KERNEL METHODS AND FACTORIZATION FOR IMAGE AND VIDEO ANALYSIS

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science

by

Ranjeeth Dasineni
200507017
ranjith_d@students.iiit.ac.in

International Institute of Information Technology
Hyderabad, India
November 2007

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Kernel Methods and Factorization for Image and Video Analysis" by Ranjeeth Dasineni, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Advisor: Dr. C. V. Jawahar

To IIIT Hyderabad,
where I learnt all that I know of Computers
and much of what I know of Life

Acknowledgments

I would like to thank Dr. C. V. Jawahar for introducing me to the fields of computer vision and machine learning. I gratefully acknowledge the motivation, technical and philosophical guidance that he has given me throughout my undergraduate and graduate education. His knowledge and valuable suggestions provided the foundation for the work presented in this thesis. I thank Dr P. J. Narayanan for providing an excellent research environment at CVIT, IIIT Hyderabad. His advice on research methodology helped me in facing the challenges of research. I am grateful to the GE Foundation and CVIT for funding my graduate education at IIIT Hyderabad.

I am also grateful to fellow lab mates at CVIT and IIIT Hyderabad for their stimulating company during the past years. While working on this thesis, few researchers across the world lent their valuable time validate my work, check the correctness of my implementations, and provide critical suggestions. I thank all of them for their help.

Finally, I am grateful to my parents who made several sacrifices to make my education a reality. Without their continuous support, I would not have completed this work.

Abstract

One of the important developments in the field of Machine Learning in recent years is the emergence of Kernel Methods. This new class of algorithms combine the stability and efficiency of linear algorithms with the descriptive power of nonlinear features. They form a powerful set of tools for detecting complex nonlinear relationships in the input data while ensuring the statistical stability of the detected relationships. In addition, the kernel paradigm gives the flexibility to treat different types of data (such as numerical and categorical, structured and unstructured) under a single framework and to incorporate prior information in the kernel function. Owing to the efficacy of kernel algorithms, they have been used extensiely to solve a number of problems in various domains such as computer vision, data mining and bioinformatics. Factorization is one of the most extensively used mathematical tools in several domains including computer vision, machine learning and physics. In computer vision, it has been used for infering latent variables that characterize a generative process from the observable outputs. Challenging tasks such as extraction of structure of a scene and motion of the camera from image sequences and extracting prinipal components of the data are solved in an elegant manner using factorization.

Kernel methods allow data to be mapped (implicitly) to a different space, which is often very high dimensional compared to the input space, so that complex patterns in the data become simpler to detect and learn. Factorization, on the other hand, helps in extraction of a small set of latent variables that aid in reducing the redundancy in the representation of data. The two methods apparently work in orthogonal manner: kernels derive more features from existing ones and increase the dimensionality of the data while factorization eliminates redundacies and compresses the data. However, this also suggests that they are two complementary tools that aid in comprehensive analysis of data. In this thesis, we employ these tools to solve problems in image and video analysis. Novel kernel algorithms are used to perform biometric authentication and planar shape recognition. Factorization of tensor representation of video data is used to devise methods for facial expression transer, facial expression recognition and face morphing. We demonstrate that kernelization and factorization form a complementary set of tools through a method to classify typograhical content that uses the two methods.

Chapter 2 reviews kernel methods and demonstrates how the kernel trick can be used to enrich several linear algorithms. The central idea behind kernel methods is to recode the data implicitly using the so called kernel functions. Algorithms that operate using the information provided by the kernel function alone, thus, can be implemented in the transformed space. The chapter introduces the fundamental ideas behind the kernel trick along with elementary theory of kernel functions. We demonstrate how several linear algorithms can be kernelized in this manner. Popular kernel-based methods such as the Support Vector Machine and Kernel Principal Component Analysis are discussed.

In chapter 3, we tackle two separate problems i.e. feature selection and modeling using kernel algorithms. Biometric Authentication using weak features such as hand-geometry needs a powerful feature selection algorithm that extracts the most discriminatory information. The single class framework for authentication, which is used for the sake of efficiency, makes the problem even tougher. We propose the Kernel Multiple Exemplar Biased Discriminant Analysis (KMEBDA) for this task along with a classifier design that handle errors due to similar classes efficiently. The second algorithm, Kernel Linear Predicitve Coding (KLPC), is used for nonlinear modeling of time

series. The prediction coefficients obtained using the method are used as features to perform model-based recognition of planar shapes (from their silhouttes) and handwritten characters. In either case, the kernel variants lead to improvement in performance as compared to their linear versions.

In chapter 4, we represent videos using tensors and use it to alter and synthesize new videos and images that posess certain properties of interest. Tensor representation helps in capturing the structure and redundancies present in videos better. The factors obtained by a positive factorization of such tensors can be viewed as generative models for regions in the image. They also help in identifying dynamic and static content in the image which enables simple solutions to problems such as facial expression transfer. Methods for identification of factors of interest and aligning factors of two different videos are proposed. These methods are used to devise techniques to perform facial expression transfer across different subjects, recognize facial expression and for generation of morph sequence between two face images. The results demonsrate that the tensor representation and tensorial factorization are powerful tools for video analysis.

In chapter 5, we give insights in to the complementary nature of kernelization and factorization for analysis of data. We argue that factorization of kernel matrices is the key to analysis of data using kernel functions. Factorization of kernel matrices gives rise to methods such as kernel principal component analysis, kernel discrimimant analysis. Low rank approximation of kernel matrices and their effect on complexity of kernel algorithms are discussed. We also demonstrate a novel application in typograhpic content classification using kernelization and factorization together.

The contributions of this thesis are : i) Development of two new kernel algorithms that are used to solve two problems in image analysis ii) Using tensorial representation and tensor factorization for face video analysis and iii) Investigations into the role of factorization in the context of kernel methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

1.1.1 Kernel Functions and Factorization in Computer Vision

Image and Video Analysis is one of the most active research areas in computer science with a large number of applications in security, surveillance, broadcast video processing etc. Prior to the past two decades, the primary focus in this domain was on efficient processing of image and video data. However, with the increase in computational power and advancements in Machine Learning, the focus has shifted to a wide range of other problems. Machine learning techniques have been widely used to perform higher level tasks such as recognizing faces from images, facial expression analysis in videos, printed document recognition and video understanding which require extensive analysis of data. The field of Machine Learning itself, witnessed the evolution of Kernel Methods as a principled and efficient approach to analyze nonlinear relationships in the data. The new algorithms are computationally efficient and statistically stable. This is in stark contrast with the previous methods used for nonlinear problems, such as neural networks and decision trees, which often suffered from overfitting and computational expense [1]. In addition, kernel methods provide a natural way to treat heterogeneous data (like categorical data, graphs and sequences) under a unified framework. These advantages led to their immense popularity in many fields, such as computer vision, data mining and bioinformatics. In computer vision, the use of kernel methods such as support vector machine [2], kernel principal component analysis [3] and kernel discriminant analysis [4] resulted in remarkable improvements in performance at tasks such as classification, recognition and feature extraction. Like Kernel Methods, Factorization techniques enabled elegant solutions to many problems in computer vision such as eliminating redundancy in representation of data and analysis of their generative processes. Structure from Motion [5] and Eigen Faces for feature extraction [6] are examples of successful applications of factorization in vision. However, factorization, so far, has been used on the traditional matrix representation of image collection and videos. This representation fails to completely exploit the structure in 2D images as each image is represented using a single 1D vector. Tensors are more natural representations for such data and recently gained wide attention in computer vision [7, 8]. Factorization becomes an even more useful tool with such representations [9, 10].

