Kernel-Based Feature Extraction and Speech Technology Applications

András Kocsor

Research Group on Artificial Intelligence

December 2003

A thesis submitted for the degree of doctor of philosophy of the University of Szeged

University of Szeged
Doctoral School in Mathematics and Computer Science
Ph.D. Program in Informatics

"Plant a kernel of wheat and you reap a pint;
plant a pint and you reap a bushel.
Always the law works to give you back more than you give."

Anthony Norwell

Preface

Let us begin this thesis, constructed around the notion of the popular "kernel-idea" which researchers are now applying to machine learning, with the interpretation of the quotation above. But to do so we first have to explain the kernel idea in a nutshell:

> We can say that the kernel idea is nothing more than the implicit transformation of a problem into a space probably more suitable for the solution.

It has been proved in a number of studies that the extra computation needed for the transformation pays off in the solving of numerous machine learning problems.

Structure of the thesis. The goal of this thesis is twofold, and a separate part is devoted to each. The first one summarizes the kernel idea and its properties, then presents some linear feature extraction algorithms and their non-linear counterparts derived with the help of the kernel functions. We will discuss both supervised and unsupervised feature extraction – Principal Component Analysis (PCA) and Independent Component Analysis (ICA) belonging to the first group, Linear Discriminant Analysis (LDA) and Springy Discriminant Analysis (SDA) belonging to the latter one. The non-linear versions of these methods (Kernel-PCA, Kernel-ICA, Kernel-LDA, Kernel-SDA) can, of course, be categorized the same way. We also demonstrate the behavior of each method on artificial data sets in order to help the reader gain a visual impression of how they work. The second part of the thesis is about speech technology applications, namely speech recognition, speech therapy and teaching reading. We shall demonstrate that the feature extraction methods defined in the first part do indeed increase classification performance.

Acknowledgement. First of all, I would like to thank my supervisor, Prof. János Csirik for supporting my work with useful comments and letting me work at an inspiring department, the Research Group on Artificial Intelligence. I would also thank all my colleagues László Tóth, Kornél Kovács, László Felföldi, Dénes Paczolay and Gábor Gosztolya – the members of the Speech Technology Team – for striving so hard on those applications – such as the OASIS speech recognition system and the SpeechMaster speech-therapy, reading therapy and teaching reading package – that provided the framework for testing the feature extraction algorithms of this thesis in real-life applications. I would also like to express my gratitude to Prof. Imre Bálint for useful comments on the topic and to David Curley for scrutinizing and correcting this thesis from a linguistic point of view.

Last, but not least, my heart-felt thanks goes to my wife Mariann for providing a secure family background during the time spent writing this work.

András Kocsor, December 2003.

Notation

$\mathbb{N}$	natural numbers
$\mathbb{R}$	real numbers
$\mathbb{R}_+$	positive reals
$\mathcal{X}$	space of input patterns; the input space
n	dimension of input patterns
k	number of input patterns
$\mathbf{x}_1, \ldots, \mathbf{x}_k$	input patterns; column vectors
$\{1, \ldots, r\}$	possible class labels of input patterns; $r \in \mathbb{N}$
$\mathcal{L}$	indicator function; $\mathcal{L}(i)$ gives the class label of the sample $\mathbf{x}_i$
$\mathcal{F}$	kernel feature space
X	matrix of the input patterns; $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^\top$
$\mathbf{x}^\top,\ X^\top$	transpose of a vector, matrix
ϕ	mapping to a kernel feature space; $\phi : \mathcal{X} \to \mathcal{F}$
$\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)$	image patterns under ϕ
F	image matrix; $F = (\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_k))$
$\mathbf{x} \cdot \mathbf{z}$	inner product or dot product between $\mathbf{x}$ and $\mathbf{z}$
κ	kernel function; $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
K	kernel matrix; $[K]_{ij} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$

List of Figures

List of Tables

Contents

To my son,
Dániel

Chapter 1

Introduction – Machine Learning: Pattern Recognition Systems

"The problem of learning is arguably at the very core of the problem of intelligence, both biological and artificial."

T. Poggio and C. R. Shelton

In this thesis we concentrate on two key topics in artificial intelligence (AI): machine learning (ML) and its application to speech technology (ST).

Creating intelligent machines is an old dream of mankind. It was realized back in the middle of the last century that the construction of intelligent systems requires automatized learning and decision making [24; 71]. These topics, however, gave rise to considerable problems. "Learning" in the machine learning sense means the application of the model method. That is, we aim at creating models that correctly simulate human thinking. The best way of doing this is to specify the model by means of a large amount of training patterns; decisions regarding a new pattern are made based on this model. But what is a "pattern"? Watanabe defines pattern "as the opposite of a chaos; it is an entity, vaguely defined, that could be given a name" [98]. For example, a speech signal, a portrait, a piece of writing, a fingerprint or a DNA sequence can all be regarded as patterns. And a typical example of decision making based on these patterns is the identification of persons. At the present time, in most sophisticated pattern recognition tasks humans still outperform computers. However, in certain specific tasks that require only a limited level of intelligence, computer models can do better than humans. In speech recognition, for example, humans currently seem unbeatable, but when it comes to the classification of phonemes (the building blocks of speech) based on 20-30 ms signal excerpts, computers have the advantage. Increasing the pattern length from phonemes to syllables, words or sentences, the task of classification becomes increasingly involved, and the computer gradually loses its superiority. Several fields of science like philosophy (as the science of sciences), physics, mathematics, biology, chemistry and theoretical computer science have all contributed to those tools that AI researchers build their models with. Such building blocks are, for example, short and

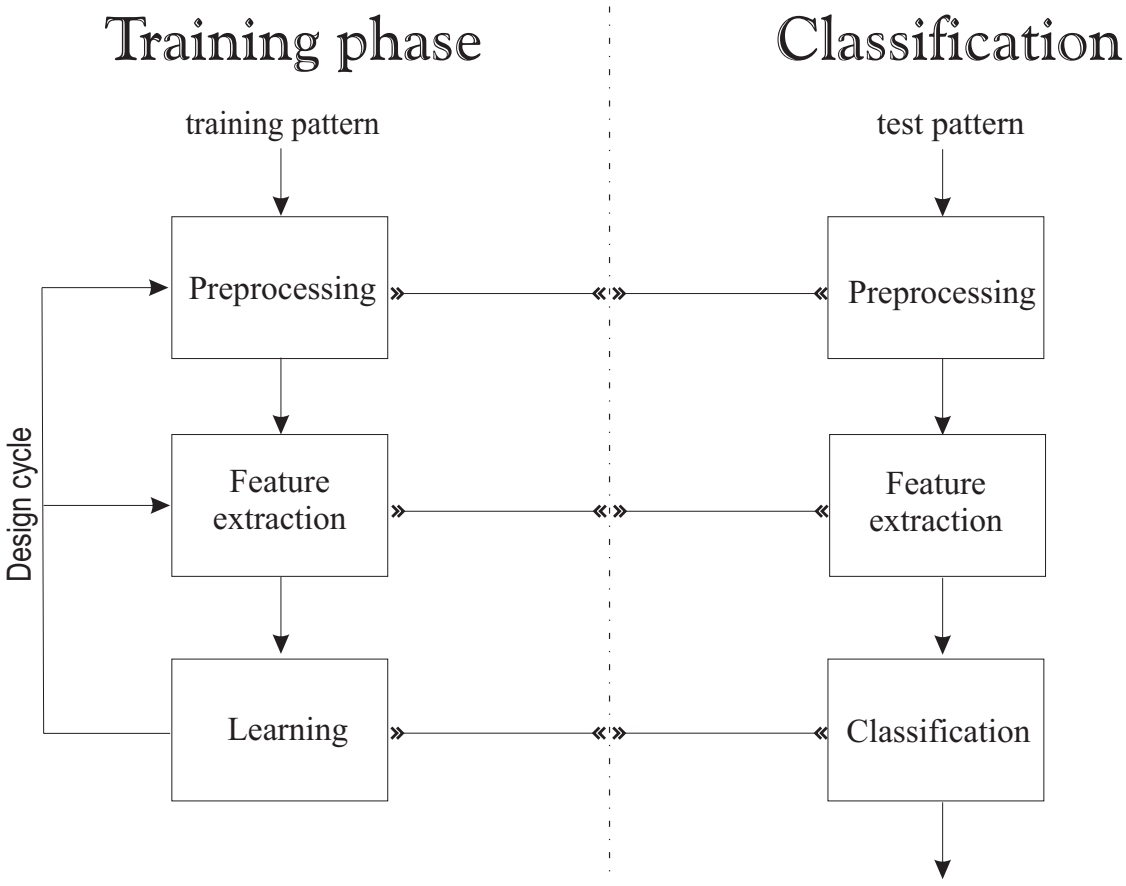

Figure 1.1: Model for pattern recognition.

long-term memory, hierarchical model construction, model hybridization, clustering, data-invariant methods, optimization and approximation. The main problem currently is that, although we are able to construct pattern classification systems that outperform humans on simple subtasks with the help of these tools, the proper way of organizing them into metastructures capable of solving more complex – and thus more "human" problems – is still unknown. Another critical issue deals with the massive parallelism of the human brain, and the collection and handling of the enormous amount of training pattern required for the tuning of the models. Fortunately, the digital revolution, the rapid development of computers and finally the Internet have together made many considerable innovations possible. In fact, these rapid developments have made possible pattern recognition systems that can attain or even exceed human performance. The results of this thesis are also based on the new advances of one such rapidly developing field, namely, non-linear approximation theory.

1.1 Pattern Recognition

The are many well-known approaches of pattern recognition, such as the template matching [6; 36], the syntactical and structural [25; 79], and the neural and statistical approaches [11; 19–21; 68; 91]. What they all have in common is that they are

all comprised of two steps, a learning and a classification phase (see Fig. 1.1). The learning phase means the systematic tuning of the model. That is, after preprocessing the training samples, informative features are extracted and, making use of the training samples, we set the model parameters such that they enable the system to be capable of making good decisions about new samples. When designing the system, the parameters of certain steps may require several re-adjustments for optimal performance. As a result of the training phase we obtain a parameter-tuned model, capable of classifying previously unseen samples. During classification, we virtually repeat the same calculations as in the training stage, but now by employing those parameter values obtained from the training.

In this thesis we will adopt the statistical pattern recognition approach. The samples will be represented as points of an n-dimensional vector space, and our goal is the optimal separation of those points which belong to different classes.

1.2 Summary by Chapters

The present thesis consists of two main parts. The first part (Chapters 2, 3 and 4) discusses four linear and four non-linear feature extraction algorithms, while the second (Chapter 5 and 6) deals with the application of these methods to speech technology.

The second chapter holds the key to the non-linear feature extraction algorithms discussed later. It presents the kernel idea – a method currently being used in machine learning research – that allows us to turn the linear feature extraction algorithms into non-linear ones. The chapter gives a concise overview of some mathematical concepts on the topic. It summarizes the properties of kernel functions, the methods of kernel construction, and provides an overview of the dot product spaces induced by these functions. Finally, at the end of the chapter simple examples are provided to illustrate the application of the kernel idea.

The third chapter discusses four linear feature extraction methods from a unified stand point. Three of them – the Principal Component Analysis (PCA), the Independent Component Analysis (ICA) and the Linear Discriminant Analysis (LDA) – are well-known, while the Springy Discriminant Analysis (SDA) is based on a novel idea, which we first proposed in [60]. These methods linearly transform the initial feature space in order to make a subsequent classification more efficient, faster and less sensitive to noise. They can be dealt with in a unified framework owing to the fact that all of them find the optimal parameters of the row vectors of the transformation matrix by optimizing an objective function given in the form of a Rayleigh quotient. This optimization is relatively straightforward to do as it involves solving a (generalized) eigenvalue problem, and numerous off-the-shelf library routines are available for solving them. The Rayleigh-quotient based approach that we follow during the thesis is based on two papers of ours, the first one [57] presenting PCA, ICA and LDA, while the second one is a summary paper [62] that deals with all the four linear methods in the unified scheme. This chapter virtually provides the whole background for the non-linear methods of the following chapter.

The fourth chapter exploits what the kernel idea and the Rayleigh-quotient based unified view both offer. However, bringing the problem into a Rayleigh-quotient form is not sufficient in itself. We need some further insight. We may soon realize, that the four linear methods lead to the optimization of a Rayleigh quotient which has a special form, where the matrices appearing in the numerator and denominator are both a unique function of the input vectors of the algorithm. Afterwards we derive this special form of the Rayleigh quotient in the dot product space induced by the kernel functions and, analogous to the kernel generalization of PCA suggested by B. Schölkopf et al. [87], the three other methods (ICA, LDA and SDA) can be generalized as well. We proposed the latter kernel based methods in monographs [59],[58] and [60], respectively. Apart from the derivation, the chapter demonstrates the behavior of these methods on artificial 2 and 3 dimensional examples.

In Chapter 5 we commence with the second part of the thesis. In the framework of our OASIS speech recognition system we prepare and execute a segmental phoneme classification test [56; 57; 62]. The aim of the test is to study how the application of the feature extraction algorithms discussed in the previous chapters affects the classification performance of a number of standard classification algorithms (Timbl, OC1, C4.5, GMM, ANN). After briefly introducing the OASIS system we present the conditions and results of the experiment.

The topic of Chapter 6 is again phoneme recognition, but here the application is real-time. The framework of the tests is our "SpeechMaster" software package, an application for speech therapy, teaching reading and reading therapy. The benefits of feature transformation are again studied in combination with several classification methods (ANN, SVM, PPL, GMM). The experiments are based on the techniques we presented in [58–60].

Chapter 7 provides a short summary of the thesis. Lastly, we round off the work with an Appendix containing a partial list of kernel functions and mathematical proofs, followed by a brief summary of the principal results of the thesis.

1.3 Summary by Results

As the dissertation consists of two main parts, the results will separated into two main parts.

The results of the first group of the thesis includes the construction of novel feature extraction algorithms applicable to machine learning problems. These are presented in detail in the first part of the dissertation in Chapters 3 and 4.

I/1. Eight feature extraction algorithms, 4 linear (PCA, ICA, LDA, SDA) and 4 non-linear (Kernel-PCA, Kernel-ICA, Kernel-LDA, Kernel-SDA) are discussed – in a uniform framework – via the optimization of Rayleigh quotient formulas [56–62]. The 4 non-linear methods are derived by non-linearizing the corresponding linear algorithms applying the so-called kernel non-linearization technique.

$\mathcal{N}o.$	PCA	ICA	LDA	SDA	Kernel-PCA	Kernel-ICA	Kernel-LDA	Kernel-SDA	framework
[56]	•				•				OASIS
[57]	•	•	•						OASIS
[58]			•				•		SpeechMaster
[59]		•				•			SpeechMaster
[60]								•	SpeechMaster
[62]	•	•	•	•	•	•	•	•	OASIS

Table 1.1: The relation between the thesis topics and the corresponding publications.

I/2. We constructed a novel linear method called SDA [62], which fits in nicely with 3 linear methods (PCA, ICA, LDA) well-known from the literature.

I/3. Making use of the kernel idea we non-linearized the ICA, LDA and SDA linear algorithms. This resulted in the non-linear methods Kernel-ICA [59], Kernel-LDA [58] and Kernel-SDA [60].

The topic of the second group of the thesis is the use of the methods of the first part in speech technology applications. These are presented in the fifth and sixth chapters of the dissertation.

II/1. We designed and executed several segmental phoneme classification tests within the framework of the OASIS speech recognizer [56; 57; 62]. The goal of these tests was to study how the feature extraction methods affect classification performance.

II/2. To improve the real-time phoneme classification accuracy of the "SpeechMaster" speech therapy, teaching reading and reading therapy software package, we designed and conducted several additional classification tests [58–60].

Finally, Table 1.1 summarizes which publication covers which method of the thesis and which software environment was used in tests carried out.

Part I

Kernel-Based Feature Extraction

Chapter 2

The Kernel Idea

Theoretical developments generally have their own very different, unique histories before they find any practical application. One such example is the "kernel-idea", which had appeared in several fields of mathematics [38; 78] and mathematical physics before it became a key notion in machine learning. The basic idea behind the kernel technique was originally introduced for pattern recognition in [2] and was again employed in the general purpose Support Vector Machine [12; 96; 97], which was followed by a great number of novel kernel-based methods [5; 7; 15; 52; 58–62; 87–90].

Already back in the 60s researchers found it necessary to stress that examples demonstrating the limited computational capabilities of linear functions are very easy to construct. But complex real-world applications in general require more efficient computational models. These models, however, in most cases are much more time consuming than the linear ones. For this reason there is serious need for methods that extend the flexibility of simpler models without significantly increasing their computational complexity.

The kernel idea does exactly this. It can be applied to almost any case when the input of an algorithm are pairwise dot products of n-dimensional vectors over a dot product space. In such cases properly non-linearizing the two-operand dot product operation allows us to perform the algorithm in a new dot product space, hopefully with a higher degree of freedom. This relatively simple idea of operand replacement should, however, be performed with care. To facilitate this here we give a brief overview of the "kernel-idea" based on classic works [15; 16]. The technical part of this chapter is as follows. After outlining the basic properties of kernel functions, we discuss their construction possibilities and the kernel feature space induced by them. Finally, at the end of the chapter two examples are provided to illustrate the application of the kernel idea.

2.1 Mercer kernels

First we define the so-called Mercel kernels, next we devote ourselves to the issue of their existence, then we finish with a discussion of the invariance properties of kernels. In the following we will assume that $\mathcal{X}$ is a compact set in an n-dimensional Euclidean space.

Definition 2.1 *A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a 'Mercer kernel', if and only if it is*

a) continuous

b) symmetric

c) positive definite

Definition 2.2 *Let us define $\mathcal{K}(\mathcal{X})$ as a set of every 'Mercer kernel' over $\mathcal{X} \times \mathcal{X}$.*

The notion of continuity and symmetry is well-known, but positive-definiteness is probably not. Hence we supply a definition of the latter in the following.

Definition 2.3 *A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive definite if for every finite set $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \mathcal{X}$ the $k \times k$ matrix*

$$\begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_k) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_k, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_k, \mathbf{x}_k) \end{pmatrix} \tag{2.1}$$

is positive semi-definite.

First, we recall that a symmetric matrix K is positive semi-definite if and only if $\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \geq 0$, for all $\boldsymbol{\alpha}$. Second, throughout this work, we call the positive semi-definite matrix defined in Eq. (2.1) a kernel matrix.

Definition 2.4 *Let $\kappa(X, Z)$ be the short-hand notation for the $k \times t$ matrix*

$$\begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{z}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{z}_t) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_k, \mathbf{z}_1) & \cdots & \kappa(\mathbf{x}_k, \mathbf{z}_t) \end{pmatrix}, \tag{2.2}$$

where $X = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is a k-tuple, while $Z = (\mathbf{z}_1, \ldots, \mathbf{x}_t)$ is a t-tuple of $\mathcal{X}$.

With this notation we can simply refer to the kernel matrix as $\kappa(X, X)$. Similarly, the ith row and jth column of the kernel matrix will be referred to by the notations $\kappa(\mathbf{x}_i, X)$ and $\kappa(X, \mathbf{x}_j)$, respectively. Sometimes, when it is not confusing, we will refer to the kernel matrix simply by K.

Finding out whether a function is continuous and symmetric is a relatively straightforward task. But checking its positive-definiteness is far from trivial. Are there any Mercer kernels at all? We will reply to this question now as it is an important one.

Proposition 2.1 *The simple dot product $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}, \quad \mathbf{x}, \mathbf{z} \in \mathcal{X}$ is in $\mathcal{K}(\mathcal{X})$.*

From Proposition 2.1 it is obvious that $\mathcal{K}(\mathbf{x})$ contains at least one element. But we can say much more. In fact, $|\mathcal{K}(\mathbf{x})| = \infty$, which can be proved from the following statement on the closure properties of $\mathcal{K}(\mathbf{x})$.

Proposition 2.2 *$\mathcal{K}(\mathcal{X})$ is closed under addition, multiplication, positive scalar addition and positive scalar multiplication, i.e. the following functions are in $\mathcal{K}(\mathcal{X})$:*

$$\begin{array}{lllll} \textit{i)} & \kappa(\mathbf{x},\mathbf{z}) = \kappa_1(\mathbf{x},\mathbf{z}) + \kappa_2(\mathbf{x},\mathbf{z}), & & \kappa_1,\kappa_2 \in \mathcal{K}(\mathcal{X}), \\ \textit{ii)} & \kappa(\mathbf{x},\mathbf{z}) = \kappa_1(\mathbf{x},\mathbf{z})\kappa_2(\mathbf{x},\mathbf{z}), & & \kappa_1,\kappa_2 \in \mathcal{K}(\mathcal{X}), \\ \textit{iii)} & \kappa(\mathbf{x},\mathbf{z}) = \lambda + \kappa_1(\mathbf{x},\mathbf{z}), & \lambda \in \mathbb{R}_+, & \kappa_1 \in \mathcal{K}(\mathcal{X}), \\ \textit{iv)} & \kappa(\mathbf{x},\mathbf{z}) = \lambda\kappa_1(\mathbf{x},\mathbf{z}), & \lambda \in \mathbb{R}_+, & \kappa_1 \in \mathcal{K}(\mathcal{X}). \end{array}$$

Corollary 2.1 *Let $p(x)$ be a polynomial with positive coefficients and κ_1 be a kernel over $\mathcal{X} \times \mathcal{X}$. Then the following functions are also in $\mathcal{K}(\mathcal{X})$:*

i) $\kappa(\mathbf{x},\mathbf{z}) = p(\kappa_1(\mathbf{x},\mathbf{z}))$,

ii) $\kappa(\mathbf{x},\mathbf{z}) = \exp(\kappa_1(\mathbf{x},\mathbf{z})/\sigma)$ *with* $\sigma > 0$.

Proposition 2.3 *Elementary properties of Mercer kernels.*

$$\begin{array}{rl} \textit{Positivity on the diagonal:} & \kappa(\mathbf{x},\mathbf{x}) \geq 0 \quad \mathbf{x} \in \mathcal{X}, \\ \textit{Cauchy-Schwarz inequality:} & \kappa(\mathbf{x},\mathbf{z})^2 \leq \kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{z},\mathbf{z}) \quad \mathbf{x},\mathbf{z} \in \mathcal{X}, \\ \textit{Vanishing diagonals:} & \kappa(\mathbf{x},\mathbf{x}) = 0 \textit{ for all } \mathbf{x} \in \mathcal{X} \Longrightarrow \\ & \kappa(\mathbf{x},\mathbf{z}) = 0 \textit{ for all } \mathbf{x},\mathbf{z} \in \mathcal{X}. \end{array}$$

One or other kernel function may exhibit some of the following invariance properties.

Definition 2.5 *Invariance properties.*

$$\begin{array}{rlllll} \textit{Scaling invariance:} & \kappa(\mathbf{x},\mathbf{z}) & = & \kappa(\lambda_1\mathbf{x},\lambda_2\mathbf{z}) & \lambda_1,\lambda_2 & \in \mathbb{R}_+ \\ \textit{Translation invariance:} & \kappa(\mathbf{x},\mathbf{z}) & = & \kappa(\mathbf{x}+\mathbf{d},\mathbf{z}+\mathbf{d}) & \mathbf{d} & \in \mathbb{R}^n \\ \textit{Unitary invariance:} & \kappa(\mathbf{x},\mathbf{z}) & = & \kappa(U\mathbf{x},U\mathbf{z}) & U^\top = U^{-1} & \in \mathbb{R}^{n\times n} \end{array}$$

2.2 Construction of Kernels

This chapter furnishes recipes for creating kernel functions. But first we need some definitions.

Definition 2.6 *A function $f : [0,\infty) \to \mathbb{R}$ is completely monotonic if it is $\mathcal{C}^\infty$ and $(-1)^k f^{(k)}(r) \geq 0$ for all $r > 0$ and $k \geq 0$. Here $f^{(k)}$ denotes the kth derivative of f.*

Definition 2.7 *A real function $g : (a,b) \to \mathbb{R}$ is convex on (a,b) iff for all $u_1, u_2 \in (a,b)$ and $\alpha \in (0,1)$*

$$f(\alpha u_1 + (1-\alpha)u_2)) \leq \alpha f(u_1) + (1-\alpha)f(u_2). \tag{2.3}$$

Proposition 2.4 *Let $\mathcal{X}$ be a compact domain in $\mathbb{R}^n$, $f : [0,\infty) \to \mathbb{R}$ a completely monotonic function, g a continuous, even function on $\mathbb{R}$, which is convex and decreasing on $(0,\infty)$, $h : \mathcal{X} \to \mathbb{R}$ and $\psi : \mathcal{X} \to \mathbb{R}^m$ are continuous functions, $\kappa_0 \in \mathcal{K}(\mathbb{R}^m)$, and B is a symmetric positive definite $n \times n$ matrix. Then the following functions are Mercer kernels:*

i) $\kappa(\mathbf{x}, \mathbf{z}) = f(||\mathbf{x} - \mathbf{z}||^2)$,

ii) $\kappa(\mathbf{x}, \mathbf{z}) = g(h(\mathbf{x}) - h(\mathbf{z}))$,

iii) $\kappa(\mathbf{x}, \mathbf{z}) = h(\mathbf{x})h(\mathbf{z})$,

iv) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_0(\psi(\mathbf{x}), \psi(\mathbf{z}))$,

v) $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top B\mathbf{z}$.

Now we present some concrete examples of kernel functions popular among the researchers of the kernel methods. Based on the propositions above, it is easy to prove that these functions are indeed kernels.

Examples of kernel functions:

- Gaussian RBF Kernel:

$$\exp\left(-\frac{||\mathbf{x} - \mathbf{z}||^2}{\sigma}\right) \quad \sigma \in \mathbb{R}_+. \tag{2.4}$$

  This function is a kernel, because $f(t) = exp(-t/\sigma)$ is completely monotonic. As regards the invariance properties, it is translation and unitary invariant.

- Polynomial Kernel:

$$(\mathbf{x}^\top \mathbf{z} + \sigma)^q \quad q \in \mathbb{N}, \quad \sigma \in \mathbb{R}_+. \tag{2.5}$$

  The polynomial kernel is obtained from applying $i)$ of Corollary 2.1 on the positive coefficient polynomial $(x + \sigma)^q$. This kernel is unitary invariant.

- Cosine Polynomial Kernel:

$$\kappa(\mathbf{x}, \mathbf{z}) = (\cos(\angle(\mathbf{x}, \mathbf{z})) + \sigma)^q \quad q \in \mathbb{N}, \quad \sigma \in \mathbb{R}_+. \tag{2.6}$$

  To show that this function is a Mercer kernel, making use of Corollary 2.1 it is sufficient to prove that

$$\cos(\angle(\mathbf{x}, \mathbf{z})) = \frac{\mathbf{x}^\top \mathbf{z}}{||\mathbf{x}||||\mathbf{z}||} \tag{2.7}$$

  is a Mercer kernel. The proof follows in exactly the same way as that for Proposition 2.1, but the vectors have to be normalized. A Cosine Polynomial Kernel has scaling and unitary invariance.

- Inverse Multi-Quadratic Kernel:

$$\frac{1}{\sqrt{||\mathbf{x} - \mathbf{z}||^2 + \sigma}} \quad \sigma \in \mathbb{R}_+. \tag{2.8}$$

  This function is indeed a Mercer kernel, as the function $(t + \sigma)^{-1/2}$ is completely monotonic. The invariance properties of it are the same as those of the Gaussian RBF Kernel, that is it is translation and unitary invariant.

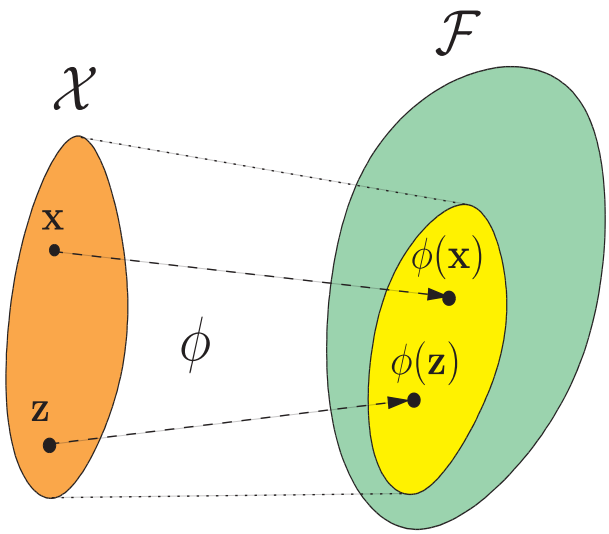

Figure 2.1: The "kernel-idea". The dot product in the kernel feature space $\mathcal{F}$ is implicitly defined.

Finally we note that a somewhat more extensive list of kernel functions can be found in Appendix A.1. A figure for the Gaussian and Polynomial Kernels is also given in Appendices A.2 and A.3.

2.3 Kernel-Induced Feature Spaces

The previous sections defined the Mercer kernels and discussed their properties and construction. Now we will examine what kind of feature spaces the Mercel kernels implicitly induce and how can these be exploited in the non-linearization of certain types of algorithms. First we commence with the following theorem.

Theorem 2.1 *For a Mercer kernel κ over $\mathcal{X} \times \mathcal{X}$ there exists a dot product space $\mathcal{F}$ with a map $\phi : \mathcal{X} \to \mathcal{F}$, such that for all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$*

$$\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}). \tag{2.9}$$

Usually $\mathcal{F}$ is called the kernel feature space and ϕ is the feature map. We have two immediate consequences. When ϕ is the identity, the function $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x} \cdot \mathbf{z}$ (the simple dot product over the space $\mathcal{X}$) is symmetric, continuous and positive definite, so it constitutes a proper Mercer kernel. Going the other way, when applying a general Mercer kernel we can assume a space $\mathcal{F}$ over which we perform dot product calculations. This space and dot product calculations over it are defined only implicitly via the kernel function itself. The space $\mathcal{F}$ and map ϕ may not be explicitly known. We need only define the kernel function, which then ensures an implicit evaluation.

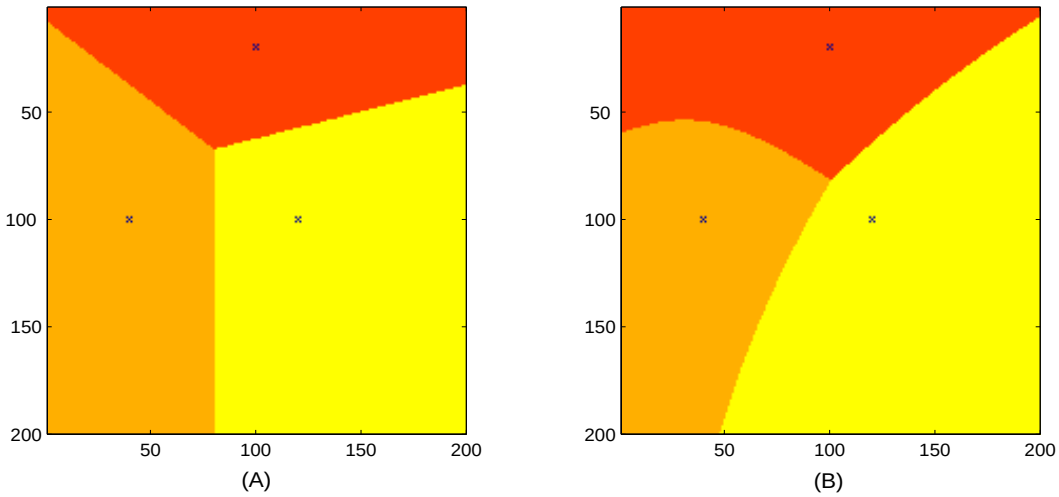

Figure 2.2: Example: Voronoi partitioning in the input space (A) and, in a kernel feature space (B).

Based on Theorem 2.1, the essence of the *kernel trick* can be summarized as follows: *If the output of an algorithm is formulated in terms of a Mercer kernel, then alternative algorithms can be constructed by replacing the kernel with a different Mercer kernel.*

Now we will provide two example applications of the kernel idea. The first one is for demonstration purposes only, while the latter will form the basis of the non-linearization of the feature extraction algorithms presented in the next chapter.

Example: Voronoi partitioning. Let there be k given centers on the $2D$ Euclidean plane, and let us partition the points of the plane by labelling them according to the label of the nearest center point. Those points for which the labelling is ambiguous form the borders of the partitions called a Voronoi polygon. Let us define the distance measure as the Euclidian distance, that is

$$||\mathbf{x}-\mathbf{z}||_2^2 = \mathbf{x}^\top\mathbf{x} + \mathbf{z}^\top\mathbf{z} - 2\mathbf{x}^\top\mathbf{z}. \tag{2.10}$$

Clearly, on the right hand side there are only Mercer kernels. This fact allows us to apply the kernel idea. Let there be a kernel map $\phi : \mathbb{R}^2 \to \mathcal{F}$ and the corresponding kernel function κ. Then the Euclidean distance in the kernel feature space $\mathcal{F}$ can be computed as

$$\begin{aligned} ||\phi(\mathbf{x})-\phi(\mathbf{z})||_2^2 &= \phi(\mathbf{x})\cdot\phi(\mathbf{x}) + \phi(\mathbf{z})\cdot\phi(\mathbf{z}) - 2\phi(\mathbf{x})\cdot\phi(\mathbf{z}) \\ &= \kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathbf{z},\mathbf{z}) - 2\kappa(\mathbf{x},\mathbf{z}) \end{aligned} \tag{2.11}$$

Figure 2.2A shows the result of the partitioning using Eq. (2.10), while Figure 2.2B demonstrates the partitioning applying Eq. (2.11) with a 3rd order polynomial kernel. As can be seen, in the second case the separating points are not straight lines but non-linear curves, owing to the effect of the kernel function. However, one should not forget that, according to Theorem 1, these curves are straight lines in the kernel feature space $\mathcal{F}$ and form a real Voronoi polygon.

