The ability to successfully distinguish tumors from normal tissue using gene expression data shows its potential to improve cancer diagnosis. To date, most published papers, apply a single approach to a single gene expression data set. It is impossible to evaluate different approaches without a comprehensive comparison. Dudoit and colleagues [19] compare the performance of different discriminant methods for the classification of tumor based on gene expression profiles. They conclude that simpler classifiers perform remarkably well compared with more sophisticated ones. They also point out that the selection of “marker” genes is still an important issue to be addressed. One should be aware of the statistical versus biological significance too. Results of purely statistical approach may be opposed to the biological interests.

Because the number of samples in gene expression data is usually small, cross-validation is extensively used to evaluate the performance of classification and gene selection. The genes most differently expressed between classes are selected and used to construct the classifier using training data and then the proportion of misclassification is estimated with these genes on an independent validation set. In [53], Michiels and colleagues expand this unique-training-and-validation-set idea using multiple random validation strategy to assess the extend to which the selected genes depend on the constitution of training samples and to
study the distribution of misclassification rate across validation. They show that the proportion of misclassification could be improved with large training-set size. Their results also show that the selection of genes strongly depends on the training sets. Every training set leads to a different subset of “marker” genes. They, therefore, suggest the use of validation by repeated random sampling.

In most cases, methods (gene selection and classification) that ignore the correlation and interaction among genes, have performed equally or better than those that corresponded more closely to the biological complexity of the data [19, 54]. This phenomenon does not, however imply that gene correlation and interaction do not exist. This is so because the data-sets studied do not contain sufficient information to provide accurate estimate of the underlying biological complexity of the system. In this circumstances, complex models often model random noise instead of the true biology inferences [54].

### 4.4 Problem Statement

A microarray of $p$ genes for $n$ samples is represented by an $p \times n$ matrix, where $x_{ji}$ is the expression level of gene $j$ in sample $i$. In microarray-based tumor classification, the classifier labels the new sample with tumor or normal tissue by monitoring its gene expression level.

In general, two possible situations with respect to $x$ and its class label may occur: 1) We may have complete statistical knowledge of the distribution of observation $x$ and its category. In this case, a standard Bayes classifier yields an optimal decision procedure. 2) We may have no knowledge of the distribution of observation $x$ and category aside from that provided by a set of pre-classified samples. In this case, a decision to classify $x$ will depend only on a collection of correctly classified samples. For the tumor classification
based on microarray, we are in the second situation, i.e., the classification has to be based on the pre-classified samples – training data. Although the knowledge of distribution is unknown in this situation, we can estimate the distribution parameters from the training data and construct the classifier using these estimated parameters. Algorithms using this strategy are called parametric method. Examples are: maximum likelihood and fisher discriminant analysis. As a comparison, algorithms without any distribution assumption are called non-parametric, e.g., nearest neighbor rule and support vector machines.

A microarray data-set is characterized by its large number of genes (thousands or tens of thousands) and limited number samples available (usually dozens). This implies a serious over-fitting problem – i.e., one can easily find a perfect decision boundary for the training data but perform poorly on the independent testing vectors. To visualize this problem, in Fig. 4.2, we show an example with an application to breast cancer classification. In this example, we have 22 samples. The first 7 are from tumor tissue and the remaining 15 are from normal tissue. We use the leave-one-out cross validation test to show how it works. In the figure, the $x$-axis represents the index of the sample left out. For each test, we use LDA on the training data, and obtain a one-dimensional projection vector. We then apply this projection to all training data and the testing sample (the one left out). It is noticed that all training samples belonging to normal tissue are projected to one single point (square), while the samples belonging to cancer tissue are projected to another single point (star). The dotted line represents the average of the two classes. However, the testing samples are projected somewhere in the middle, away from the training data. Therefore, due to the over-fitting problem, classifiers built solely on training data can separate training samples perfectly but are unable to correctly classify new samples.
Figure 4.2: Plotted here is an illustration of the over-fitting problem. The $x$-axis represents the experiment number. In each experiment, the training samples are perfectly separated (samples from class 1 and classes 2 are projected onto two distinct points – shown as ⋆ and □), while the testing sample generally lies in the middle and cannot be classified correctly.

To avoid this problem, researches usually search for the “marker genes” that will improve classification. In Chapter 4.3, we have introduced several well-known gene selection methods described in the literature. It is impossible to claim the superiority of a specific algorithms over all others. The best technique in one data-set may be inferior in a different set. In addition, the number of “marker genes” is generally determined by cross-validation. It is really not known why such a number of genes is sufficient. It has been emphasized by Ntzani [56] that the success of molecular classification also depends on the number of genes selected. Although in rare cases, exploring restricted categories of genes may be justified (reflecting specific cell function), molecular function is correlated to diverse gene categories. Therefore, without a well-defined reason to limit the number of genes, we are left to use a large numbers of genes to maximize the information generated. For this reason, in this chapter, we only focus on those classification algorithms that can deal with large
number of features. We will introduce these algorithms in Section 4.5. However, with such a large number of features, these methods cannot avoid the over-fitting problem. We thus propose a new classification algorithm in Chapter 4.6 which not only can be applied to a large number of genes but also aims to relieve the influence of over-fitting.

4.5 Current Algorithms

In this section, we briefly introduce some well-known algorithms that we are going to use for comparison when we show the superiority of our approach.

4.5.1 $k$ Nearest Neighbor ($k$NN)

In many instances, it is reasonable to assume that observations which are close to each other in our feature space – under some appropriate metric – belong to the same class. The Nearest Neighbor rule is the simplest non-parametric decision procedure to adapt this form. Specifically, the label of sample $x$ is $k$ if

$$
C(\tilde{x}) = k, \text{ where } \tilde{x} = \arg \min_{x_i} d(x_i, x), \ i = 1, \cdots, n,
$$

where $C(x)$ denotes the class label of $x$, and $d(\cdot, \cdot)$ is a distance measurement. Generally, the Euclidean distance is used and will be the one considered in this chapter. Notice that the NN-rule only uses the nearest neighbor as a classifier, while ignoring the remaining pre-labelled data points. If the number of pre-classified points is large it makes good sense to use the majority vote of the nearest $k$ neighbors. This method is referred to as the $k$-NN rule.