1.1.2 Problem Statement

While both Kernel Methods and Factorization both aid in analysis of the data and detection of inherent regularities, they do so in orthogonal manner. The central idea in kernel methods is to

work with new sets of features derived from the input set of features. Factorization, on the other hand, operates by eliminating redundant or irrelevant information. Thus, they form a complementary set of tools to analyze data. This thesis addresses the problem of effective manipulation of dimensionality of representation of visual data, using these tools, for solving problems in image analysis. The purpose of this thesis is three fold: i) Demonstrating useful applications of kernel methods to problems in image analysis. New kernel algorithms are developed for feature selection and time series modeling. These are used for biometric authentication using weak features, planar shape recognition and handwritten character recognition. ii) Using the tensor representation and factorization of tensors to solve challenging problems in facial video analysis. These are used to develop simple and efficient methods to perform expression transfer, expression recognition and face morphing. iii) Investigating and demonstrating the complementary nature of Kernelization and Factorization and how they can be used together for analysis of the data.

The applications, on which we demonstrate the methods developed in the thesis, are of practical importance and have been gaining wide attention in computer vision in recent years. Biometric authentication with features such as hand geometry, which are easily obtainable and non-invasive, is being preferred to more invasive methods. However, such features are not strong and need powerful feature selection methods and carefully designed classifiers. Chapter 3 deals with this problem along with planar shape recognition and handwritten character recognition. The latter two problems are solved using a novel kernel algorithm for time series modeling. Chapter 4 deals with three challenging problems in facial video processing : facial expression transfer, facial expression recognition and morphing. The problems are all of practical value with applications in gaming, virtual worlds, entertainment etc. The solutions proposed are complementary to existing methods which use richer information and complex modeling schemes. Typographic style and content classification is useful in printed document analysis and recognition and can be used for font-independent recognition etc. The results obtained using the methods proposed in this thesis indicate that kernelization and factorization are powerful tools for analysis of image and video data.

The following section gives a brief account of evolution of kernel methods and presents, informally, the key ideas behind the kernel approach. Section 1.3 presents the central ideas behind factorization and its applications. Previous work related to the algorithms and applications in thesis is surveyed in section 1.4. Section 1.5 presents the organization of the thesis.

1.2 Evolution of Kernel Methods

Detection of regularities (or patterns) in data generated by a process, natural or artificial, by observing the data is one of the important problems in Machine Learning. Such regularities can be exploited to perform tasks such as prediction, recognition, classification, compression and modeling the process itself. While efficient and stable algorithms, such as the perceptron [11], for detection of simple linear relationships in data are known, the approaches to detection of nonlinear relationships have been less principled. The introduction of backpropagation in neural networks [12] and decision trees [13] during the 1980s and their use for learning nonlinear functions made significant impact in practical applications. The expressive power of these methods led to significant improvement in applications ranging from medical diagnosis to automatic vehicle driving [14]. However, the methods lacked the theoretical elegance and practical simplicity of the linear methods. They depended on greedy and heuristic, search algorithms often suffering from local minima problems [1]. Thus, handling nonlinear relationships involved significantly more effort than handling linear relationships. Practical applications often had to sacrifice the power of nonlinear methods for the simplicity and efficiency of linear methods or vice versa.

However, the scenario changed with the emergence of the theory of learning machines [15] as a principled way to analyze learning algorithms. Learning theory opened up the possibility of improving the statistical stability of existing algorithms. Linear methods were modified to improve their chances of success on unseen data (the so called *large margin* learning machines). This along with a representation of nonlinear functions using certain special class of functions (known as kernel functions), known much earlier but less used in machine learning [16], led to the discovery of the support vector machine [17] for classification. The support vector machine(SVM) found remarkably wide application in practical problems [18, 19, 20] owing to several advantages : i) The support vector algorithm is designed to maximize the performance on unseen samples by reduction of the structural risk of the function inferred. ii) Use of kernel functions allows to handle nonlinear relationships while keeping the risk of overfitting low. iii) The use of kernel functions allows the solution to be searched using convex optimization which avoid local minima.

The expressive power of the SVM and its robustness to local minima is primarily due to the use of representation of nonlinearities using kernel functions. This soon gave rise to a number of other linear methods being used in conjunction with the representation. Many successful linear methods such as principal component analysis and fisher discriminant analysis are *kernelized* [3, 4, 21]. The central idea behind the representation is simple : since learning nonlinearities in the input space of features is difficult, transform the features nonlinearly such that the sought regularities become linear in the transformed domain. Then, the traditional algorithms could be used in the transformed domain. However, the transformation itself is often complex, computationally intensive and requires prohibitively large amount of space. The kernel trick is to use a kernel function, that is simple to compute, which gives the inner product between the two elements in the transformed space. The kernel function takes two elements in the input space and gives the inner product between the images of those elements in the transformed space. This implicit representation is compact, efficient and allows the transformed space to be accessed via the inner product and the input data. However, this also means that algorithms can never access the transformed samples explicitly and therefore have to be reformulated to use only the inner product between pairs of elements. Figure 1.1 outlines the kernel paradigm. Although this appears very restrictive, mathematical properties of linear functions, such as the duality of lines and points, allow linear algorithms to be implemented in the transformed space easily. This can be seen from the number of linear algorithms kernelized in recent years [1].

In addition to the expressive power and statistical stability, kernel methods offer a number of other advantages that make them suitable tools for pattern analysis. Kernel functions do not require the input elements to be vectors. This allows the treatment of different kinds of data such as numerical, symbolic, sequences under a unified framework. Likewise, modifying the kernel function changes the feature set while keeping the algorithm fixed making feature extraction easier. Multiple kernel functions can be used together to combine the power of different feature sets. In addition, the statistical robustness of the algorithm itself stays fixed. In effect, problems such as model selection, incorporation of prior and bootstrapping are all decoupled from the learning algorithm and can be addressed by designing the appropriate kernel. Selection of a kernel function that is optimal for a given task is one of the most challenging problems in the area of kernel methods. Despite this, kernel methods continue to be the preferred scheme for nonlinear pattern analysis. Chapters 2 and 3 are dedicated to the study of these methods and development of new kernel algorithms.