Example: Linear Mappings in Kernel Feature Spaces. Let us start with the definition of linear mappings from $\mathcal{X}$ to $\mathbb{R}$. In this case we map the vectors by applying a mapping $\mathbf{z} \rightarrow \boldsymbol{a}^\top \mathbf{z}$, where $\mathbf{z}$ is an arbitrary element of $\mathcal{X}$ and $\boldsymbol{a}$ is a fixed, real n-dimensional vector. If we assume that $\boldsymbol{a}$ is a linear combination of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathcal{X}$, i.e. $\boldsymbol{a} = \alpha_1 \mathbf{x}_1 + \cdots + \alpha_k \mathbf{x}_k$, the linear mapping will have the following dot product form:

$$\mathbf{z} \rightarrow \alpha_1 \mathbf{x}_1^\top \mathbf{z} + \cdots + \alpha_k \mathbf{x}_k^\top \mathbf{z}. \tag{2.12}$$

Let us now choose a fixed Mercer Kernel κ and let $\phi : \mathcal{X} \rightarrow \mathcal{F}$ be a mapping that satisfies Theorem 2.1. Let us also consider the analogue of the linear mapping of Eq. (2.12) in the kernel feature space $\mathcal{F}$

$$\phi(\mathbf{z}) \rightarrow \alpha_1 \phi(\mathbf{x}_1) \cdot \phi(\mathbf{z}) + \cdots + \alpha_k \phi(\mathbf{x}_k) \cdot \phi(\mathbf{z}), \tag{2.13}$$

which – taking into account the fact that $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{z}) = \kappa(\mathbf{x}_i, \mathbf{z})$ – is equivalent to

$$\phi(\mathbf{z}) \rightarrow \alpha_1 \kappa(\mathbf{x}_1, \mathbf{z}) + \cdots + \alpha_k \kappa(\mathbf{x}_k, \mathbf{z}). \tag{2.14}$$

Since the mapping $\mathbf{z} \rightarrow \phi(\mathbf{z})$ is generally non-linear, the composite mapping

$$\mathbf{z} \rightarrow \alpha_1 \kappa(\mathbf{x}_1, \mathbf{z}) + \cdots + \alpha_k \kappa(\mathbf{x}_k, \mathbf{z}) \tag{2.15}$$

yields a non-linear function. In this way linear mappings may easily be extended to non-linear ones. Later this generalization scheme will be employed in Chapter 4 to extend a set of linear feature extraction methods.

2.4 Summary

In this chapter we reviewed a non-linearization idea (which is becoming evermore widespread) because it will be a key part for the developments described in the later chapters. We should like to stress that here we did not intend to give a detailed overview of the theory behind kernels. We deliberately insisted on working with Mercer kernels in our discussions, but Theorem 2.1 also holds for conditionally positive definite (CPD) functions, a class which is wider than Mercer kernels [67; 90]. Though from the theory of CPD functions [22; 63; 101] one could gain a deeper insight into the relations underlying the kernel functions, and though they would provide access to a further set of kernel functions, from a practical point of view the Mercer kernels have just the degree of freedom necessary for solving real-world problems.

Chapter 3

Linear Feature Extraction

In most classification problems it is normal to view the objects to be classified as points in a feature space of proper dimensions. The space has to have a sufficient degree of freedom so that the object classes are sufficiently 'separable'. Making use of superfluous components, however, can confuse classification algorithms. A general practical observation is that it is worth decreasing the dimensionality of the given feature space until we can still guarantee that the overall structure of the data points remains intact. A simple way to do this is by means of a linear transformation that linearly maps an initial feature space into a new features space, usually one with fewer dimensions. Along with dimension reduction, the transformation may also aim at simplifying or emphasizing the structure of the data at the same time.

The mathematical goal of a linear transformation intended for feature extraction can be defined several ways. For example, we may choose the basis vectors of the transformed space as those directions of the original space along which the data shows a large variance (PCA – Principal Component Analysis), or when their distribution greatly differs from a Gaussian one (ICA – Independent Component Analysis). Since these methods ignore the class information of the data, they are called unsupervised. In the opposite case, when an algorithm makes use of the class labels it is called supervised. These algorithms all try to push the classes apart, while keeping the data belonging to the same class close together. Besides the two unsupervised method (PCA, ICA) the chapter presents two supervised methods as well, one of them – Linear Discriminant Analysis (LDA) – is well-known, while the other – Springy Discriminant Analysis (SDA) – is based on a novel idea. This method is based on our springy model, which creates attractive and repulsive forces between the pairs of data points and selects those directions with a high potential [60].

We should say here that, as in the following, we intend to non-linearize the four methods just mentioned using kernel functions. The goal of this chapter is to place the linear algorithms in a unified framework and prepare the groundwork for the coming chapter. We used this unified view for describing PCA, ICA and LDA in [57] and for PCA, ICA, LDA and SDA in [62].

3.1 Introduction

Now without loss of generality we shall assume that, as a realization of multivariate random variables, there are n-dimensional real attribute vectors in a compact set $\mathcal{X}$ over $\mathbb{R}^n$ describing objects in a certain domain, and that we have a finite $n \times k$ sample matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ containing k random observations. Actually, $\mathcal{X}$ constitutes the initial feature space and X is the input data for the linear feature extraction algorithms which defines a linear mapping

$$\begin{aligned} h: \quad & \mathcal{X} \to \mathbb{R}^m \\ & \mathbf{z} \to V\mathbf{z} \end{aligned} \tag{3.1}$$

for the extraction of a new feature vector. The $m \times n$ ($m \leq n$) matrix of the linear mapping – which may inherently include a dimension reduction – is denoted by V, and for any $\mathbf{z} \in \mathcal{X}$ we will refer to the result $h(\mathbf{z}) = V\mathbf{z}$ of the mapping as $\mathbf{z}^*$.

With the linear feature extraction methods we search for an optimal matrix V, where the precise definition of optimality can vary from method to method. Although it is possible to define functions that measure the optimality of m directions (i.e. the row vectors of V) *all together*, here we will find each particular direction of the optimal transformations *one-by-one*, employing a $\tau : \mathbb{R}^n \to \mathbb{R}$ objective function for each direction separately. Intuitively, if larger values of τ indicate better directions and the chosen m directions need to be independent in some ways, then choosing stationary points that have the m largest function values is a reasonable strategy. Obtaining the above stationary points of a general objective function is a difficult global optimization problem. But if τ is defined by a Rayleigh quotient formulae, i.e.

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top B_1 \mathbf{v}}{\mathbf{v}^\top B_2 \mathbf{v}}, \tag{3.2}$$

where B_1 and B_2 are symmetric $n \times n$ matrices and B_2 is positive definite – finding the solution is relatively quick and straightforward when formulated as a simple eigenvalue problem. The above well-known property is asserted in the following proposition.

Proposition 3.1 *The stationary points of $\tau(\mathbf{v})$ are precisely the eigenvectors of matrix $B_2^{-1}B_1$, and the corresponding eigenvalues are the values $\tau(\mathbf{v})$ takes at these points*[1].

Actually, the Rayleigh quotient-based approach offers a unified view of the linear transformation methods discussed in this chapter.

Because two of the four methods to be discussed belong to the supervised family,[2] we should expect to make use of the class labels. To do this let us assume as well that we have r classes and an indicator function $\mathcal{L} : \{1, \ldots, k\} \to \{1, \ldots, r\}$, where $\mathcal{L}(i)$ gives the class label of the sample $\mathbf{x}_i$. Let k_j further denote the number of vectors associated with label j in the sample data.

[1] We note that since B_2 is positive definite, and thus invertible, the simple eigenvalue-eigenvector problem $B_2^{-1}B_1\mathbf{x} = \lambda\mathbf{x}$ is equivalent to the generalized eigenproblem $B_1\mathbf{x} = \lambda B_2\mathbf{x}$

[2] The two types of feature vector space transformations (supervised or unsupervised) can be distinguished by whether they utilize an indicator function containing the class information or not.

3.2 Principal Component Analysis

Principal component analysis is a ubiquitous statistical method for unsupervised feature extraction, data analysis, and compression. It has a very extensive literature, an overview of which can be found in several monographs (e.g. [20; 49]). Some further classic discussions of the technique are given in [42; 51; 72; 73; 80]. From among the widespread applications of PCA we could mention the recognition of objects [53], image processing and image compression [34]. We also applied successfully this method to speech technology in [56; 57; 62].

3.2.1 Derivation of the Method

Normally in PCA the objective function τ for selecting new directions is defined by

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top C \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}, \tag{3.3}$$

where

$$C = E\{(\mathbf{x} - E\{\mathbf{x}\})(\mathbf{x} - E\{\mathbf{x}\})^\top\} \tag{3.4}$$

is the sample covariance matrix. Here E denotes the mean value. The following proposition asserts a well-known property of Eq. (3.3).

Proposition 3.2 *Eq. (3.3) defines $\tau(\mathbf{v})$ as the variance of the centralized sample vectors*

$$\mathbf{x}_1 - E\{\mathbf{x}\}, \ldots, \mathbf{x}_k - E\{\mathbf{x}\} \tag{3.5}$$

projected onto vector $\mathbf{v}/||\mathbf{v}||$.

Hence this method prefers directions which have a large variance. From Proposition 3.1 we know that the stationary points of Eq. (3.3) correspond to the right eigenvectors of the sample covariance matrix C where the eigenvalues form the corresponding optimum values. If we assume that the eigenpairs of C are $(\mathbf{c}_1, \lambda_1), \ldots, (\mathbf{c}_n, \lambda_n)$ and $\lambda_1 \geq \cdots \geq \lambda_n$, then the transformation matrix V will be $[\mathbf{c}_1, \ldots, \mathbf{c}_m]^\top$, i.e. the eigenvectors with the largest m eigenvalues. Notice that since the sample covariance matrix is symmetric and positive semidefinite its eigenvalues are nonnegative and its eigenvectors are orthogonal. Moreover, after applying the above orthogonal transformation V on the sample data X, we find that VX is uncorrelated, i.e. its covariance matrix is diagonal with $\lambda_1, \ldots, \lambda_m$ being along the diagonal:

$$E\{(V\mathbf{x} - E\{V\mathbf{x}\})(V\mathbf{x} - E\{V\mathbf{x}\})^\top\} = VCV^\top = diag(\lambda_1, \ldots, \lambda_m). \tag{3.6}$$

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the PCA transformation can be performed using $\mathbf{z}^* = V\mathbf{z}$. But if the output data needs to be centralized we can apply $V(\mathbf{z} - E\{\mathbf{x}\})$ as well.

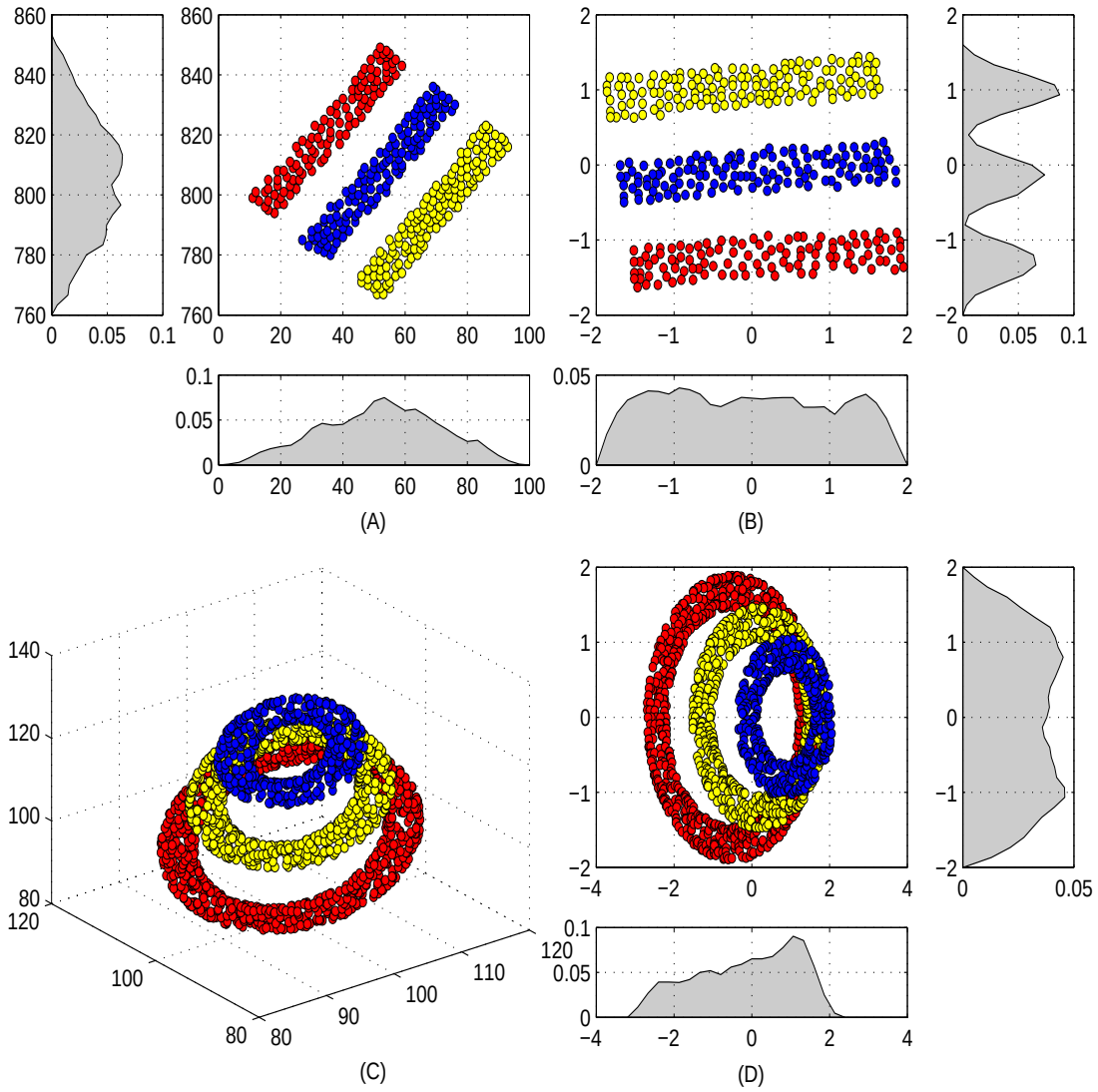

Figure 3.1: The effect of PCA on 2D and 3D data sets. The artificial data sets of Fig. (A) and (C) were transformed using PCA and the results are shown in Fig. (B) and (D), respectively. Besides the data sets themselves we also plotted their distribution along the axes.

3.2.2 2D and 3D Examples

For the purpose of demonstration we generated two artificial data sets shown in Fig. 3.1. The set of Fig. (A) is 2-dimensional, while the set shown in Fig. (C) is of 3 dimensions. Although PCA does not make use of the class information, we chose to represent the class labels using the colors red, blue and yellow. We did so because, after all, we transform the data in order to make the classes more separable, so it is important to see how the transform affects class separability. For ease of understanding the effect of the transformation, the distribution of the data along the x and y-axis was also plotted. When examining Fig. (B) showing the data of Fig. (A) after PCA, we can see that PCA indeed found the direction with the largest variance in Fig. (A), however, this direction is not the most useful for class separability. The same can be said about the PCA transform of (C), shown in Fig. (D).

3.3 Independent Component Analysis

Independent Component Analysis [9; 13; 18; 23; 45–48; 50] is a general purpose statistical method that originally arose from the study of blind source separation (BSS). A typical BSS problem is the cocktail-party problem where several people are speaking simultaneously in the same room and several microphones record a mixture of speech signals. The task is to separate the voices of different speakers using the recorded samples. Another application of ICA is unsupervised feature extraction, where the aim is to linearly transform the input data into uncorrelated components, along which the distribution of the sample set is the least Gaussian. The reason for doing this is that, along these directions, the data is supposedly easier to classify. We also tried to exploit this property in speech technology applications in [57; 59; 62].

For optimal selection of the independent directions, several objective functions were defined using approximately equivalent approaches. Here we follow the way proposed by A. Hyvärinen et al. [23; 45–48]. Generally speaking, we expect these functions to be non-negative and have a zero value for the Gaussian distribution. Negentropy is a useful measure having just this property, which is used for assessing non-Gaussianity (i.e. the least Gaussianity). Since obtaining this quantity via its definition is computationally rather difficult, a simple easily-computable approximation is normally employed. The negentropy of a variable η with zero mean and unit variance is estimated by using the formula

$$J_G(\eta) \approx \left(E\{G(\eta)\} - E\{G(\nu)\}\right)^2 \tag{3.7}$$

where $G : \mathbb{R} \to \mathbb{R}$ is an appropriate non-quadratic function, E again denotes the expectation value and ν is a standardized Gaussian variable. The following three choices of G are conventionally used:

$$\begin{aligned} G_1(\eta) &= \eta^4, \\ G_2(\eta) &= \log\left(\cosh\left(\eta\right)\right), \\ G_3(\eta) &= -\exp\left(-\eta^2/2\right). \end{aligned} \tag{3.8}$$

It should be mentioned that in Eq. (3.7) the expectation value of $G(\nu)$ is a constant, its value only depending on the selected function (e.g. $E\{G_1(\nu)\} = 3$).

3.3.1 Derivation of the Method

In A. Hyvärinen's FastICA algorithm for the selection of a new direction $\mathbf{v}$ the following τ objective function is used:

$$\tau_G(\mathbf{v}) = \left(E\{G(\mathbf{v}^\top \mathbf{z})\} - E\{G(\nu)\}\right)^2, \tag{3.9}$$

which can be obtained by replacing η in the negentropy approximant Eq. (3.7) with $\mathbf{v}\cdot\mathbf{z}$, the dot product of the direction $\mathbf{v}$ and sample $\mathbf{z}$. FastICA is an approximate Newton iteration procedure for the local optimization of the function $\tau_G(\mathbf{v})$. Before running

the optimization procedure, however, the raw input data X must first be preprocessed – by centering and whitening it[3].

Centering. An essential step is to shift the original sample set $\mathbf{x}_1, \ldots, \mathbf{x}_k$ with its mean $E\{\mathbf{x}\}$, to obtain data $\mathbf{x}'_1 = \mathbf{x}_1 - E\{\mathbf{x}\}, \ldots, \mathbf{x}'_k = \mathbf{x}_k - E\{\mathbf{x}\}$, with a mean of $\mathbf{0}$.

Whitening. The goal of this step is to transform the centered samples $\mathbf{x}'_1, \ldots, \mathbf{x}'_k$ via an orthogonal transformation Q into vectors $\mathbf{z}_1 = Q\mathbf{x}'_1, \ldots, \mathbf{z}_k = Q\mathbf{x}'_k$ where the covariance matrix $E\{\mathbf{z}\mathbf{z}^\top\}$ is the unit matrix. Since standard Principal Component Analysis transforms the covariance matrix into a diagonal form [49], where the diagonal elements are the eigenvalues of the original covariance matrix $E\{\mathbf{x}'\mathbf{x}'^\top\}$, it only remains to transform each diagonal element to one. Now if we assume that the eigenpairs of $E\{\mathbf{x}'\mathbf{x}'^\top\}$ are $(\mathbf{c}_1, \lambda_1), \ldots, (\mathbf{c}_n, \lambda_n)$ and $\lambda_1 \geq \ldots \geq \lambda_n$, the transformation matrix Q will take the form $[\mathbf{c}_1\lambda_1^{-1/2}, \ldots, \mathbf{c}_m\lambda_m^{-1/2}]^\top$. If m is less than n a dimensionality reduction is employed.

Following the approach proposed by A. Hyvärinen et al., it is now worth noting some basic properties of the preprocessing stage.

Proposition 3.3 *Properties of the preprocessing stage. After centering and whitening the following properties are fulfilled.*

- $i)$ *For every normalized* $\mathbf{v}$ *the mean of* $\mathbf{v}^\top\mathbf{z}_1, \ldots, \mathbf{v}^\top\mathbf{z}_k$ *is set to zero, and its variance is set to one.*
- $ii)$ *For any matrix* W *the covariance matrix of the transformed, preprocessed points* $W\mathbf{z}_1, \ldots, W\mathbf{z}_k$ *will remain a unit matrix if and only if* W *is orthogonal.*

Actually property $i)$ is essential since Eq. (3.7) requires that η should have a zero mean and variance of one hence, with the substitution $\eta = \mathbf{v}^\top\mathbf{z}$, the projected data $\mathbf{v} \cdot \mathbf{z}$ must also have this property. Moreover, after preprocessing based on property $ii)$ it is sufficient to look for a new orthogonal base W for the preprocessed data, where the values of the non-Gaussianity measure τ_G for the base vectors are large. Note that since the data remains whitened after an orthogonal transformation, ICA can be considered an extension of PCA.

Now we briefly outline how the optimization procedure of the FastICA algorithm works (cf. [23; 47]). The input for this algorithm (see the next page) is the centered & whitened sample $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_k]$ and the non-linear function G, while the output is the transformation matrix W. The first and second order derivatives of G are denoted by G' and G''. In the pseudo-code $(W_iW_i^\top)^{-1/2}W_i$ means a symmetric decorrelation, where $(W_iW_i^\top)^{-1/2}$ can be readily obtained from its eigenvalue decomposition[4].

Transformation of test vectors. For an arbitrary test vector $\mathbf{x} \in \mathcal{X}$ the ICA transformation can be performed using $\mathbf{z}^* = WQ\mathbf{x}$. Here W denotes the orthogonal transformation matrix we obtained as the output from FastICA, while Q is the matrix obtained from whitening. Much like that in PCA, if we require that the output data be centralized then $\mathbf{z}^* = WQ(\mathbf{x} - E\{\mathbf{x}\})$.

[3] There are many other iterative methods for performing Independent Component Analysis, some of these (similar to FastICA) do require centering and whitening, while others do not. In general, experience has taught us that all these algorithms should converge faster on centered and whitened data, even with those which do not really require it.

[4] If $W_iW_i^\top = VDV^\top$, then $(W_iW_i^\top)^{-1/2}$ is equal to $VD^{-1/2}V^\top$.

procedure FastICA_Opt(Z, G);
  % initialization
  let W_0 be a random $m \times m$ matrix;
  $W_0 = (W_0 W_0^\top)^{-1/2} W_0$;
  $i = 0$;
  % approximate Newton iteration
  While W has not converged;
    for $j = 1$ to m
      let the column vector $\mathbf{s}_j$ be the jth row vector of W_i;
      $\mathbf{w}_j = E\{\mathbf{z}G'(\mathbf{s}_j^\top \mathbf{z})\} - E\{G''(\mathbf{s}_j^\top \mathbf{z})\}\mathbf{s}_j$;
    end;
    $i = i + 1$;
    $W_i = [\mathbf{w}_1, \ldots, \mathbf{w}_m]^\top$;
    $W_i = (W_i W_i^\top)^{-1/2} W_i$;
  do
End procedure

3.3.2 2D and 3D Examples

We will demonstrate the behavior of ICA on the same data sets that were used for PCA. The results are shown in Fig. 3.2, organized in a fashion similar to Fig. 3.1. Inspecting Fig. (B), we can say that in this case the distribution along the x-axis seems more convenient from a classification point of view than it was in the case of PCA. We arrive at similar conclusion from examining the 3-torus example of Figs. (C-D). From this we may conclude that forcing the data projections to be the least Gaussian might be a better idea from a classification point of view.

Let us make a useful remark here. As we have seen, the preprocessing step of ICA is a PCA, possibly including a dimension reduction. The next step is the FastICA_Opt routine, resulting in an orthogonal transformation that further transforms our data. Here we again have an opportunity for dimension reduction, that is to omit those components along with the data distribution similar to a Gaussian one. It is important not to reduce the dimensions too much during the preprocessing, otherwise the FastICA_Opt procedure will not have enough degrees of freedom to find a proper direction. This problem may arise especially in situations when we are working with a small number of dimensions. For instance, in the case of the 3-torus example, if we had kept only 2 dimensions instead of 3 after preprocessing (see Fig. 3.1D), we could have not found the direction optimal for class separation, as shown in Fig. (D).

In addition to the points above, we note that the task of distinguishing features optimal for classification from those that may confuse the classifier belongs to the topic of feature selection. There are many possible feature selection strategies both for PCA and ICA. Unfortunately, for optimal performance all subsets of the features should be examined [14], but in practice there are quite good heuristics available. Such strategies, like the SFFS [81] method, are available in the literature.

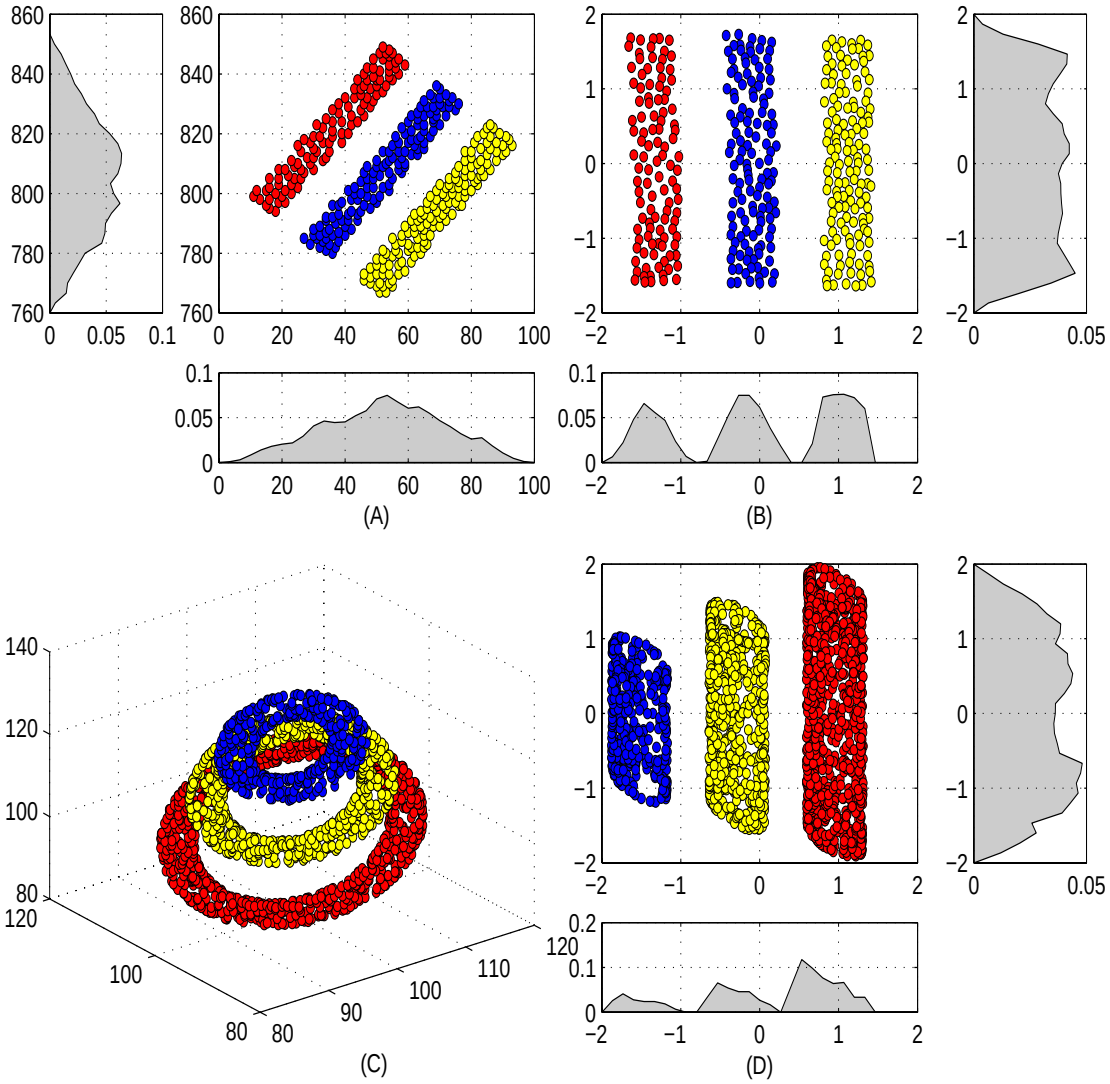

Figure 3.2: The effect of ICA on 2D and 3D data sets. The artificial data sets of Figs. (A) and (C) were ICA transformed and the results depicted in Figs. (B) and (D), respectively. Besides the data sets themselves the axis-wise distributions are also plotted.

3.4 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a traditional supervised feature extraction method [24; 27] which has proved to be one of the most successful preprocessing techniques for classification[5]. The goal of LDA is to find a new (not necessarily orthogonal) basis for the data which provides the optimal separation between groups of sample points (classes)[6]. This method was also applied in our speech recognition studies [57; 58; 62]. Now we continue with a derivation of the method.

[5]One should note here that the technique can be directly used for classification as well.

[6]The mathematical background underlying the LDA method has been extensively studied. It turned out that for two classes having the same covariance matrix the direction found by LDA is Bayes optimal [21]. However, if the covariance matrices are different the optimal decision surface is quadratic [27]. It has also been shown that LDA is closely related to the artificial neural nets with one hidden layer [11].

3.4.1 Derivation of the Method

In order to define the transformation matrix of LDA we first define the objective function $\tau : \mathbb{R}^n \to \mathbb{R}$ which depends not only on the sample data X, but also on the indicator function $\mathcal{L}$ owing to the supervised nature of this method. Let us define

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top B \mathbf{v}}{\mathbf{v}^\top W \mathbf{v}}, \tag{3.10}$$

where B is the *Between-class Scatter Matrix*, while W is the *Within-class Scatter Matrix*. Here the *Between-class Scatter Matrix* B shows the scatter of the class mean vectors $\mathbf{m}_j$ around the overall mean vector $\mathbf{m}$:

$$\begin{aligned} B &= \textstyle\sum_{j=1}^{r} \frac{k_j}{k} (\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^\top \\ \mathbf{m} &= \textstyle\frac{1}{k} \sum_{i=1}^{k} \mathbf{x}_i \\ \mathbf{m}_j &= \textstyle\frac{1}{k_j} \sum_{\mathcal{L}(i)=j} \mathbf{x}_i \end{aligned} \tag{3.11}$$

The *Within-class Scatter Matrix* W represents the weighted average scatter of the covariance matrices C_j of the sample vectors having label j:

$$\begin{aligned} W &= \textstyle\sum_{j=1}^{r} \frac{k_j}{k} C_j \\ C_j &= \textstyle\frac{1}{k_j} \sum_{\mathcal{L}(i)=j} (\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^\top. \end{aligned} \tag{3.12}$$

The value of the function $\tau(\mathbf{v})$ is large when its nominator is large and its denominator is small or, equivalently, when the within-class averages of the sample projected onto $\mathbf{v}$ are far from each other and the variance of the classes is small. The larger the value of $\tau(\mathbf{v})$ the farther the classes will be spaced and the smaller their spreads will be. As we saw earlier (cf. Proposition 3.1) the stationary points of Eq. (3.10) correspond to the right eigenvectors of $W^{-1}B$, where the eigenvalues form the corresponding function values. Before defining the LDA transformation, we will assert a well-known result which gives a maxima for the rank of $W^{-1}B$.

Proposition 3.4 *The rank of $W^{-1}B$, where the matrices are defined in Eqs. (3.11) and (3.12), is less than the class number r.*

Since $W^{-1}B$ is not necessarily symmetrical and its rank is at most the class number minus one $(r-1)$, the number of its real, not necessarily orthogonal eigenvectors is less than r. This means that if $r < n$ we have a limit for the number of new features which can be extracted. Besides this, numerical problems can arise when computing W^{-1} if $det(W)$ is zero or the condition number of W is too large. The most probable cause for this could be the redundancy of feature components. But we know W is positive semidefinite. So if we add a small positive constant ϵ to its diagonal, that is we work with $W + \epsilon I$ instead of W, this matrix is guaranteed to be positive definite and hence should always be invertible. If we assume that the real eigenvectors with the largest $m(< r)$ real eigenvalues of $(W+\epsilon I)^{-1}B$ are $\mathbf{c}_1, \ldots, \mathbf{c}_m$, then the transformation matrix V will be $[\mathbf{c}_1, \ldots, \mathbf{c}_m]^\top$.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the LDA transformation can be performed using $\mathbf{z}^* = V\mathbf{z}$.