The value of $k$ should be: 1) large to minimize the probability of misclassifying $x$, and 2) small (with respect to the number of samples) so that the points are close enough to $x$. 

90
to give an accurate estimate of its true class. In practice, the number of \( k \) is determined by cross-validation.

### 4.5.2 Maximum Likelihood

Maximum Likelihood (ML) is a parametric method. It assumes the distribution forms \( p_i(x) \) for samples of each class is a prior known (Gaussian distributions being the most common). The parameters of the distribution are estimated by means of the training samples.

ML assigns the sample \( x = (x_1, \cdots, x_p)^T \) to the class which gives the largest likelihood to \( x \), i.e.,

\[
C(x) = \arg \max_i p_i(x).
\]

When samples are Gaussian distributed, i.e., \( x \sim \mathcal{N}(\mu_i, \Sigma_i) \), then

\[
C(x) = \arg \min_i \{ (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) + \log |\Sigma_i| \}.
\]

Since \( \Sigma_i \) is a \( p \)-by-\( p \) matrix, it is singular in all cases. We can simplify the problem by assuming \( \Sigma_i \) is a diagonal matrix, i.e., \( \Sigma_i = \text{diag}(\sigma_{i1}, \cdots, \sigma_{ip}) \). Or, simplify further, assuming every class has an identical covariance matrix \( \Sigma = \frac{1}{C} \sum_{i=1}^{C} \Sigma_i = \text{diag}(\sigma_1, \cdots, \sigma_p) \). In these two special cases,

\[
C(x) = \arg \min_i \sum_{j=1}^{p} \left\{ \frac{(x_j - \mu_{ij})^2}{\sigma_{ij}^2} + \log \sigma_{kj}^2 \right\}, \tag{4.1}
\]

and

\[
C(x) = \arg \min_i \sum_{j=1}^{p} \frac{(x_j - \mu_{ij})^2}{\sigma_j^2}. \tag{4.2}
\]

When each class has the same prior probability, ML is equivalent to maximum a posteriori (MAP).
4.5.3 Weighted Voting

Golub et al. [27] propose a weighted-voting (WV) scheme for binary classification. Each gene casts a weighted vote for one of the classes, and the final decision is made based on the summation of all individual votes. Specifically, for each gene $g$, they define a correlation value $P(g)$ measuring the correlation between its expression level and the class distinction as $P(g) = [\mu_1(g) - \mu_2(g)]/[\sigma_1(g) + \sigma_2(g)]$, where $\{\mu_1(g), \sigma_1(g)\}$ and $\{\mu_2(g), \sigma_2(g)\}$ denote the mean and standard deviation of the log of the expression level of gene $g$. Large values of $|P|$ indicate high correlation between the gene expression and class distinction. Then, the vote of each gene is $v(g) = P(g)(x(g) - b(g))$, where $b(g) = (\mu_1(g) + \mu_2(g))/2$, with a positive value indicating vote for class 1 and negative value indicating vote for class 2. The final decision is given by

$$C(x) = \begin{cases} 
1 & \sum_{g=1}^{p} v(g) > 0, \\
2 & \sum_{g=1}^{p} v(g) < 0.
\end{cases}$$

It is shown by Dudoit [19] that this method is a minor variant of the sample maximum likelihood rule when all genes are all uncorrelated. For the two class problem, ML classifies sample $x$ as 1 if

$$\sum_{j=1}^{p} \frac{(x_j - \mu_{2j})^2}{\sigma_{j}^2} \geq \sum_{j=1}^{p} \frac{(x_j - \mu_{1j})^2}{\sigma_{j}^2},$$

which is equivalent to

$$\sum_{j=1}^{p} \frac{(\mu_{1j} - \mu_{2j})}{\sigma_{j}^2} \left( x_j - \frac{(\mu_{1j} + \mu_{2j})}{2} \right).$$

4.5.4 Fisher Discriminant Analysis

As discussed in previous chapters, Fisher’s LDA is used to reduce the dimensionality such that the between-class variance is maximized while the within-class variance is minimized. Here, again, since $S_B$ and $S_X$ are both singular, we use Eq. (2.3) in Theorem 1 to
make it applicable for microarray data. After the projection vector $v$ is obtained, a nearest class mean classifier can be easily implemented in the low-dimensional space:

$$C(x) = \arg \min_i ((x - \mu_i)^T v)^2.$$

### 4.5.5 Support Vector Machines

Although SVM can handle non-linear decision boundary when one uses the kernel trick, we will limit ourselves to linear SVM because of the nature of the datasets under investigation, which is linearly separable. If the data-set is linearly separable, a linear SVM is a maximum margin classifier [71].

Consider the linear classifier

$$f(x) = \langle w, x \rangle + b = \begin{cases} \geq 1, & \text{for } x \in \text{class 1} \\ \leq -1, & \text{for } x \in \text{class 2}, \end{cases}$$

maximizing the margin is equivalent to solving the following optimization problem

$$\begin{cases} \text{minimize} & J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ \text{subject to} & \sum_{i=1}^{n} y_i \alpha_i = 0, \alpha_i \geq 0, i = 1, \cdots, n. \end{cases}$$

The weight vector $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ is a linear combination of the training patterns. Because the optimal $\alpha$, $w$ and $b$ must satisfy

$$\alpha_i [y_i(\langle w, x_i \rangle + b) - 1] = 0,$$

only for the input $x_i$ on the boundary (i.e., $y_i(\langle w, x_i \rangle + b) - 1 = 0$) the corresponding $\alpha_i$ are non-zero. Therefore, those input samples are called support vectors. The bias $b = \sum_{\alpha_i \neq 0} (y_i - w \cdot x_i)$ is an average over the support vectors.
Microarray data usually has thousands of genes, but generally only dozens of samples. Therefore, we face a serious problem of over-fitting when we try to separate tumor from normal tissue based on microarrays. “Maker Genes” is a solution, but has received criticism because the molecular function is correlated with a diversity of genes, ignoring some of the genes may cause information loss.