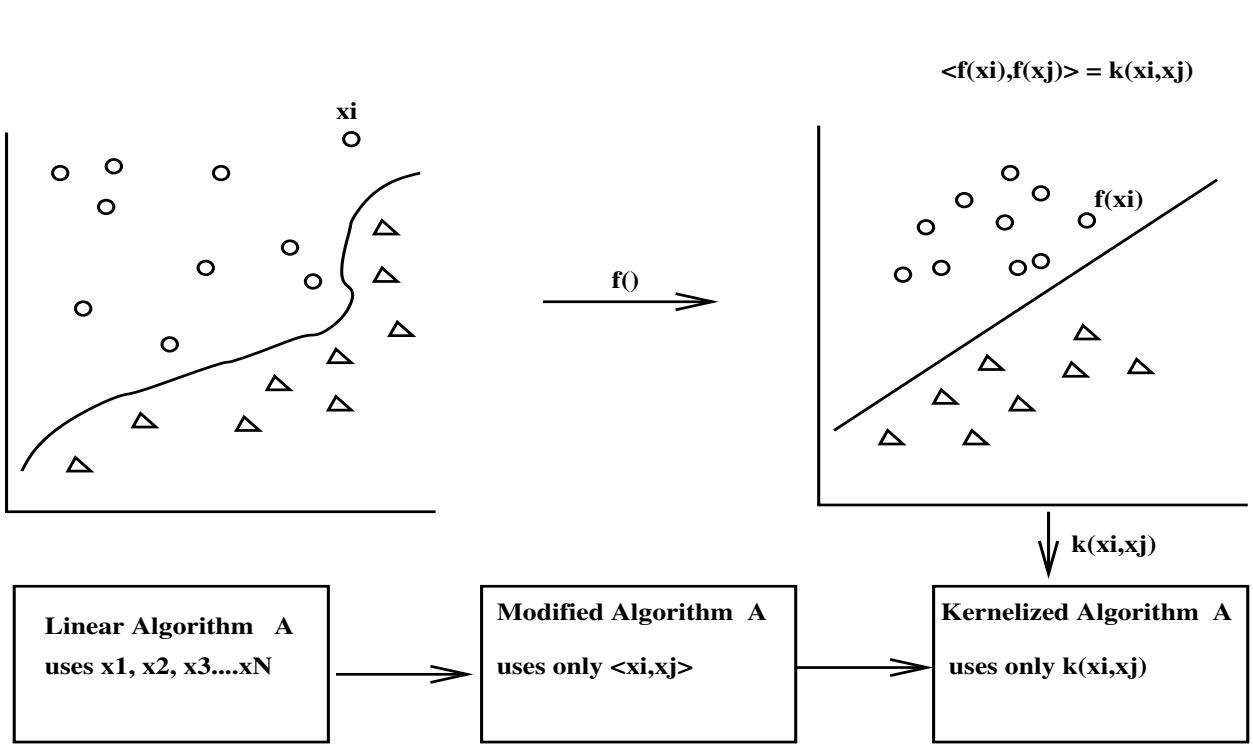


Figure 1.1: Overview of the kernel approach. Given input samples the map $\mathbf{f}(.)$ is the nonlinear mapping that transforms the data. However, the map is never explicitly computed. Instead, it is accessed via the kernel function $\kappa(.,.)$ that gives the inner product in the transformed space. Algorithms use this information alone to learn the required function in the transformed space.

1.3 Factorization for Data Analysis

Factorization is the second tool studied in this thesis. It is a popular tool used for a number of applications in computer vision. Although the term *factorization* typically refers to factorization of matrices into factors, the term is used in a slightly different sense in the current context. Factors are variables that characterize the generative process and can be inferred from the observations. The factors along with a *factor model* i.e., the manner in which they interact to produce the outputs complete the description of the generative process (and thus, of the data). Extraction of these factors is factorization (also called factor analysis). For instance, the images in a video can be reconstructed, given the scene geometry, motion of the camera and the projection model from world coordinates to the image coordinates [5]. Here, the scene geometry and motion of the camera are the factors and the projection model is the factor model. For simple factor models such as the bilinear model used in the above example, factors can be extracted by factoring certain measurement matrices. In such cases, matrix factorization is a technique used to learn the factors. However, other factor models and more complex methods to infer such factors exist [22]. The factorization methods considered in this thesis use bilinear or multilinear factor models. Hence, the focus is primarily on matrix and tensor factorization. Figure 1.2 shows the key ideas behind factorization. Factorization helps in identifying a small set of variables that characterize the source process generating the data. These variables aid in a number of tasks : i) Along with the factor model they provide a compact representation of the data. ii) Generation of additional synthetic observations by simulating the model iii) Given a new sample, verifying whether it was generated by the source process. In addition, the factors themselves may have meaningful interpretations

and can be used for other tasks as in the case of Structure from Motion [5] and Style and Content Separation [23]. Matrix factorization is also a robust method to solve overconstrained system of linear equations and has been used for many tasks such as fundamental matrix estimation in computer vision [24].

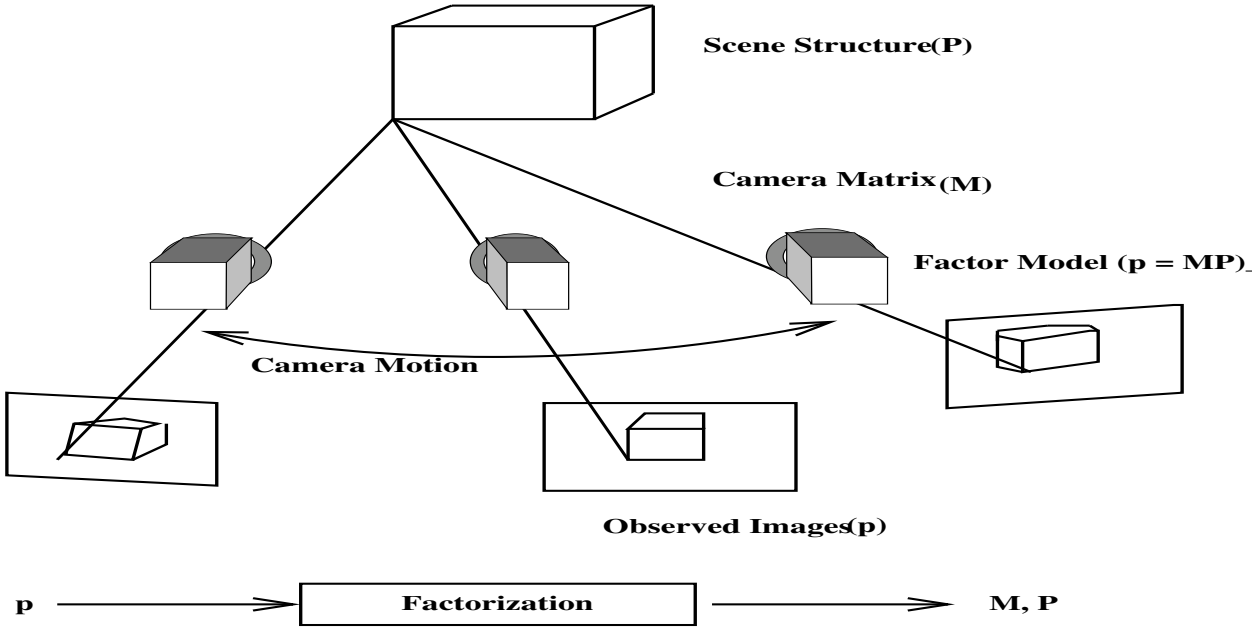


Figure 1.2: Overview of Factorization. Given input samples and factor model the factors are learnt from the observed data. The factors can then be used for a number of tasks such as compression, prediction and verification.