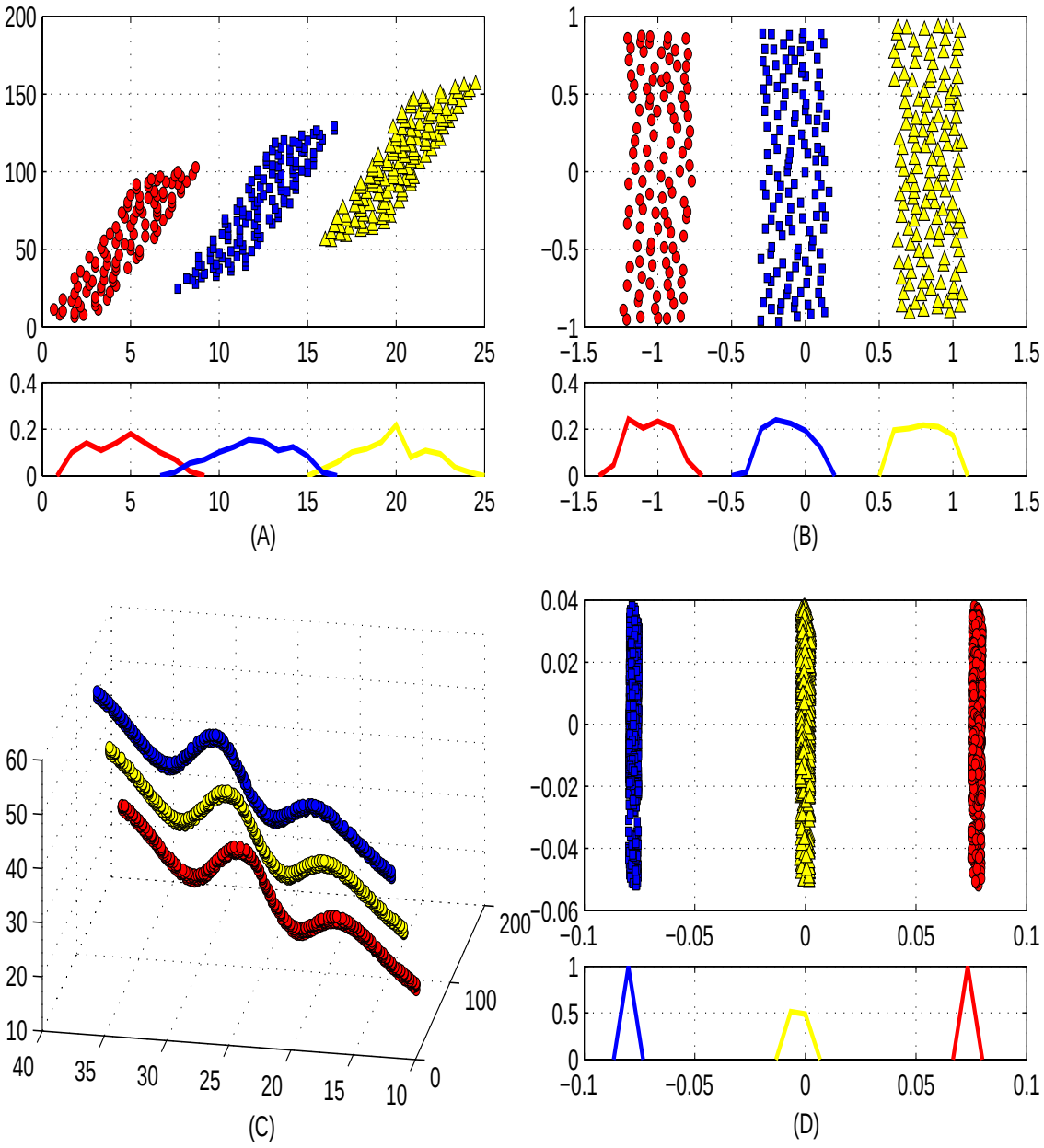

Figure 3.3: The effect of LDA on 2D and 3D data sets. Figures (A) and (C) show two artificial data sets, while Figs. (B) and (D) depict both after LDA. Besides the data, its axis-wise distribution is also plotted using one curve for each class.

3.4.2 2D and 3D Examples

Figure 3.3. depicts the conception of our first supervised feature extraction algorithm, LDA. As can be seen in Figs. (B) and (D), after LDA the data become easily separable along the x-axis in both the 2D and 3D cases. One should also realize that in the case of data set (A) no orthogonal transform could have resulted in a similarly well separated data distribution.

3.5 Springy Discriminant Analysis

We saw that LDA can become numerically unstable because of the invertibility problem of the *Within-class Scatter Matrix*. Furthermore, the nonorthogonality and the limited dimension of the resulting transformation matrix may prove disadvantageous. These issues give rise to the need for an objective function τ that leads to a supervised transformation which yields similar results to LDA, but the transformation matrix is orthogonal and avoids the numerical problems mentioned earlier. Springy Discriminant Analysis (SDA) is a method of ours that takes all these issues into consideration. Although we originally proposed it in a non-linear form [60; 61], we later published the corresponding linear version as well [62].

3.5.1 Derivation of the Method

The name Springy Discriminant Analysis stems from the utilization of a spring & anti-spring model, which involves searching for directions with optimal potential energy using attractive and repulsive forces. In our case sample pairs in each class are connected by springs, while those of different classes are connected by antisprings. New features can be easily extracted by taking the projection of a new point in those directions where a small spread in each class is obtained, while different classes are spaced out as much as possible.

Let $\delta(\mathbf{v})$, the potential of the spring model along the direction $\mathbf{v}$, be defined by

$$\delta(\mathbf{v}) = \sum_{i,j=1}^{k} \left((\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{v} \right)^2 [M]_{ij} \,, \tag{3.13}$$

where

$$[M]_{ij} = \begin{cases} -1, & \text{if } \mathcal{L}(i) = \mathcal{L}(j) \\ 1, & \text{otherwise} \end{cases} \qquad i,j = 1,\ldots,k. \tag{3.14}$$

Naturally, the elements of matrix M can be initialized with values different from ± 1 as well. The elements can be considered as a kind of force constant and can be set to a different value for any pair of data points.

It is easy to see that the value of δ is largest when those components of the elements of the same class that fall in the given direction $\mathbf{v}$ ($\mathbf{v} \in \mathbb{R}^n$) are close, and the components of the elements of different classes are far at the same time.

Now with the introduction of the matrix

$$D = \sum_{i,j=1}^{k} (\mathbf{x}_i - \mathbf{x}_j) (\mathbf{x}_i - \mathbf{x}_j)^\top [M]_{ij} \tag{3.15}$$

we immediately obtain the result that $\delta(\mathbf{v}) = \mathbf{v}^\top D \mathbf{v}$. Based on this, the objective

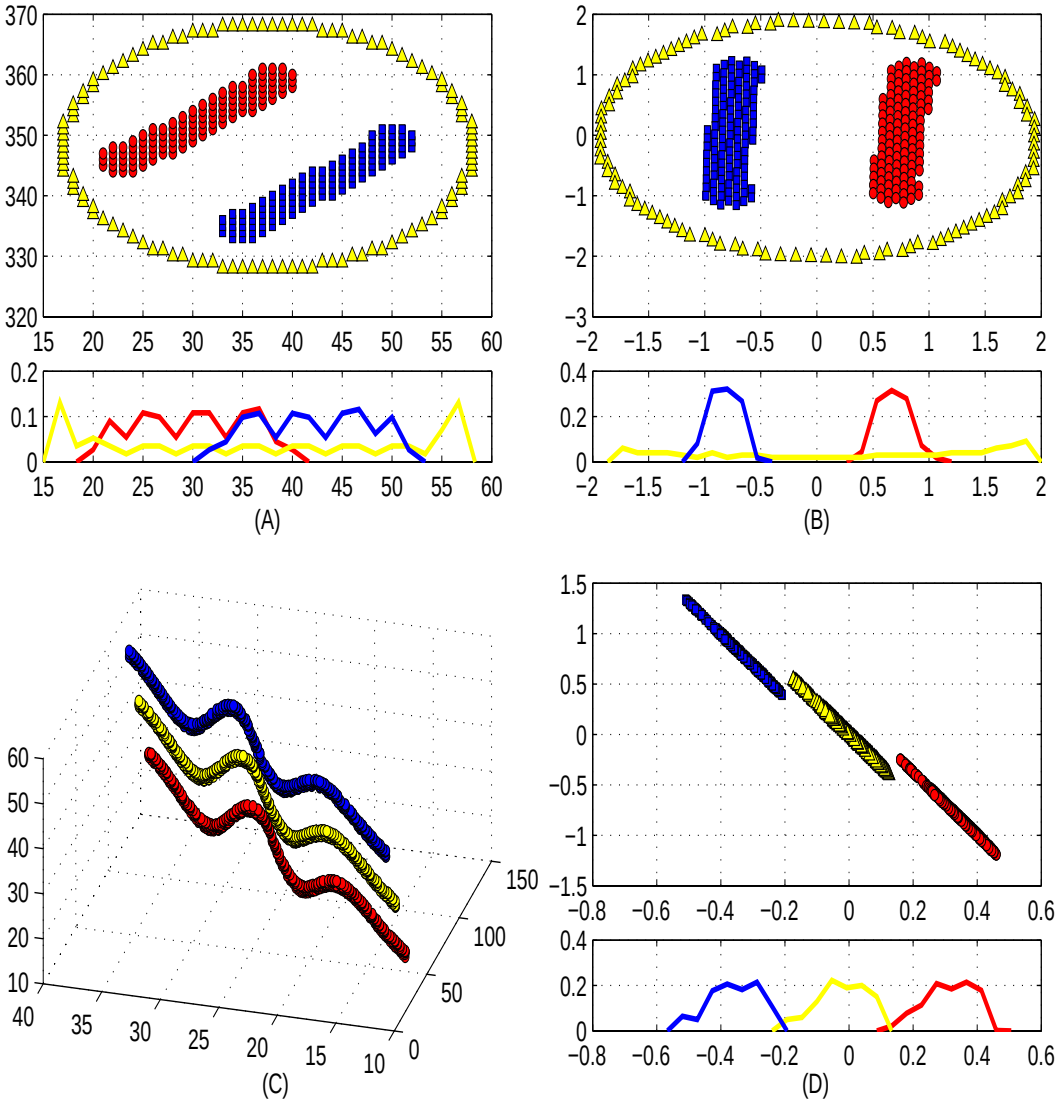

Figure 3.4: The effect of SDA on 2D and 3D data sets. The artificial data sets of Figs. (A) and (C) were transformed using SDA and the results are shown in Figs. (B) and (D). Besides the data the axis-wise distributions are also shown using one curve for each class.

function τ can be defined as the Rayleigh quotient

$$\tau(\mathbf{v}) = \frac{\delta(\mathbf{v})}{\mathbf{v}^\top \mathbf{v}} = \frac{\mathbf{v}^\top D \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}. \tag{3.16}$$

Obviously, the optimization of τ leads to the eigenvalue decomposition of D, just like in the case of PCA. Because D is symmetric, its eigenvalues are real and its eigenvectors are orthogonal. The matrix V of the SDA transformation is defined using those eigenvectors corresponding to the m dominant eigenvalues of D.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the SDA transformation can be performed using $\mathbf{z}^* = V\mathbf{z}$.

3.5.2 2D and 3D Examples

Figure 3.4 readily shows the behavior of SDA. By adjusting the force constant matrix M we can shift the emphasis between the minimization of inter-class variance and the maximization of between-class distance. For the data set (A) SDA found a direction (see Fig. (B)) along which the red and blue points are easily separable, but the yellow ones are not. But, of course, we cannot expect any linear algorithm to separate this latter class. Fig. (D) shows the data set of Fig. (C) after SDA. We can see that, compared to LDA, the distributions associated with these classes are less concentrated than in the case of LDA. This means that minimizing the inter-class variance was less important for SDA than for LDA. On the other hand, the gain is that now the classes are separable not only along the x-axis, but also along the y-axis.

3.6 Summary

In this chapter four linear feature extraction algorithms were presented, all of them working by optimizing a Rayleigh quotient in order to find those directions that, after projecting the data on them, result in a new better feature set [62]. We saw that, thanks to the Rayleigh quotient formulation, the optimization can be performed very efficiently by solving a (generalized) eigenvalue problem.

The general concept of all the four methods is summarized in the following:

PCA concentrates on those independent directions with the largest variances.

ICA besides keeping the directions independent, chooses directions along which the non-Gaussianity is large.

LDA prefers those directions along which the class centers are far away and the average variance of the classes is small.

SDA creates attractive forces between the samples belonging to the same class and repulsive forces between samples of different classes. Then it chooses those directions along which the potential energy of the system is maximal.

In the next chapter we will non-linearize these methods, one after the other, using the kernel idea.

Chapter 4

Non-linear Feature Extraction with Kernels

The approach of feature extraction could be either linear or non-linear, but it seems the kernel-idea is, in some sense, breaking down the barrier between the two types. As we have already seen in Chapter 2 if some linear method uses only the pairwise dot product of its input vectors during its computations then, just by altering the dot product operation in a suitable way, we can create a non-linear version of it. The effect of the replacement of the operation is that the original linear method will implicitly be performed in a space of more (possibly even infinite) dimensions, and thus with a higher degree of freedom. In the following this notion is also used to derive the non-linear counterparts of PCA, ICA, LDA and SDA. The non-linear version of PCA (Kernel-PCA) was first proposed by B. Schölkopf et al. [87]. We applied this algorithm to speech technology in [56; 62], the method itself also being described in this article in detail with a minor modification of the derivation. The main result of this chapter is the three other kernel based feature extraction methods (Kernel-ICA, Kernel-LDA, Kernel-SDA), which we proposed in monographs [59],[58] and [60], respectively.

4.1 Introduction

Let us assume that there are n-dimensional real attribute vectors in a compact set $\mathcal{X}$ over $\mathbb{R}^n$ describing objects belonging to r classes in a certain domain. Furthermore, we will assume that we have a finite $n \times k$ sample matrix

$$X = (\mathbf{x}_1, \ldots, \mathbf{x}_k) \tag{4.1}$$

containing k random observations and an indicator function

$$\mathcal{L} : \{1, \ldots, k\} \to \{1, \ldots, r\}, \tag{4.2}$$

where $\mathcal{L}(i)$ gives the class label of the sample $\mathbf{x}_i$. Here k_j denotes the number of vectors associated with label j in the sample data.

Our goal is now to perform the linear transformations of the previous chapter in kernel feature spaces. To this end let the dot product be defined by a Mercer kernel κ, which induces non-linearly a dot product space denoted by $\mathcal{F}$ via a feature map ϕ (see Theorem 2.1). First, let us examine the Rayleigh quotient of Eq. (3.2) in this space. Formally,

$$\tau(\boldsymbol{v}) = \frac{\boldsymbol{v}^\top \mathcal{B}_1 \boldsymbol{v}}{\boldsymbol{v}^\top \mathcal{B}_2 \boldsymbol{v}}, \quad \boldsymbol{v} \in \mathcal{F} \tag{4.3}$$

where $\mathcal{B}_1$ and $\mathcal{B}_2$ are symmetric matrices of size $\dim(\mathcal{F}) \times \dim(\mathcal{F})$, and $\mathcal{B}_2$ is positive definite. Unfortunately, this form is not restrictive enough so that we could express $\tau(\boldsymbol{v})$ as a function of the kernel κ.

Let us now observe that in the case of all the linear transformations the matrices in the nominator of the Rayleigh quotient were always a unique function of the sample matrix X. They all had the common form

$$X\Theta X^\top = \sum_{i=1}^{j} [\Theta]_{ij} \mathbf{x}_i \mathbf{x}_j^\top, \quad [\Theta]_{ij} \in \mathbb{R} \tag{4.4}$$

(cf. Eqs. (3.3), (3.10) and (3.16)), where Θ was a symmetric real matrix specific to the method. The matrix in the denominator was the unity matrix for PCA, ICA, SDA, and in the case of LDA it was again of the form of Eq. (4.4) (see Eq. (3.12)). Based on this observation, the linear methods of Chapter 3 take the following special Rayleigh quotient form

$$\frac{\mathbf{v}^\top X \Theta_1 X^\top \mathbf{v}}{\mathbf{v}^\top (X \Theta_2 X^\top + \delta I) \mathbf{v}}, \tag{4.5}$$

where $\mathbf{v} \in \mathcal{X}$, X is the matrix containing the samples, Θ_1, Θ_2 are method-dependent real symmetric matrices of size $n \times n$, and $\delta \in \mathbb{R}_+$. In the case of LDA $\delta = 0$, and for PCA, ICA and SDA $\delta = 1$ and Θ_2 is the zero matrix.

Now we can formalize the Rayleigh quotient that corresponds to Eq. (4.5) in the kernel feature space. For this we simply have to substitute matrix

$$F = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)) \tag{4.6}$$

for

$$X = (\mathbf{x}_1, \ldots, \mathbf{x}_k) \tag{4.7}$$

and vector $\boldsymbol{v} \in \mathcal{F}$ for $\mathbf{v} \in \mathcal{X}$:

$$\tau(\boldsymbol{v}) = \frac{\boldsymbol{v}^\top F \Theta_1 F^\top \boldsymbol{v}}{\boldsymbol{v}^\top (F \Theta_2 F^\top + \delta I) \boldsymbol{v}}. \tag{4.8}$$

Now let us have a look at the stationary points of $\tau(\boldsymbol{v})$.

Proposition 4.1 *$\boldsymbol{v} \in span(\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_n))$ holds for all stationary points of $\tau(\boldsymbol{v})$*

That is, we can assume that $\boldsymbol{v} = \alpha_1 \phi(\mathbf{x}_1) + \cdots + \alpha_n \phi(\mathbf{x}_n) = F\boldsymbol{\alpha}$. With this we arrive

at the following form of the Rayleigh quotient defined in Eq. (4.8), now depending on vector $\boldsymbol{\alpha}$:

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top F^\top F \Theta_1 F^\top F \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top F^\top (F \Theta_2 F^\top + \delta I) F \boldsymbol{\alpha}} \tag{4.9}$$

And because $F^\top F$ is equivalent to the kernel matrix $K = \kappa(X, X)$, we obtain the formula

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top K \Theta_1 K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top (K \Theta_2 K + \delta K) \boldsymbol{\alpha}}, \tag{4.10}$$

where, according to Proposition 2.1, the eigenvectors of the following matrix

$$(K \Theta_2 K + \delta K)^{-1} K \Theta_1 K \tag{4.11}$$

are the stationary points. Let us define the matrix of the eigenvectors corresponding to the m dominant eigenvalues by

$$\mathcal{A} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m)^\top. \tag{4.12}$$

Now we may formalize the analogue of the linear mapping of Eq. (3.1) in the kernel feature space $\mathcal{F}$ by

$$\phi(\mathbf{z}) \to \mathcal{U} \phi(\mathbf{z}), \tag{4.13}$$

where $\mathcal{U} = \mathcal{A} F^\top$. Taking into account that $F^\top \phi(\mathbf{z}) = \kappa(X, \mathbf{z})$ and that the kernel function induces a $\mathbf{z} \to \phi(\mathbf{z})$ feature map, for the Rayleigh-quotient based kernel feature extraction scheme we have the following composite mapping

$$\mathbf{z} \to \mathcal{A} \kappa(X, \mathbf{z}). \tag{4.14}$$

Having obtained the uniform framework, we can now proceed with the derivation of the 'kernelized' methods, one after the other.

4.2 Kernel Principal Component Analysis

After the Support Vector Machine [97], the second-best-known kernel algorithm is perhaps the Kernel Principal Component Analysis (Kernel-PCA) proposed by Schölkopf et al. [87; 88]. This method fits nicely into the unified scheme presented here. It has been employed in a wide range of practical problems such as handwritten digit recognition [65] and human face recognition. We also applied this method to speech technology applications [56; 62].

The following derivation of the method will use our unified scheme, and thus will differ slightly from the original one. However, the resulting formula will obviously be equivalent to the one proposed in the original derivation.

4.2.1 Derivation of the Method

Having chosen a proper (according to Theorem 2.1) κ kernel function for which

$$\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}), \qquad \mathbf{x}, \mathbf{z} \in \mathcal{X}, \tag{4.15}$$

holds for a mapping $\phi : \mathcal{X} \to \mathcal{F}$, we now give the PCA transformation in $\mathcal{F}$.

First, let us examine the form of the τ objective function of PCA (Eq. (3.3)) in the kernel feature space $\mathcal{F}$:

$$\tau(\boldsymbol{v}) = \frac{\boldsymbol{v}^\top \mathcal{C} \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}}, \tag{4.16}$$

where $\mathcal{C}$ is the covariance matrix of the sample $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)$:

$$\mathcal{C} = E\{(\phi(\mathbf{x}) - E\{\phi(\mathbf{x})\})\,(\phi(\mathbf{x}) - E\{\phi(\mathbf{x})\})^\top\}. \tag{4.17}$$

Much like the PCA approach, we define the Kernel-PCA transformation based on the stationary points of Eq. (4.16), which are given as the eigenvectors of the symmetric positive semidefinite matrix $\mathcal{C}$. However, since this matrix is of the form

$$\mathcal{C} = \sum_{i,j}^{k} [\Theta_1]_{ij} \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^\top = F \Theta_1 F^\top, \tag{4.18}$$

we may suppose the following equation holds during the analysis of the stationary points (cf. Proposition 4.1):

$$\boldsymbol{v} = \sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}_i). \tag{4.19}$$

Based on the above assumption the variational parameters of τ can be the vector $\boldsymbol{\alpha}$ instead of $\boldsymbol{v}$:

$$\tau(\boldsymbol{\alpha}) = \frac{\left(\sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}_i)^\top\right) \mathcal{C} \left(\sum_{j=1}^{k} \alpha_j \phi(\mathbf{x}_j)\right)}{\left(\sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}_i)^\top\right) \left(\sum_{j=1}^{k} \alpha_j \phi(\mathbf{x}_j)\right)}. \tag{4.20}$$

In the following proposition we derive Eq. (4.20) making use of the kernel matrix.

Proposition 4.2 *The τ function of Kernel-PCA is of the form*

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top \frac{1}{k} K (I - \hat{I}) K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K \boldsymbol{\alpha}}, \tag{4.21}$$

where $[K]_{ij} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is the Kernel matrix and

$$\hat{I} = \frac{1}{k} \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}. \tag{4.22}$$

Having defined τ, we proceed with the derivation of its stationary points. The stationary points of Eq. (4.21) are the solution vectors of the general eigenvalue problem

$$\frac{1}{k}K(I-\hat{I})K\boldsymbol{\alpha}=\lambda K\boldsymbol{\alpha}. \tag{4.23}$$

However, to find solution of Eq. (4.23) we may solve the eigenvalue problem

$$\frac{1}{k}(I-\hat{I})K\boldsymbol{\alpha}=\lambda\boldsymbol{\alpha} \tag{4.24}$$

for nonzero eigenvalues. Although the matrix $(I-\hat{I})K$ is not symmetric, its eigenvalues are real and non-negative, and those eigenvectors that correspond to positive eigenvalues are orthogonal. It is straightforward to see since, if we left-multiply Eq. (4.24) by $\hat{I}$

$$\frac{1}{k}\hat{I}(I-\hat{I})K\boldsymbol{\alpha}=\lambda\hat{I}\boldsymbol{\alpha} \tag{4.25}$$

and, taking into account the fact that $\hat{I}\hat{I}=\hat{I}$ and $\hat{I}(I-\hat{I})K=(\hat{I}-\hat{I})K$ is the zero matrix, we have that $\hat{I}\boldsymbol{\alpha}=\mathbf{0}$. This means that (4.24) is equivalent to the problem

$$\frac{1}{k}(I-\hat{I})K(I-\hat{I})\boldsymbol{\alpha}=\lambda\boldsymbol{\alpha}. \tag{4.26}$$

for solutions with nonzero eigenvalues. Since the matrix $\frac{1}{k}(I-\hat{I})K(I-\hat{I})$ is positive semidefinite the eigenvalues are non-negative and the eigenvectors are orthogonal.

In fact the best approach for solving Eq. (4.24) is to solve the symmetric eigenproblem defined in Eq. (4.26). We note that we would have arrived at the same eigenproblem had we assumed that

$$\boldsymbol{v}=\sum_{i=1}^{k}\alpha_i\left(\phi(\mathbf{x}_i)-E\{\phi(\mathbf{x})\}\right) \tag{4.27}$$

(as B. Schölkopf et al. did) instead of Eq. (4.19). With this assumption we would have obtained the function

$$\tau(\boldsymbol{\alpha})=\frac{\boldsymbol{\alpha}^\top\frac{1}{k}(I-\hat{I})K(I-\hat{I})(I-\hat{I})K(I-\hat{I})\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top(I-\hat{I})K(I-\hat{I})\boldsymbol{\alpha}}. \tag{4.28}$$

Now let the m positive dominant eigenvalues of Eq. (4.26) be denoted by $\lambda_1\geq\ldots\geq\lambda_m$ and the corresponding eigenvectors be $\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m$. Then the matrix $\mathcal{A}$ of the non-linear transformation in Eq. (4.14) can be calculated as:

$$\mathcal{A}=(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m)^\top. \tag{4.29}$$

Transformation of test vectors. For an arbitrary test vector $\mathbf{z}\in\mathcal{X}$ the Kernel-PCA transformation can be done using

$$\mathbf{z}^*=\mathcal{A}F^\top\phi(\mathbf{z})=\mathcal{A}\kappa(X,\mathbf{z}). \tag{4.30}$$

Of course, if it is desired the norm of the $\boldsymbol{\alpha}$ eigenvectors can easily be chosen such that the two-norm (i.e. Euclidean norm) of the column vectors of $\mathcal{A}F^\top$ becomes 1.

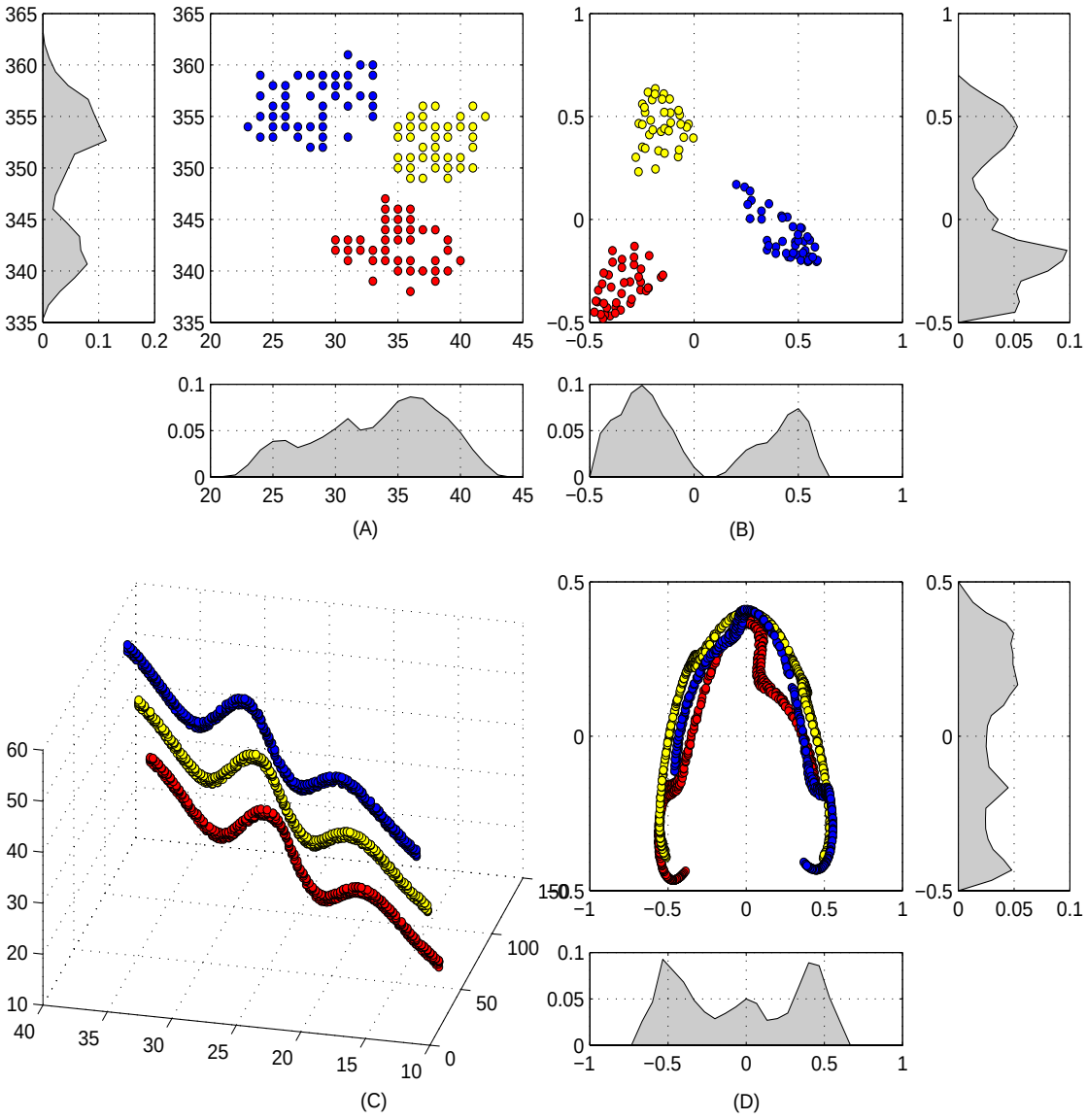

Figure 4.1: The effect of Kernel-PCA on 2D and 3D data sets. The Kernel-PCA transformed versions of the artificial data sets (A) and (C) are shown in Figs. (B) and (D). Besides the data sets themselves their distributions along the axes have also been represented in visual form.

4.2.2 2D and 3D Examples

Figure 4.1 shows the Kernel-PCA transform of a 2D and a 3D data set, respectively. Fig. (B) shows what shape the distribution of the data set of Fig. (A) took after transformation. It can be easily seen that the transform is indeed non-linear. The distributions along both the x and y-axis confirm the fact that, we obtained large-variance directions with Kernel-PCA that could not have been achieved with a linear method. The transformation seems beneficial from a classification point of view as well.

Now let us examine the 3D data set of Fig. (C), along with its transformed version in Fig. (D). We again obtain directions corresponding to large variances. In this case, however, the transformation is absolutely unhelpful for a subsequent classification. It suggests that a dimension reduction by Kernel-PCA should be performed with caution.

4.3 Kernel Independent Component Analysis

The idea of non-linearizing Independent Component Analysis was proposed quite early, with the aim of non-linearly separating mixed signals [76; 77; 100]. We first mentioned applying the 'kernel trick' to the non-linearization of ICA in [56], and discussed the Kernel-ICA method in detail in [59]. Possibly in parallel or perhaps later and in a different context, other ICA algorithms were kernelized as well [5; 39]. In this thesis we follow our own derivation to present the Kernel-ICA algorithm.

4.3.1 Derivation of the Method

In this section we derive the kernel counterpart of *FastICA* [59]. To this end, let the inner product be implicitly defined by the kernel function κ in $\mathcal{F}$ with associated transformation ϕ. As we saw earlier, the FastICA algorithm consists of two main blocks: centering & whitening, and the subsequent approximate Newton algorithm. In these we will extend non-linearly only the centering and whitening procedure of the data, since afterwards we get uncorrelated data in the kernel feature space $\mathcal{F}$ in a non-linear way. However, although it could be done,[1] the second, iterative part of FastICA will not be non-linearized here.

Let us first examine how the centering and whitening preprocessing steps can be performed in the kernel feature space.

Centering in $\mathcal{F}$. We shift the data $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)$ with its mean $E\{\phi(\mathbf{x})\}$, to obtain the data

$$\begin{aligned} \phi'(\mathbf{x}_1) &= \phi(\mathbf{x}_1) - E\{\phi(\mathbf{x})\} \\ &\vdots \\ \phi'(\mathbf{x}_k) &= \phi(\mathbf{x}_k) - E\{\phi(\mathbf{x})\} \end{aligned} \tag{4.31}$$

with a mean of $\mathbf{0}$.

Whitening in $\mathcal{F}$. Much like that in linear ICA, the goal of this step is to find a transformation matrix Q such that the covariance matrix

$$\hat{\mathcal{C}} = \frac{1}{k} \sum_{i=1}^{k} \hat{\phi}(\mathbf{x}_i) \hat{\phi}(\mathbf{x}_i)^\top \tag{4.32}$$

of the sample

$$\begin{aligned} \hat{\phi}(\mathbf{x}_1) &= Q\phi'(\mathbf{x}_1) \\ &\vdots \\ \hat{\phi}(\mathbf{x}_k) &= Q\phi'(\mathbf{x}_k) \end{aligned} \tag{4.33}$$

is a unit matrix. Since standard principal component analysis [49] – just like its kernel-based counterpart – transforms the covariance matrix into a diagonal form, where the

[1] Obviously Eq. (3.9) could be very easily non-linearized using kernels, as the formula contains only one dot product, $\mathbf{v}^\top \mathbf{z}$. We chose to disregard this step because it did not fit into our Rayleigh quotient based non-linearization approach.

diagonal elements are the eigenvalues of the data covariance matrix

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \phi'(\mathbf{x}_i)\phi'(\mathbf{x}_i)^\top, \tag{4.34}$$

it only remains to transform each diagonal element to 1. Based on this observation, the required whitening transformation is obtained by slightly modifying the formulas presented in the section on Kernel-PCA. That is, based on the formula we got in Kernel-PCA (cf. Eq. (4.29)), the centering & whitening transformation matrix can be defined by

$$\begin{aligned} Q &= \mathcal{A}F^\top, \\ \mathcal{A} &= (\lambda_1^{-1/2}\boldsymbol{\alpha}_1, \ldots, \lambda_m^{-1/2}\boldsymbol{\alpha}_m)^\top, \\ F &= (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)), \end{aligned} \tag{4.35}$$

where $(\boldsymbol{\alpha}_1, \lambda_1), \ldots, (\boldsymbol{\alpha}_m, \lambda_m)$ are the dominant m eigenpairs of Eq. (4.26). The preprocessing phase is again followed by the approximate Newton iteration that we already discussed when we presented the ICA algorithm.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-ICA transformation can be performed using

$$\mathbf{z}^* = WQ\phi(\mathbf{z}) = W\mathcal{A}\kappa(X, \mathbf{z}). \tag{4.36}$$

Here W denotes the orthogonal transformation matrix we obtained as the output from the iterative section of FastICA[2], while Q is the matrix obtained from kernel centering & whitening. Practically speaking, here Kernel-FastICA = Kernel-Centering + Kernel-Whitening + iterative section of the original FastICA.