Therefore, our objective is to build a classifier using all genes available, and at the same time, relieve the influence of over-fitting.

The reason why classifiers built on training data generally work poorly on testing data is because the distribution of the training data does not represent that of the testing data. In that case, testing samples are just passive objects. People usually feed testing samples to the classifier and wait for the best. It is however possible to make testing samples more active in the whole procedure of classification. In what follows, we will propose a new classification strategy where testing samples are not passive. Instead, we take advantage of the information embedded in the testing sample $x$ and involve it in the construction of the classification rule. Using this strategy, the over-fitting problem is relieved.

Our solution originates from the discriminant power (DP) of linear discriminant analysis defined in Eq. (2.8), where DP measures how well the classes are separated in the subspace spanned by $V$. Here, we want to measure the separability of the data given when a testing sample is classified into different classes. Because it is reasonable to assume that samples belonging to one class are close to each other in the feature space, the best separability of the data when $x$ is considered to belong to a specific class, should imply a higher probability of getting the correct answer. Fig. 4.3 shows a simple procedure that
can be used for classification. The testing sample is classified to the class with maximum discriminant power.

However, when the number of genes (features) $p$ is much larger than the number of samples $n$, the values of $DP_i$ are identical for all classes. This is specified in the following result.

**Result 14.** When $p \geq n - 1$, the discriminant power in the direction found by the Fisher-Rao’s criterion $\mathbf{v} = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T\mathbf{S}_B\mathbf{a}}{\mathbf{a}^T\Sigma_X\mathbf{a}}$ is equal to one.

The prove is in Appendix C.9.

This result is illustrated by the example shown in Fig. 4.4. We synthetically generate $n$ samples in $\mathbb{R}^{10}$ for a total of two classes. For visualization purpose, we only show the data using the three dimensions with largest variance in Fig. 4.4(a). Red circles represent samples from class 1, blue squares for class 2 and the black star is the testing sample $\mathbf{x}$. Then, we project the data onto the direction found by the Fisher-Rao’s criterion as shown in Fig. 4.4(b)-(e). Fig. 4.4(a) and (b) show the distributions of the data when $n = (p + 1) + 4$, i.e., $p < n - 1$. (a) is the distribution when $\mathbf{x}$ is labelled as belonging to class 1 and (b) is the distribution when $\mathbf{x}$ is labelled 2. It is clear from this plot that the data are better separated when $\mathbf{x}$ is assumed to be from class 2. This is also indicated by $DP_1 = 0.329$ and $DP_2 = 0.814$. (d) and (e) show the same projection for $n = p + 1$, i.e., $p \geq n - 1$. It is noticed that all samples from class 1 are projected onto a single point and the samples
from class 2 to another point. In those two cases, \( DP_1(x) = DP_2(x) = 1 \). Therefore, the discriminant power cannot help with the classification. However, since the within-class scatter is zero, the between-class variance itself is the measure of separability. We denote \( d_k(x) \) as the distance between two classes when \( x \) is labelled as class \( k \) and define it as follows

\[
d_k(x) = v^T S_B v, \quad (4.3)
\]

where \( v \) is the direction determined by Fisher-Rao’s criterion when \( x \) belongs to class \( k \).

The larger the distance, the more separated the two classes are. We classify \( x \) to the class with highest value of \( d \), i.e.,

\[
C(x) = \arg \max_k d_k(x).
\]

We denote this new discriminant power based method as DP-based algorithm. This algorithm is summarize in Fig. 4.5.

### 4.7 Experimental Result

In this section, we compare our DP-based algorithm with other classifiers described in Section 4.5. We use a variety of databases comprising four different data-sets that includes gene expression data from different kinds of tumors.
Figure 4.4: (a) Visualizes the data generated in $\mathbb{R}^{10}$ using the 3 dimensions with largest variance. In (b)-(e), we show the distribution of the data when projected onto the direction of $v$ found by Fisher-Rao’s criterion. In (b) and (d), the testing sample is assumed to be drawn from class 1, and in (c) and (e) the testing sample is labelled class 2. When $n > p + 1$, the $DP$ is a good measure of separability. When $n \leq p + 1$, $d$ is the appropriate measure.
4.7.1 Data Sets

Lymphoma

Diffuse large B-cell lymphoma (DLBCL) is the most common lymphoid malignancy in adults, curable in less than 50% of patients. This data-set is constructed to distinguish DLBCL from a related GC B-cell lymphoma, follicular lymphoma (FL) [62]. In DLBCL-FL research, the microarray contains gene expression profiles for 77 patients (58 with DLBCL and 19 with FL) for a total of 6817 genes. Accepting the suggestion of [62], we use the value 16,000 as a ceiling and 20 as the lower threshold. All expression levels beyond this range are filtered out because they cannot be reliably measured. The variation filter is used to exclude genes showing small variation across samples. Two types of variations are used as the measurements: fold-change and absolute variation. For each gene, these two variations are calculated from \( \frac{\text{max}}{\text{min}} \) and \( \text{max} - \text{min} \) respectively, where \( \text{max} \) and
*min* refer to the maximum and minimum value of expression level for this gene across all samples. For this data-set, we exclude all genes with \( \frac{\text{max}}{\text{min}} < 3 \) or \( \text{max} - \text{min} < 100 \).

**Leukemia**

This data-set is introduced in [27] and its specification is available at http://www.genome.wi.mit.edu/MPR. This data-set is used to study two types of acute leukemia – acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML) – based on their gene expressions. This microarray contains 6817 genes. The initial data consists of 38 bone marrow samples (27 ALL and 11 AML). The independent testing set contains another 34 samples (20 ALL and 14 AML). We carried out two experiments on this data set: leave-one-out cross validation on the training data (initial data), and class prediction of the testing data using the classifier built from the training data. A similar filtering procedure as in the Lymphoma data-set is employed.