Although factorization is very useful tool, computer vision applications used the method, thus far, in conjunction with the matrix representation of image collections. This representation is unnatural for image data as images are vectorized, which disrupts their natural 2D structure. The tensor representation of image data captures the spatial coherencies better. Tensor representation and multilinear algebra are used extensively in computer vision during recent times [7, 8]. The discovery of methods to factorize tensors into low rank factors [25, 26, 9] led to powerful applications such as expression decomposition [27] and compression [10]. With the emergence of these new representations and efficient algorithms, factorization continues to be an elegant technique for analysis of data. Chapter 4 uses the tensor representation and non-negative tensor factorization to perform face video alternation.

1.4 Related Work

1.4.1 Kernel Methods

The literature concerning kernel methods can be broadly classified in to three categories : i) The Theory of kernel methods and kernel functions: Work in this class primarily deals with representation of nonlinearities using kernel functions, representation of linear functions in feature spaces, properties of kernel functions etc. ii) Development of new kernel algorithms: This class deals with kernelization of linear algorithms for various tasks such as classification, regression, feature selection and modeling. iii) Design of kernels : This class deals with challenging problems in kernel design

such as selection of optimal kernel function for a task, incorporation of prior and invariances in to kernel functions etc. While the first class laid foundations for kernel methods, the second furthered the use of kernel trick extensively and the third class attempts to tackle the future challenges.

The Theory of Kernel Functions: The origin of kernel functions dates back 1904, used by David Hilbert in the context of integral equations [28]. Mercer's theorem, the result key to the kernel trick, was first proposed and proved in 1909 [29]. Interpretation of the kernel function as inner products in a feature space, using the Mercer theorem, was done by Aizerman *et.al* [16]. Although [16] is the closest work to the modern use of kernel functions, kernels were used in machine learning in the context of neural networks [30] and regularization theory [31] extensively. Other theoretical results such as the positivity of the kernel matrix, when Mercer kernels are used, also have been used in machine learning [32]. However, the reintroduction of kernel functions into learning machines was done by Boser *et.al* [17]. They combined kernel function with large margin hyperplanes, leading to new learning algorithm : the Support Vector Machine(SVM). This landmark paper marked the beginning of the use of kernel functions with linear algorithms leading to a new class of algorithms.

Kernelization of Linear Algorithms: The success of support vector machine was primarily due to the expressive power combined with high generalization ability. This led to the use of the kernel trick with a number of linear algorithms for a variety of tasks such as feature extraction, feature selection, classification and modeling. Kernel Principal Component Analysis [3] is the notable among them. The development of kernel PCA not only gave rise to powerful feature extraction technique but also demonstrated that the second order statistics of the data in the feature space can be accessed using the kernel matrix. Using this fact, other linear algorithms, which analyzed the second order statistics to infer relationships, were kernelized. Kernel Fisher Discriminant Analysis [4], Kernel Independent Component Analysis [21] are two popular examples of such algorithms. Gaussian Mixture Modeling, which also uses the second order statistics, is a popular tool for many tasks in computer vision, such as background modeling and tracking. It has been kernelized by Wang *et.al* [33]. The kernelization of algorithms was catalyzed by the success of such methods in various practical applications [19, 34, 35, 36, 37]. The kernel variant of Biased Discriminant Analysis [38], the algorithm used in chapter 3 was used for content-based image retrieval [39, 40] systems. An extensive of list kernelized algorithms can be found [1].

Kernel Design and Selection: One of the important implications of the kernel paradigm is that problems such as feature extraction, model selection, bootstrapping can all be handled by using an appropriate kernel. Hence, kernel design and selection are the most actively pursued problems in the area of kernel methods. Table 1.1 lists the popular kernels designed for several kinds of data. The recently developed Binet-Cauchy kernels [41] provide a unifying framework for many of these kernels. Kernel design by incorporation of prior information, about the domain of features and the relationships sought, is a simpler task than kernel selection i.e., deciding the optimal kernel to be used for a given task, which remains largely unsolved. However, solutions for kernel selection under certain constraints have been proposed in the past. Lanckreit *et. al* [42] use semi-definite programming to learn the kernel matrix in a transductive setting. Convex algorithms for kernel selection suitable for Support Vector Machines were proposed by Jebara [43]. Kim*et.al* [44] formulate the kernel selection problem as a tractable convex optimization problem and select the optimal kernel for Kernel Fisher Discriminant Analysis. Despite these developments, the selection of kernel function optimal for a given task is still open and is promising direction for future research.

Category	Kernels	References
Feature Vectors	Polynomial Kernels, Gaussian Radial Basis Kernels, Hyberbolic Tangent Kernels	[15, 17]
Strings and Sequences	Bag-of-words kernel (Joachims 1998), Fisher kernel (Jaakkola and Haussler 1999), Dynamic Alignment (Watkins 2000), Spectrum kernel(Leslie *et.al* 2002)	[36, 45, 46, 47]
Sets and Distributions	P-kernels (Haussler 1999), Diffusion Kernels(Kondor and Lafferty 2002), Tree kernel(Vert 2002), All Subsets kernel(Takimoto and Warmuth 2002), Probability product kernel(Jebara and Kondor)	[48, 49, 50, 51, 52]
Image and Object Retrieval	Hausdorff kernel(Barla *et.al*), Histogram Intersection kernel(Boughorbel *et.al*)	[53, 54]

Table 1.1: Popular kernels for various types of data and applications.

1.4.2 Factorization

Factorization and Factor Analysis are extensively used in computer vision and machine learning for a number of tasks. The most popular among the applications is the extraction of structure of a scene and motion of the camera from images taken with a moving camera(known as structure from motion). The method presented by Tomasi and Kanade [5] uses singular value decomposition of a measurement matrix composed of tracked feature points. The low rank factors represent the scene structure and camera motion. The simple bilinear factor model enables an elegant solution to the problem. This landmark work led to number of algorithms that used factorization for similar problems. Poelman and Kanade [55] use Cholesky decomposition for recovery of shape and motion under the assumption that the camera is paraperspective. A related family of methods was proposed by Triggs [56]. Extensions of the method to dynamic scenes [57], projective cameras [58] and with missing data [59] have made extensive use of factorization.