4.3.2 2D and 3D Examples

Figure 4.2 demonstrates the effect of Kernel-ICA on artificial data sets. The 2D set of Fig. (A) is the same as the one used to test Kernel-PCA. The result of the transformation is shown in Fig. (B). As we can see, the direction found to have a distribution very different from Gaussian and chosen as the new x-axis is very good from a classification point of view, despite the fact that the method is unsupervised. Fig. (C) is the example of three toruses with the result of the transforms shown in Fig. (D). When comparing the results with those of Fig. 3.2 – that is, linear ICA – the difference owing to the non-linear nature of the feature extraction is obvious. However, after observing the distribution of the data sets along the x-axis, we should remark that the separability of the classes is now somewhat worse than that obtained in the linear case.

Similar to the linear case, performing a radical dimension reduction is recommended only after the approximate Newton iteration and not immediately after kernel centering & whitening. For the dimension reduction it is worth ordering the directions in accordance with the corresponding non-Gaussianity, and keeping only those directions with the m largest values. For example, in Fig. (D) we retained only two directions of the three, according to this scheme.

[2]Matlab code is available in [23].

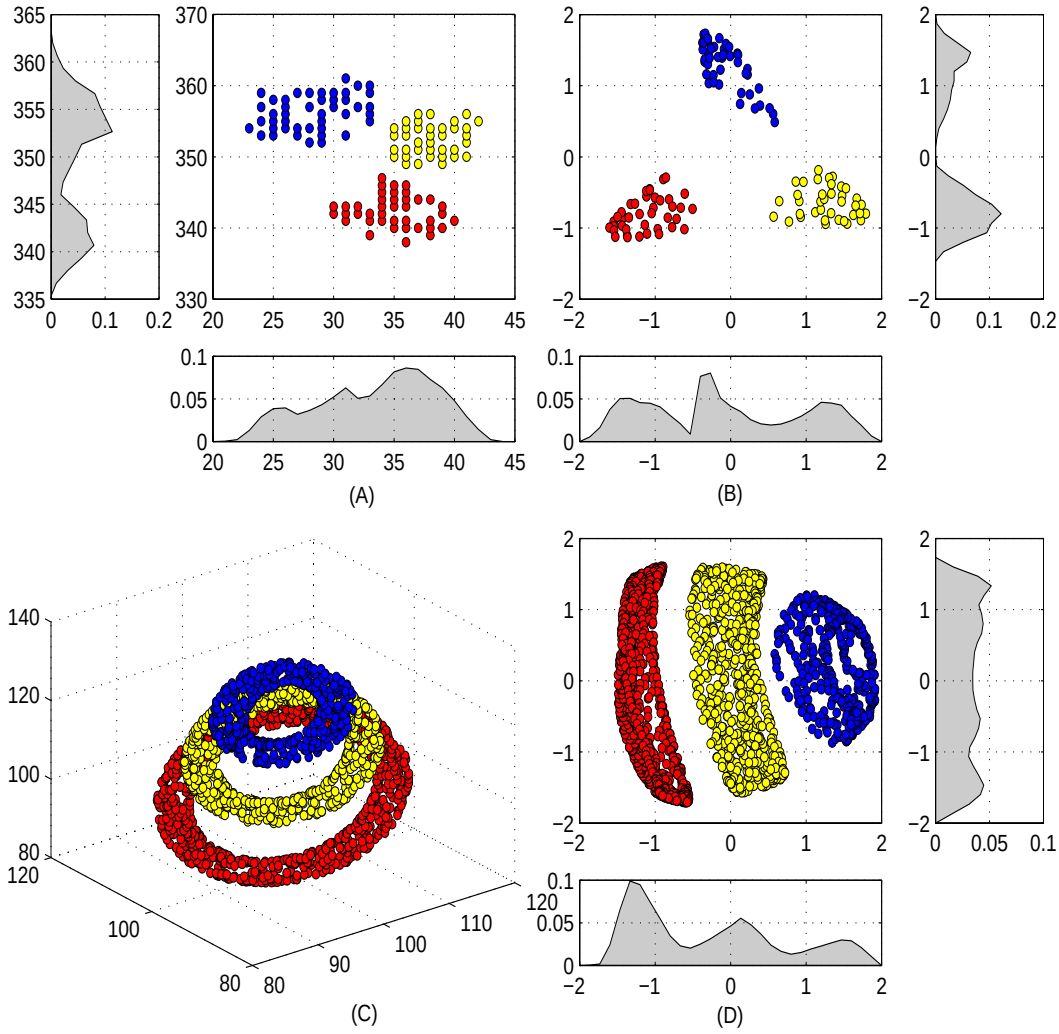

Figure 4.2: The effect of Kernel-ICA on 2D and 3D data sets. The artificial data sets of Fig. (A) and (C) were transformed with Kernel-ICA, and the resulting distributions are shown in Figs. (B) and (D), respectively. Besides the data, their distribution along the axes are depicted as well.

4.4 Kernel Linear Discriminant Analysis

The 'kernelized' counterpart of Linear Discriminant Analysis, the Kernel-LDA is a supervised feature extraction method very suitable for classification. There are many authors who have studied this method in parallel. For the two class case S. Mika et al. were the first to present a formulation [66]; the same was later introduced by S. A. Billings [10]. We proposed the multi-class version of the method to be used for phoneme recognition [95], publishing the technical details only a little later [58]. The multi-class version, with minor modifications, can also be found in [7; 84]. In addition several authors [30; 99] have reported that Kernel-LDA is practically equivalent to the LSSVM method elaborated by J. A. K. Suykens et al. [93].

We will now continue with the derivation of Kernel-LDA, of course following our unified scheme.

4.4.1 Derivation of the Method

Let us consider the following function for a fixed kernel function κ, a kernel map ϕ and a kernel feature space $\mathcal{F}$.

$$\tau(\boldsymbol{v}) = \frac{\boldsymbol{v}^\top \mathcal{B} \boldsymbol{v}}{\boldsymbol{v}^\top \mathcal{W} \boldsymbol{v}}, \tag{4.37}$$

where the matrices needed for LDA (see Eqs. (3.11) and (3.12)) are now given in $\mathcal{F}$:

$$\begin{aligned}
\mathcal{B} &= \sum_{j=1}^{r} \frac{k_j}{k} (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^\top \\
\mathcal{W} &= \sum_{j=1}^{r} \frac{k_j}{k} \mathcal{C}_j \\
\boldsymbol{\mu} &= \tfrac{1}{k} \sum_{i=1}^{k} \phi(\mathbf{x}_i) \\
\boldsymbol{\mu}_j &= \tfrac{1}{k_j} \sum_{\mathcal{L}(i)=j} \phi(\mathbf{x}_i) \\
\mathcal{C}_j &= \tfrac{1}{k_j} \sum_{\mathcal{L}(i)=j} (\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j)(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j)^\top
\end{aligned} \tag{4.38}$$

We suppose without loss of generality here that $\boldsymbol{v} = \sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}_i)$ holds during the search for the stationary points of Eq. (4.37). With this assumption, after some algebraic rearrangement we obtain the following.

Proposition 4.3 *The $\tau(\boldsymbol{\alpha})$ function of Kernel-LDA has the form:*

$$\frac{\boldsymbol{\alpha}^\top K (R - \hat{I}) K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K (I - R) K \boldsymbol{\alpha}}, \tag{4.39}$$

where K is the kernel matrix, $\hat{I}$ is defined in Proposition 4.2 and

$$[R]_{ij} = \begin{cases} \frac{1}{k_t} & \text{if } t = \mathcal{L}(i) = \mathcal{L}(j) \\ 0 & otherwise. \end{cases} \tag{4.40}$$

This means that Eq. (4.37) can be expressed as dot products of $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)$ and that the stationary points of this equation can be computed using the real eigenvectors of $(K(I-R)K)^{-1}K(R-\hat{I})K$. Since $K(I-R)K$ is in general a positive semidefinite matrix, it can be forced to be invertible using the technique presented in the section on LDA. For defining the transformation matrix $\mathcal{A}$ of Kernel-LDA we will use only those eigenvectors which correspond to the m dominant real eigenvalues, denoted by $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-LDA transformation can be performed using $\mathbf{z}^* = \mathcal{A} F^\top \phi(\mathbf{z}) = \mathcal{A} \kappa(X, \mathbf{z})$.

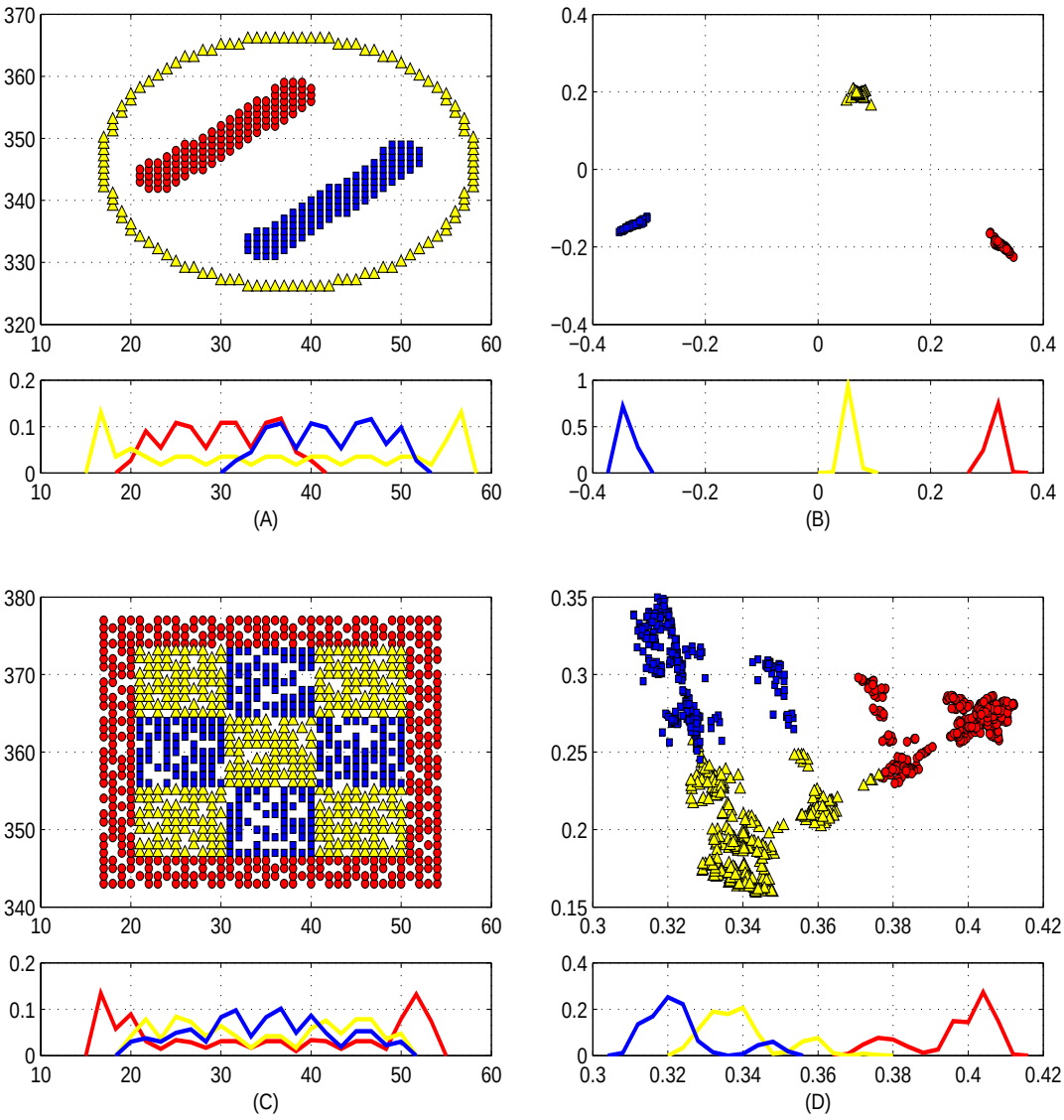

Figure 4.3: The effect of Kernel-LDA on 2D data sets. Figs. (A) and (C) show artificial 2D input sets. The result of Kernel-LDA applied on these sets are depicted in Figs. (B) and (D), respectively.

4.4.2 2D and 3D Examples

To examine how Kernel-LDA affects the distribution of data sets let us analyze Fig. 4.3. The 3-class sets of Figs. (A) and (C) were transformed with Kernel-LDA, and the results are shown in Figs. (B) and (D). It can be clearly seen that in Figs. (A) and (C) separating the classes linearly is no longer possible. However, in Fig. (B) the classes are perfectly separable along the x-axis. In accordance with the Kernel-LDA criterion, the variance of each class is small, while the distance among the classes is large. Similar observations can be made regarding Fig. (D). Once again the separability of the classes is surprisingly good.

We can reasonably conclude that applying the non-linear approach made the LDA criterion even more effective for the separation of the classes.

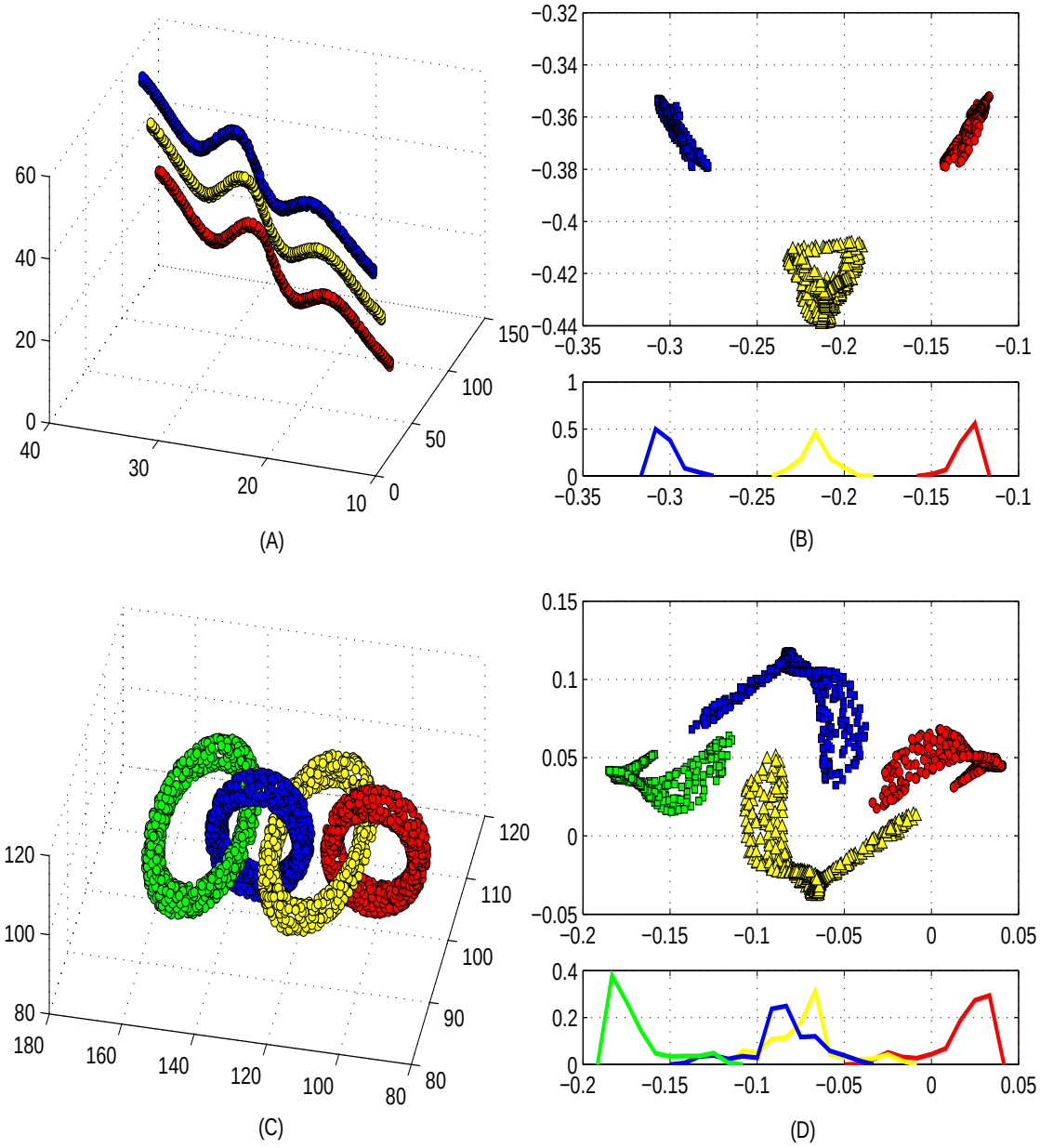

Figure 4.4: The effect of Kernel-LDA on 3D data sets. Figs. (A) and (C) depict 3-dimensional data sets, and their distribution after Kernel-LDA in 2 dimensions is shown in Figs. (B) and (D), respectively.

The behavior of Kernel-LDA on 3-dimensional artificial data is shown in Fig. 4.4. The data set of Fig. (A) consists of the usual 'snake-like parallel curves'. The result of using the Kernel-LDA method, depicted in Fig. (B) is similar to those of the 2-D examples. The example of Fig. (C) contains 4 classes in the form of 4 intertwined toruses. As illustrated in Fig. (D), Kernel-LDA is able to non-linearly separate them. These 3D examples justify our claim that Kernel-LDA is very effective in class separation.

Based on these 2-dimensional and 3-dimensional examples we can expect that, with the help of Kernel-LDA, even more sophisticated topological constructions may be readily separable.

4.5 Kernel Springy Discriminant Analysis

The non-linear version of Springy Discriminant Analysis (Kernel-SDA) was invented with goals very similar to those of Kernel-LDA. We first published the method in [60], which was followed by a summary paper containing Kernel-SDA together with the seven algorithms discussed up to this point.

Following our unified framework, we now transform the formulas of linear SDA into ones for the kernel feature space.

4.5.1 Derivation of the Method

Now let the dot product be implicitly defined by the kernel function κ in some finite or infinite dimensional feature space $\mathcal{F}$ with associated transformation ϕ:

$$\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}), \qquad \mathbf{x}, \mathbf{z} \in \mathcal{F}. \tag{4.41}$$

Let $\delta(\boldsymbol{v})$, the potential of the spring model along the direction $\boldsymbol{v}$ in $\mathcal{F}$, be defined by

$$\sum_{i,j=1}^{k} ((\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top \boldsymbol{v})^2 [M]_{ij}, \quad \boldsymbol{v} \in \mathcal{F} \tag{4.42}$$

where

$$[M]_{ij} = \begin{cases} -1, & \text{if } \mathcal{L}(i) = \mathcal{L}(j) \\ 1, & \text{otherwise} \end{cases} \qquad i, j = 1, \ldots, k. \tag{4.43}$$

Naturally, different M settings may also be reasonable when defining δ, just as we saw in the case of the linear version.

In space $\mathcal{F}$ the τ function takes the form

$$\tau(\boldsymbol{v}) = \frac{\delta(\boldsymbol{v})}{\boldsymbol{v}^\top \boldsymbol{v}} = \frac{\boldsymbol{v}^\top \mathcal{D} \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}}, \tag{4.44}$$

where

$$\mathcal{D} = \sum_{i,j=1}^{k} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top [M]_{ij}. \tag{4.45}$$

Technically speaking, with the above τ definition Kernel-SDA searches for those directions $\boldsymbol{v}$ along which a large potential is obtained.

Let us now examine the stationary points of $\tau(\boldsymbol{v})$. Without loss of generality we may assume that vector $\boldsymbol{v}$ has the form

$$\boldsymbol{v} = \sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}_i) = F^\top \boldsymbol{\alpha} \tag{4.46}$$

With this assumption we will consider the form of $\tau(\boldsymbol{v})$ where the variable is $\boldsymbol{\alpha}$. In the following proposition we will express Eq. (4.44) as a function of $\boldsymbol{\alpha}$.

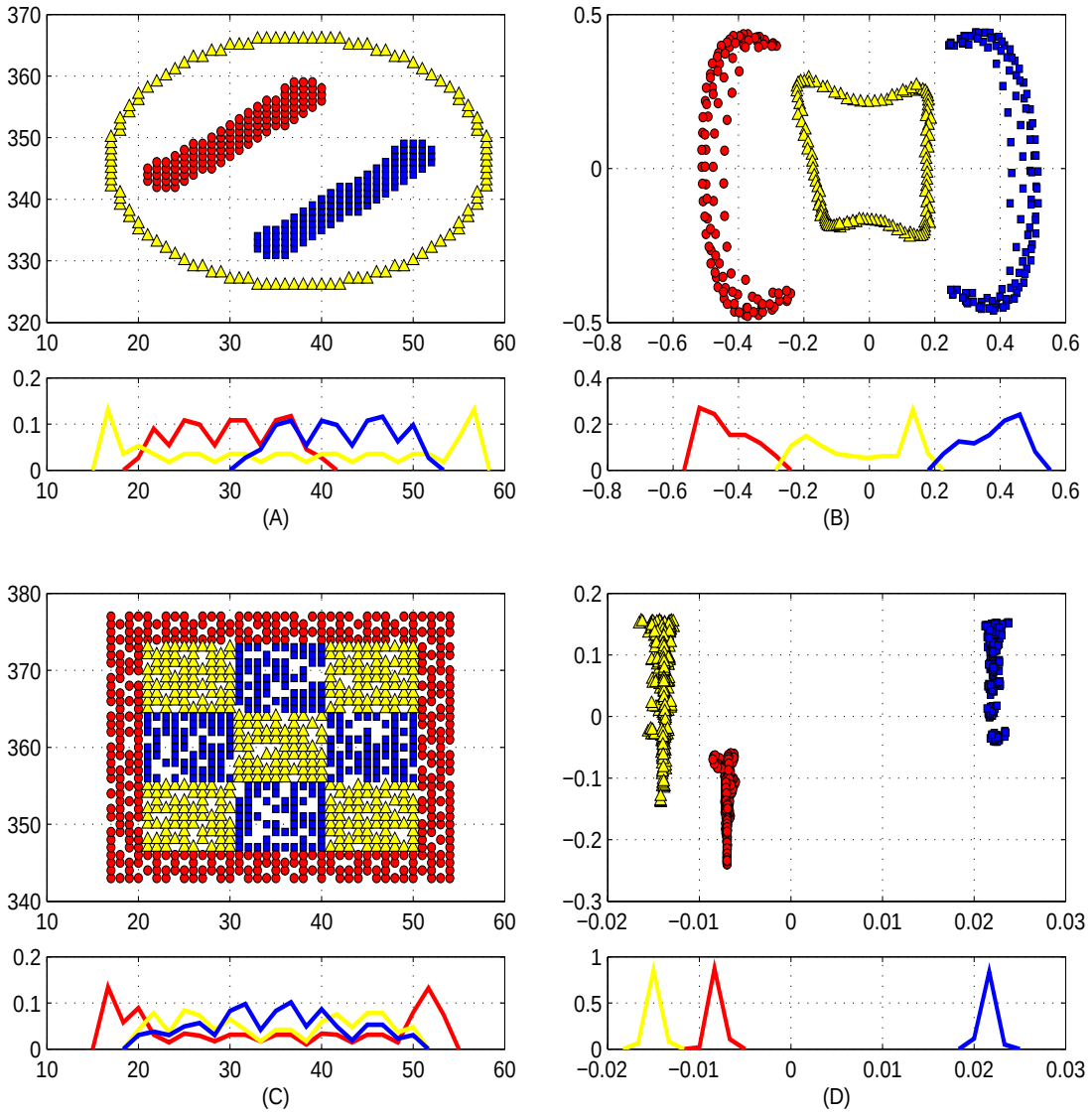

Figure 4.5: The effect of Kernel-SDA on 2D data sets. Figs. (A) and (C) depict 2-dimensional data sets. Their distribution after Kernel-SDA in 2 dimensions are shown in Figs. (B) and (D), respectively.

Proposition 4.4 *The Rayleigh quotient for Kernel-SDA has the form:*

$$\tau(\boldsymbol{\alpha}) = 2\frac{\boldsymbol{\alpha}^\top K(\tilde{M} - M)K^\top \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K \boldsymbol{\alpha}}, \tag{4.47}$$

where K is the kernel matrix and $\tilde{M}$ is a diagonal matrix with the sum of each row of M in the diagonal.

As we have already seen the stationary points of $\tau(\boldsymbol{\alpha})$ can be obtained via an eigen-analysis of the following generalized eigenproblem:

$$(K(\tilde{M} - M)K^\top)\boldsymbol{\alpha} = \lambda K \boldsymbol{\alpha}. \tag{4.48}$$

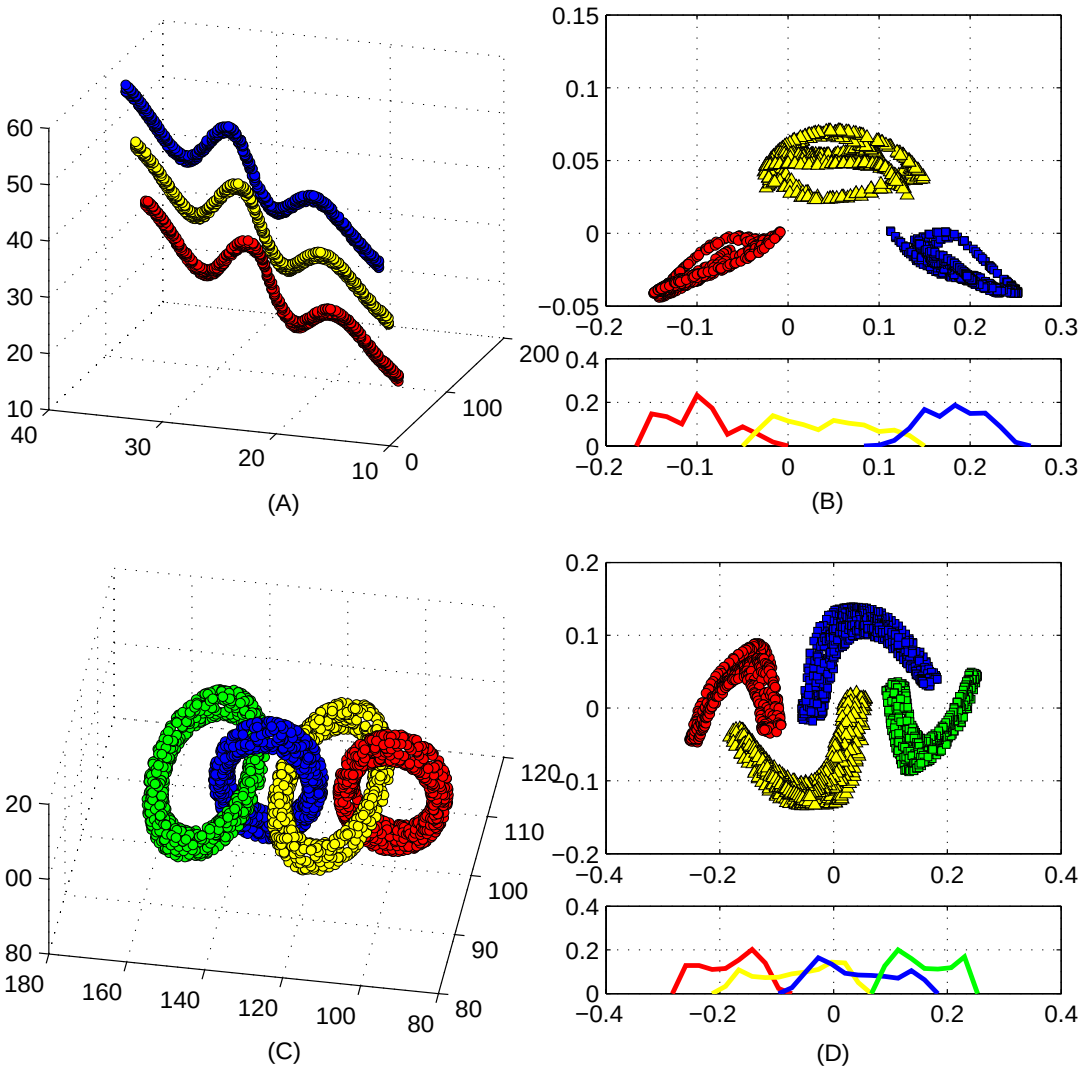

Figure 4.6: The effect of Kernel-SDA on 3D data sets. Figs. (A) and (C) depict 3-dimensional data sets. Their distributions after Kernel-SDA in 2 dimensions are shown in Figs. (B) and (D), respectively.

If we assume that the dominant m eigenvectors are $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_m$ then the transformation matrix $\mathcal{A}$ is defined by $(\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_m)^\top$.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-SDA transformation can be performed using $\mathbf{z}^* = \mathcal{A}F^\top \phi(\mathbf{z}) = \mathcal{A}\kappa(X, \mathbf{z})$.

Finally, we remark that if we had not insisted on using the form Eq. (4.44) of the Rayleigh quotient and had defined $\tau(\boldsymbol{\alpha})$ as $\delta(F^\top\boldsymbol{\alpha})/||\boldsymbol{\alpha}||^2$ instead of $\delta(F^\top\boldsymbol{\alpha})/||F^\top\boldsymbol{\alpha}||^2$), we would have arrived at the relatively simple symmetric eigenproblem

$$(K(\tilde{M} - M)K^\top)\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha} \tag{4.49}$$

instead of a generalized eigenvalue problem. As in the case of the Kernel-SDA the numerator contains information about both shrinking the classes and moving them apart, hence we have the option to modify the denominator if we wish.

4.5.2 2D and 3D Examples

To test Kernel-SDA we employed the same data sets as those used for the testing of Kernel-LDA. The results in the 2D case are shown in Fig. 4.5, and in the 3D case of the 3D data in Fig. 4.6. One can see that the transformation was almost as effective as that for Kernel-LDA. The only difference is that the Kernel-SDA approach emphasized the partitioning of the classes rather than shrinking their data points. Of course, this behavior can be adjusted by a proper redefinition of the force constant matrix M. This flexibility is a great advantage of Kernel-SDA and could be exploited here.

4.6 Reducing the Computational Cost

As we have already seen, all four methods lead to a (generalized) eigenproblem that involves finding the stationary points of the objective function $\tau(\boldsymbol{v})$, defined in the form of a Rayleigh quotient. During optimalization, the vector $\boldsymbol{v}$ consists of the linear combinations of the images of the data points X in the kernel feature space. Without doubt, if the amount of data points (k) is large, the $k \times k$ sized matrices that are needed for constructing $\tau(\boldsymbol{v})$ – hence for solving the eigenproblem – can be so big that they pose serious computational and memory management problems.

Fortunately, in most practical problems good $\boldsymbol{v}$ directions can be found even if we use only $q << k$ data points instead of k when constructing the linear combinations. Let us denote the indices of these r samples by $1 \leq i_1 < \cdots < i_q \leq k$. It is easy to check that by just using these data items the formulas we obtain for the function $\tau(\boldsymbol{\alpha})$ can be expressed as the following:

$$\text{KPCA, KICA:} \qquad \frac{\boldsymbol{\alpha}^\top \frac{1}{k} K_1^\top (I - \hat{I}) K_1 \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K_2 \boldsymbol{\alpha}}, \tag{4.50}$$

$$\text{KLDA:} \qquad \frac{\boldsymbol{\alpha}^\top K_1^\top (R - \hat{I}) K_1 \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K_1^\top (I - R) K_1 \boldsymbol{\alpha}}, \tag{4.51}$$

$$\text{KSDA:} \qquad \frac{\boldsymbol{\alpha}^\top \left(K_1^\top (2\tilde{M} - 2M) K_1 \right) \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K_2 \boldsymbol{\alpha}}, \tag{4.52}$$

where $\boldsymbol{\alpha}$ is a vector of dimension q, K_1 is the matrix constructed from the columns $i_1, \cdots, i_q$ of the kernel matrix K, and K_2 is the minor matrix determined by the rows and columns of K with indices $i_1, \cdots, i_q$. Based on these formulas, the eigenproblems to be solved are now reduced to a matrix of size $q \times q$. In practice, this matrix usually has no more than a couple of dozen or a couple of hundred rows and columns.

Of course, a key issue here is the strategy for choosing the q indices. Numerous selection strategies are possible from the random selection to the exhaustive search approach. At the time of writing we are working on the kernel counterparts of some standard feature selection methods suitable for dimension reduction. The discussion of this techniques is, however, out of the scope of this thesis.

4.7 Summary

This chapter presented the non-linear counterparts of the linear methods of the previous chapter, obtained with the help of the kernel idea. For the non-linearization we exploited the fact that the Rayleigh quotients defining the linear methods all have a special form (see Eq. (4.8)). This allowed us to derive the corresponding counterparts of these functions in the kernel feature space $\mathcal{F}$ so that they depend only implicitly – via the kernel functions – on the points of $\mathcal{F}$ (see Eq. (4.10)).

We also demonstrated the efficiency of the methods on 2D and 3D artificial data sets. The following table summarizes what the methods focus on during non-linear feature extraction.

Kernel-PCA concentrates on those non-linear directions along which the variance of the data set is large.

Kernel-ICA searches non-linearly for those directions which are independent, and along which the distribution of the data significantly differs from a Gaussian one.

Kernel-LDA performs non-linear feature extraction with the aim of class separation. The data in different classes are pushed apart while the data points belonging to the same class are pulled together.

Kernel-SDA non-linearly maps the initial feature space, and in the space obtained it seeks to separate classes just as Kernel-LDA does, but by means of defining attractive and repulsive forces.