**Breast cancer**

In [59], Radmacher propose a general framework for prediction of predefined tumor classes using gene expression profiles from microarray experiments. They use the application on human breast cancer as the prediction paradigm. The samples are generated from 22 primary human breast tumors (7 BRCA1-mutation-positive, 8 BRCA2-mutation-positive and 7 sporadic patient samples). The interest of the experiment is in determining whether hereditary breast cancers could be classified based solely on their gene-expression profile. The 22 samples are grouped in two ways. The first grouping labels the 22 tumor samples according to \( BRCA1 \)– mutation status (positive or negative), and the second grouping labels the samples according to \( BRCA2 \)– mutation status (positive or negative). There are 3226 genes in this data-set. The data is obtained through request to the authors.
This data-set contains the total of 2000 gene expressions of 40 tumor and 22 normal colon tissue samples [1]. This data-set was original produced for testing the clustering method. In this chapter, we will apply different supervised learning methods on it and compare performances. Following the suggestion from the author of [59], we employ the following preprocessing on this data-set: 1) compute the median of each array (an array corresponds to a specimen); 2) determine the median of the medians computed in step 1; call this value $M$; 3) for a given array, add or subtract an appropriate constant to each expression value to re-center the median of the array to be $M$; 4) log-transform the entire data-set.

### 4.7.2 Experiments

Because of the small number of samples, cross-validation error estimation has been quite popular for microarray classification. In our work, we use the leave-one-out-test cross-validation. In each iteration, one sample is left out for testing. We count $+1$ if the sample is successfully classified. This process is repeated $n$ times, one for each sample that we can leave out. The experimental results on these data-sets are listed in Table 4.8. In the $k$-NN rule, we use cross-validation to determine the the number of neighbors. This value is shown in the parentheses in the column of $k$-NN. In the case of the ML classifier, we consider two cases: where the two classes have the same variation (Eq. (4.2)) or different variation (Eq. (4.1)). This two cases are represented by MLs and MLd in Table 4.8. We use bolded numbers to represent the best recognition rate. In almost all experiments, our DP-based method works best.
4.8 Conclusion

Over-fitting is a common problem in discriminant analysis. When the number of features are much larger than the number of samples, the classifier built on training data usually works poorly on the independent testing set because the data available used for training cannot represent the real underlying distribution of the data. In order to avoid this problem, people tend to apply feature selection before classification. However, in some cases, such as tumor classification based on gene expression, since the class distinction is correlated to a diversity of features, without a well-defined reason, it may not be appropriate for us to limit the number of features. In this chapter, we proposed a new classification strategy aimed to relieve the over-fitting problem while keeping the large number of features. It solves this problem by taking advantage of the information embedded in the testing samples, changing the role of the testing samples from passive objects to active ones. We compare our approach with several well-known discriminant methods for tumor classification on a variety of data-sets. The experimental results confirm our expectation.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>genes</th>
<th>samples</th>
<th>DP</th>
<th>NN</th>
<th>WV</th>
<th>LDA</th>
<th>SVM</th>
<th>ML-s</th>
<th>ML-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLBCL-FL</td>
<td>6817</td>
<td>52 diffuse large B-cell lymphoma 25 follicular lymphoma</td>
<td>74/77</td>
<td>71/77(7)</td>
<td>63/77</td>
<td>74/77</td>
<td>74/77</td>
<td>65/77</td>
<td>58/77</td>
</tr>
<tr>
<td>ALL-AML</td>
<td>6817</td>
<td>27 acute myeloid leukemia 11 acute lymphoblastic leukemia</td>
<td>38/38</td>
<td>37/38(3)</td>
<td>38/38</td>
<td>38/38</td>
<td>38/38</td>
<td>30/38</td>
<td>27/38</td>
</tr>
<tr>
<td>ALL-AML-independent</td>
<td>6817</td>
<td>20 acute myeloid leukemia 14 acute lymphoblastic leukemia</td>
<td>31/34</td>
<td>31/34(3)</td>
<td>31/34</td>
<td>32/34</td>
<td>32/34</td>
<td>31/34</td>
<td>20/34</td>
</tr>
<tr>
<td>BRCA1</td>
<td>3226</td>
<td>7 BRCA1-mutation-positive tumor 15 BRCA1-mutation-negative tumor</td>
<td>21/22</td>
<td>18/22(1)</td>
<td>18/22</td>
<td>18/22</td>
<td>18/22</td>
<td>19/22</td>
<td>16/22</td>
</tr>
<tr>
<td>BRCA2</td>
<td>3226</td>
<td>8 BRCA2-mutation-positive tumor 14 BRCA2-mutation-negative tumor</td>
<td>21/22</td>
<td>21/22(1)</td>
<td>17/22</td>
<td>19/22</td>
<td>18/22</td>
<td>17/22</td>
<td>17/22</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison results of different classifiers for gene-based tumor classification. The value shown in the parentheses in the column of k-NN is the number of neighbors determined by cross-validation.
CHAPTER 5

CONCLUSION

Statistical pattern recognition is an active area of research. The related technologies have been widely used in a variety of engineering and scientific discipline such as psychology, biology, computer vision and remote sensing. In this thesis, three new feature extraction and classification algorithms are proposed, which have been successfully used in many applications, including face recognition, object classification and tumor classification.

The first algorithm (Chapter 2) is a correlation based criterion used to prune noisy bases in LDA without eliminating those bases associated to the true discriminant power of the underlying structure of the data. Experimental results using five different databases have illustrated the robustness of the proposed approach.

The second algorithm is called Subclass Discriminant Analysis (SDA). Unlike other discriminant analysis algorithms which are tuned to specific data distributions, SDA can be applied to most distribution types. This is so, because SDA approximates the underlying distribution of each class with a mixture of Gaussians (subclasses). The key problem in SDA is 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 Chapter 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 address this problem, we introduced a second criterion which takes advantage of the fact that any generalized eigenvalue decomposition equation 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.

The third algorithm developed (DP-based classification, Chapter 4) is aimed to relieve the influence of over-fitting without reducing the number of features. A direct application of this algorithm is tumor classification based on gene expression, because these data-sets tend to have thousands of features but only dozens of samples are usually available. As a result, the classifiers built on training data generally work poorly on the independent testing data. In order to avoid this problem, our algorithm takes advantage of the information embedded in the testing sample, changing the role of testing data from the passive object to an active one in the process of classification. The superiority of the new algorithm is illustrated by the comparisons against well-known methods on several databases.