Factorization also forms the basis of many linear subspace methods such as Principal Component Analysis [6] and Fisher Discriminant Analysis [60]. These methods have been used widely for feature extraction [61] and extraction of discriminative information [62]. While factors obtained using such methods are used for tasks such as compression, recognition and discrimination, the factor themselves do not have meaningful interpretations as in the case of structure from motion. Freeman and Tanenbaum [23] model the interaction of style and content (the factors) using a bilinear model (the factor model) and develop algorithms to extract the factors. The factors are then used to perform tasks such as classification according to style or content, transfer of style or content and extrapolating style and content. Another popular use of factorization is in solving overconstrained systems of linear equations. Eigen/Singular vectors corresponding to the least eigen/singular value of the coefficient matrix corresponding to system of equations are used as good initial values for iterative estimation of the solution vectors. In computer vision this is used for tasks such as estimation of the fundamental matrix between two views [24]. More recently, positive factorization of matrices was used to obtain sparse basis for representation of images [63]. The fact that sparse basis corresponds to a local parts decomposition evoked great interest in positive factorization methods.

While the matrix representation of data and their factorization was used widely in the past,

the tensor representation of image data is gaining wide attention in computer vision during recent years [7, 64]. The tensor representation is more natural for image and video representation and the successful use of the representation for face recognition [7] and image encoding [64] demonstrates this. Factorization of tensors has been used in the past for face transfer [8] and expression decomposition [27]. As in the case of matrices, positive factorization result in sparse basis. However, as the tensor representation captures redundancies better, the basis was sparser than that in the matrix case [9, 26]. As shown in chapter 3 this enables solutions to challenging problems such as expression transfer.

Factor models that are linear or multilinear in factors are popular due to the elegant solutions they enable. However, such factor models are not rich enough to capture complex interactions involving multiple factors. Other methods for factor analysis do exist [22, 65]. However, the solutions are typically complex using computationally intensive methods such as Expectation Maximization. They have been used for applications like clustering [65], probabilistic principal component analysis [66], dynamic event analysis [67] etc. The efficiency and simplicity of linear/multilinear factor models make them preferred choice despite their limited capacity.

1.4.3 Applications

The applications considered in this thesis have been investigated in the past by several researchers. Below, we indicate how the methods proposed in this thesis differ from previous methods.

Biometric Authentication: With the growing emphasis on security, biometric-trait based authentication is one of the active research areas with number of practical applications. Traits such as finger prints, iris and face form strong personal traits that can be used for authentication [68, 69, 70]. However, the acquisition of these features is often tedious as care must be taken to sense them with the required precision. The acquisition process is often invasive and uncomfortable to the person, making authentication systems based on these traits less acceptable for general use. Noninvasive features such as hand-geometry pose a different problem: such traits inherently do not have enough discriminatory information to distinguish between persons (particularly if the population, on which they are used, is large). Their use for authentication requires strong feature selection techniques that can boost their performance. In addition, efficiency concerns, small number of samples, multimodal distributions complicate the problem further. Although feature selection methods addressing the problems individually were proposed in the past [38, 40, 71, 72], there is no single solution that handles all these problems. The method proposed in this thesis consists of strong nonlinear feature selection technique that is robust to the number of samples and multimodal distributions. In addition, a classifier that can handle highly similar classes is used in conjunction with the feature selection scheme to enhance the performance of the authentication system.

Planar Shape and Online Handwriting Recognition: Planar shape recognition has been done in many ways using a different sets of features [73, 74, 75, 76]. The focus, in the current work, is on the use of time series representation of the silhouettes and model-based recognition of such series. This representation is used earlier in conjunction with autoregressive models [77, 78]. However, the linear nature of the models used in these methods restricts the expressive power. We use the kernel trick to enrich autoregressive models and use them to improve the recognition accuracy.

Expression Analysis and Morphing: The appearance of facial expression in images/videos is a result of a complex phenomenon that can be characterized only with the use of rich information such as muscular motion, texture, shape etc. Past methods for expression transfer [79, 80, 81, 82, 27, 8], expression recognition [83, 84, 85, 86] and morphing [80, 87, 88] use such information

extensively. The methods are computationally expensive and it is not feasible, in all cases, to extract the information they use. The method proposed in the thesis uses appearance information alone to perform these takes. The methods not only provide simpler alternatives to the existing methods, but also are complementary to these methods. In addition the novel use of tensor representation and tensor factorization is a distinguishing feature of the methods proposed.

Typographic Content Classification: The method proposed is a kernelized version of the style and content separation using an asymmetric bilinear model proposed by Freeman and Tanenbaum [23]. The nonlinearity introduced by the kernel function enriches the model. The results indicate that the kernel variant of the method is superior to the linear one.

1.5 Organization of the Thesis

Chapter 1 provides a broad overview of the thesis. The two important techniques used in the thesis, kernelization and factorization, are introduced, informally. Previous work related to these techniques and the various applications considered in thesis are reviewed. Chapter 2 reviews kernel methods and demonstrates how the kernel trick can be used to enrich several linear algorithms. The central idea behind kernel methods is to recode the data implicitly using the so called kernel functions. Algorithms that operate using the information provided by the kernel function alone, thus, can be implemented in the transformed space. The chapter introduces the fundamental ideas behind the kernel trick along with the elementary theory of kernel functions. Our presentation of the kernel trick makes extensive use of the data matrix notation and demonstrates how this notation makes kernelization elegant. Several algorithms are kernelized as examples of the kernel trick. Popular kernel-based methods such as the Support Vector Machine and Kernel Principal Component Analysis are discussed.

In chapter 3, we tackle two separate problems i.e. feature selection and modeling using kernel algorithms. Biometric Authentication using weak features such as hand-geometry needs a powerful feature selection algorithm that extracts the most discriminatory information. The single class framework for authentication, which is used for the sake of efficiency, makes the problem even tougher. We propose the Kernel Multiple Exemplar Biased Discriminant Analysis (KMEBDA) for this task along with a classifier design that handle errors due to similar classes efficiently. The second algorithm, Kernel Linear Predicitve Coding (KLPC), is used for nonlinear modeling of time series. The prediction coefficients obtained using the method are used as features to perform model-based recognition of planar shapes (from their silhouettes) and handwritten characters. In either case, the kernel variants lead to improvement in performance as compared to their linear versions.

In chapter 4, we represent videos using tensors and use the represenation, along with tensor factorization, to alter and synthesize new videos and images that possess certain properties of interest. Tensor representation helps in capturing structure and redundancies present in videos better. The factors obtained by a positive factorization of such tensor can be viewed as generative models for regions in the image. They also help in identifying dynamic and static content in the image which enables simple solutions to problems such as facial expression transfer. Methods for identification of factors of interest and aligning factors of two different videos are proposed. These methods are used to devise techniques to perform facial expression transfer across different subjects, recognize facial expression and for generation of morph sequence between two face images. The results demonstrate that the tensor representation and tensorial factorization are powerful tools for video analysis.