Part II

Speech Technology Applications

Chapter 5

Speech Recognition

Automatic speech recognition is a special pattern classification problem which aims to mimic the perception and processing of speech in humans. For this reason it clearly belongs to the fields of machine learning and artificial intelligence. For historical reasons, however, it has mostly been ranked as a sub-field of electrical engineering, with its own unique technologies, conferences and journals. In the last two decades the dominant method for speech recognition has been the hidden Markov modeling approach. Since then the theory of machine learning has developed considerably and now has a wide variety of learning and classification algorithms for pattern recognition problems. The main goal of this chapter is to study the applicability of some of these methods to phoneme classification.

In essence, this chapter deals with the linear (PCA, ICA, LDA, SDA) and kernel-based feature extraction methods (Kernel-PCA, Kernel-ICA, Kernel-LDA, Kernel-SDA), described in Chapters 3 and 4, applied prior to learning in order to improve classification rates. Their effect on classification performance is tested in combination with classifiers like the IB1 algorithm (TiMBL), ID3 tree learning (C4.5), oblique tree learning (OC1), artificial neural nets (ANN) and Gaussian mixture modeling (GMM). We compare these methods with a traditional hidden Markov phoneme model (HMM), working with the linear prediction-based cepstral coefficient features (LPCC). The empirical issues described in this chapter are based on our three experimental studies [56; 57; 62].

5.1 The Task of Phoneme Classification

Speech recognition is a pattern classification problem in which a continuously varying signal has to be mapped to a string of symbols (the phonetic transcription). Speech signals display so many variations that attempts to build *knowledge-based speech recognizers* have mostly been abandoned. Currently researchers tackle speech recognition only with statistical pattern recognition techniques. Here, however a number of special problems arise that have to be dealt with. The first one is the question of the recognition unit. The basis of the statistical approach is the assumption that we have a finite set of units (in other words, classes), the distribution of which is modelled statistically

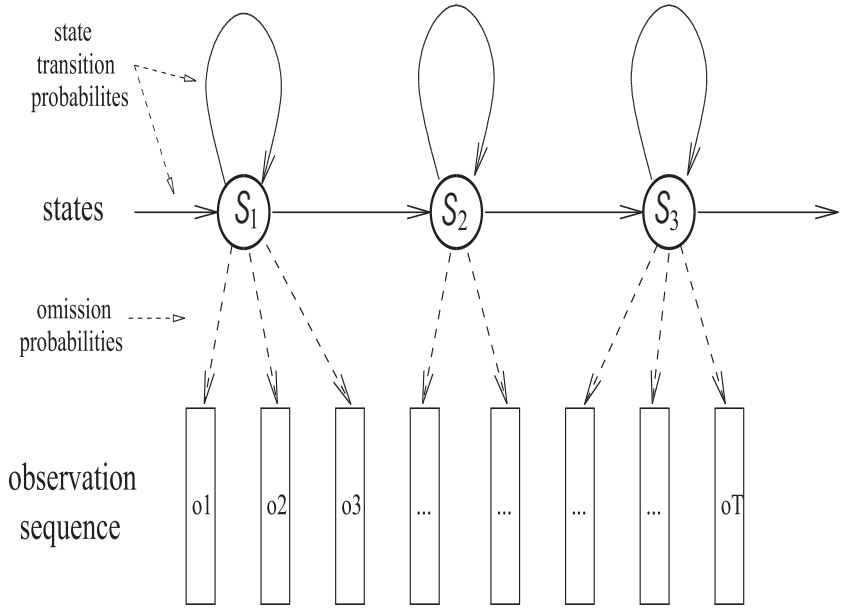

Figure 5.1: The three-state left-to-right phoneme HMM.

from a large set of training examples. During recognition an unknown input is classified as one of these units using some kind of similarity measure. Since the number of possible sentences or even words is 'potentially infinite', some sort of smaller recognition units have to be chosen in a general speech recognition task. The most commonly used unit of this kind is the phoneme, hence this chapter deals with the classification problem of phonemes.

The other special problem is that the length of the units may vary, that is utterances get warped along the time axis. The only known way of solving this is to perform a search in order to locate the most probable mapping between the signal and the possible transcriptions. Normally a depth-first search is applied (implemented with dynamic programming), but a breadth-first search with a good heuristic is a viable option as well.

5.2 Phoneme Modeling

Hidden Markov Models [83] synchronously handle both the problems mentioned above. Each phoneme in the speech signal is given as a series of observation vectors

$$O = \mathbf{o}_1, \ldots, \mathbf{o}_T, \tag{5.1}$$

and one has one model for each unit of recognition. These models eventually return a class-conditional likelihood $P(O|c)$, where c refers to these units. The models are composed of states, and for each state we model the probability that a given observation vector belongs to ("was omitted by") this state. Time warping is handled by state transition probabilities, that is the probability that a certain state follows the given state. The final 'global' probability is obtained as the product of the proper omission

and state-transition probabilities.

When applied to phoneme recognition, the most common state topology is the three-state left-to-right model (see Fig. 5.1). We use three states s_1, s_2 and s_3 because the first and last parts of a phoneme are usually different from the middle due to coarticulation. This means that in a sense we do not really model phonemes but rather phoneme thirds.

Because the observation vectors usually have continuous values the state omission probabilities have to be modeled as multidimensional likelihoods. The usual procedure is to employ a mixture of weighted Gaussian distributions for all state s_j of the form

$$p(\sigma|s_j) = \sum_{i=1}^{k} \alpha_i \mathcal{N}(\sigma, \boldsymbol{\mu}_i, \Sigma_i), \tag{5.2}$$

where $\mathcal{N}(\sigma, \boldsymbol{\mu}_i, \Sigma_i)$ denotes the multidimensional normal distribution with mean $\boldsymbol{\mu}_i$ and covariance matrix Σ_i, k is the number of mixtures, and α_i are non-negative weighting factors which sum to 1.

A possible alternative to HMM are the Stochastic Segmental Models. The more sophisticated segmental techniques fit parametric curves to the feature trajectories of the phonemes [74]. There is, however, a much simpler methodology [33; 54; 55] that applies non-uniform smoothing and sampling in order to parametrize any phoneme with the same number of features, independent of its length. The advantage of this uniform parametrization is that it allows us to apply any sort of machine learning algorithm for the phoneme classification task. This is why we chose this type of segmental modeling for the experiments performed and also for our speech recognition system [95].

Hidden Markov Models describe the class conditional likelihoods $P(O|c)$. These type of models are called *generative*, because they model the probability that an observation O was generated by a class c. However, the final goal of classification is to find the most probable class c. We can readily compute the posterior probabilities $P(c|O)$ from $P(O|c)$ using Bayes' law since

$$P(c|O) = \frac{P(O|c)P(c)}{P(O)}. \tag{5.3}$$

Another approach is to model the posteriors directly. This is how *discriminative* learners work. Instead of describing the distribution of the classes, these methods model the surfaces that separate the classes and usually perform slightly better than generative models.

In this chapter in the phoneme classification tests we work with both generative and discriminative methods. But before coming to the point, we shall briefly introduce the OASIS speech recognition system [54; 55], developed at the Research Group on Artificial Intelligence, which served as a framework for all the tests.

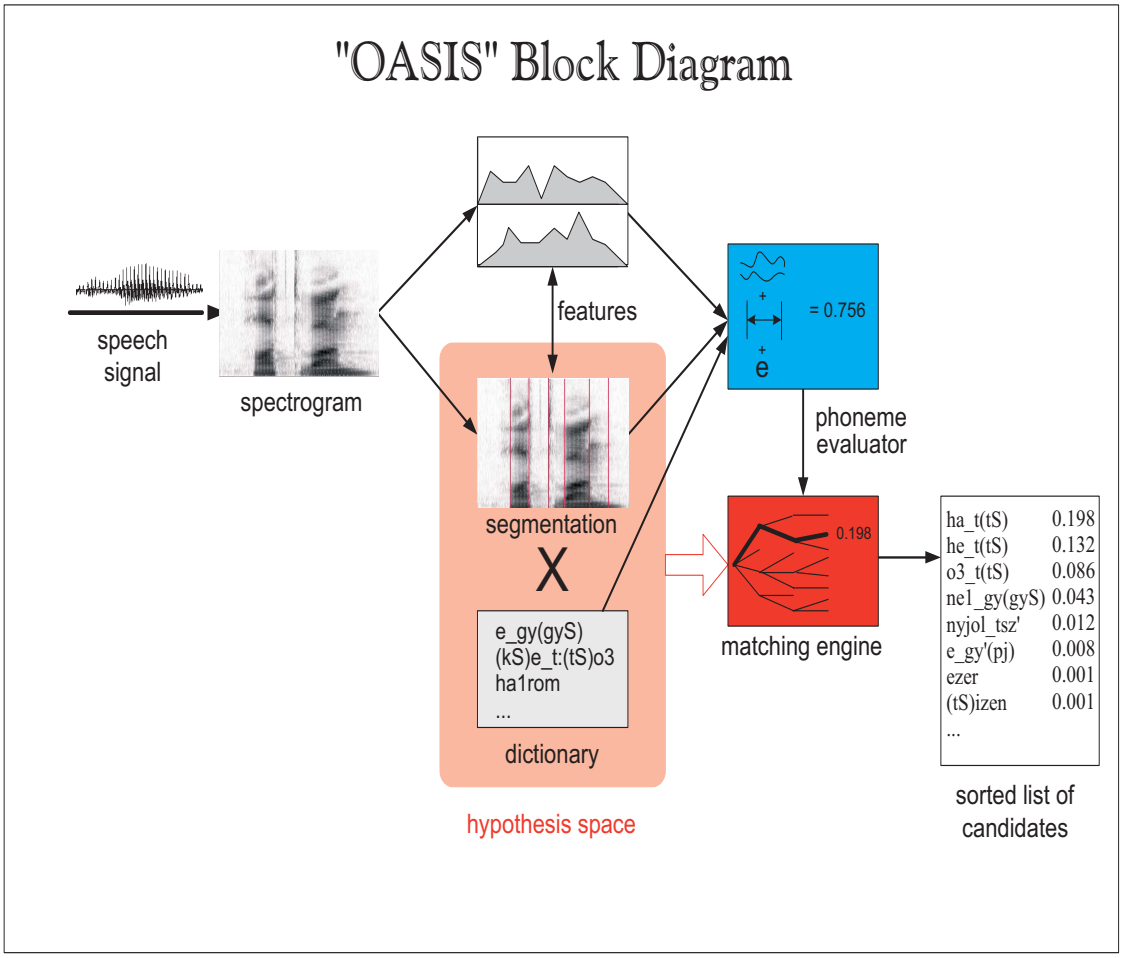

Figure 5.2: Block diagram for the OASIS speech recognizer.

5.3 The Oasis System

When choosing the directions of our speech recognition research, we decided to focus on Hungarian with the hope that we could address some special issues concerning the processing of our national language, and also that we could make use of our previous experience with NLP (Natural Language Processing) for Hungarian. In addition, we were looking for a flexible framework that allowed experimentation with different preprocessing techniques, feature-space transformation methods and machine learning algorithms. These expectations led us to the stochastic segmental approach which, in a certain sense, can be viewed as an extension of hidden Markov modeling. Our recognition system, OASIS[1], was designed to be as modular as possible, so we can easily conduct experiments with combining different techniques for the several subtasks of recognition. The first module is the usual frame-based preprocessing phase of the speech signal. After this an additional step, the modeling of phonetic segments was inserted. This allows the combination of different preprocessing techniques and also

[1]The acronym comes from "**O**ur **A**coustics-based **S**peaker-**I**ndependent **S**peech recognizer".

the incorporation of phonetic knowledge. The output of this module are the so-called segmental features, which are used by the phoneme classifier for the classification of a segment. For this task, again several possible learning algorithms can be applied. Finally, the segmental classification scores are combined by the matching engine, which performs an utterance-level search in the graph of the possible segmentations. Its goal is to find the best matching transcription from those provided by the language model. The system is flexible enough for use in trying out many search techniques and strategies. The main steps of the recognition process are illustrated in the block diagram (see Fig. 5.2).

5.4 Phoneme Classification Results

Now we continue with a description of the experiments. First, we describe the evaluation domain. Then we present the initial frame-based features and how these are converted into segmental features. This is followed by an elaboration of how the feature extraction techniques of Chapters 3 and 4 were applied along with a brief explanation of the learning algorithms used. Finally we discuss the test settings, the results and, of course, their evaluation.

5.4.1 Evaluation Domain

The classification techniques combined with feature extraction methods were compared using a relatively small corpus which consists of several speakers pronouncing Hungarian numbers. More precisely, 20 speakers were used for training and 6 for testing, and 52 utterances were recorded from each person. The ratio of male and female talkers was 50%-50% in both the training and testing sets. The recordings were made using a cheap commercial microphone in a reasonably quiet environment, at a sample rate of 22050Hz. The whole corpus was manually segmented and labelled. Since the corpus contained only numbers, we had occurrences of only 32 phones, which is approximately two thirds of the Hungarian phoneme set. Since some of these labels represented only allophonic variations of the same phoneme, some labels were fused, and so we actually worked with a set of 28 labels. We made tests as well with two other groupings where the labels were grouped into 11 and 5 classes, respectively, based on phonetic similarity. We had two good reasons for doing experiments with these gross phonetic classes. Firstly, this way we could increase the number of training examples in each class and monitor the effects of this on the learning algorithms. Secondly, our speech recognition system has a first-pass procedure, where the segments are classified into gross phonetic categories only.

Hence we had three phonetic groupings, which henceforth will be denoted by *grp1*, *grp2* and *grp3*. With the first grouping the number of occurrences of the different labels in the training set was between 40 and 599. This value was between 120 and 1158 for the second, and between 716 and 2158 for the third grouping.

5.4.2 Acoustic Features

In the following we describe the initial feature extraction techniques which were employed in our tests. For each test a certain subset of these features was chosen. The only exception was the recognizer, which used its own traditional features (for details see 5.4.5).

Critical Band Log-Energies (CBLE)

Before the initial feature extraction the energy of each word was normalized. After this the signals were processed in 512-point frames (23.2 ms), where the frames overlapped by a factor of 3/4. A Fast Fourier Transform was applied on each frame. After that critical band energies were approximated by the use of triangular-shaped weighting functions. 24 such filters were used to cover the frequency range from 0 to 11025Hz, the bandwidth of each filter being 1 bark. The energy values were then measured on a logarithmic scale.

Mel-Frequency Cepstral Coefficients (MFCC)

In order to incorporate the most common preprocessing method, that is *MFCC* into our features, we made additional tests after taking the discrete cosine transform *(DCT)* of the critical band log-energies calculated above. The test used the first 16 coefficients (including the zeroth one).

A point which should be mentioned here is that since the spectrum has already been smoothed by the critical band filters, the calculation of the cepstrum does not fulfil its original task of removing the effect of pitch. Instead, its supposed benefit is the decorrelation of features. In fact, it can be proved that the *DCT* approximates the *PCA* quite closely for most signals [3], so it is worth comparing the classification results for *MFCC* with critical band log-energies plus *PCA*.

Formant Band Gravity Centers (FBGC)

Besides the above ones we also wanted to do experiments with some more phonetically based features like formants. However, since we had no reliable formant tracker (which is known to be a very difficult task anyway), we instead used gravity centers as a crude approximation for formants [4]. The gravity centers were calculated from the power spectrum in the following four frequency bands:

$$\begin{array}{c} [200\text{Hz}, 1400\text{Hz}], \\ [1000\text{Hz}, 3000\text{Hz}], \\ [2500\text{Hz}, 4000\text{Hz}], \\ [3000\text{Hz}, 11025\text{Hz}]. \end{array} \tag{5.4}$$

The formula for the gravity center $\mathcal{G}(a,b)$ of a band [a,b] is

$$\mathcal{G}(a,b) = \frac{\int_a^b fS(f)df}{\int_a^b S(f)df}, \tag{5.5}$$

where $S(f)$ denotes the power spectrum.

Since the gravity centers can give misleading values at parts which have no clear formant structure (e.g. silence), the "spreading ratio" $\mathcal{D}(a,b)$ of the gravity center was employed as a kind of measure for the strength or reliability of the formant. We defined this as the ratio of the deviance of the spectrum in intensity $\mathcal{D}_i(a,b)$ and frequency $\mathcal{D}_j(a,b)$. That is,

$$\mathcal{D}(a,b) = \frac{\mathcal{D}_i(a,b)}{\mathcal{D}_f(a,b)} = \frac{\sqrt{\frac{1}{b-a}\int_a^b S^2(f)df - \left(\frac{1}{b-a}\int_a^b S(f)df\right)^2}}{\sqrt{\frac{\int_a^b f^2S(f)df}{\int_a^b S(f)df} - \mathcal{G}^2(a,b)}}. \tag{5.6}$$

Thus the four gravity centers and four spreading ratios gave eight additional features.

5.4.3 Segmental Features

From Frame-Based to Segmental Measurements

Although *HMM* is the most widely used technology for speech recognition, it is not without its critics. When it comes to phoneme modeling it has a number of deficiencies. The most serious criticisms levelled against *HMM* as a phoneme model (cf. also [75]) are that

- its duration modeling abilities are poor
- as it works with uniform-sized frames, it does not permit the incorporation of long-term (segmental) measurements
- it assumes that the frames which belong to a given state are independent

One possible way of overcoming these limitations is to work within a segmental framework like that of [33] or [74]. Since working with variable-sized segments instead of frames introduces many new problems, switching to such an approach requires very strong justification. One convincing proof would be if we found phoneme models which were not only intuitively more appealing, but also led to better classification results. Recently great efforts have been made to find good segmental phoneme models [26; 28; 32; 75].

Rather than employ these sophisticated techniques we decided to follow a very simple procedure, the idea having been taken from [37]. For each feature we took the average of the frame-based measurements for the first quarter, the central part and the last quarter of the phoneme. In this way we obtained $3n$ features for each phoneme, where n is the number of the features for one frame. Our reasons for modeling the segments this way were twofold. On one hand we needed to describe each phoneme with the same number of features otherwise the discriminative learners could not have been applied at all. On the other we wanted to use features which were very closely related to the original spectrum, and learn what the transformations being studied would produce from them.

Duration

In *HMM* the duration of the phonemes is modeled only implicitly: the usual 3-state left-to-right model remains in a state with an exponentially decaying probability, which is quite a poor approximation of how the length of phonemes (or rather, phoneme thirds) is distributed in the real world. In segmental models phonemic duration can be modeled explicitly, which we think is especially important in languages like Hungarian, where most of the phonemes have a "short" and a "long" version (that is, duration has a characteristic role). Our experiments showed that adding duration to the segmental feature set indeed increased classification accuracy, so duration was used in all of our experiments. However, our statistical measurements indicated that the duration of the phonemes has a huge scatter. This means that speaking rate normalization would be very beneficial for recognition.

5.4.4 Feature Extraction

Applying the linear and non-linear transformations of Chapter 3 and 4 we transform the segmental features, hoping for a better classification. In the case of PCA, ICA, SDA, Kernel-PCA, Kernel-ICA and Kernel-SDA the original feature space was reduced to 32 dimensions, while in the case of LDA and Kernel-LDA the number of features kept was always the number of classes minus one.

Naturally when we applied a certain transformation on the training set before learning, we applied the same transformation on the test data during testing.

5.4.5 Learning Methods

The following sections briefly present the applied generative and discriminative learning techniques along with their evaluation method in the experiments.

Applied Generative Learners

Hidden Markov Model (HMM)

Hidden Markov Modeling is currently the dominant technology in speech recognition [83]. This is why in the tests the *HMM* was trained on its "standard" features and not on those utilized in all the other experiments. The intention behind this was to have a reference point for the current state-of-the-art technology, something to judge things by.

The hidden Markov models for the experiments were trained using the *FlexiVoice* speech engine [94]. The system used a feature vector of 34 components, which consisted of 16 lpc-derived cepstrum coefficients plus the frame energy, and the first derivatives of these. The frame size was 30 msec while the step size was 10 msec.

One model was assigned to each of the phonemes, that is 28, 11 and 5 models were trained for the groupings *grp1*, *grp2* and *grp3*, respectively. The phoneme models were of the 3-state strictly left-to-right type, that is each state had one self transition and one transition to the next state. In each case the observations were modeled using a mixture of 4 Gaussians with diagonal covariance matrices. The models were trained using the Viterbi training algorithm [40].

Gaussian Mixture Model (GMM)

Gaussian Mixture Modeling [21] assumes that the class-conditional probability distribution $p(\mathbf{x}|c)$ can be well approximated by a distribution of the form

$$f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i \mathcal{N}(\mathbf{x}, \boldsymbol{\mu}_i, \Sigma_i), \tag{5.7}$$

where $\mathcal{N}(\mathbf{x}, \boldsymbol{\mu}_i, \Sigma_i)$ denotes the multidimensional normal distribution with mean $\boldsymbol{\mu}_i$ and covariance matrix Σ_i, k is the number of mixtures and α_i are non-negative weighting factors which sum to 1. This is virtually the same as the state omission formula of HMM (see Eq. (5.2)), but in this case it is used to parametrize a whole phoneme and not frame-based observation vectors, i.e. $\mathbf{x}$ represents an element of the feature space describing phonemes.

Unfortunately there is no closed formula for the optimal parameters of the mixture model, so normally the expectation maximization (*EM*) algorithm is used to find proper parameters, but it guarantees only a locally optimal solution. This iterative technique is very sensitive to initial parameter values, so we used k-means clustering [83] to find a good starting parameter set. Since k-means clustering again guaranteed finding only a local optimum, we ran it 15 times with random parameters and used the one with the highest log-likelihood to initialize the *EM* algorithm. After experimenting, the best value for the number of mixtures k was found to be 3, 7 and 20 for the three groupings used in the tests.

In all cases the covariance matrices were forced to be diagonal, since training full matrices would have required much more training data (and computation time as well).

Applied Discriminative Learners

C4.5

C4.5 [82] is based on the well-known *ID3* tree learning algorithm. It is able to learn pre-defined discrete classes from labelled examples. The result of the learning process is an axis-parallel decision tree. This means that, during the training, the sample space is divided into subspaces by hyperplanes that are parallel to every axis but one. In this way we get many n-dimensional rectangular regions that are labelled with class labels and organized in a hierarchical way, which can then be encoded into the tree. Since *C4.5* considers attribute vectors as points in an n-dimensional space, using continuous-valued sample attributes naturally makes sense.

Although the learning is quite fast, the results are often unsatisfactory. This is mainly due to two reasons. Firstly, *C4.5* uses the "divide and conquer" technique, which means that regions are split during learning whenever they are insufficiently homogeneous. Under certain circumstances this strategy results in a huge number of regions that are needlessly split. The second reason is the knowledge representation itself – it would seem that axis parallel decision trees are not flexible enough for phoneme recognition.

Three main cases should be mentioned where *C4.5* apparently faces serious difficulties. There are

- **Non-rectangular regions.** Even with reasonable feature space transformations the phoneme classes are found in non-rectangular regions. To achieve the desired accuracy *C4.5* should increase the number of regions without limit, but some reduction is inevitable because of time and space considerations. No matter how carefully it is performed, the reduction of tree size increases the misclassification rate.

- **Poorly separated regions.** When the algorithm divides the search space, its goal is to create near-homogeneous regions. In addition, early splits determine the direction towards which the whole procedure progresses. However if the samples are scattered or noisy (e.g. classes are distributed randomly along certain axes) there is scarce guidance for the initial divisions. The overall result is a set of numerous regions (in other words large decision trees) with a substantial number of misclassifications, whereas only a few well-placed regions would assure the same accuracy.

- **Fragmented regions.** Irrelevant attributes force *C4.5* to create too many regions as the examples become mixed along one or more axes. The consequence of this are errors similar to those mentioned above.

Despite the problems outlined here, *C4.5* achieved good results when it had to decide from among only a few, well separated classes. The greatest advantage of the method is time complexity. In the worst case it is $O(dn^2)$, where d is the number of features and n is the number of samples.

OC1

The *OC1* (Oblique Classifier 1) algorithm [70] learns by creating *oblique* decision trees. The advantages and drawbacks are similar to those of *C4.5*, although in many cases *OC1* produces better results. Having more freedom when splitting regions not surprisingly increases accuracy and decreases the tree size. Then some non-rectangular regions become efficiently learnable.

OC1 chooses oblique splits through its perturbation algorithm by performing random jumps. There are two consequences of this. One is that it eliminates the problem of early splits being so crucial. The other is that it assures an efficient algorithm, the time complexity being of $O(dn^2 \log n)$. This is only $\log n$ times more than that seen in *C4.5*.

TiMBL

TiMBL [17] is a Memory Based Learner which means a new example is evaluated by consuming the previous examples stored in the memory. Since no rule or decision is made before the actual classification, this approach is called *lazy learning*. Typically, this kind of machine learning has a very short training time but the classification of new data takes rather a long time. The storing and processing of millions of examples can also be a serious handicap.

TiMBL is based on *IB1*, which is a version of the k-Nearest Neighbour algorithm with a special difference metric. *TiMBL* has many advanced tools to get the most out of the *k-NN* approach. Information theory is applied to both the difference metrics and attribute weighting. Measuring the information gain ratio of different attributes yields valuable information about useful information-rich and information-poor attributes as well. In this way irrelevant features can be skipped over. Due to the underlying strategy, however, redundant features evidently run the risk of being overweighted, which may corrupt the classification by excessively dominating the metric.

TiMBL uses a tree storing model as a solution for the time and space problems mentioned previously. Training samples are stored in a tree-like manner to decrease both computational time and the memory required for data storage.

Although the results given by *TiMBL* are satisfactory, there are obvious reasons why we cannot suppose a better classification. One of the principal problems is that *IB1* is designed to run on discrete features. Generally speaking, it can deal with numerical features only by discretising them.

Artificial Neural Networks (ANN)

Artificial Neural Networks [91] now count among the conventional pattern recognition tools, so we will not describe them here. In the experiments we employed the most common feed-forward multilayer perceptron network with the backpropagation learning rule. The number of neurons in the hidden layer was set to be three times the number of features (i.e. the number of input neurons). Training was stopped when, for the last 20 iterations, the decrease in the error between two consecutive iteration steps stayed below a certain threshold.

Evaluation Method

The task of pattern classification is to map a given feature vector $\mathbf{x}$ to one of the classes. Some of the standard machine learning algorithms we tried (*C4.5*, *OC1*, *TiMBL*) do just this, that is they return a label (i.e. a class) for each test vector. With these methods the calculation of the recognition rate is quite straightforward.

The learning methods which model the a posteriori probabilities $P(c|\mathbf{x})$ return a probability value for each class c given a test vector $\mathbf{x}$. The so-called Bayes's decision rule states [91] that the optimal way of converting these values to a class label is to choose the class with the maximum a posteriori probability. We used this rule to calculate the classification error for ANN.

Finally, other learning techniques such as *HMM* and *GMM* model the class-conditional probabilities $P(\mathbf{x}|c)$. From this $P(c|\mathbf{x})$ can be obtained by employing the rule

$$P(c|\mathbf{x}) = \frac{P(\mathbf{x}|c)P(c)}{P(\mathbf{x})}. \tag{5.8}$$

Thus, according to the Bayes decision rule, we have to choose that class for which $P(\mathbf{x}|c)P(c)$ is maximal. ($P(\mathbf{x})$ is independent of c and so can be omitted.) Instead of doing this we did not multiply by the factor $P(c)$ in the evaluation, since handling this probability calculation traditionally belongs to the language model. Moreover, preliminary tests showed that multiplication with $P(c)$ led only to marginal improvements, clearly because the relative frequencies of the phonemes were quite well balanced.

5.4.6 Experimental Results

The experiments were performed on five feature sets as described below. As all sets contained duration, we do not mention them separately. *Set1* consisted of the *MFCC* coefficients as these are the most commonly used features. To have the opportunity of studying the importance of the cosine transform we also conducted tests on the filter bank energies themselves *(Set3)*. By augmenting *Set1* and *Set3* with the gravity center features we acquired two further sets *Set2* and *Set4*. We expected the addition of these phonetics-based features to lead to a slight improvement. Lastly, the largest set (*Set5*) contained all the features, that is filter bank energies, *MFCC* coefficients and gravity centers. Our interest was in seeing whether the transformations could effectively select the important ones, and in finding out whether combining all the features would confuse the learning algorithms.

The same experiments were carried out on the three phoneme groupings *grp1*, *grp2*, *grp3*, all the learning methods being tested not just on each set but with each transformation technique. The only exception was *HMM*, which we used as a comparison using the current "standard" technique, so it used its own feature extraction method (see 5.4.5) but, of course, with the same training and test corpus.

The tables below depict the recognition accuracies for *grp1*, *grp2* and *grp3*, respectively. The columns show the five feature sets and the rows correspond to the transformation methods (including "none") applied. The numbers in the diagonal correspond to the recognition accuracies of *TiMBL*, *C4.5*, *OC1*, *ANN* and *GMM*, respectively.

TiMBL C4.5 OC1 ANN GMM	MFCC *(Set1)*	MFCC+FBGC *(Set2)*	CBLE *(Set3)*	CBLE+FBGC *(Set4)*	all *(Set5)*
PCA	75.29 52.30 60.64 83.45 72.22	72.22 50.60 55.79 84.81 73.88	80.37 60.90 64.83 85.82 71.93	77.89 63.80 64.48 86.82 72.93	75.23 56.20 60.17 84.46 74.82
ICA	75.79 53.90 59.81 79.02 72.70	76.39 54.30 55.67 72.87 72.75	78.09 55.30 62.71 82.92 70.57	71.89 48.60 55.67 77.90 72.70	69.70 44.70 52.90 78.13 72.70
LDA	83.33 67.30 72.04 88.48 85.82	82.62 69.20 71.28 86.41 85.34	83.56 66.70 71.63 87.00 85.87	83.21 69.10 71.16 86.94 85.52	83.33 67.80 70.86 86.94 86.23
SDA	80.67 66.70 69.39 87.94 80.02	79.50 67.60 71.93 84.93 79.02	79.84 64.20 68.56 85.22 78.14	80.61 66.80 70.33 84.99 77.13	80.49 66.90 68.91 83.22 79.85
Kernel-PCA	81.41 55.50 63.43 82.94 82.94	77.42 58.40 68.28 84.89 79.27	83.59 64.00 71.06 83.76 85.24	81.59 62.80 70.35 86.13 84.36	80.42 63.20 70.76 84.65 83.82
Kernel-ICA	80.81 55.00 67.69 83.06 82.05	82.23 58.70 63.61 82.05 75.49	84.33 61.00 69.82 78.74 80.93	77.61 61.40 68.16 76.14 77.09	77.11 61.10 67.22 82.76 77.38
Kernel-LDA	88.78 81.50 76.38 89.44 88.26	90.26 83.60 78.68 90.15 88.38	90.38 80.80 77.74 90.86 88.67	89.85 83.10 80.22 89.97 88.55	89.32 83.80 75.55 90.86 89.20
Kernel-SDA	86.60 77.80 74.54 86.19 86.54	87.48 77.80 74.84 87.02 83.65	87.96 78.60 77.68 86.66 85.60	87.60 76.90 74.96 86.19 83.17	88.07 78.90 78.56 87.67 82.94

Table 5.1: Recognition accuracies for *Grp1* (28 phonemes). *HMM* scored 90.66% for this grouping.

TiMBL C4.5 OC1 ANN GMM	MFCC *(Set1)*	MFCC+FBGC *(Set2)*	CBLE *(Set3)*	CBLE+FBGC *(Set4)*	all *(Set5)*
PCA	81.14 57.60 68.91 86.76 84.04	80.37 63.10 68.44 87.41 78.66	86.17 73.90 77.25 89.78 75.77	86.46 75.20 77.48 90.96 80.73	82.74 69.30 74.41 90.60 80.91
ICA	79.13 67.00 72.70 85.46 84.46	80.26 67.20 71.39 78.96 78.37	83.09 72.20 80.50 89.13 79.96	77.18 64.40 71.45 84.34 83.10	76.83 61.00 70.04 85.05 83.45
LDA	84.81 79.80 86.11 90.78 88.89	85.57 84.00 85.87 90.07 86.88	83.03 81.00 82.74 90.25 87.29	86.17 82.40 84.40 88.83 87.12	86.11 83.00 85.22 89.78 87.12
SDA	86.46 80.60 79.26 90.25 83.63	87.29 79.80 82.98 88.89 82.15	84.81 79.30 78.01 88.30 82.51	86.05 80.50 80.44 89.48 85.99	84.21 79.60 81.56 89.18 84.04
Kernel-PCA	83.18 63.80 76.47 86.75 90.77	85.31 69.30 75.52 88.76 84.21	88.50 77.50 83.50 91.72 85.45	86.73 74.90 82.97 91.60 89.12	85.13 75.80 81.55 90.95 88.41
Kernel-ICA	84.51 68.00 74.10 87.70 90.65	86.10 69.70 74.34 88.17 82.61	87.92 75.90 78.12 86.87 88.11	81.91 77.40 79.90 90.24 85.51	83.68 75.30 78.95 89.23 86.99
Kernel-LDA	92.66 88.10 87.22 90.59 91.42	90.82 89.80 88.58 91.01 90.18	93.25 90.10 88.17 93.78 90.71	93.96 92.50 91.54 94.73 92.13	93.96 92.20 88.58 92.84 91.89
Kernel-SDA	91.24 88.10 87.40 90.30 89.88	90.47 89.80 84.74 90.83 90.30	93.01 90.70 87.93 93.31 92.19	93.01 91.60 87.46 93.49 89.59	93.13 90.40 87.22 92.48 86.87

Table 5.2: Recognition accuracies for *Grp2* (11 phonetic categories). *HMM* scored 95.27% for this grouping.