Finally, we would like to mention the problems left open in in the third algorithm. To date, people only know that molecular variation is correlated to some genes. However, which genes are correlated to a specific cancer is still unclear. To avoid the loss of discriminant information, we tend to use all available genes. Inevitably, noisy genes are included. For a fixed number of genes, the best we can do to deal with over-fitting is to use the testing
sample information, as our algorithm does. However, its performance is expected to improve if noisy genes are correctly detected and excluded. This could be addressed with an algorithm similar to that presented in Chapter 2. To successfully do this, one would need to depend on an algorithm that can prune noisy bases even when the number of samples is much smaller than the number of features. This is still an open problem.
APPENDIX A

NOTATION

\( C \) \quad \text{Number of classes}
\( p \) \quad \text{Dimensionality of the feature space}
\( n \) \quad \text{Number of samples}
\( n_i \) \quad \text{Number of samples in class } i
\( n_{ij} \) \quad \text{Number of samples in the } j^{th} \text{ subclass of class } i
\( H_i \) \quad \text{Number of subclasses in class } i
\( H = \sum_{i=1}^{C} H_i \) \quad \text{Total number of subclasses}
\( H_o \) \quad \text{Optimal number of subclasses}
\( x_{ij} \) \quad \text{The } j^{th} \text{ sample of class } i, \text{ where } 1 \leq i \leq C \text{ and } 1 \leq j \leq n_i
\( x_i \) \quad \text{The } i^{th} \text{ sample, where } 1 \leq i \leq n
\( x_{ji} \) \quad \text{The } j^{th} \text{ feature of sample } i, \text{ where } 1 \leq i \leq n \text{ and } 1 \leq j \leq p
\( \mu \) \quad \text{The sample mean for all the data}
\( \mu_i \) \quad \text{The sample mean for class } i
\( \mu_{ij} \) \quad \text{The sample mean for subclass } j \text{ in class } i
\( S_B \) \quad \text{Between-class scatter matrix}
\( S_W \) \quad \text{Within-class scatter matrix}
\( \Sigma_X \) \quad \text{Sample Covariance matrix}
\( \Sigma_B \) \quad \text{Between-subclass scatter matrix}
\( p_X \) \quad \text{Rank of } \Sigma_X
\( p_B \) \quad \text{Rank of } \Sigma_B
\( W = \{ \mathbf{w}_1, \ldots, \mathbf{w}_{p_B} \} \) \quad \text{Eigenvectors of } S_B
\( U = \{ \mathbf{u}_1, \ldots, \mathbf{u}_{p_X} \} \) \quad \text{Eigenvectors of } \Sigma_X
\( \text{rank}(A) \) \quad \text{Rank of matrix } A
\( \text{ran}(A) \) \quad \text{Range space of matrix } A
APPENDIX B

BAYES RULE AND FISHER’S CRITERION

In this section, we show that LDA is equivalent to the Bayes rule when the data distributions are homoscedastic.

Bayes Decision Rule

Define the conditional risk \( R(\omega_i|x) \) as follows:

\[
R(\omega_i|x) = \sum_{j=1}^{C} \lambda(\omega_i|\omega_j) P(\omega_j|x),
\]

where \( \lambda(\omega_i|\omega_j) \) is the loss function describing the loss incurred for taking \( \omega_i \) when \( \omega_j \) is the correct choice.

The Bayes decision rule will select the class \( \omega_i \) for which the conditional risk \( R(\omega_i|x) \) is minimized.

If

\[
\lambda(\omega_i|\omega_j) = \begin{cases} 
1, & \text{for } i \neq j \\
0, & \text{for } i = j,
\end{cases}
\]

then \( R(\omega_i|x) = \sum_{i \neq j} P(\omega_j|x) = 1 - P(\omega_i|x) \). The Bayes decision rule is the one that minimizes the classification error.

Define the discriminant function \( g_i = -R(\omega_i|x) \), which is equivalent to \( g_i = P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{P(x)} \), and equivalent to \( g_i = \ln p(x|\omega_i) + \ln P(\omega_i) \).
If the densities \( p(x|\omega_i) \) are multivariate Normal distribution, i.e.,

\[
p(x|\omega_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left[ -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right],
\]

then, \( g_i \) is equivalent to:

\[
g_i = -\frac{1}{2} (x - \mu)^T \Sigma_i^{-1} (x - \mu) - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i).
\]

If the data is homoscedastic, i.e., \( \Sigma_i = \Sigma \) for \( i = 1 \cdots , C \), then

\[
g_i = \ln P(\omega_i) - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + x^T \Sigma^{-1} \mu_i.
\]

In the two-class problem, we classify \( x \) as \( \omega_1 \) if

\[
v^T x + v_0 > 0,
\]

where the intercept \( v_0 = -\ln \frac{P(\omega_2)}{P(\omega_1)} - \frac{1}{2}(\mu_1 + \mu_2) v \), and the projection vector is

\[
v = \Sigma^{-1}(\mu_1 - \mu_2). \tag{B.1}
\]

**Fisher’s criterion**

The approach adopted by Fisher was to find a linear combination of the variables that separate the two classes as much as possible. The criterion proposed by Fisher is the ratio of between-class to within-class variance. Formally, we seek a direction \( v \) such that

\[
J_F = \frac{|v^T (\mu_1 - \mu_2)|^2}{v^T S_W v} \tag{B.2}
\]

is maximum. Where \( S_W = \frac{1}{n}(n_1 \Sigma_1 + n_2 \Sigma_2) \). The solution for \( v \) that maximizes \( J_F \) can be obtained by taking the derivative of \( J_F \) with respect to \( v \) and making it zero. This yields

\[
\frac{v^T (\mu_1 - \mu_2)}{v^T S_W v} \left\{ 2(\mu_1 - \mu_2) + \left( \frac{v^T (\mu_1 - \mu_2)}{v^T S_W v} \right) S_W v \right\} = 0.
\]
Hence,

\[ v \propto S_W^{-1}(\mu_1 - \mu_2). \quad (B.3) \]