In chapter 5, we give insights into the complementary nature of kernelization and factorization for analysis of data. We argue that factorization of kernel matrices is the key to analysis of data using

kernel functions. Factorization of kernel matrices gives rise to methods such as kernel principal component analysis, kernel discriminant analysis. Low rank approximation of kernel matrices and their effect on complexity of kernel algorithms are discussed. We also demonstrate a novel application in typographic content classification using kernelization and factorization together.

Thus, the contributions of this thesis are : i) Development of two new kernel algorithms that are used to solve two problems in image analysis ii) Using tensorial representation and tensor factorization for face video analysis and iii) Investigations into the role of factorization in the context of kernel methods.

1.5.1 Note to the Reader

Chapter 2 is written as a tutorial introduction to kernel methods. It is not a prerequisite for understanding the following chapters. It is intended for readers who are unfamiliar with kernel methods. Readers who are initiated to the field may skip the chapter without losing continuity. Section 2.2 provides sufficient information to understand and appreciate the central ideas concerning kernel methods. Section 2.3 and the following sections provide additional details on the theory of kernel methods and kernelization. Readers seeking an elementary introduction to kernel methods may read till section 2.2 and skip the rest of the chapter. The figures in the thesis are best appreciated when viewed in the electronic format. These can be viewed in the soft copy of thesis available at *http://research.iiit.ac.in/˜ ranjith/thesis.pdf*.

Chapter 2

A Review Of Kernel Methods

2.1 Introduction

Kernel Methods received wide attention in the last one and a half decade and are used extensively in a number of domains such as computer vision, bio-informatics, data mining, pattern recognition etc [19, 34, 20, 36, 89]. Traditionally, linear algorithms are the first choice for a large number of tasks such as classification, unsupervised learning etc. This is due to the numerical and statistical stability of linear algorithms. Linear relationships are easier to detect from data and are the simplest and most natural estimate of an unknown relationship among several variables. A number of linear methods such as Principal Component Analysis (PCA) [6], Linear Discriminant Analysis(LDA) [60], Linear Predictive Coding [90] are used widely for tasks involving compression, recognition, detection, modeling and synthesis. However, as the complexity of the tasks increases the low descriptive power of linear functions turns out to be a handicap. Nonlinear functions, on the other hand, are more descriptive but the estimation of such functions is fraught with problems concerning stability (both numerical and statistical) and convergence. The approaches to detection of nonlinear relationships in data were rather less principled than linear algorithms [1]. For a long time, in both research world and the industry, one of the complementary benefits of these two classes of algorithms was sacrificed in favor of the other (based on the complexity of the task at hand) since there was no way of combining these two advantages.

The introduction of the Support Vector Machine(SVM) [17] for the classification problem first demonstrated the use of *kernel functions* [16, 91] to combine the advantages of linear and nonlinear functions. The essence of the method is to implement a linear classifier in a feature space that is nonlinearly related to the input space *without* explicitly accessing the feature space. The fundamental idea is that a complex relationship in the input data can be simplified by recoding the data in an appropriate manner. Although this idea was in vogue in the domain of nonlinear modeling for a long time, explicit recoding of data is prohibitively expensive in a number of applications. The paradigm was, thus, of little use for problems involving high-dimensional data. However, using the *kernel function* to indirectly access the recoded data via the inner product makes estimation of linear functions feasible. Ever since the introduction of SVM, a number of successful linear algorithms such as PCA, LDA are *kernelized* i.e. used the kernel trick to incorporate the power of nonlinearity [3, 4, 33]. The resulting algorithms are superior to their linear counterparts in terms of descriptive power and are as stable.

Descriptive power is only one among the many advantages brought by the use of kernel functions. A kernelized algorithm is akin to template definitions in software : Any kernel function can be used with a kernelized algorithm without effecting the statistical properties, such as generalization

capability, of the algorithm (in case of classification algorithms). This allows domain specific knowledge (or *prior*) to be incorporated in to the kernel function without changing the algorithm. This modularity makes the development of powerful and stable algorithms feasible. The theory of kernels does not place any restrictions on the input space: the input data samples can be numerical values, vectors, text strings, sets, graphs etc. Thus, the kernel trick allows a number of algorithms that are designed for numeric data to be applied to non-numerical data. This enhances the applicability of the these algorithms and increases the number of tools available for analysis of heterogeneous data. Several other advantages of kernel methods will be described in the following sections. The underlying theory of kernel methods is covered in a number of books [1, 92, 93, 94]. In the following section, the kernel trick is introduced with the example of *kernel perceptron*. The essentials of theory of kernel methods, popular kernel methods, their advantages and disadvantages, current challenges in the field are reviewed in the later sections.

2.2 The kernel trick : An example

Pattern analysis refers to the task of finding inherent regularities in the input data from a finite sample. Such regularities characterize the source distribution generating the data and aid in a number of tasks such as i) compression of data, ii) classification of unseen samples, iii) prediction of unseen values, iv) detecting outliers i.e. samples that are not likely from the source distribution etc. Linear relationships are the simplest of such regularities. Detecting such relationships is both numerically and statistically stable. One such method, the *perceptron* algorithm [11] is described below. The use of kernel trick to increase the descriptive power of the perceptron algorithm is demonstrated.

2.2.1 The perceptron algorithm

Binary pattern classification is one of the popular problems in machine learning. Given an input data set : $\mathcal{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathcal{R}^m$ and $y_i \in \pm 1$ for $i = 1, 2, \cdots, N$, the task is to find a function $f : \mathcal{R}^m \rightarrow \pm 1$ which learns the relationship between $\mathbf{x}_i$ and y_i. Each $\mathbf{x}_i$ is an *input pattern* and y_i is its *label*. The function $f(.)$ can be used for a number of purposes such as i) prediction of label y_t for a test sample $\mathbf{x}_t$ ii) verifying the authenticity of a new pattern-label pair $(\mathbf{x_t}, y_t)$ etc. The perceptron algorithm [11] sets out by assuming that a function, $f(.)$, that is linear in $\mathbf{x}$ can satisfy the constraints $f(\mathbf{x}_i) = y_i, \forall\ i \in \{1, 2, \cdots, N\}$. If such a function existsi, the set $\mathcal{X}$ is said to be *linearly separable*. The function $f(.)$, is of the following form:

$$y = sign(\langle \mathbf{w}, \mathbf{x} \rangle + w_0) \tag{2.1}$$

where $\mathbf{w} \in \mathcal{R}^m$ and w_0 is a scalar. In the following discussion the scalar w_0 is absorbed in to the weight $\mathbf{w}$ by augmenting the samples with a constant term: $\mathbf{x}_i^t = [\mathbf{x}_i^t \quad 1]$ and $\mathbf{w}^t = [\mathbf{w}_i^t \quad w_0]$. This allows us to deal with functions of the form $sign(\langle \mathbf{w}, \mathbf{x} \rangle)$ i.e., functions linear in $\mathbf{x}$. The intuition behind this form is that similar objects are mapped to similar classes, since

$$|\langle \mathbf{w}, \mathbf{x}_1 \rangle - \langle \mathbf{w}, \mathbf{x}_2 \rangle| = |\langle \mathbf{w}, \mathbf{x}_1 - \mathbf{x}_2 \rangle| \leq ||\mathbf{w}|| \cdot ||\mathbf{x_1} - \mathbf{x}_2|| \tag{2.2}$$

where the inequality is a result of the Cauchy-Schwartz inequality. The solution does not change if the solution vector $\mathbf{w}$ is rescaled since for $\lambda \neq 0$,

$$sign(\langle \lambda \mathbf{w}, \mathbf{x} \rangle) = sign(\lambda \langle \mathbf{w}, \mathbf{x} \rangle) = sign(\langle \mathbf{w}, \mathbf{x} \rangle) \tag{2.3}$$