TiMBL C4.5 OC1 ANN GMM	MFCC *(Set1)*	MFCC+FBGC *(Set2)*	CBLE *(Set3)*	CBLE+FBGC *(Set4)*	all *(Set5)*
PCA	84.75 70.40 82.15 92.43 90.78	84.75 74.70 81.80 92.67 86.11	90.24 85.40 87.23 93.68 84.34	87.64 83.90 88.30 93.44 89.60	88.17 79.10 86.88 93.09 90.13
ICA	84.45 73.80 82.57 91.37 89.13	83.35 78.20 82.74 85.76 82.92	87.41 83.00 84.99 93.03 85.05	86.46 78.60 81.74 87.71 85.40	86.99 76.40 81.66 91.13 88.59
LDA	89.47 89.40 91.78 91.43 91.37	89.47 90.40 91.55 92.26 91.84	88.71 87.60 89.66 90.31 88.42	91.07 91.50 93.09 93.09 90.69	90.95 92.50 92.67 93.09 92.26
SDA	90.48 89.70 87.59 92.55 87.29	90.24 90.80 89.48 93.56 86.17	89.53 87.10 87.59 93.62 83.69	90.36 90.90 88.83 92.61 89.24	89.47 90.00 86.35 92.20 81.73
Kernel-PCA	87.64 74.90 83.51 92.37 92.96	88.23 81.70 86.58 92.78 89.42	91.48 84.80 90.66 95.80 88.53	91.42 85.60 88.71 95.09 93.32	89.59 83.10 89.59 94.68 92.55
Kernel-ICA	85.81 79.10 82.15 91.96 92.73	85.63 80.20 84.57 91.54 87.76	90.36 86.20 90.07 93.02 90.89	87.46 87.60 87.29 92.37 90.13	89.41 85.10 88.23 93.49 90.48
Kernel-LDA	95.32 93.60 93.85 95.15 93.20	94.26 95.30 94.32 94.44 95.39	92.01 89.90 90.54 95.39 92.31	93.90 93.60 94.32 93.20 93.26	95.62 94.30 95.39 94.68 93.61
Kernel-SDA	93.90 94.60 91.66 94.44 92.73	94.67 94.30 95.09 94.74 92.55	91.24 91.10 93.14 94.68 92.08	94.49 93.70 93.85 95.74 93.14	95.62 96.60 92.84 95.80 89.06

Table 5.3: Recognition accuracies for *Grp3* (5 gross phonetic categories). *HMM* scored 96.75% for this grouping.

Discussion

When inspecting the results the first thing one notices is that only the neural net could attain the efficiency of the *HMM*, all the other learners producing an error rate of 1.5 to 2 times bigger. We attribute this to our very simple segment model which used only the feature averages for the three pre-selected segment parts. Actually, the fact that the neural net could achieve the results of the *HMM* in spite of this drastic data reduction clearly showed the advantages of discriminative learning over generative types. The drawbacks of our primitive segment modeling becomes even more apparent when we compare the results of the *HMM* with those of the *GMM*, since *HMM* uses Gaussian mixtures too, but in a more refined way. The results of our comparison obviously indicate that we must look for a more sophisticated segment representation later on.

As for the other algorithms, from the results it seems that *C4.5* and *OC1* are quite unsuitable for the task of phoneme recognition, at least in this form. Nevertheless for *grp3* (the case of gross phonetic categories) they worked reasonably well and their fast learning may be a justification for their use in this case.

Finally *TiMBL* (namely, the *IB1* algorithm) worked quite well, and it appears that in the case of sparse training data it may turn out to work better than the parametric learners. Its major drawback, however, is its long evaluation time.

On examining the features the first thing to notice is that each learner performed the same or better with the filter bank energies (*Set1*) than with their corresponding cosine-transformed version (*Set3*). The only exception was *GMM* where the decorrelating effect of the *DCT* proved beneficial – but *PCA* was always performed much better.

Another thing we realized was that including the gravity center features did not bring about any general improvement. The main exception was *grp3*, where they helped in many cases. It seems that they are fairly useful in identifying gross phonetic categories, but not consistent enough for classifying phonemes.

With the feature set "all", we found that in general it was neither better nor worse than the other sets. It seems that while using all the features together did not confuse the learners, it could not significantly help them either.

As regards the feature extraction algorithms we reached the following overall conclusion. Independent of the learning method, the non-linear methods always outperformed their linear counterparts by some extent. As expected, the supervised methods were almost always better than the unsupervised ones. In most cases the dimension reduction inherent in the transforms was also beneficial or, at least, did not noticeably degrade performance. The feature space transformations, especially the supervised non-linear ones, lead to a significant increase in the classification performance of the weaker learners.

Finally we note that the goals of feature extraction and learning are practically the same. That is, if we have a very efficient learner then there is almost no need for feature extraction. Put the other way round, a proper transformation of the feature space may make the data so easily separable that quite simple learners will suffice. These are, of course, extreme examples. In practice this simply means that solving the problem in two steps by first transforming and then learning should usually prove to be the best approach.

5.5 Beyond the Phoneme Level

Up to this point we have been concerned only with phonetic classification. That is, in our experiments we supposed that the start and end points of the phonemes were known, and that the classifiers return a "hard-label". However, when using the phonetic classifiers in a speech recognizer the phonetic boundaries are not actually known, and the phoneme models are supposed to return probabilities. As regards the former problem – finding "the" correct phonetic segmentation of an utterance (if there is such thing at all) – remains unsolved. So if we insist on working with segments the best we can do is to try many possible segmentations, assign scores to them and pick the best one according to some evaluation criterion. If the evaluation means mapping probabilities to the segmentations, the best one means the most probable one – and we arrive at a probabilistic framework [33].

Other authors arrived at a segmental probabilistic structure in a different way. They had been trying to find models that overcome the limitations of the *HMM*, yet keep its advantages. These authors seek to present a mathematically unified framework where the *HMM* is just a special case [74]. A survey of these probabilistic segmental recognizers is beyond the scope of this work, but we suppose that our phonetic classifiers can be incorporated in such a recognition system. For a description of our segmental recognizer see [54] or [95].

The second problem we mentioned above is that the usual speech recognition frameworks (be they frame-based or segmental) expect the acoustic module to return probabilities and not "hard labels". This clearly does not cause a problem for the methods which model the class-conditional or the a posteriori probabilities, but we have to do something in the case of those non-parametric models which return only a class label. Some of these algorithms can be modified to return probabilities, but some of them cannot. In the latter case a possible solution is to train a set of classifiers on randomly chosen subsets of the training data, and approximate the probability of a class based on the votes of these. Although this seems plausible, we are unaware of any such theoretical study in the machine learning literature.

Although word recognition is beyond the scope of this thesis we should mention that when it comes to recognizing Hungarian numbers our OASIS speech recognition system performs very well indeed [95].

5.6 Summary

This chapter sought to study the effects of the linear and non-linear feature transformation methods of the previous chapters on phoneme classification, a basic task of speech recognition. During the tests the phoneme classification performance was studied as a function of the three main steps of statistical pattern recognition, that is initial feature extraction (acoustic/segmental features), feature space transformation (PCA, ICA, etc.) and the application of some machine learning algorithm. After viewing the test results we may confidently say that it is worth experimenting with feature transformations in order to obtain better classification results.

Chapter 6

Phonological Awareness Teaching

An important clue to the process of learning to read in alphabet-based languages is the ability to separate and identify consecutive sounds that make words and to associate these sounds with their corresponding written form [1; 85]. To learn to read in a fruitful way young learners must, of course, also be aware of the phonemes and be able to manipulate them. Many children with learning disabilities have problems in their ability to process phonological information. Furthermore, phonological awareness teaching has also great importance for the speech and hearing handicapped, along with developing the corresponding articulatory strategies of tongue movement.

In this chapter we study the application of automatic phoneme classification to the computer-aided training of the speech and hearing handicapped as well as to the learning of reading. Since a highly efficient automatic phoneme recognizer can make the teaching system more reliable, we decided to examine whether the feature extraction techniques could improve the recognition accuracy. The material in this chapter is based on results of the experimental studies we described in articles [58–60].

6.1 Introduction – The SpeechMaster

Before going into detail of the experiments performed, first we will give a concise overview of a real-world application which induced us to perform the tests discussed in this Chapter.

The "SpeechMaster" package (Fig. 6.1) seeks to apply speech recognition technology to speech therapy (Fig. 6.2) and the teaching of reading (Fig. 6.3), where the role of speech recognition is to provide visual phonetic feedback. In the first case it is intended to replace the missing auditive feedback of the hearing impaired, while in the latter case it is to reinforce the 'correct' association between the phoneme-grapheme pairs.

With the aid of a computer children can practice without the need for the continuous presence of the teacher. This is very important because the therapy of the hearing impaired requires a long and tedious fixation phase. Furthermore, our experiments and general observations show that young people are more willing to practice with the computer than with traditional drills. We found we could make impressive progress in

Figure 6.1: A screenshot of the start screen of the "SpeechMaster" software package. As the menu shows, the user can select one of the two main modules, the reading therapy – reading teaching one or the speech therapy one.

a very short training period.

The "SpeechMaster" program was designed after listening to the advice from teachers who had a lot of experience of reading teaching and/or the speech training of the hearing impaired. It contains exercises for each main phase of the methodologies traditionally employed in these areas. Since both groups of our intended users consist mostly of young children it was most important that the design of the software interface be made attractive and novel with interesting, playful graphics. Although the reading teaching part also has the usual letter-to-phoneme and word-to-letter association drills using images, the core of the software is the real-time phoneme recognition component. Here the pronounced phones immediately appear on the screen in the form of flickering letters, their identity and brightness being related to the speech recognizer's output.

We realized early on that the real-time visual feedback the software provides must be kept simple, otherwise the human eye cannot follow it. Basically this is why the output of a speech recognizer seems better suited to this goal than the usual method where only the short-time spectrum is displayed: a few flickering discrete symbols are much easier to follow than a spectrum curve, which requires further mental processing. This is especially the case with very young children.

Figures 6.2 and 6.3 show the user interface of "SpeechMaster" in the speech therapy and teaching reading applications, respectively. As one can see, in the first case the flickering letter is positioned over a web camera image (Fig. 6.2D) to help the impaired student learn the proper articulator positions while in the second case it is combined with a traditional picture for associating the word and word sound (Fig. 6.3A).

From the speech recognition point of view the need for a real-time output poses a number of special problems. Owing to the need for very fast classification we cannot

Figure 6.2: Some screenshots from the speech therapy part of the "SpeechMaster" software package. (A) – a vocalization drill, (B) – a drill for loudness control, (C) – a drill for pitch control, (D) – real-time phoneme recognition, (E) – learning the phoneme positions within words, (F) – a reading drill.

delay the response even until the end of phonemes, hence we cannot employ complicated long-term models. The algorithm should process no more than a few neighboring frames. Furthermore, since the program has to recognize vowels pronounced in isolation as well, a language model cannot be applied.

So that the real-time recognition performance can be improved the "SpeechMaster" package was designed to be able to use the output of the following machine learning algorithms: Artificial Neural Networks, Gaussian Mixture Models, Support Vector Machines, Projection Pursuit Learners and the family of feature extraction methods described in Chapters 2 and 3.

All these methods require training data as input, which in a speech recognition application means collecting a large amount of speech recordings. We carefully designed the contents of this speech corpus before making the recordings. To this end we studied the vocabulary of 13 reading textbooks used in the reading teaching of children. For

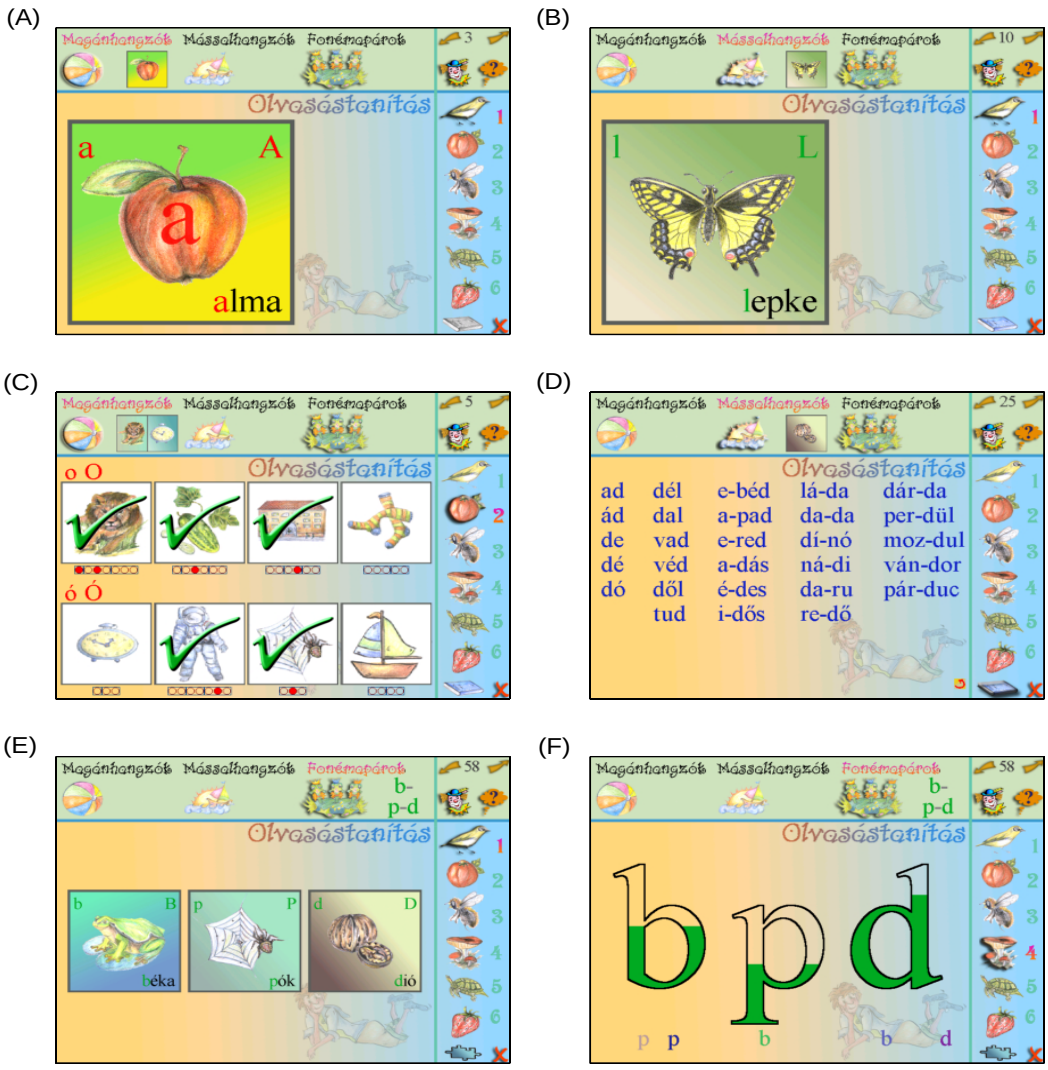

Figure 6.3: Some screenshots from the teaching reading part of the "SpeechMaster" software package. (A) – practicing vowels with the help of calling pictures, (B) – practicing consonants with the help of calling pictures, (C) – a drill to practice the positions of a letter within a word, (D) – a syllable reading exercise, (E-F) drills for excluding the rotation of letters.

the speech training of the hearing impaired, the vocabulary used in the speech drills of the deaf was collected and analyzed. This study resulted in a frequency dictionary for both the reading textbooks and the hearing impaired exercises. These lists influenced the selection of the words that were recorded for the speech corpus. Altogether 700 recordings were made which were phonetically segmented and labelled and formed the proper input for the machine learning algorithms.

The "SpeechMaster" program is not yet fully developed; it will be tested in several elementary schools, and in several schools for the hearing impaired all around in Hungary. On the basis of these tests, we will assemble the best methodology of how to use the software package in reading teaching and in improving the speech of the hearing impaired. These methodologies and, of course, the software itself will be freely available in 2004 so that all the teachers will have the opportunity to try it out for themselves.

6.2 Experimental Results

In this section we discuss the real-time phoneme recognition tests. We will talk about the extraction of the acoustic features, the way the transformations were applied, the learners we employed and, finally, about the setup and evaluation of the experiments.

6.2.1 Evaluation Domain

For training and testing purposes we recorded samples from 160 children aged between 6 and 8. The ratio of girls and boys was 50% - 50%.The speech signals were recorded and stored at a sampling rate of 22050Hz in 16-bit quality. Each speaker uttered all the Hungarian vowels, one after the other, separated by a short pause. Since we decided not to discriminate their long and short versions, we only worked with 9 vowels altogether. The recordings were divided into a train and a test set in a ratio of 50% - 50%.

6.2.2 Acoustic Features

There are numerous methods for obtaining representative feature vectors from speech data [43], but their common property is that they are all extracted from 20-30 ms chunks or "frames" of the signal in 5-10 ms time steps. The simplest possible feature set consists of the so-called bark-scaled filterbank log-energies (FBLE). This means that the signal is decomposed with a special filterbank and the energies in these filters are used to parameterize speech on a frame-by-frame basis. In our tests the filters were approximated via Fourier analysis with a triangular weighting, as described in [43]. Altogether 24 filters were necessary to cover the frequency range from 0 to 11025 Hz. Although the resulting log-energy values are usually sent through a cosine transform to obtain the well-known mel-frequency cepstral coefficients, we abandoned it for two reasons. First, the transforms we were going to apply have a similar decorrelating effect. Second, we observed earlier that the learners we work with - apart from GMM - are not sensitive to feature correlation so, consequently, the cosine transform would bring no significant improvement [57].

The filterbank log-energies seem to be a proper feature set for a general speech recognition task as their spectro-temporal modulation is supposed to carry all the speech information [69]. But in the special task of classifying vowels pronounced in isolation it is only the gross spectral shape that carries the phonetic information. More precisely, it is known from phonetics that the spectral peaks (called formants) code the identity of vowels [69]. To estimate the formants, we implemented a simple algorithm that calculates the gravity centers and the variance of the mass in certain frequency bands [4]. The frequency bands are chosen so that they cover the possible place of the first, second and third formants. This resulted in 6 new features altogether.

A more sophisticated option for the analysis of the spectral shape would be to apply some kind of auditory model [35]. Unfortunately, most of these models are too slow for a real-time application. For this reason we experimented with the In-Synchrony-Bands-Spectrum of Ghitza, because it is computationally simple and attempts to model the

dominance relations of the spectral components [31]. The model analyses the signal using a filterbank that is approximated by weighting the output of a FFT - quite similar to the FBLE analysis. In this case, however, the output is not the total energy of the filter, but the frequency of the component that has the maximal energy, and so dominates the given frequency band. Obviously, the output resulting from this analysis contains no information about the energies in the filters, but only about their relative dominance. Hence we supposed that this feature set complements the FBLE features in a certain sense.

6.2.3 Feature Extraction

When applying the feature extraction methods (see Chapters 3 and 4) we invariably kept only 8 of the new features. We performed this severe dimension reduction in order to show that, when combined with the transformations, the classifiers can yield the same scores in spite of the reduced feature set. To study the effects of non-linearity, the linear version of each transformation was also used on each feature set. Naturally, when we applied a certain transformation on the training set before learning, we applied the same transformation on the test data during testing.

6.2.4 Learners

Describing the mathematical background of the learning algorithms applied is beyond the scope of this work. Besides, we believe that they are familiar to those who are acquainted with pattern recognition. So in the following we specify only the parameters and the training algorithms employed with each learner, respectively.

Gaussian mixture modeling

The most widely used method for modeling the class-conditional (continuous) distribution of the features is to approximate it by means of a weighted sum of Gaussians [21]. In the GMM experiments, three Gaussian components were used and the expectation-maximization (EM) algorithm was initialized by k-means clustering [27]. To find a good starting parameter set we ran it 15 times and used the one with the highest log-likelihood. In every case the covariance matrices were forced to be diagonal.

Artificial neural networks

Since it was realized that, under proper conditions, ANNs can model the class posteriors [11], neural nets are becoming evermore popular in the speech recognition community. In the ANN experiments we used the most common feed-forward multilayer perceptron network with the backpropagation learning rule. The number of neurons in the hidden layer was set at 18 in each experiment (this value was chosen empirically, based on preliminary experiments). Training was stopped based on the cross-validation of 15% of the training data.

Projection Pursuit Learning

Projection pursuit learning is a relatively little-known modelling technique. It can be viewed as a neural net where the rigid sigmoid function is replaced by an interpolating polynomial. With this modification the representation power of the model is increased, so less units are necessary. Moreover, there is no need for additional hidden layers: one layer plus a second layer with linear combinations will suffice. During learning the model looks for directions in which the projection of the data points can be well approximated by its polynomials, thus the mean square error will have the smallest value (hence the name 'projection pursuit'). Our implementation is based on the results described in paper [44]. In each experiment, a model with 8 projections and a 5th-order polynomial was applied.

Support Vector Machines

Support vector machines is a classifier algorithm that is based on the same kernel idea we presented earlier. It first maps the data points into a high-dimensional feature space by applying some kernel function. Then, assuming that the data points have become more easily separable in the kernel-space, it performs linear classifications to separate the classes. A linear hyperplane is chosen with a maximal margin. For further details on SVM the reader may peruse [97]. In all the experiments with SVM the radial basis kernel function was applied.

6.2.5 Experiments

In the experiments 5 feature sets were constructed from the initial acoustic features described in Section 6.2.2. *Set1* contained the 24 FBLE features. In *Set2* we combined *Set1* with the gravity center features, so *Set2* contained 30 measurements. *Set3* was composed of the 24 SBS features, while in *Set4* we combined the FBLE and SBS sets. Lastly, in *Set5* we added all the FBLE, SBS and gravity center features, thus obtaining a set of 54 values.

In the classification experiments every transformation was combined with every classifier on every feature set. This resulted in the large table of Table 6.1. In the header of the table PCA, ICA, LDA and SDA stand for the linear transformations (i.e. the kernel $\mathbf{x}^\top\mathbf{z}$ was used), while KPCA, KICA, KLDA and KSDA stand for the non-linear transformations (with an exponential kernel), respectively. The numbers shown are the recognition errors on the test data. The number in parentheses denotes the number of features preserved after a transformation. The best scores of each set are given in bold.

6.2.6 Results and Discussion

Upon inspecting the results the first thing one notices is that the SBS feature set (*Set3*) did about twice as badly as the other sets, no matter what transformation or classifier was tried. When combined with the FBLE features (*Set1*) both the gravity center and the SBS features brought some improvement, but this improvement is quite small and varies from method to method.

feature set	classifier	none (*all*)	PCA (8)	ICA (8)	LDA (8)	SDA (8)	KPCA (8)	KICA (8)	KLDA (8)	KSDA (8)
	GMM	16.38	13.81	16.45	14.37	15.06	15.20	13.68	12.43	12.70
	ANN	10.34	9.86	9.93	10.97	9.58	9.86	9.58	8.05	7.98
Set1 (24)	PPL	11.04	10.06	10.69	9.51	9.93	8.95	9.51	7.98	8.75
	SVM	9.93	10.00	8.95	8.05	8.05	8.88	8.26	**6.73**	7.22
	GMM	13.33	11.38	13.33	12.84	13.33	13.47	12.36	10.27	11.31
	ANN	7.43	8.05	7.36	7.77	6.18	6.52	8.19	5.69	6.66
Set2 (30)	PPL	9.37	8.59	6.54	6.11	6.45	6.59	6.45	**4.93**	6.66
	SVM	8.33	6.66	6.66	6.45	5.13	7.36	6.11	5.27	5.34
	GMM	25.90	23.19	25.90	22.91	24.37	25.13	24.65	23.05	21.45
	ANN	20.00	18.88	19.58	21.45	20.00	21.04	18.54	18.26	17.84
Set3 (24)	PPL	20.48	20.69	19.58	20.00	20.76	18.88	19.16	17.84	18.54
	SVM	19.65	20.69	18.88	17.36	19.58	19.79	18.33	16.52	**16.45**
	GMM	13.95	12.01	15.90	13.81	14.16	15.34	12.08	10.00	9.93
	ANN	10.27	9.86	8.05	9.02	8.95	7.36	9.86	**5.55**	7.56
Set4 (48)	PPL	10.48	8.95	9.37	8.95	9.44	7.36	9.09	6.18	7.98
	SVM	9.09	9.79	8.26	6.04	7.56	8.75	5.97	5.76	6.25
	GMM	15.48	12.29	13.33	11.04	13.75	11.73	11.87	10.83	11.59
	ANN	8.68	7.01	6.45	10.00	7.56	9.09	6.59	7.15	4.93
Set5 (54)	PPL	8.26	9.23	7.36	6.52	7.29	8.05	7.77	6.18	7.77
	SVM	9.37	8.54	5.76	4.65	5.62	6.11	5.76	6.18	**4.23**

Table 6.1: Recognition accuracies for each feature set as a function of the transformation and classification applied.

When focusing on the performance of the classifiers, we see that ANN, PPL and SVM yielded very similar results. They, however, consistently outperformed GMM, which is still the method most commonly used in speech technology today. Firstly, this can be attributed to the fact that the functions that a GMM (with diagonal covariances) is able to represent are more restricted in shape than those of ANN or PPL. Secondly, it is a consequence of modeling the classes separately, rather than in the case of the other three classifiers that optimize a discriminative error function.

As regards the transformations, an important observation is that after the transformations the classification scores did not get worse compared to the classifications when no transformation was applied. This is so in spite of the dimension reduction, which shows that the features are highly redundant. Removing this redundancy by means of a transformation can make the classification more robust and, of course, faster.

Comparing the linear and the kernel-based algorithms, there is a slight preference towards the supervised transformations rather than the unsupervised ones. Similarly, the non-linear transforms yielded somewhat better scores than the linear ones. The best transformation-classifier combination, however, varies from set to set. This warns us that no such broad claim can really be made about one transformation being superior to the others. This is always dependent on the feature set and the classifier. This is, of course, in accordance with the "no free lunch" theorem which claims that, for different learning tasks, a different inductive bias can be beneficial [21].

Finally, we should make some general remarks. First of all, we must emphasize that both the transformations and the classifiers have quite a lot of adjustable parameters, and to examine all parameter combinations is practically impossible. Changing some of these parameters can sometimes have a significant effect on the classification scores. Keeping this (and the no free lunch theorem) in mind, our goal in this work was to show that the non-linear supervised transformations have the tendency to perform better (with any given classifier) than the linear and/or unsupervised methods. The results here seem to justify our hypothesis.

6.3 Summary

The main purpose of this chapter was to compare several classification and transformation methods applied to real-time phoneme classification. The goal of applying a transformation can be dimension reduction, improvement of the classification scores, or increasing the robustness of the learning by removing the noisy and redundant features.

We found that non-linear transformations in general lead to better classifications than the non-linear ones, and thus are a promising new direction for research. We also found that the supervised transformations are usually better than the unsupervised ones. These transformations greatly improved our phonological awareness teaching system by offering a robust and reliable real-time phoneme classification.

Finally, we should mention that finding the optimal parameters both for the transformations and the classifiers is quite a difficult problem. A combined optimization should probably produce better results if done.

Chapter 7

Conclusions

In the first part of this thesis we non-linearized several linear feature extraction methods with the help of the kernel idea. In the second part, to demonstrate the effect of the methods, we performed phoneme recognition tasks within the framework of speech technology applications.

In the case of the linear feature extraction methods employed here, their non-linearization was made possible by the property that they all lead to the optimization of a Rayleigh quotient, both its numerator and denominator being a special function of the training samples. Thanks to this special form the Rayleigh quotient could be formulated as the function of the pairwise dot product of the samples. Exploiting this, to non-linearize these formulas we simply had to redefine the dot product operation making use of kernel functions. The effect of this operation replacement (see Theorem 2.1) is that the whole calculation can be performed implicitly in a different dot product space. The linear transformations themselves still remain linear in this other space but, when viewing the transformation as the function of the original space elements, it is non-linear. This is because owing to the kernel functions, the connection between the original and the new space is non-linear.

Practically speaking, with the application of the kernel functions we open a door into 'parallel' dot product spaces, the 'passage' to which is non-linear. The kernel idea and the linear and non-linear methods presented in this thesis demonstrate that perhaps the gap between linear and non-linear models is not that big at all. More precisely, a subset of the non-linear models is linear but in another space.

The results of the speech technology applications demonstrated that, in order to increase classification performance, it is worth decomposing the classification problem into a feature extraction and a learning step. Albeit both the feature extraction and the learning algorithms aim to separate the classes, performing it in two steps usually proves more efficient.

Next we should remark that as the construction of pattern recognition systems requires the construction of proper models, it is very important to exhaustively study the possible building blocks of these models, or even to create new ones. The kernel idea is undoubtedly a good example of the latter, seeing how much progress it has already brought to the discipline.

Appendix A

Kernel Functions

A.1 List of Kernel Functions

Gaussian RBF Kernel	$\exp\left(-\frac{\|\|\mathbf{x}-\mathbf{z}\|\|^2}{\sigma}\right)$	$\sigma \in \mathbb{R}_+$
Polynomial Kernel	$(\mathbf{x}^\top\mathbf{z}+\sigma)^q$	$q \in \mathbb{N}, \quad \sigma \in \mathbb{R}_+$
Cosine Polynomial Kernel	$\left(\frac{\mathbf{x}^\top\mathbf{z}}{\|\|\mathbf{x}\|\|\|\|\mathbf{z}\|\|}+\sigma\right)^q$	$q \in \mathbb{N}, \quad \sigma \in \mathbb{R}_+$
Inverse Multi-Quadratic Kernel	$\frac{1}{\sqrt{\|\|\mathbf{x}-\mathbf{z}\|\|^2+\sigma}}$	$\sigma \in \mathbb{R}_+$
Rational Quadratic Kernel	$1-\frac{\|\|\mathbf{x}-\mathbf{z}\|\|^2}{\|\|\mathbf{x}-\mathbf{z}\|\|^2+\sigma}$	$\sigma \in \mathbb{R}_+$
Sigmoidal Kernel*	$\tanh(c_1\mathbf{x}^\top\mathbf{z}+c_2)$	$c_1, c_2 \in \mathbb{R}$
Triangular Kernel	$1-\frac{\|\|\mathbf{x}-\mathbf{z}\|\|}{\beta}$	$\beta \geq \sup_{\mathbf{x}_1,\mathbf{x}_2\in\mathcal{X}} \|\|\mathbf{x}_1-\mathbf{x}_2\|\|$

Table A.1: A partial list of useful kernel functions.

* With the exception of the Sigmoidal kernel (cf. [92]), all the functions listed here are Mercer kernels with any parameter value in the given range.

A.2 Graph of Radial Basis Kernel Functions

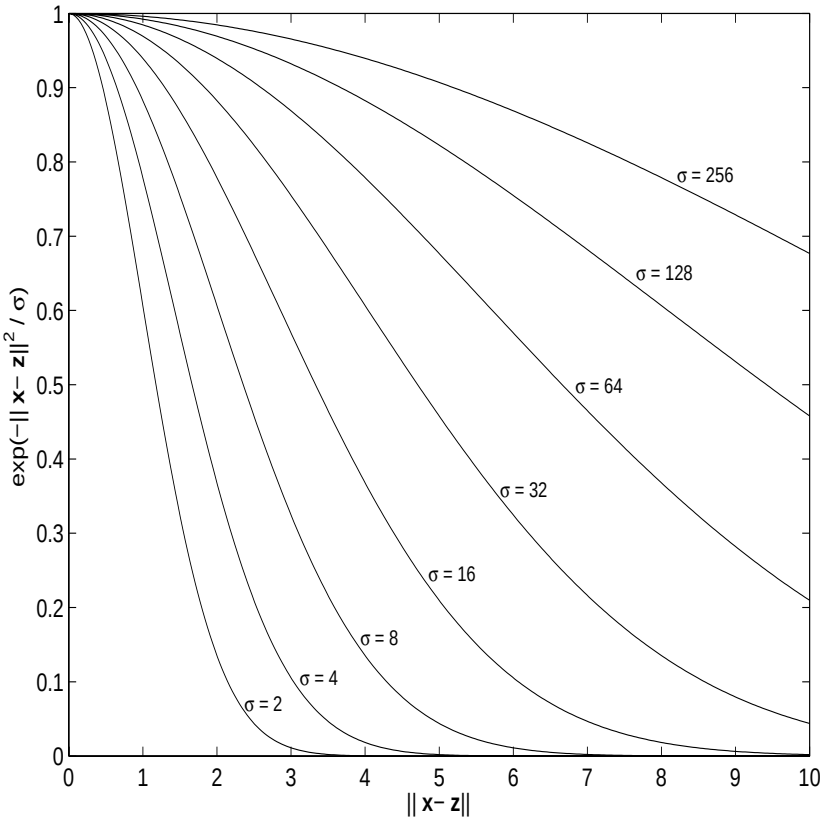

Figure A.1: Graph of $exp(-||\mathbf{x}-\mathbf{z}||/\sigma)$ for $\sigma \in \{2, 4, 8, 16, 32, 64, 128, 256\}$.