We note that when \( \Sigma_1 = \Sigma_2 \), (B.3) is equal to (B.1).
APPENDIX C

PROOF OF LEMMAS AND THEOREMS

C.1 Proof of Result 2

Proof. The eigenvalue decomposition of $S_B$ and $\Sigma_X$ can be written as:

$$
\Sigma_X = \sum_{j=1}^{p_X} \lambda_{Xj} u_j u_j^T = U \Lambda_X U^T,
$$

$$
S_B = \sum_{i=1}^{p_B} \lambda_{B_i} w_i w_i^T,
$$

where $\lambda_{X1} \geq \cdots \geq \lambda_{Xp_X}$, $U = (u_1, \ldots, u_{p_X}) = (U_s, U_{p_X-s})$, $U_s = \{u_j | \lambda_{Xj} \neq 0\}$ and $U_{p_X-s}$ contains the remaining $p_X - s$ eigenvectors of $\Sigma_X$. We first project the data onto the subspace spanned by $U_s$. If in the reduced space the two matrices corresponding to $\Sigma_X$ and $S_B$ are

$$
\tilde{S}_B = U_s^T S_B U_s,
$$

$$
\tilde{\Sigma}_X = U_s^T \Sigma_X U_s = \Lambda_{U_s},
$$

then the generalized eigenvalue decomposition of $\tilde{S}_B$ with respect to $\tilde{\Sigma}_X$ is:

$$
\tilde{S}_B \tilde{V} = \tilde{\Sigma}_X \tilde{V} \tilde{\Lambda}
$$

$$
U_s^T S_B U_s \tilde{V} = \Lambda_{U_s} \tilde{V} \tilde{\Lambda}.
$$

(C.1)
Here, the diagonal matrix $\Lambda_{U_s}$ represents the eigenvalues of the $s$ eigenvectors in $U_s$.

Using basic linear algebra, (C.1) can be rewritten as

$$\Lambda_{U_s}^{-1} U_s^T S_B U_s \tilde{V} = \tilde{V} \tilde{\Lambda}$$
$$U_s \Lambda_{U_s}^{-1} U_s^T S_B U_s \tilde{V} = U_s \tilde{V} \tilde{\Lambda}$$

\begin{align*}
\left( u_1 \cdots u_s \right) \begin{pmatrix}
\frac{1}{\lambda_{X_1}} \\
\vdots \\
\frac{1}{\lambda_{X_s}}
\end{pmatrix} \begin{pmatrix}
u_1^T \\
\vdots \\
u_s^T
\end{pmatrix} \begin{pmatrix}
\sum_{i=1}^{p_B} \lambda_{B_i} u_i^T w_i^T U_s \tilde{V} = U_s \tilde{V} \tilde{\Lambda}
\end{pmatrix}
\end{align*}

\begin{align*}
\left( \frac{1}{\lambda_{X_1}} u_1 \cdots \frac{1}{\lambda_{X_s}} u_s \right) \begin{pmatrix}
\sum_{i=1}^{p_B} \lambda_{B_i} u_i^T w_i^T \\
\vdots \\
\sum_{i=1}^{p_B} \lambda_{B_i} u_i^T w_i^T
\end{pmatrix}
\begin{pmatrix}
\sum_{j=1}^{s} \sum_{i=1}^{p_B} \frac{\lambda_{B_i}}{\lambda_{X_j}} (u_j, w_i) u_j w_i^T U_s \tilde{V} = U_s \tilde{V} \tilde{\Lambda}
\end{pmatrix}
\end{align*}

\begin{align*}
\sum_{\lambda_{X_j} \neq 0} \sum_{i=1}^{p_B} \frac{\lambda_{B_i}}{\lambda_{X_j}} (u_j, w_i) u_j w_i^T U_k \tilde{V} = U_k \tilde{V} \tilde{\Lambda}.
\end{align*}

Comparing (C.2) and (2.3), we note that $V$ is identical to $U_s \tilde{V}$. \qed

\section*{C.2 Proof of Theorem 3}

\textit{Proof.} Since $\Sigma_X = U \Lambda_X U^T = U \Lambda_X^{1/2} \Lambda_X^{1/2} U^T = \left( U \Lambda_X^{1/2} \right) \left( U \Lambda_X^{1/2} \right)^T$, we can rework Eq. (3.1) as follows

$$S_B V = \left( U \Lambda_X^{1/2} \right) \left( U \Lambda_X^{1/2} \right)^T V \Lambda$$

$$\left( U \Lambda_X^{1/2} \right)^{-1} S_B \left[ \left( U \Lambda_X^{1/2} \right)^T \right]^{-1} \left( U \Lambda_X^{1/2} \right)^T V = \left( U \Lambda_X^{1/2} \right)^T V \Lambda.$$

Let $Y = \left( U \Lambda_X^{1/2} \right)^T V$. Since $U$ is a matrix of orthonormal vectors, we can write the above equation as

111
\[ \Lambda_X^{-1/2} \mathbf{U}^T \mathbf{S}_B \mathbf{U} \Lambda_X^{-1/2} = \mathbf{Y} \Lambda. \] \hspace{1cm} (C.3)

Eq. (3.1) and Eq. (C.3) have the same eigenvalues \( \Lambda \), but different eigenvectors, which are related by \( \mathbf{Y} = \left( \mathbf{U} \Lambda_X^{-1/2} \right)^T \mathbf{V} \).

\[ \Lambda_X^{-1/2} \begin{pmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_p^T \end{pmatrix} \left( \sum_{i=1}^{p_B} \lambda_{B_i} \mathbf{w}_i \mathbf{w}_i^T \right) \mathbf{U} \Lambda_X^{-1/2} \mathbf{Y} = \mathbf{Y} \Lambda \]

where \( a_{st} = \sum_{i=1}^{p_B} \frac{\lambda_{B_i}}{\sqrt{\Lambda_{X_s} \Lambda_{X_t}}} \langle \mathbf{u}_s, \mathbf{w}_i \rangle \mathbf{w}_i^T \mathbf{u}_t \).