Require: $\exists \mathbf{w} \ni y_i = sign(\langle \mathbf{w}, \mathbf{x}_i \rangle)$ for $i = 1, 2, \cdots, N$

$\mathbf{w} = \mathbf{0}$
repeat
  for $i = 1$ to N **do**
    if $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
    end if
  end for
until no sample is misclassified

Algorithm 1: PerceptronPrimal($\{(\mathbf{x}_i, y_i)\}$ for $i = 1, 2, \cdots, N$)

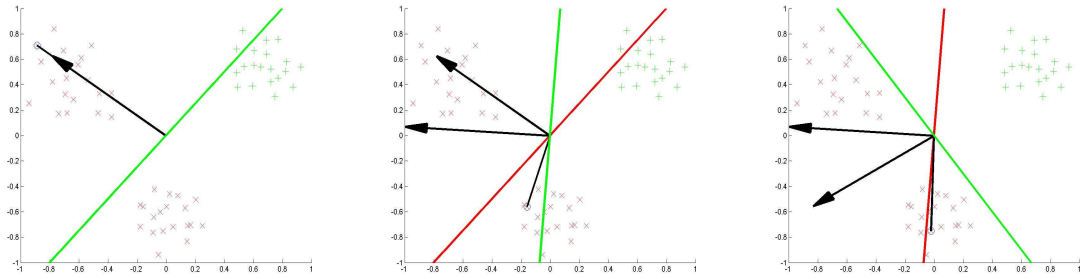


Figure 2.1: The adjustment of the plane $\mathbf{w}$ in the perceptron algorithm. Each misclassified sample moves the plane (plane in red) to reduce the number of errors in the next iteration (plane in green).

This ambiguity is removed by enforcing an additional constraint that $||\mathbf{w}|| = 1$. An iterative procedure to estimate $\mathbf{w}$ is proposed by Rosenblatt [11] which operates as shown in Algorithm 1.

The intuition behind the perceptron algorithm is simple : if a sample is misclassified i.e. it lies on the *wrong* side of the plane described by $\mathbf{w}$, then the plane is moved *towards* the sample either by adding/subtracting that sample to the vector $\mathbf{w}$. This reduces the distance between the sample and the plane. The plane, eventually, moves past the misclassified sample so that it is classified correctly. Figure 2.1 shows how this adjustment of $\mathbf{w}$ operates. The figure also shows how a plane in this space can be viewed as equivalent to point. The vector describing this point is perpendicular to the plane. This duality of points and planes allows many problems to be recast in an alternate form known as the *dual form*. Algorithm 1 is the *primal form* of the perceptron algorithm. An interesting feature of the perceptron algorithm is that the final solution vector $\mathbf{w}$ is always a linear combination of the points in the input data i.e.

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \tag{2.4}$$

Estimating the coefficients $\alpha_1, \cdots, \alpha_N$, is equivalent to estimating $\mathbf{w}$. Additionally, the projection of a sample $\mathbf{x}_t$ on to $\mathbf{w}$ can be computed using the coefficients α_i and the input samples $\mathbf{x}_i$ where $i \in \{1, 2, \cdots, N\}$ since

$$\langle \mathbf{w}, \mathbf{x}_t \rangle = \langle \sum_{i=1}^{N} \alpha_i \mathbf{x}_i, \mathbf{x}_t \rangle = \sum_{i=1}^{N} \alpha_i \langle \mathbf{x}_i, \mathbf{x}_t \rangle \tag{2.5}$$

This computation in the dual form is more expensive than it is in the primal form. However, the

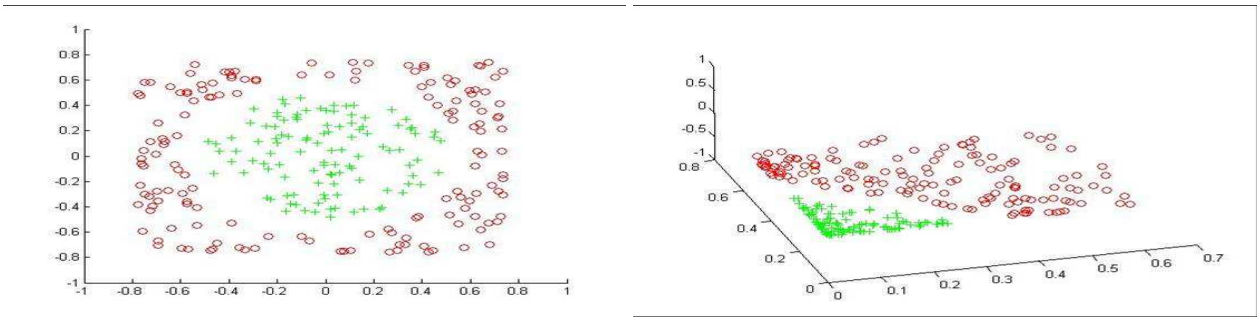


Figure 2.2: Linearization of nonlinear relationships by recoding the data. The circular boundary in $\mathcal{R}^2$ becomes a plane in $\mathcal{R}^3$.

dual form results in other advantages in terms of expressive power. The perceptron algorithm in its dual form is given in algorithm 2. To speeden the estimation of α_i, the inner products between the input data samples can be precomputed and stored in a matrix $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, know as the Gram Matrix.

Require: $\exists \alpha_j \ni y_i = sign(\langle \sum_{j=1}^{N} \alpha_j \mathbf{x}_j, \mathbf{x}_i \rangle)$ for $i, j \in \{1, 2, \cdots, N\}$
1: $\alpha_i = 0$ for $i = 1, 2, \cdots, N$
2: compute the Gram Matrix $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
3: **repeat**
4: **for** $i = 1$ to N **do**
5: **if** $y_i \sum_{j=1}^{N} \alpha_j \mathbf{G}_{ij} \leq 0$ **then**
6: $\alpha_i \leftarrow \alpha_i + y_i$
7: **end if**
8: **end for**
9: **until** no sample is misclassified

Algorithm 2: PerceptronDual($\{(\mathbf{x}_i, y_i)\}$ for $i = 1, 2, \cdots, N$)

2.2.2 Linearization of Nonlinearities

While the simplicity of the perceptron algorithm makes it very attractive, the lack of expressive power is a big disadvantage. Many practical applications involve nonlinear relationships and hence can not use the perceptron algorithm. An example of linearly non-separable data is shown in figure 2.2. The perceptron algorithm is guaranteed to converge whenever the data is linearly separable, but is not guaranteed to converge for data that is not linearly separable. This limits its applicability to a very small number of cases.