A.3 Graph of Polynomial Kernels

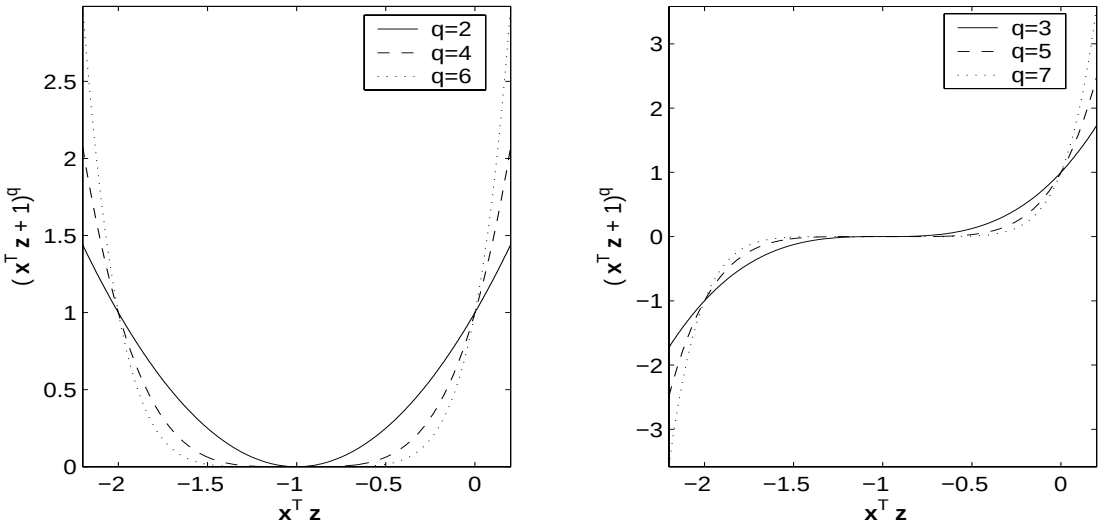

Figure A.2: Graph of $(\mathbf{x}^T\mathbf{z}+1)^q$ for $q \in \{2, 4, 6\}$ and $q \in \{3, 5, 7\}$, respectively.

Appendix B

Proofs

This appendix contains the proofs of the propositions mentioned in Chapters 2 – 4.

B.1 Proof of Proposition 2.1

Proof
Let $X = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ be a k-tuple of $\mathcal{X}$. It is sufficient to prove that the following K matrix – the Gram matrix –

$$\begin{pmatrix} \mathbf{x}_1^\top \mathbf{x}_1 & \cdots & \mathbf{x}_1^\top \mathbf{x}_t \\ \vdots & \ddots & \vdots \\ \mathbf{x}_k^\top \mathbf{x}_1 & \cdots & \mathbf{x}_k^\top, \mathbf{x}_t \end{pmatrix} \tag{B.1}$$

is positive semidefinite. And this is easy to see because

$$\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} = \boldsymbol{\alpha}^\top \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_k^\top \end{pmatrix} \Big(\mathbf{x}_1, \cdots, \mathbf{x}_k \Big) \boldsymbol{\alpha} = \| X\boldsymbol{\alpha} \|^2 \geq 0 \tag{B.2}$$

□

holds for any arbitrary $\boldsymbol{\alpha}$.

B.2 Proof of Proposition 2.2

Proof
It is trivial to prove that the functions defined in the proposition are continuous and symmetric. To prove that they are positive definite, first we define a finite set $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \mathcal{X}$. Next, let us denote the matrices $\kappa(X, X)$, $\kappa_1(X, X)$ and $\kappa_2(X, X)$ by K, K_1 and K_2.

i) Since $\boldsymbol{\alpha}^\top (K_1 + K_2)\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top K_1 \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top K_2 \boldsymbol{\alpha} \geq 0$ the matrix $K = K_1 + K_2$ is positive semidefinite, hence κ is a Mercer kernel.

ii) Let the entry-wise product of the matrices K_1 and K_2 be denoted by $K_1 \circ K_2$. This product is usually called the Hadamard or Schur product. According to the Schur product theorem [41] which states that the Hadamard product of positive semidefinite matrices is positive semidefinite[1] the $K = K_1 \circ K_2$ matrix is also positive semidefinite, proving our initial statement.

iii) Because matrix K_1 is positive semidefinite, matrix $K_1 + \lambda I$ is obviously also positive semidefinite for an arbitrary non-negative λ.

iv) Similar to that in *iii)*, the positive semidefiniteness of matrix K_1 implies the positive semidefiniteness of matrix λK_1 for any non-negative λ.

B.3 Proof of Corollary 2.1

Proof

i) It is an immediate corollary of Proposition 2.2 since, applying the operations in $i) - iv)$, one can construct any polynomial of the kernel function that has positive coefficients.

ii) Since the exponential function can be arbitrarily closely approximated by polynomials with positive coefficients and the kernels are closed under taking point-wise limits, the result follows.

B.4 Proof of Proposition 2.3

Proof

i) *Positivity on the diagonal.* This is trivial since the kernel matrix is positive semidefinite and the diagonal elements of a positive semidefinite matrix are non-negative.

ii) *Cauchy-Schwarz inequality.* The kernel matrix

$$\begin{pmatrix} \kappa(\mathbf{x},\mathbf{x}) & \kappa(\mathbf{x},\mathbf{z}) \\ \kappa(\mathbf{z},\mathbf{x}) & \kappa(\mathbf{z},\mathbf{z}) \end{pmatrix} \tag{B.3}$$

constructed for the $X = (\mathbf{x}, \mathbf{z})$ 2-tuple of $\mathcal{X}$ is positive semidefinite and, consequently, its determinant is non-negative. This, after noting the symmetry of the kernel function, proves the statement.

[1] To give a fast proof to the Schur product theorem, first let $\mathbf{x}$ and $\mathbf{y}$ be independent random vectors with covariance matrices A and B, respectively. Next it is easy to verify that the covariance matrix of the random vector $\mathbf{z} = \mathbf{x} \circ \mathbf{y}$ is $A \circ B$. Then, recalling that every covariance matrix is positive definite, the theorem directly follows.

iii) *Vanishing diagonals.* If both $\kappa(\mathbf{x},\mathbf{x})$ and $\kappa(\mathbf{z},\mathbf{z})$ equal 0 then, from the Cauchy-Schwarz inequality,

$$\kappa(\mathbf{x},\mathbf{z})^2 \leq \kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{z},\mathbf{z}) \quad \mathbf{x},\mathbf{z} \in \mathcal{X} \tag{B.4}$$

we get that $\kappa(\mathbf{x},\mathbf{z}) = 0$. □

B.5 Proof of Proposition 2.4

Proof

i) This statement of the proposition was originally proved by Schoenberg, together with its converse, which is much more difficult to prove [86]. Later Baxter constructed a shorter geometric proof [8]. We will not repeat the proof here, because it is rather lengthy and technical.

ii) The proof readily follows from a well-know result of Polya. A real function $g(t_1 - t_2)$, $t_1, t_2 \in \mathbb{R}$ is positive definite if g is continuous, even function on $\mathbb{R}$, which is convex and decreasing on $(0,\infty)$.

iii) The following derivation proves the statement.

$$\begin{aligned}
\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} &= \boldsymbol{\alpha}^\top [\kappa(\mathbf{x}_i,\mathbf{x}_j)]_{i,j=1}^k \boldsymbol{\alpha} \\
&= \boldsymbol{\alpha}^\top [h(\mathbf{x}_i)h(\mathbf{x}_j)]_{i,j=1}^k \boldsymbol{\alpha} \\
&= \boldsymbol{\alpha}^\top \begin{pmatrix} h(\mathbf{x}_1) \\ \vdots \\ h(\mathbf{x}_k) \end{pmatrix} \Big(h(\mathbf{x}_1), \cdots, h(\mathbf{x}_k) \Big) \boldsymbol{\alpha} \\
&= \Big\| \Big(h(\mathbf{x}_1), \cdots, h(\mathbf{x}_k) \Big) \boldsymbol{\alpha} \Big\|^2 \geq 0
\end{aligned} \tag{B.5}$$

iv) It is known from topology that a continuous mapping preserves the compactness of a set. Let the image of $\mathcal{X}$ under ψ be denoted by $\mathcal{Z}$. Since $\kappa_0 \in \mathcal{K}(\mathbb{R}^m)$ it also follows that $\kappa_0 \in \mathcal{K}(\mathcal{Z})$. Thus $\kappa \in \mathcal{K}(\mathcal{X})$.

v) Let us examine the eigenvalue decomposition $V^\top DV$ of matrix B. Since matrix B is symmetric positive definite and its eigenvalues are thus non-negative, we can assume that $V^\top DV = V^\top D^{1/2} D^{1/2} V$. Applying the notation $A = D^{1/2}V$ we find that

$$\kappa(\mathbf{x},\mathbf{y}) = \mathbf{x}^\top B \mathbf{z} = (A\mathbf{x})^\top (A\mathbf{z}). \tag{B.6}$$

Using the latter form of the kernel function for an arbitrary $X = (\mathbf{x}_1, \cdots, \mathbf{x}_k)$ finite k-tuple of $\mathcal{X}$ the kernel matrix takes the form $X^\top A^\top A X$. Its positive semidefiniteness immediately follows from

$$\boldsymbol{\alpha}^\top X^\top A^\top A X \boldsymbol{\alpha} = \| AX\boldsymbol{\alpha} \|^2 \geq 0. \tag{B.7}$$

□

B.6 Proof of Theorem 2.1

Proof

Here we sketch two possible proofs (see [64; 90] for details). In fact, we present two different methods for creating a Hilbert space $\mathcal{F}$ which satisfies the proposition[2].

The first one is obtained from a theorem of Mercer's from 1909, while the construction of the second one is based on the reproducing property of Mercer kernels. Obviously, giving two different constructions is unnecessary, because any two separable Hilbert spaces are isometrically isomorph. But we present both of them here for historical, rather then mathematical reasons. When the undoubtedly most popular kernel algorithm - the SVM - was published, kernel functions were introduced following the first, operator-theoretic approach. The second approach, however, is more general and, in fact, does not require the compactness of $\mathcal{X}$. We note that in this thesis we always assume the compactness[3] for the sake of simplicity, in accordance with the results of Cucker and Smale [16].

a.) **Mercer's Theorem.** *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^n$, ν is a Borel measure, and κ is a Mercer kernel over $\mathcal{X} \times \mathcal{X}$. Let λ_k be the k*th *eigenvalue of the integral operator $L_\kappa : L_2(\mathcal{X}, \nu) \to L_2(\mathcal{X}, \nu)$,*

$$(L_\kappa f)(\mathbf{x}) = \int_{\mathcal{X}} \kappa(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) d\nu(\mathbf{z}), \tag{B.8}$$

and $\{\psi_k\}_{k \geq 1}$ the orthonormal eigenfunctions. For all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$

$$\kappa(\mathbf{x}, \mathbf{z}) = \sum_{k=1}^{\infty} \lambda_k \psi_k(\mathbf{x}) \psi_k(\mathbf{z}) \tag{B.9}$$

where the convergence is absolute and uniform.

If we now substitute the Hilbert space ℓ_2 for the space $\mathcal{F}$ of Theorem 2.1, and we define the function $\phi : \mathcal{X} \to \ell_2$ as $\phi(\mathbf{x}) = (\sqrt{\lambda_k}\psi_k(\mathbf{x}))_{k=1,\ldots,\infty}$ then, for any arbitrary $\mathbf{x}, \mathbf{z} \in \mathcal{X}$, we get that $\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$.

[2] A complete dot product space is a Hilbert space.

[3] Had we not insisted on compactness, we would have had to elaborate on positive definite kernels in general instead of Mercer kernels.

b.) Let us have a positive definite kernel function κ over $\mathcal{X} \times \mathcal{X}$. We then define a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$,

$$\begin{aligned} \phi: \quad & \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \\ & \mathbf{x} \to \kappa(., \mathbf{x}). \end{aligned} \tag{B.10}$$

Here, $\mathbb{R}^{\mathcal{X}} = \{f | f : \mathcal{X} \to \mathbb{R}\}$ and $\phi(\mathbf{x})$ denotes the function which assigns the value $\kappa(\mathbf{z}, \mathbf{x})$ to $\mathbf{z} \in \mathcal{X}$, in other word, $\phi(\mathbf{x})(.) = \kappa(., \mathbf{x})$. It can be readily seen that the set of all linear combinations of the form

$$f(.) = \sum_{i=1}^{m} \alpha_i \kappa(., \mathbf{x}_i), \tag{B.11}$$

where $m \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$, $\mathbf{x}_i \in \mathcal{X}$ are arbitrary, forms a vector space. The dot product between f and

$$g(.) = \sum_{j=1}^{m'} \beta_j \kappa(., \mathbf{x}'_j), \tag{B.12}$$

where $m' \in \mathbb{N}$, $\beta_i \in \mathbb{R}$, $\mathbf{x}'_j \in \mathcal{X}$, is defined by

$$f \cdot g = \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j \kappa(\mathbf{x}_i, \mathbf{x}'_j). \tag{B.13}$$

It is straightforward to prove that it is well-defined, i.e. Eq. (B.13) is symmetric, bilinear and strictly positive definite. Up to this point we have only defined a dot product space. To turn it into a Hilbert space we need to complete it with the limit points of sequences which are convergent in the norm $||\mathbf{x}|| = \sqrt{f \cdot f}$. Finally, let $\mathcal{F}$ be defined by this Hilbert space, and, by the dot product definition

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{z}) = \kappa(., \mathbf{x}) \cdot \kappa(., \mathbf{z}) = \kappa(\mathbf{x}, \mathbf{z}), \tag{B.14}$$

which proves the theorem. □

B.7 Proof of Proposition 3.1

Proof

Let us take the derivative of $\tau(\mathbf{v})$

$$\begin{aligned} \nabla\tau(\mathbf{v}) &= \nabla \frac{\mathbf{v}^\top B_1 \mathbf{v}}{\mathbf{v}^\top B_2 \mathbf{v}} \\ &= \frac{(\nabla \mathbf{v}^\top B_1 \mathbf{v}) \mathbf{v}^\top B_2 \mathbf{v} - (\nabla \mathbf{v}^\top B_2 \mathbf{v}) \mathbf{v}^\top B_1 \mathbf{v}}{(\mathbf{v}^\top B_2 \mathbf{v})^2} \\ &= \frac{2 B_1 \mathbf{v}\mathbf{v}^\top B_2 \mathbf{v} - 2 B_2 \mathbf{v}\mathbf{v}^\top B_1 \mathbf{v}}{(\nabla \mathbf{v}^\top B_2 \mathbf{v})^2}. \end{aligned} \tag{B.15}$$

Now if we examine whether the numerator can become the zero vector, we have that

$$\begin{aligned} 2B_1\mathbf{v}\mathbf{v}^\top B_2\mathbf{v} - 2B_2\mathbf{v}\mathbf{v}^\top B_1\mathbf{v} &= \mathbf{0} \\ B_1\mathbf{v}\mathbf{v}^\top B_2\mathbf{v} &= B_2\mathbf{v}\mathbf{v}^\top B_1\mathbf{v} \\ B_1\mathbf{v} &= \frac{\mathbf{v}^\top B_1\mathbf{v}}{\mathbf{v}^\top B_2\mathbf{v}} B_2\mathbf{v} \\ B_2^{-1}B_1\mathbf{v} &= \frac{\mathbf{v}^\top B_1\mathbf{v}}{\mathbf{v}^\top B_2\mathbf{v}}\mathbf{v}, \end{aligned} \tag{B.16}$$

which means that the stationary points satisfy the eigenvalue-eigenvector equation. Put the other way round: if, for some λ and $\boldsymbol{v}$,

$$B_2^{-1}B_1\mathbf{v} = \lambda\mathbf{v} \tag{B.17}$$

holds, then

$$B_2^{-1}B_1\mathbf{v} = \frac{\mathbf{v}^\top B_2\mathbf{v}}{\mathbf{v}^\top B_2\mathbf{v}}\lambda\mathbf{v} = \frac{\mathbf{v}^\top B_2 B_2^{-1}B_1\mathbf{v}}{\mathbf{v}^\top B_2\mathbf{v}}\mathbf{v} = \frac{\mathbf{v}^\top B_1\mathbf{v}}{\mathbf{v}^\top B_2\mathbf{v}}\mathbf{v}. \tag{B.18}$$

Finally, from Eq. (B.16) and Eq. (B.18) we find that $\mathbf{v}$ must necessarily be a stationary point. □

B.8 Proof of Proposition 3.2

Proof

For ease of notation we introduce a new notation for the centralized data:

$$\begin{aligned} \mathbf{z}_1 &= \mathbf{x}_1 - E\{\mathbf{x}\}, \\ &\vdots \\ \mathbf{z}_k &= \mathbf{x}_k - E\{\mathbf{x}\}. \end{aligned} \tag{B.19}$$

If we now project the $\mathbf{z}_1, \cdots, \mathbf{z}_k$ samples onto a vector $\frac{\mathbf{v}^\top}{||\mathbf{v}||}$ and examine the variance of the resulting points, we find that

$$\begin{aligned} \frac{1}{k}\sum_{i=1}^{k}\left(\frac{\mathbf{v}^\top}{||\mathbf{v}||}\mathbf{z}_i\right)^2 &= \frac{1}{k}\sum_{i=1}^{k}\left(\frac{\mathbf{v}^\top}{||\mathbf{v}||}\mathbf{z}_i\right)\left(\mathbf{z}_i^\top\frac{\mathbf{v}}{||\mathbf{v}||}\right) \\ &= \frac{\mathbf{v}^\top}{||\mathbf{v}||}\left(\frac{1}{k}\sum_{i=1}^{k}\mathbf{z}_i\mathbf{z}_i^\top\right)\frac{\mathbf{v}}{||\mathbf{v}||} \\ &= \frac{\mathbf{v}^\top\mathcal{C}\mathbf{v}}{\mathbf{v}^\top\mathbf{v}}. \end{aligned} \tag{B.20}$$

□

B.9 Proof of Proposition 3.3

Proof

$i)$ The following equalities prove the statement.

$$E\{\mathbf{v}^\top \mathbf{z}\} = \mathbf{v}^\top E\{\mathbf{z}\} = \mathbf{v}^\top \mathbf{0} = 0 \tag{B.21}$$

$$E\{(\mathbf{v}^\top \mathbf{z})^2\} = E\{\mathbf{v}^\top \mathbf{z}\mathbf{z}^\top \mathbf{v}\} = \mathbf{v}^\top E\{\mathbf{z}\mathbf{z}^\top\}\mathbf{v} = \mathbf{v}^\top I\mathbf{v} = 1 \tag{B.22}$$

$ii)$ Let W be an arbitrary $n \times n$ matrix. Then, for the covariance matrix C_W of the set $W\mathbf{z}_1, \cdots, W\mathbf{z}_k$, we find that

$$\begin{aligned} C_W &= E\{W\mathbf{z}(W\mathbf{z})^\top\} \\ &= WE\{\mathbf{z}\mathbf{z}^\top\}W^\top \\ &= WIW^\top \\ &= WW^\top. \end{aligned} \tag{B.23}$$

Finally, $WW^\top$ is a unit matrix if and only if W is orthogonal. □

B.10 Proof of Proposition 3.4

Proof

According to the rank theorem of product matrices, it is sufficient to show that one of the factors of product matrix $W^{-1}B$ has a rank smaller than r. In this case we will demonstrate that the rank of matrix

$$B = \sum_{j=1}^{r} \frac{k_j}{k}(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^\top \tag{B.24}$$

cannot be more than $r - 1$. This simply follows from the fact that the vectors

$$\mathbf{m}_1 - \mathbf{m}, \cdots, \mathbf{m}_k - \mathbf{m} \tag{B.25}$$

are dependent which, in turn, follows from the equation

$$k_1(\mathbf{m}_1 - \mathbf{m}) + \cdots + k_r(\mathbf{m}_k - \mathbf{m}) = \mathbf{0}. \tag{B.26}$$

□

B.11 Proof of Proposition 4.1

Proof
From the proof of Proposition 3.1 we know that the stationary points of $\tau(\boldsymbol{v})$ can be obtained as the solution of the generalized eigenvalue-eigenvector problem

$$F\Theta_1 F^\top \boldsymbol{v} = \lambda(F\Theta_2 F^\top + \delta I)\boldsymbol{v} \tag{B.27}$$

Now, if $\boldsymbol{v} = \boldsymbol{v}_1 + \boldsymbol{v}_2$, where $\boldsymbol{v}_1 \in SPAN(\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_k))$ and $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$, then, exploiting the fact that $F^\top \boldsymbol{v} = F^\top \boldsymbol{v}_1$, we may obtain the result

$$F\Theta_1 F^\top \boldsymbol{v}_1 = \lambda F\Theta_2 F^\top \boldsymbol{v}_1 + \lambda\delta\boldsymbol{v}, \tag{B.28}$$

where, in the case of $\lambda\delta \neq 0$, equality is possible if and only if $\boldsymbol{v} = \boldsymbol{v}_1$. □

B.12 Proof of Proposition 4.2

Proof
Let us examine what form the covariance matrix takes in $\mathcal{F}$.

$$\begin{aligned}
\mathcal{C} &= E\{(\phi(\mathbf{x}) - E\{\phi(\mathbf{x})\})(\phi(\mathbf{x}) - E\{\phi(\mathbf{x})\})^\top\} \\
&= E\{\phi(\mathbf{x})\phi(\mathbf{x})^\top\} - E\{E\{\phi(\mathbf{x})\}\phi(\mathbf{x})^\top\} \\
&\quad -E\{\phi(\mathbf{x})E\{\phi(\mathbf{x})^\top\}\} + E\{E\{\phi(\mathbf{x})\}E\{\phi(\mathbf{x})^\top\}\} \\
&= E\{\phi(\mathbf{x})\phi(\mathbf{x})^\top\} - E\{\phi(\mathbf{x})\}E\{\phi(\mathbf{x})^\top\} \\
&= \frac{1}{k}FIF^\top - \frac{1}{k}F\hat{I}F^\top \\
&= \frac{1}{k}F(I - \hat{I})F^\top.
\end{aligned} \tag{B.29}$$

Thus

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top F^\top \frac{1}{k} F(I - \hat{I})F^\top F\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top F^\top F\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}^\top \frac{1}{k} K(I - \hat{I})K\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}}. \tag{B.30}$$

□

B.13 Proof of Proposition 4.3

Proof

Let us first examine $\mathcal{B}$ and $\mathcal{W}$.

$$\begin{aligned}
\mathcal{B} &= \sum_{j=1}^{r} \frac{k_j}{k}(\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^\top \\
&= \left(\sum_{j=1}^{r} \frac{k_j}{k}\boldsymbol{\mu}_j\boldsymbol{\mu}_j^\top\right) - \left(\sum_{j=1}^{r} \frac{k_j}{k}\boldsymbol{\mu}_j\boldsymbol{\mu}^\top\right) \\
&\quad - \left(\sum_{j=1}^{r} \frac{k_j}{k}\boldsymbol{\mu}\boldsymbol{\mu}_j^\top\right) + \left(\sum_{j=1}^{r} \frac{k_j}{k}\boldsymbol{\mu}\boldsymbol{\mu}^\top\right) \\
&= \frac{1}{k}FRF^\top - \frac{1}{k}F\hat{I}F^\top - \frac{1}{k}F\hat{I}F^\top + \frac{1}{k}F\hat{I}F^\top \\
&= \frac{1}{k}F(R - \hat{I})F^\top
\end{aligned} \tag{B.31}$$

$$\begin{aligned}
\mathcal{W} &= \sum_{j=1}^{r} \frac{k_j}{k}\frac{1}{k_j}\sum_{\mathcal{L}(i)=j}(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j)(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j)^\top \\
&= \frac{1}{k}\sum_{j=1}^{r}\sum_{\mathcal{L}(i)=j}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top - \frac{1}{k}\sum_{j=1}^{r}\sum_{\mathcal{L}(i)=j}\phi(\mathbf{x}_i){\boldsymbol{\mu}_j}^\top \\
&\quad - \frac{1}{k}\sum_{j=1}^{r}\sum_{\mathcal{L}(i)=j}\boldsymbol{\mu}_j\phi(\mathbf{x}_i)^\top + \frac{1}{k}\sum_{j=1}^{r}\sum_{\mathcal{L}(i)=j}\boldsymbol{\mu}_j{\boldsymbol{\mu}_j}^\top \\
&= \frac{1}{k}FIF^\top - \frac{1}{k}FRF^\top - \frac{1}{k}FRF^\top + \frac{1}{k}FRF^\top \\
&= \frac{1}{k}F(I - R)F^\top
\end{aligned} \tag{B.32}$$

Then,

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top F^\top \frac{1}{k}F(R - \hat{I})F^\top F\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top F^\top \frac{1}{k}F(I - R)F^\top F\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}^\top K(R - \hat{I})K\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K(I - R)K\boldsymbol{\alpha}}. \tag{B.33}$$

□

B.14 Proof of Proposition 4.4

Proof

Let us first examine $\mathcal{D}$.

$$\begin{aligned}
\mathcal{D} &= \sum_{i,j=1}^{k} \left(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\right)\left(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\right)^\top [M]_{ij} \\
&= \sum_{i,j=1}^{k} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top [M]_{ij} - \sum_{i,j=1}^{k} \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^\top [M]_{ij} \\
&\quad - \sum_{i,j=1}^{k} \phi(\mathbf{x}_j)\phi(\mathbf{x}_i)^\top [M]_{ij} + \sum_{i,j=1}^{k} \phi(\mathbf{x}_j)\phi(\mathbf{x}_j)^\top [M]_{ij} \\
&= F\tilde{M}F^\top - FMF^\top - FMF^\top + F\tilde{M}F^\top \\
&= F(2\tilde{M} - 2M)F^\top.
\end{aligned} \tag{B.34}$$

Then,

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top F^\top F(2\tilde{M} - 2M)F^\top F\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top F^\top F\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}^\top \frac{1}{k} K(2\tilde{M} - 2M)K\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}}. \tag{B.35}$$

□

Appendix C

Summary

C.1 Summary in English

Introduction

In this thesis we concentrate on two key topics in artificial intelligence (AI): machine learning (ML) and its application to speech technology (ST).

"Learning" in the machine learning sense means the application of the model method. That is, we aim at creating models that simulate human thinking correctly. The best way of doing this is to specify the model by means of a large amount of training patterns; decisions regarding a new pattern are made based on this model.

This summary is structured similar to the thesis itself. That is, it consists of two main parts. In the first one we constructed algorithms that may form the building blocks of certain models used in intelligent systems [58–62]. We have defined a unified mathematical framework for a set of linear feature extractions algorithms. With the application of the kernel idea - a method currently being used in machine learning research - this unified framework allowed us to non-linearize all the linear methods in an analogous way [62]. In the second part we demonstrated the usefulness of the methods derived in the first part. We did so by performing phoneme classification tests in the framework of speech technology applications, namely the OASIS speech recognizer [56; 57; 62] and the "SpeechMaster" speech therapy and reading teaching software [58–60].

Kernel-Based Feature Extraction

The Kernel Idea

Theoretical findings generally have their own very different, unique histories before they find any practical application. One such example is the "kernel-idea", which had appeared in several fields of mathematics [38; 78] and mathematical physics before it became a key notion in machine learning. The basic idea behind the kernel technique was originally introduced for pattern recognition in [2] and was again employed in the general purpose Support Vector Machine [12; 96; 97], which was followed by other kernel-based methods [52].

The kernel idea can be applied in any case when the input of some algorithm consists of the pairwise dot (scalar) products of the elements of an n-dimensional dot product

space. In this case, simply by a proper redefinition of the two-operand operation of the dot product, we can achieve that the algorithm will now be executed in a different dot product space, which is probably more suitable for solving the original problem. Of course, when replacing the operand, we have to satisfy certain criteria, because not every function is suitable to implicitly generate a dot product space. The family of the Mercer Kernels is an appropriate choice (according to Mercer's theorem) [16; 64]. The thesis summarizes the properties of this function class, along with the possibilities of constructing such functions, and we also give an overview of the dot product spaces (also called kernel feature spaces) induced by them.

Linear Feature Extraction

In most classification problems it is normal to view the objects to be classified as points in a feature space of proper dimensions. The space has to have a sufficient degree of freedom so that the object classes are separable enough. Making use of superfluous components, however, can confuse classification algorithms. A general practical observation is that it is worth slightly decreasing the dimensionality of the given feature space so that we can still guarantee that the overall structure of the data points remains intact. A simple way to do this is by means of a linear transformation that linearly maps an initial feature space into a new features space, usually one with fewer dimensions. Along with dimension reduction, the transformation may also aim at simplifying or emphasizing the structure of the data at the same time.

The mathematical goal of a linear transformation intended for feature extraction can be defined several ways. For example, we may choose the basis vectors of the transformed space as those directions of the original space along which the data shows a large variance (PCA – Principal Component Analysis), or when their distribution greatly differs from a Gaussian one (ICA – Independent Component Analysis). Since these methods ignore the class information of the data, they are called unsupervised. In the opposite case, if an algorithm makes use of the class labels it is called supervised. These algorithms all try to push the classes apart, while keeping the data belonging to the same class close together. Two supervised methods will be presented in this thesis, one of them – Linear Discriminant Analysis (LDA) – is well-known, while the other – Springy Discriminant Analysis (SDA) – is based on a novel idea [60; 61].

Handling all the four linear feature extraction methods in a unified framework is made possible by the fact that in all cases the row vectors of the matrix of the linear mapping can be obtained by optimizing a function given in the form of a Rayleigh quotient. This otpimization can be performed relatively easily, as it leads to a (generalized) eigenvalue-eigenvector problem, and many reliable and fast libraries are available to solve these kind of tasks.

Non-linear Feature Extraction with Kernels

Bringing the problem into a Rayleigh-quotient form is not sufficient in itself. We need some further insight. We may soon realize, that all methods lead to the optimization of a Rayleigh quotient that has a special form, where the matrices appearing in the

numerator and denominator are both a unique function of the input vectors of the algorithm. In fact, this property allows us to express the formulas as the function of the pairwise inner products of the sample vectors. After this all that is left is to redefine the inner product operation by using proper kernel functions. The effect of this substitution of the operation - as we already know - is that the algorithm will be executed implicitly in a different inner product space. The linear feature extraction will still perform a linear transformation, but now in this new inner product space. So, viewing the transformations as a function of the vectors in the original space, this function will be non-linear, because the connection between the original and the new space is non-linear, thanks to the kernel functions.

Besides deriving the Kernel-PCA, Kernel-ICA, Kernel-LDA and Kernel-SDA algorithms [58–62], we also demonstrate their effect in the thesis by plotting their results when using simple artificial 2- and 3-dimensional data sets as input data.

Speech Technology Applications

Speech Recognition

Automatic speech recognition is a special pattern classification problem which aims to mimic the perception and processing of speech in humans. For this reason it clearly belongs to the fields of machine learning and artificial intelligence. For historical reasons, however, it is mostly ranked as a sub-field of electrical engineering, with its own unique technologies, conferences and journals. In the last two decades the dominant method for speech recognition has been the hidden Markov modeling approach. Meanwhile, the theory of machine learning has developed considerably and now has a wide variety of learning and classification algorithms for pattern recognition problems. The primary goal of this Chapter is to study the applicability of some of these methods to phoneme classification.

When choosing the directions of our speech recognition research, we decided to focus on Hungarian with the hopes that we can address some special issues concerning the processing of our national language, and also that we can make use of our previous experience with NLP for Hungarian. Furthermore, we were looking for a flexible framework that allows experimentation with different preprocessing techniques, feature-space transformation methods and machine learning algorithms. These expectations led us to the stochastic segmental approach which, in a certain sense, can be viewed as an extension of hidden Markov modeling. Our recognition system, OASIS[1], was designed to be as modular as possible, so we can easily conduct experiments with combining different techniques for the several subtasks of recognition.

In the thesis we designed and performed a segmental phoneme classification tests within the framework of the OASIS speech recognizer [56; 57; 62]. The goal of this test was to examine how the feature extraction algorithms - when combined with classification algorithms (Timbl, OC1, C4.5, GMM, ANN) - influence the classification accuracy.

Examining the results of the many tests performed, we can claim that in the hope of better classification it is worth applying feature extraction algorithms prior to learning.

[1] The acronym is from "**O**ur **A**coustics-based **S**peaker-**I**ndependent **S**peech recognizer".

Phonological Awareness Teaching

An important clue to the process of learning to read in alphabet-based languages is the ability to separate and identify consecutive sounds that make words and to associate these sounds with its corresponding written form [1; 85]. To learn to read in a fruitful way young learners must, of course, also be aware of the phonemes and be able to manipulate them. Many children with learning disabilities have problems in their ability to process phonological information. Furthermore, phonological awareness teaching has also great importance for the speech and hearing handicapped, along with evolving the corresponding articulatory strategies of tongue movement.

The "SpeechMaster" software developed by our team seeks to apply speech recognition technology to speech therapy and the teaching of reading. Both applications require a real-time response from the system in the form of an easily comprehensible visual feedback. With the simplest display setting feedback is given by means of flickering letters, their identity and brightness being adjusted to the speech recognizer's output. In speech therapy it is intended to supplement the missing auditive feedback of the hearing impaired, while in teaching reading it is to reinforce the correct association between the phoneme-grapheme pairs. With the aid of a computer children can practice without the need for the continuous presence of the teacher. This is very important because the therapy of the hearing impaired requires a long and tedious fixation phase. Furthermore, experience shows that most children prefer computer exercises to conventional drills.

In the tests performed in this thesis within the "SpeechMaster" software package we again studied how the combination of feature extraction algorithms with classifiers (ANN, PLL, GMM) affects classification [58–60]. We found that non-linear transformations in general lead to a better classification than the non-linear ones, and thus are a promising new direction for research. We also found that the supervised transformations are usually better than the unsupervised ones. These transformations greatly improved our phonological awareness teaching system by offering a robust and reliable real-time phoneme classification.

Conclusion

The kernel idea and the linear and non-linear feature extraction methods presented in this thesis demonstrate that perhaps the gap between linear and non-linear models is not that big at all. More precisely, a subset of the non-linear models is linear but in another space.

The results of the speech technology applications demonstrated that, in order to increase classification performance, it is worth decomposing the classification problem into a feature extraction and a learning step. Albeit both the feature extraction and the learning algorithms aim to separate the classes, performing it in two steps usually proves more efficient.