For any matrix \( \mathbf{M} \) (of \( p \) rows and \( p \) columns) the tr \( (\mathbf{M}) = \sum_{i=1}^p \lambda_i \) and, therefore,

\[ \text{tr} (\Sigma_X^{-1} \mathbf{S}_B) = \sum_{j=1}^p \sum_{i=1}^{p_B} \frac{\lambda_{B_i}}{\sqrt{\Lambda_{X_j} \Lambda_{X_j}}} \langle \mathbf{u}_j, \mathbf{w}_i \rangle \mathbf{w}_i^T \mathbf{u}_j \]

\[ = \sum_{i=1}^{p_B} \sum_{j=1}^p \frac{\lambda_{B_i}}{\Lambda_{X_j}} (\mathbf{u}_j^T \mathbf{w}_i)^2 \]

When \( \Sigma_X \) is not full rank, i.e., \( p_X < p \),

\[ \sum_{i=1}^{p_B} \sum_{j=1}^p \frac{\lambda_{B_i}}{\Lambda_{X_j}} (\mathbf{u}_j^T \mathbf{w}_i)^2 = \sum_{i=1}^{p_B} \sum_{j=1}^p \frac{\lambda_{B_i}}{\Lambda_{X_j}} (\mathbf{u}_j^T \mathbf{w}_i)^2 \]

because \( \mathbf{u}_j^T \mathbf{w}_i = 0 \) for \( \lambda_{X_j} = 0 \).

\[ \square \]

**C.3 Proof of Result 4**

*Proof.* We can rework Eq. (3.4) as follows,

\[ \hat{\Sigma}_B = \sum_{i=1}^C H_i \sum_{j=1}^{H_i} \frac{n_{ij}}{n} (\mu_{ij} - \mu_i + \mu_i - \mu)(\mu_{ij} - \mu_i + \mu_i - \mu)^T = \]

\[ \sum_{i=1}^C H_i \sum_{j=1}^{H_i} \frac{n_{ij}}{n} (\mu_{ij} - \mu_i)(\mu_{ij} - \mu_i)^T + \sum_{i=1}^C H_i \sum_{j=1}^{H_i} \frac{n_{ij}}{n} (\mu_i - \mu)(\mu_i - \mu)^T + \sum_{i=1}^C H_i \sum_{j=1}^{H_i} \frac{n_{ij}}{n} (\mu_{ij} - \mu_i)(\mu_i - \mu)^T. \] \hspace{1cm} (C.10)
The first summing term can be further simplified as,

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

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

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

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

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

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

\[\square\]

C.4 Proof of Result 5

Proof.

$$
2\hat{\Sigma}_B = 2 \sum_{i=1}^{H} p_i (m_i - m)(m_i - m)^T
$$

$$
= \sum_{i=1}^{H} \sum_{j=1}^{H} p_i p_j (m_i - m_j)(m_i - m_j)^T
$$

$$
= \sum_{i=1}^{C} \sum_{j=1}^{h_i} \left[ \sum_{k=1}^{C} h_k \sum_{l=1}^{C} p_{ij} p_{kl} (m_{ij} - m_{kl})(m_{ij} - m_{kl})^T \right]
$$

$$
= \sum_{i=1}^{C} \sum_{j=1}^{h_i} \left[ h_i \sum_{l=1}^{h_i} p_{ij} p_{il} (m_{ij} - m_{il})(m_{ij} - m_{il})^T + \sum_{k \neq i}^{h_k} \sum_{l=1}^{h_i} p_{ij} p_{kl} (m_{ij} - m_{kl})(m_{ij} - m_{kl})^T \right]
$$

$$
= \sum_{i=1}^{C} \sum_{j=1}^{h_i} \left[ h_i \sum_{l=1}^{h_i} p_{ij} p_{il} (m_{ij} - m_{il})(m_{ij} - m_{il})^T \right] + \sum_{i=1}^{C} \sum_{j=1}^{h_i} \sum_{k \neq i}^{h_k} h_k p_{ij} p_{kl} (m_{ij} - m_{kl})(m_{ij} - m_{kl})^T
$$

$$
= A + B,
$$

113
where

\[
A = \sum_{i=1}^{C} \sum_{j=1}^{h_i} \sum_{l=1}^{h_i} \frac{n_{ij} n_{il}}{n} \left( \frac{n_i}{n} \right)^2 (m_{ij} - m_{il})(m_{ij} - m_{il}^T)
\]

\[
= \sum_{i=1}^{C} \frac{n_i^2}{n^2} \sum_{j=1}^{h_i} \sum_{l=1}^{h_i} \frac{n_{ij} n_{il}}{n_i} (m_{ij} - m_{il})(m_{ij} - m_{il})^T
\]

\[
= \sum_{i=1}^{C} \frac{n_i^2}{n^2} \frac{h_i}{2} \sum_{j=1}^{h_i} \frac{n_{ij}}{n_i} (m_{ij} - m_i)(m_{ij} - m_i)^T
\]

\[
= 2 \sum_{i=1}^{C} \frac{n_i^2}{n^2} Q_i,
\]

and

\[
B = \sum_{i=1}^{C} \sum_{j=1}^{h_i} \sum_{k=i+1}^{h_i} \sum_{l=1}^{h_k} p_{ij} p_{kl} (m_{ij} - m_{kl})(m_{ij} - m_{kl})^T
\]

\[
= \sum_{i=1}^{C} \frac{h_i}{n} \sum_{j=1}^{h_i} \sum_{k=i+1}^{C} \frac{h_k}{n} \sum_{l=1}^{h_k} p_{ij} p_{kl} (m_{ij} - m_{kl})(m_{ij} - m_{kl})^T
\]

\[
= 2 \sum_{i=1}^{C} \frac{n_i}{n} Q_i.
\]

\[
\hat{\Sigma}_B = \sum_{i=1}^{C} \frac{n_i^2}{n^2} Q_i + \Sigma_B. \tag{C.11}
\]

From Result 4, we have

\[
\hat{\Sigma}_B = S_B + \sum_{i=1}^{C} \frac{n_i}{n} Q_i. \tag{C.12}
\]

As a result,

\[
\Sigma_B = S_B + \sum_{i=1}^{C} \frac{n_i(n - n_i)}{n^2} Q_i. \tag{C.13}
\]

Since \( \frac{n_i}{n} > \frac{n(n-n_i)}{n^2} \), the intra-subclass scatter is less weighted in \( \Sigma_B \) than that in \( \hat{\Sigma}_B \).