C.2 Summary in Hungarian

Bevezetés

A tézis témaköre tágabb értelemben a mesterséges intelligencia, szorosabb értelemben pedig a gépi tanulás.

A mesterséges intelligenciában a tanulás a modell-módszer alkalmazását jelenti. Megprobálunk olyan modelleket létrehozni, amelyek jól szimulálják az emberi intelligenciát. Ennek a legjobb lehetséges módja az, hogy minták sokaságát figyelembe véve specifikáljuk a modellünket, és a döntéseinket új minták esetére ezen modell alkalmazasával hozzuk meg.

Ez az összefoglaló követi a tézis felépítését. A tézis két részre tagolódik. Az első részben olyan algoritmusok konstrukcióját adtuk meg, amelyek építőköveit képezhetik intelligens rendszerekben használt modelleknek [58–62]. Egy olyan egységes matematikai keretet definiáltunk lineáris tulajdonságkinyerő algoritmusok egy halmazához, amely egy, a gépi tanulási kutatások fókuszában álló ötlet, a kernel ötlet alkalmazásával lehetővé tette ezen eljárások nemlinearizálását [62]. A második részben pedig konkrét beszédtechnológiai alkalmazások, az OASIS beszédfelismerő [56; 57; 62], illetve a SpeechMaster beszédjavítás-terápiai és olvasásfejlesztő rendszer keretein belül [58–60], fonémafelismerési tesztekkel demonstráltuk az első rész eljárásainak hasznosságát.

Tulajdonságkinyerés kernel függvényekkel

A kernel ötlet

Mire egy-egy elméleti felfedezés eljut a gyakorlati alkalmazásig, az sokszor hosszas folyamat eredménye lehet. Erre példa a kernel ötlet is, amely a matematika [38; 78], illetve matematikai fizika számos területén felbukkant, mielőtt a gépi tanulási kutatások fókuszába került. Az alapötlet mintafelismerési alkalmazását eredetileg majd 40 éve Aizerman javasolta [2]. A módszer azonban igazán ismertté csak sokkal később, a support vektor gépek publikálásakor vált [12; 96; 97]. Az ötlet alkalmazása nem állt meg, sőt igazából manapság éli fénykorát, sorra jelennek meg újabb és újabb kernel eljárások [52].

A kernel ötlet olyan esetekben alkalmazható, amikor egy algoritmus bemenetét egy n-dimenziós belső szorzat tér vektorainak páronkénti belső szorzata jelenti. Ekkor pusztán a belső szorzat, vagy skaláris szorzat, kétoperandusú művelet alkalmas, nemlineáris megváltoztatásával elérhetjük, hogy az előbbi algoritmus immáron egy másik, esetleg az algoritmus céljainak jobban megfelelő belső szorzat térben hajtódjon végre. Persze nem minden függvény alkalmas az előbbi kritériumnak megfelelő műveletcserére, hiszen nem minden függvény generál implicit módon egy másik belső szorzat teret. Viszont a Mercer kernelek családja egy lehetséges jó választás (Mercer tétele) [16; 64]. A tézisben összefoglaljuk ennek a függvényosztálynak a tulajdonságait, konstrukciós lehetőségeit és

áttekintjük az általuk indukált belső szorzat tereket, vagy másnéven kernel tulajdonság tereket.

Lineáris tulajdonságkinyerés

A legtöbb klasszifikációs probléma megoldása során a klasszifikálandó komplex objektumokat célszerű egy, a dimenzióját tekintve megfelelően nagy tulajdonságtér pontjaival ábrázolni. Ennek a térnek elég szabadsági fokának kell lennie ahhoz, hogy a különböző osztályokhoz tartozó objektumok elégségesen elszeparálhatóak legyenek, azonban a felesleges komponensek megzavarhatják a klasszifikációs algoritmusok működését. Általános gyakorlati tapasztalat, hogy érdemes a tulajdonságtér dimenzióját csökkenteni mindaddig, amíg az adatok struktúrája nem sérül. Ennek egy egyszerű lehetősége a lineáris lekepezések használata, amikor is az iniciális tulajdonságteret lineárisan leképezzük egy új, rendszerint kisebb dimenziós tulajdonságtérbe. Ennek a transzformációnak nemcsak a dimenziócsökkentés lehet a célja, hanem az adatok struktúrájának kiemelése, világosabbá, egyszerűbbé tétele.

A lineáris transzformációkat a tulajdonságkinyerés érdekében különféle céllal végezhetjük. Az új tér bázisvektorai például lehetnek olyan irányok az eredeti térben, amelyek mentén az adatok nagy varianciát mutatnak – ilyen módszer a PCA (Principal Component Analysis), vagy esetleg nagyon eltérnek a Gaussz-eloszlástól, mint az ICA (Independent Component Analysis) esetében. Mivel ezen irányok meghatározásához az osztályinformációkat nem kell számításba venni, ezek a módszerek az ún. nem felügyelt tulajdonságkinyerő eljárások családjába tartoznak. Ellenkező esetben, ha az osztályinformációkat is felhasználjuk a számítások során, akkor felügyelt lineáris tulajdonságkinyerő eljárásokról beszélhetünk. Ebben az esetben a transzformáció alkalmazásával igyekszünk a különböző osztályokhoz tartozó objektumokat eltávolítani, miközben elvárjuk, hogy az azonos osztályba tartozó elemek közeledjenek egymáshoz. Felügyelt tulajdonságkinyerésre is két eljárást mutatunk be a dolgozatban, az egyik a közismert LDA (Linear Discriminant Analysis), a másik pedig az újszerű SDA (Springy Discriminant Analysis) [60; 61].

A tézisben a négy lineáris tulajdonságkinyerésre alkalmas módszer egységes tárgyalásmódját az adja, hogy minden eljárás esetében a lineáris leképezés mátrixának sorvektorait egy-egy Rayleigh-hányados alakban megadott függvény optimalizálásával határozhatjuk meg. Ez az optimalizáció azonban viszonylag egyszerűen elvégezhető, hiszen (általános) sajátérték-sajátvektor problémára vezet, amely megoldására rendelkezésre állnak megbízható, relatíve gyors könyvtári rutinok.

Nemlineáris tulajdonságkinyerés kernel függvényekkel

Az előbbi lineáris tulajdonságkinyerő eljárások kernel függvényes nemlinearizálásához a Rayleigh-hányados alak önmagában nem elég. További észrevételek szükségesek. Könnyen belátható, hogy ezek a módszerek olyan speciális Rayleigh-hányados alakú formulák optimalizálására vezetnek, ahol a számlálóban és a nevezőben szereplő mátrixok mindegyike az algoritmusok bemenetét képező mintavektorok sajátos függvénye.

Tulajdonképpen ennek köszönhető, hogy a formulákat ki tudjuk fejezni a mintavektorok páronként vett belső szorzatának függvényeként. Ezek után a nemlinearizáláshoz nem kell mást tenni, mint a belső szorzat műveletet átdefiniálni alkalmas kernel függvények segítségével. Ennek a műveletcserének – mint tudjuk – az a hatása, hogy az eljárás implicit módon egy másik belső szorzat térben fog végrehajtódni. A lineáris tulajdonságkinyerő eljárások továbbra is lineáris transzformációkat fognak meghatározni ebben a másik térben, viszont ha a kapott leképezéseket az eredeti tér vektorainak függvényeként tekintjük, akkor az már nemlineáris lesz, hiszen az eredeti és az új tér között a kernel függvények alkalmazása miatt a kapcsolat nemlineáris.

A tézisben a Kernel-PCA, Kernel-ICA, Kernel-LDA és Kernel-SDA [58–62] módszerek levezetésén túl egyszerű mesterséges 2-, és 3-dimenziós mintákon keresztül demonstráljuk az eljárások működését.

Beszédtechnológiai alkalmazások

Beszédfelismerés

Az automatikus beszédfelismerés egy olyan mintafelismerési probléma, amelynek a célja az ember beszédfeldolgozási képességének modellezése. Ezért nyilvánvalóan a gépi tanulással és mesterséges intelligenciával foglalkozó tudomány részét képezi. Tradícionális okokból azonban az elektromérnöki tudományok részterületeként szokás megjelölni. Az utóbbi néhány évtizedben a beszédfelismerés domináns technológiája a rejtett Markov modell (HMM) alapú megközelítés volt. Eközben viszont a gépi tanulás elmélete sokat fejlődött és számos új tanuló és klasszifikációs eljárás vált elérhetővé [11; 21; 27]. A disszertációban a beszédfelismerési fejezetek célja a bevezetett tulajdonságkinyerő eljárások alkalmazhatóságának vizsgálata a fonémafelismerés feladatán.

Amikor beszédfelismerési kutatásaink irányát megválasztottuk, elhatároztuk, hogy olyan flexibilis rendszert fejlesztünk, amely lehetővé teszi a kísérletezést különféle előfeldolgozó, tulajdonságkinyerő és gépi tanuló algoritmusokkal. Ezek az elvárások elvezettek egy sztochasztikus szegmentális beszédfelismerő rendszer, az OASIS [95] kifejlesztéséhez, amelynek a moduljai egy sajátos script nyelv segítségével vezérelhetőek. A moduláris felépítés és a magas szintű vezérlési lehetőség intenzív kutatómunkát tesz lehetővé a beszédfelismerés területén.

A disszertációban az OASIS beszédfelismerő rendszer keretein belül előkészítettünk és végrehajtottunk szegmentális fonémafelismerési teszteket [56; 57; 62]. A tesztekben azt vizsgáltuk, hogy a tulajdonságkinyerő algoritmusok kombinálva különféle klasszifikációs algoritmusokkal (Timbl, OC1, C4.5, GMM, ANN) hogyan befolyásolják a felismerési pontosságot.

A nagyszámú teszt eredményének ismeretében kijelenthetjük, hogy a hatékonyabb klasszifikáció reményében tanulás előtt érdemes tulajdonságkinyerő algoritmusokat alkalmazni.

Fonológiai tudatosság tanítás

Az alfabetikus nyelvek esetében az olvasástanulás folyamatában nagyon fontos a szavakat alkotó egymást követő hangok szeparálásának és azonosításának képessége, valamint a beszédhangok és írásjelek helyes asszociációja [1; 85]. Ahhoz, hogy az olvasástanulás eredményes legyen, a tanulókban valamiféle fonológiai tudatosságnak kell kialakulni, sőt a gyerekeknek képesnek kell lenniük manipulálni is ezeket. A legtöbb tanulási nehézségekkel küzdő gyermeknek problémája van a fonológiai információk feldolgozásával. Mindezek mellett a fonológiai tudatosság kialakításának és tanításának szintén nagy szerepe van a siket gyermekek beszédjavítás-terápiájában is.

Az általunk kifejlesztett „Beszédmester" szoftvercsomag beszédfelismerési technológiákat alkalmaz olvasásfejlesztésre és beszédjavítás-terápiára. A beszédfelismerés feladata egy vizuális fonetikai visszacsatolás megvalósítása egy megbízható valós idejű fonémafelismerő segítségével [58–60]. A képernyőn a kiejtés pillanatában a felismerő kimenete alapján megjelenik egy betű, amelynek a fényessége éppen a felismerésének valószínűségével arányos. Míg a beszédjavítás-terápiában a siket gyermekek hiányzó auditív visszacsatolásának helyettesítése a cél, addig az olvasástanításban a fonéma-graféma párok asszociációjának megerősítése. A számítógéppel segített beszédterápia lehetősége jelentős, hiszen a tanár állandó jelenléte nem szükséges hozzá. Rendszerint a siket gyermekek esetében nagyon hosszú terápiás folyamat eredményez előrehaladást, amelyet a „Beszédmester" akár jelentősen is meggyorsíthat. Továbbá a tapasztalatok azt mutatják, hogy a gyermekek előnyben részesítik a számítógépes munkát a hagyományos feladatokkal szemben.

A disszertációban, a „Beszédmester" programcsomag keretein belül elvégzett tesztek során szintén azt vizsgáltuk, hogy a javasolt tulajdonságkinyerő eljárások milyen hatást fejtenek ki klasszifikációs algoritmusok egy halmazára (ANN, PPL, GMM, SVM) a felismerési pontosság tekintetében [58–60]. Az eredmények azt mutatták, hogy a transzformációk közül a nemlineárisak rendszerint kisebb klasszifikációs hibát eredményeztek. A felügyelt és nem felügyelt módszerek viszonylatában pedig a felügyelt eljárások voltak sikeresebbek. Mindezek az eredmények nagyban hozzájárultak, hogy a „Beszédmester" szoftverünk hatékonyabb és megbízhatóbb valós idejű fonémaklasszifikációt, és ezáltal eredményesebb terápiát végezhessen.

Konklúzió

A disszertációban szereplő lineáris tulajdonságkinyerő eljárások, illetve ezek kernel függvényeket használó nemlinearizált változata jól demonstrálja, hogy talán matematikai értelemben nem is olyan nagy a különbség a lineáris és a nemlineáris modellek között. Pontosabban a nemlineáris modellek egy halmaza lineáris, de egy másik térben.

A beszédtechnológiai alkalmazások eredményeiből kiderült, hogy a hatékonyság növelése érdekében érdemes a felismerés problémáját két részre bontani: először tulajdonságkinyerésre, majd tanulásra. Noha mind a tulajdonságkinyerést végző, mind a tanulást megvalósító matematikai modellek célja a szeparáció, a végcél több lépésben történő elérése sokszor célravezetőnek bizonyul.

sorszám	PCA	ICA	LDA	SDA	Kernel-PCA	Kernel-ICA	Kernel-LDA	Kernel-SDA	keretrendszer
[56]	•				•				OASIS
[57]	•	•	•						OASIS
[58]			•				•		Beszédmester
[59]		•				•			Beszédmester
[60]								•	Beszédmester
[62]	•	•	•	•	•	•	•	•	OASIS

C.1. táblázat. A tézis témájának és az azt fedő saját publikációknak a viszonya.

A disszertáció tézisei

Mivel a disszertáció két fő részre tagolódik, az eredményeket is ennek megfelelően két csoportra fogjuk felosztani.

Az első téziscsoport eredményeit a gépi tanulás témakörébe tartozó újszerű tulajdonságkinyerő algoritmusok konstrukciója képezi, amelyeket a szerző a disszertáció első részében a 3. és 4. fejezetekben mutat be.

I/1. *A szerző egy olyan egységes matematikai keretet definiált lineáris tulajdonságkinyerő algoritmusok egy halmazához, amely egy, a gépi tanulási kutatások fókuszában álló ötlet, a kernel ötlet alkalmazásával lehetővé tette ezen eljárások nemlineáris változatának kidolgozását. A disszertációban 8 tulajdonságkinyerő eljárást, 4 lineáris (PCA, ICA, LDA, SDA) és 4 nemlineáris módszert (Kernel-PCA, Kernel-ICA, Kernel-LDA, Kernel-SDA) mutat be egy egységes megközelítésben a Rayleigh-hányados optimalizálásával [56–62].*

I/2. *Az irodalomból korábban ismert 3 lineáris módszer (PCA, ICA, LDA) kiegészítéseként a szerző megkonstruált egy újszerű lineáris eljárást az SDA-t [62].*

I/3. *A szerző megadta az ICA, LDA és SDA lineáris eljárások nemlineáris változatát, amelyek eredményeképpen 3 további új algoritmus, a Kernel-ICA [59], Kernel-LDA [58] és Kernel-SDA [60] jött létre.*

A második téziscsoport témáját az első téziscsoportban felsorolt lineáris, illetve nemlineáris tulajdonságkinyerő eljárások beszédtechnológiai alkalmazása alkotja, amely megtalálható a disszertáció 5. és 6. fejezetében.

II/1. *A szerző megtervezett, és munkatársaival együtt végrehajtott az OASIS beszédfelismerő rendszer keretében olyan szegmens-alapú fonémafelismerési teszteket, amelyek demonstrálják a kifejlesztett tulajdonságkinyerő eljárások hatását a felismerési pontosság tekintetében [56; 57; 62]. A szerző saját munkáját képezi a tervezésen kívül a tulajdonságkinyerő eljárások implementálása és futtatása is.*

II/2. *A Beszédmester beszédjavítás-terápiai, olvasásfejlesztő és olvasásterápiai rendszer valós idejű fonémafelismerési hatékonyságának megnövelése érdekében a szerző további felismerési teszteket tervezett meg és végzett el munkatársaival [58–60]. A munkavégzés itt is az előbbi tézispontban leírt munkavégzési kondícióknak megfelelően történt*[2]*.*

Végül a C.1-es táblázat összefoglalja, hogy a disszertáció eredményeihez kapcsolódó publikációk milyen tulajdonságkinyerő eljárásokat ismertetnek, és hogy az alkalmazások melyik keretrendszer felhasználásával készültek.

[2]Noha az elmúlt évek során mind az OASIS, mind a „Beszédmester" rendszerek esetében a szerző projekt- és témavezetőként vett részt a munkálatokban, a két rendszert magát nem sorolja a disszertáció eredményei közé, hiszen ezekben az esetekben a munkatársakkal közös és szétválaszthatatlan eredményekről van szó.

Bibliography

[1] M. J. Adams, Beginning to read: Thinking and learning about print, Cambridge, MA: MIT Press, 1990.

[2] M. A. Aizerman, E. M. Braverman, L. I. Rozonoer, "Theoretical foundation of the potential function method in pattern recognition learning," Automat. Remote Cont., Vol. 25, pp. 821-837, 1964.

[3] A. N. Akansu, R. A. Haddad, Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets, Academic Press, 1992.

[4] D. Albesano, R. De Mori, R. Gemello, F. Mana, "A study on the effect of adding new dimensions to trajectories in the acoustic space," Proc. of Eurospeech'99, Budapest, pp. 1503-1506, 1999.

[5] F. R. Bach, M. I. Jordan, "Kernel Independent Component Analysis," J. Machine Learning Res., Vol. 3, pp. 1-48, 2002.

[6] R. Bajcsy, S. Kovacic, "Multiresolution elastic matching," Computer Vision, Graphics and Image Processing, Vol. 46, pp. 1-21, 1989.

[7] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," Neural Comput., Vol. 12, pp. 2385-2404, 2000.

[8] B. J. C. Baxter, "Positive Definite Functions on Hilbert Space," EILAT 98 Conference on Multivariate Approximation and Interpolation with Applications in CAGD, Signal and Image Processing, Israel, September 7-11, 1998.

[9] A. J. Bell, T. J. Sejnowski, "An information maximization approach to blind separation and blind deconvolution," Neural Comp., Vol. 7, pp. 1129-1159, 1995.

[10] S. A. Billings, K. L Lee, "Nonlinear Fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm," Neural Networks, Vol. 15, No. 2, pp. 263-270, 2002.

[11] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford Univerisity Press Inc., New York, 1996.

[12] B. E. Boser, I. M. Guyon, V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in Proc. of the Fifth Annual ACM Conference on Computational Learning Theory, D. Haussler (eds.), ACM Press, Pittsburg, pp. 144-152, 1992.

[13] P. Comon, "Independent component analysis, A new concept?" Signal Processing, Vol. 36, pp. 287-314, 1994.

[14] T. M. Cover, J. M. Van Campenhout, "On the Possible Orderings in the Measurement Selection Problem," IEEE Trans. Systems, Man, and Cybernetics, Vol. 7, No. 9, pp. 657-661, 1977.

[15] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.

[16] F. Cucker, S. Smale, "On the mathematical foundations of learning," Bull. Am. Math. Soc., Vol. 39, pp. 1-49, 2002.

[17] W. Daelemans, J. Zavrel, K. Sloot, A. Bosch, "TiMBL: Tilburg Memory Based Learner version 2.0 Reference Guide", ILK Technical Report - ILK 99-01, Computational Linguistics, Tilburg University, The Netherlands, 1999.

[18] G. Deco, D. Obradovic, "Linear redundancy reduction learning," Neural Networks, Vol. 8, No. 5, pp. 751-755, 1995.

[19] L. Devroye, L. Gyorfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer-Verlag, Berlin, 1996.

[20] K. I. Diamantaras, S. Y. Kung, Principal Component Neural Networks: Theory and Applications, John Wiley, New York, 1996.

[21] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, John Wiley & Sons, New York, 2001.

[22] N. Dyn, "Interpolation and approximation by radial and related functions," In C. K. Chui, L. L. Schumaker, D. J. Ward (eds.), Approximation Theory, VI, pp. 211-234, Academic Press, New York, 1991.

[23] FastICA Web Page, http:// www.cis.hut.fi/projects/ica/fastica/index.shtml.

[24] R. A. Fisher, "The use of multiple measurements in taxonomic problems," Annals of Eugenics, Vol. 7, pp. 179-188, 1936.

[25] K. S. Fu, Syntactic Pattern Recognition and Applications, Prentice-Hall, 1982.

[26] T. Fukada, Y. Sagisaka, K. K. Paliwal, "Model Parameter Estimation for Mixture Density Polynomial Segment Models," Proc. of ICASSP'97, pp. 1403-1406, 1997.

[27] K. Fukunaga, Statistical Pattern Recognition, Academic Press, New York, 1989.

[28] M. Gales, S. Young, Segmental Hidden Markov Models, Proceedings of EuroSpeech'93, pp. 1579-1582, 1993.

[29] M. G. Genton, "Classes of Kernels for Machine Learning: A Statistics Perspective," J. Machine Learning Res., Vol. 2., pp. 299-312, 2001.

[30] T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, J. Vandewalle, "Bayesian framework for least squares support vector machine classifiers, gaussian processes and kernel fisher discriminant analysis", Neural Computation, Vol. 15, No. 5, pp. 1115-1148, 2002.

[31] O. Ghitza, "Auditory Nerve Representation Criteria for Speech Analysis/Synthesis," IEEE Trans. on ASSP, Vol. 35., No. 6., pp. 736-740, 1987.

[32] H. Gish, K. Ng, "A Segmental Speech Model with Applications to Word Spotting," Proceedings of ICASSP'93, pp. 447-450, 1993.

[33] J. Glass, J. Chang, M. McCandless, "A probabilistic framework for feature-based speech recognition," Proceedings of ICSLP, Philadelphia, pp. 2277-2280, 1996.

[34] R. C. Gonzalez, R. E. Woods, Digital image processing, Addison Wessley Publishing Company, 1992.

[35] S. Greenberg, M. Slaney, (eds.), Computational Models of Auditory Function, IOS Press, 2001.

[36] U. Grenander, General Pattern Theory, Oxford University Press, 1993.

[37] A. K. Halberstadt, Heterogeneous Measurements and Multiple Classifiers For Speech Recognition, Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, 1998.

[38] D. J. Hand, Kernel discriminant analysis, Research Studies Press, New York, 1982.

[39] S. Harmeling, A. Ziehe, M. Kawanabe, B. Blankertz, K. Muller, "Nonlinear blind source separation using kernel feature spaces," In Third Internation Conference on Independent Component Analysis and Signal Separation, 2001.

[40] P. E. Hart, N. J. Nilsson, B. Raphael, Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", SIGART Newsletter, No. 37, pp. 28-29, 1972.

[41] R. A. Horn, C. R. Johnson, Topics in matrix analysis, Cambridge University Press, Cambridge, 1991.

[42] H. Hotelling, "Analysis of a complex of statistical variables into principal components," Journal of Educational Psychology, Vol. 24, pp. 417-441 and 498-520, 1933.

[43] X. Huang, A. Acero, H. W. Hon, Spoken Language Processing, Prentice Hall, 2001.

[44] J. N. Hwang, S. R. Lay, M. Maechler, R. D. Martin, J. Schimert, "Regression Modeling in Back-Propagation and Projection Pursuit Learning," IEEE Trans. on Neural Networks, Vol. 5., No. 3., pp. 342-353, 1994.

[45] A. Hyvärinen, "A family of fixed-point algorithms for independent component analysis," Proceedings of ICASSP, Munich, Germany, 1997.

[46] A. Hyvärinen, E. Oja, "A fast fixed-point algorithm for independent component analysis," Neural Computation, Vol. 9, No. 7, pp. 1483-1492, 1997.

[47] A. Hyvärinen, "New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit," In Advances in Neural Information Processing Systems, Vol. 10, pp. 273-279, MIT Press, 1998.

[48] A. Hyvarinen, J. Karhunen, E. Oja, Independent Component Analysis, Wiley, 2001.

[49] I. J. Jolliffe, Principal Component Analysis, Springer-Verlag, New York, 1986.

[50] C. Jutten, J. Hérault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture", Signal Processing, Vol. 24, No. 1-10, 1991.

[51] K. Karhunen, "Zur Spektraltheorie Stochasticher," Prozesse Ann. Acad. Sci. Fennicae, Vol. 37, 1946.

[52] Kernel Machines Web site, http://kernel-machines.org.

[53] M. Kirby, L. Sirovich, "Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 1, pp. 103-107, 1990.

[54] A. Kocsor, A. Kuba Jr., L. Tóth, "An Overview of the OASIS speech recognition project," Proceedings of the 4th International Conference on Applied Informatics, Eger-Noszvaj, Hungary, pp. 94-102, 1999.

[55] A. Kocsor, A. Kuba Jr., L. Tóth, M. Jelasity, L. Felföldi, T. Gyimóthy, J. Csirik, "A Segment-Based Statistical Speech Recognition System for Isolated/Continuous Number Recognition," Proc. of the FUSST'99, Sagadi,Estonia, pp. 201-211, 1999.

[56] A. Kocsor, A. Kuba Jr., L. Tóth, "Phoneme Classification Using Kernel Principal Component Analysis," Periodica Polytechnica, Vol. 44, No. 1, pp. 77-90, 2000.

[57] A. Kocsor, L. Tóth, A. Kuba Jr., K. Kovács, M. Jelasity, T. Gyimóthy, J. Csirik, "A Comparative Study of Several Feature Transformation and Learning Methods for Phoneme Classification," Int. Journal of Speech Technology, Vol. 3., No. 3/4, pp. 263-276, 2000.

[58] A. Kocsor, L. Tóth, D. Paczolay, "A Nonlinearized Discriminant Analysis and its Application to Speech Impediment Therapy," in V. Matousek et al. (eds.): Proc. of Text, Speech and Dialogue 2001, Springer Verlag LNAI Series, Vol. 2166, pp. 249-257, 2001.

[59] A. Kocsor, J. Csirik, "Fast Independent Component Analysis in Kernel Feature Spaces," in: L. Pacholski, P. Ruzicka (eds.): Proc. of SOFSEM 2001, Springer Verlag LNCS Series, Vol. 2234, pp. 271-281, 2001.

[60] A. Kocsor, K. Kovács, "Kernel Springy Discriminant Analysis and Its Application to a Phonological Awareness teaching System," in P. Sojka et al. (eds.): Proceedings of Text, Speech and Dialogue 2002, Springer Verlag LNAI Series, Vol. 2448, pp. 325-328, 2002.

[61] A. Kocsor, "Kernel Springy Discriminant Analysis," NATO ASI on Learning Theory and Practice (LTP 2002), K.U. Leuven, Belgium, 8th July - 19th July, 2002.

[62] A. Kocsor, L. Tóth, "Application of Kernel-Based Feature Space Transformations and Learning Methods to Phoneme Classification," accepted for Applied Intelligence, 2003.

[63] W. R. Madych, S. A. Nelson, "Multivariate interpolation and conditionally positive definite functions", Math. of Computation, Vol. 54, No. 189, pp. 211-230, 1990.

[64] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," Philos. Trans. Roy. Soc. London, A, Vol. 209, pp. 415-446, 1909.

[65] S. Mika, B. Schölkopf, A. J. Smola, K. R. Müller, M. Scholz, G. Rätsch, "Kernel PCA and De–Noising in Feature Spaces," in: M. S. Kearns, S. A. Solla, D. A. Cohn (eds.), Advances in Neural Information Processing Systems 11, MIT Press, 1999.

[66] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller, "Fisher discriminant analysis with kernels," in: Y.-H. Hu et al. (eds.): Neural Networks for Signal Processing IX, pp. 41-48. IEEE, 1999.

[67] C. A. Micchelli, "Interpolation of scattered data : distance matrices and conditionally positive definite", Constructive Approximation, Vol. 2, pp. 1122, 1986.

[68] D. Michie, D. J. Spiegelhalter, C. C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, 1994.

[69] B. C. J. Moore, An Introduction to the Psychology of Hearing, Academic Press, 1997.

[70] S. K. Murthy, S. Kasif, S. Salzberg, "A System for Induction of Oblique Decision Trees," Journal of Artificial Intelligence Research, Vol. 2, pp. 1-32, 1994.

[71] J. V. Neumann, O. Morgenstern, Theory of Games and Economic Behavior, Princeton University Press, 1947.

[72] E. Oja, Subspace methods of pattern recognition, Volume 6 of Pattern recognition and image processing series, John Wiley & Sons, 1983.

[73] E. Oja, "Neural networks, principal components, and subspaces," International Journal of Neural Systems, Vol. 1, No. 1, pp. 61-68, 1989.

[74] M. Ostendorf, V. Digalakis, O. Kimball, "From HMMs to segment models: a unified view of stochastic modeling for speech recognition," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 4, pp. 360-378, 1996.

[75] M. Ostendorf, "From HMMs to Segment Models: Stochasting Modeling for CSR," In C. H. Lee, F. K. Soong, K. K. Paliwal (eds.): Automatic Speech and Speaker Recognition, Advanced Topics, Kluwer Academic, pp. 185-211, 1996.

[76] P. Pajunen, A. Hyvärinen, J. Karhunen, "Nonlinear blind source separation by self-organizing maps," In Proc. Int. Conf. on Neural Information Processing, pp. 1207-1210, Hong Kong, 1996.

[77] P. Pajunen, J. Karhunen, "A maximum likelihood approach to nonlinear blind source separation," In Proceedings of the 1997 Int. Conf. on Artificial Neural Networks (ICANN'97), pp. 541-546, Lausanne, Switzerland, 1997.

[78] E. Parzen, "On estimation of probability density function and mode", Annals of Mathematical Statistics, Vol. 33, pp. 1065-1076, 1962.

[79] T. Pavlidis, Structural Pattern Recognition, New York, Springer-Verlag, 1977.

[80] K. Pearson, "On lines and planes of closest fit to systems of points in space," The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, 2 (sixth series), pp. 559-572, 1901.

[81] P. Pudil, J. Novovicova, J. Kittler, "Feature Selection Based on the Approximation of Class Densities by Finite Mixtures of the Special Type," Pattern Recognition, Vol. 28, No. 9, pp. 1389-1397, 1995.

[82] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[83] L. Rabiner, B. H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.

[84] V. Roth, V. Steinhage, "Nonlinear discriminant analysis using kernel functions," In S. A. Solla, T. K. Leen, K. R. Müller (eds.), Advances in Neural Information Processing Systems, Vol. 12, pp 568-574, MIT Press, 2000.

[85] D. J. Sawyer, B. J. Fox, Phonological Awareness in Reading: The Evolution of Current Perspectives (Springer Series in Language and Communication, Vol 28), Springer-Verlag, New York, 1991.

[86] I. J. Schoenberg (1938), "Metric spaces and completely monotone functions", Ann. of Math. 39(4), 811-841.

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

[88] B. Schölkopf, A. J. Smola, K. R. Müller, "Kernel principal component analysis," in Advances in Kernel Methods - Support Vector Learning, B. Schölkopf et al. (eds.), MIT Press, Cambridge, pp. 327-352, 1999.

[89] B. Schölkopf, P. L. Bartlett, A. J. Smola, R. Williamson, "Shrinking the tube: a new support vector regression algorithm," in Advances in Neural Information Processing Systems 11, M. S. Kearns et al. (eds.), MIT Press, Cambridge, pp. 330-336, 1999.

[90] B. Schölkopf, A. J. Smola, Learning with Kernels, MIT Press, Cambridge, 2002.

[91] J. Schurmann, Pattern Classification: A Unified View of Statistical and Neural Approaches, John Wiley & Sons, New York, 1996.

[92] A. J. Smola, B. Schölkopf, K. R. Müller, "The connection between regularization operators and support vector kernels," Neural Networks, Vol. 11, pp. 637-649, 1998.

[93] J. A. K. Suykens, J. Vanderwalle, "Least squares support vector machine classifiers," Neural Processing Letters, Vol. 9, No. 3, pp. 293-300, 1999.

[94] M. Szarvas, P. Mihajlik, T. Fegyó, P. Tatai, "Automatic Recognition of Hungarian: Theory and Practice," International Journal of Speech Technology, Vol 3., No. 3/4, pp. 237-252, 2000.

[95] L. Tóth, A. Kocsor, K. Kovács, "A Discriminative Segmental Speech Model and its Application to Hungarian Number Recognition," in: P. Sojka, I Kopecek, K. Pala (eds.), TProceedings of Text, Speech and Dialogue 2002, Springer Verlag LNAI Series, Vol. 1902, pp. 307-313, 2000.

[96] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[97] V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons Inc., 1998.

[98] S. Wattanabe, Pattern Recognition: Human and Mechanical, Wiley, New York, 1985.

[99] J. Xu, X. Zhang, Y. Li, "Kernel MSE algorithm: a unified framework for KFD, LS-SVM and KRR," In Proceedings of IJCNN, pp. 1486-1491, 2001.

[100] H. H. Yang, S.I. Amari, A. Cichocki, "Information-theoretic approach to blind separation of sources in non-linear mixture," Signal Processing, Vol. 64, No. 3, pp. 291-300, 1998.

[101] J. Yoon, "Approximation by Conditionally Positive Definite Functions With Finitely Many Centers," Trends in Approximation Theory, K. Kopotun, T. Lyche, M. Neamtu (eds.), Vanderbilt University Press, Nashville, pp. 437-446, 2001.