Another observation is: if \( \frac{n_i}{n} > \frac{n_j}{n} \), then the weight of \( Q_i \) decreases faster than the weight of \( Q_j \), because \( \frac{n-n_i}{n} < \frac{n-n_j}{n} \). That means the intra-subclass scatter of group with smaller size is relatively more weighted than the group with larger size. \( \square \)
C.5 Proof of Lemma 6

Proof. For simplicity, we will first show our proof for the two dimensional case. The general case trivially follows from this one.

To get $\Sigma_B = \alpha I$, we need to construct a between-subclass scatter matrix with identical eigenvalues (i.e. $\lambda_1 = \cdots = \lambda_d$). To do that, we need to vary the expected values (means) of the subclasses. This can be achieved by grouping the data into subclasses. Assuming $H = n$, our initial Gaussian distribution equals

$$
\frac{1}{\sqrt{2\pi|\Sigma_X|}} e^{-\frac{1}{2} (x-\mu_x, y-\mu_y)^T \Sigma_X^{-1} (x-\mu_x, y-\mu_y)},
$$

where $\Sigma_X = \text{diag}(a^2, b^2)$. Assuming $0 < b \leq a$ and the data has zero mean.

If we now group the data from $-\infty$ to $-\eta$ into one subclass, we vary the expected value about the $X$-axes as follows:

$$
E_x[x \leq -\eta] = \int_{-\infty}^{\infty} \int_{-\eta}^{\infty} \frac{1}{\sqrt{2\pi ab}} e^{-\frac{1}{2} \left( \frac{x^2}{a^2} + \frac{y^2}{b^2} \right)} dxdy
$$

$$
= \frac{1}{\sqrt{2\pi ab}} \int_{-\infty}^{\infty} e^{-\frac{x^2}{2a^2}} dy \int_{-\infty}^{-\eta} x e^{-\frac{y^2}{2b^2}} dx
$$

$$
= \frac{1}{\sqrt{2\pi ab}} \left( \sqrt{2\pi} b \right) \left( -a^2 e^{-\frac{a^2}{2a^2}} \right)
$$

$$
= -ae^{-\frac{a^2}{2a^2}}.
$$

When

$$
\eta = a \sqrt{2(\log a - \log b)},
$$

we have $E_x[x \leq -\eta] = -b$. Similarly, when we group the samples from $\eta$ to $\infty$, we get $E_x[x \geq \eta] = b$. Then the distribution of the between-subclass scatter matrix has two equal eigenvalue: $\lambda_1 = \lambda_2 = b^2$. By repeating the above method for each of the $d$ dimensions of our space, we can obtain $\lambda_1 = \cdots = \lambda_d$. And, therefore, $\Sigma_B = \alpha I$ where $\alpha = b^2$. \qed
Proof of Lemma 7

Proof. When $\Sigma_B = \alpha I$, Eq. (3.1) is equivalent to $\alpha V = \Sigma_X V \Lambda$, which is the same as:
$$\Sigma_X V = V \alpha \Lambda^{-1}.$$

\[\square\]

Proof of Lemma 9

Proof. When $\Sigma_B$ and $\Sigma_X$ are full rank and have proportional eigenvectors, we can assume that both share a common eigenvector matrix $U = (u_1, u_2, \ldots, u_p)$ but have different eigenvalue matrices, $\Lambda_B$ and $\Lambda_X$. Multiplying each of the equations above by $U^T$, we obtain
$$\Sigma_B = U \Lambda_B U^T, \quad \text{and} \quad \Sigma_X = U \Lambda_X U^T.$$ Substituting the above result in Eq. (3.1) we obtain
$$U \Lambda_B U^T V = U \Lambda_X U^T V \Lambda$$
$$\Lambda_B (U^T V) = \Lambda_X (U^T V) \Lambda$$
$$(\Lambda_X^{-1} \Lambda_B) M = M \Lambda,$$
where $M = U^T V$, which is the same as $V = UM$.

\[\square\]

Proof of Lemma 10

Proof. If $\frac{\lambda_{B1}}{\lambda_{X1}} \geq \frac{\lambda_{B2}}{\lambda_{X2}} \geq \ldots \geq \frac{\lambda_{Bp}}{\lambda_{Xp}}$, Eq. (3.6) has the obvious solution $M = I$, and therefore $V = U M = (u_1, \ldots, u_p)$.

When $\frac{\lambda_{B1}}{\lambda_{X1}} \leq \frac{\lambda_{B2}}{\lambda_{X2}} \leq \ldots \leq \frac{\lambda_{Bp}}{\lambda_{Xp}}$, we have
$$M = \begin{pmatrix}
0 & 0 & \ldots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 1 & \ldots & 0 \\
1 & 0 & \ldots & 0
\end{pmatrix},$$

116
and, therefore, \( V = U M = (u_p, \ldots, u_1) \).

C.9 Proof of Result 14

Proof. It is well-known that

\[
\Sigma_X = S_B + S_W,
\]

and therefore,

\[
\text{rank}(\Sigma_X) = \text{rank}(S_B) + \text{rank}(S_W).
\]

Because \( \text{rank}(\Sigma_X) \leq n - 1 \) and \( \text{rank}(S_B) \leq C - 1 \), \( \text{rank}(S_W) \leq n - C \). Therefore, if \( p - (n - 2) \geq 1 \), i.e., \( p \geq n - 1 \), there exists at least one direction in \( \mathbb{R}^p \) where \( S_W \) has zero variance.

Fisher-Rao’s criterion \( v = \arg \max_a \frac{a^T S_B a}{a^T S_W a} \) searches for the direction in which within-class variance is minimum and between-class variance is maximum. The direction \( v \) where \( S_W \) has zero variance will be selected by this criterion, and therefore, \( v^T \Sigma_X v = v^T S_B v \).

The discriminant power \( DP = \frac{v^T S_B v}{v^T \Sigma_X v} = 1 \) always.
BIBLIOGRAPHY