A well known technique to handle nonlinear relationships using linear algorithms is to *linearize* the relationships by transforming the data appropriately. For instance, the boundary between the positive and negative samples in figure 2.2 is described by a circle of radius R centered at the origin, $x_1^2 + x_2^2 - R^2 = 0$ in $\mathcal{R}^2$ where each sample is described by $\mathbf{x} = [x_1 \quad x_2]^t$. If the data is mapped, to $\mathcal{R}^2$ using the map $\phi : \mathcal{R}^2 \mapsto \mathcal{R}^3$, such that all possible monomials of second degree form the entries of the transformed vector :

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) \mapsto \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix} \tag{2.6}$$

then the boundary becomes linear in the transformed space described by $\mathbf{W}^t\phi(\mathbf{x}) - R^2 = 0$ where $\mathbf{W} = [1 \quad 1 \quad 0]^t$. All relationships that can be expressed as second order polynomials in the input feature space can be linearized with the map $\phi()$. Similarly, more complex relationships can be linearized by using an appropriate mapping. The perceptron algorithm can be applied to the transformed data, thus, extending it to handle nonlinear relationships.

The technique presented above, however, does not scale with size and dimensionality of the input data. For instance, for input samples of dimensionality m, the number of all possible degree d monomials is

$$D = \frac{(m+d-1)!}{(m-1)!d!} \tag{2.7}$$

A typical recognition problem in computer vision involves images of size 256×256 pixels. A common representation of such data is to form a feature vector by stacking all the pixel values, resulting in vectors of dimensionality of $m = 65536$. The number of all possible second degree monomials, in this case is, $(65537 \times 65536)/2 = 214756416$. The numbers increase exponentially with the data size and the degree of the monomials, rendering their explicit evaluation prohibitively expensive. However, the perceptron algorithm in its dual form does not require the data to be explicitly recoded. Instead, only the Gram Matrix in the transformed space (called the *feature space*), $\mathbf{G}_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, is required. Similarly, classification of a test sample, $\mathbf{x}_t$ requires the inner products $\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_t)\rangle$ alone. Thus, if the inner product between two samples in the feature space can be computed efficiently, then the dual form can be used to handle complex relationships efficiently.

2.2.3 The Kernel Trick

The kernel trick enables the efficient computation of inner product in the feature space. Consider the following mapping, which is similar to the mapping in Equation 2.6 except that all ordered products of second degree are taken as the entries of the transformed vector:

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) \mapsto \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_2 x_1 \end{bmatrix} \tag{2.8}$$

This alternate mapping makes it possible to evaluate the inner product easily since,

$$\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 x_2 y_2 = (x_1 y_1 + x_2 y_2)^2 = \langle\mathbf{x}, \mathbf{y}\rangle^2 \tag{2.9}$$

The above result can be generalized to any dimension and degree [17]: If $\phi(\mathbf{x})$ maps m-dimensional vector $\mathbf{x}$ to a vector whose entries are all possible degree d ordered products of components of $\mathbf{x}$, then the inner product can be expressed as

$$\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle = \sum_{i_1,\cdots,i_d}^{m} x_{i_1} \cdots x_{i_d} y_{i_1} \cdots y_{i_d} = (\langle\mathbf{x}, \mathbf{y}\rangle)^d \tag{2.10}$$

Alternate forms of $\phi()$, that consider only the unordered products, but weight them proportional to the frequency of their occurrence result in the same inner product. For instance, for the $m = 2$

and $d = 2$ the map $\phi(\mathbf{x}) = [x_1 \quad x_2 \quad \sqrt{2}x_1x_2]$ would result in the same inner product as the map in equation 2.8. The functional form the of the inner product

$$\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{2.11}$$

is called the *kernel function*. The kernel function is simple to evaluate and bypasses the computationally intensive computation of $\phi(.)$. Further, concatenating an additional constant term 1 to $\phi(\mathbf{x})$ is equivalent to modifying the kernel function as follows

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \begin{bmatrix} \phi(\mathbf{x}) \\ 1 \end{bmatrix}, \begin{bmatrix} \phi(\mathbf{y}) \\ 1 \end{bmatrix} \rangle = 1 + \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = 1 + (\langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{2.12}$$

The transformed version of the Gram matrix, $\mathbf{G}$, called the kernel matrix $\mathbf{K}$ can now be computed efficiently, since $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{y}_j)$. The *kernelized* version of the perceptron algorithm is given in algorithm 3.

1: $\alpha_i = 0$ for $i = 1, 2, \cdots, N$
2: compute the Kernel Matrix $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$
3: **repeat**
4: **for** $i = 1$ to N **do**
5: **if** $y_i \sum_{j=1}^{N} \alpha_j \mathbf{K}_{ij} \leq 0$ **then**
6: $\alpha_i \leftarrow \alpha_i + y_i$
7: **end if**
8: **end for**
9: **until** no sample is misclassified

Algorithm 3: KernelPerceptron($\{(\mathbf{x}_i, y_i)\}$ for $i = 1, 2, \cdots, N$, $\kappa(.,.)$)

The kernel perceptron algorithm described above, takes the higher order statistics of the data in to account without explicitly computing the higher order terms (the complexity of which increases combinatorially), while the original perceptron algorithm fails to do so. Figure 2.3 shows how the kernel perceptron captures nonlinear relationships in the data. The traditional perceptron algorithm failed to converge in this case. The images in figure 2.3 show the input space and the how the relationship learned using a kernel function looks like in the input space. Such images a useful for visualizing the advantages of using a kernel function and will be used throughout this thesis for demonstrating algorithms using synthetic data.
The kernel functions of the form in equation 2.11 are called *polynomial* kernels. Even this simple family of inner products in the feature space enables us to detect a wide variety of nonlinearities since the kernels can be combined to give rise to other inner products. For instance, consider kernel function $\kappa_1(.)$ and $\kappa_2(,)$ corresponding to to two mappings $\phi_1()$ and $\phi_2()$. The inner product in the feature space that includes features from both the mappings is equivalent to :

$$\begin{aligned} \kappa_3(\mathbf{x}, \mathbf{y}) &= \langle \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} \phi_1(\mathbf{y}) \\ \phi_2(\mathbf{y}) \end{bmatrix} \rangle = \langle \phi_1(\mathbf{x}), \phi_1(\mathbf{y}) \rangle + \langle \phi_2(\mathbf{x}), \phi_2(\mathbf{y}) \rangle \\ \Rightarrow \kappa_3(\mathbf{x}, \mathbf{y}) &= \kappa_1(\mathbf{x}, \mathbf{y}) + \kappa_2(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{2.13}$$

The above result, in conjunction with the binomial expansion, results in the following kernel which takes in to account, all the monomials up degree d or less :

$$\kappa(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{2.14}$$

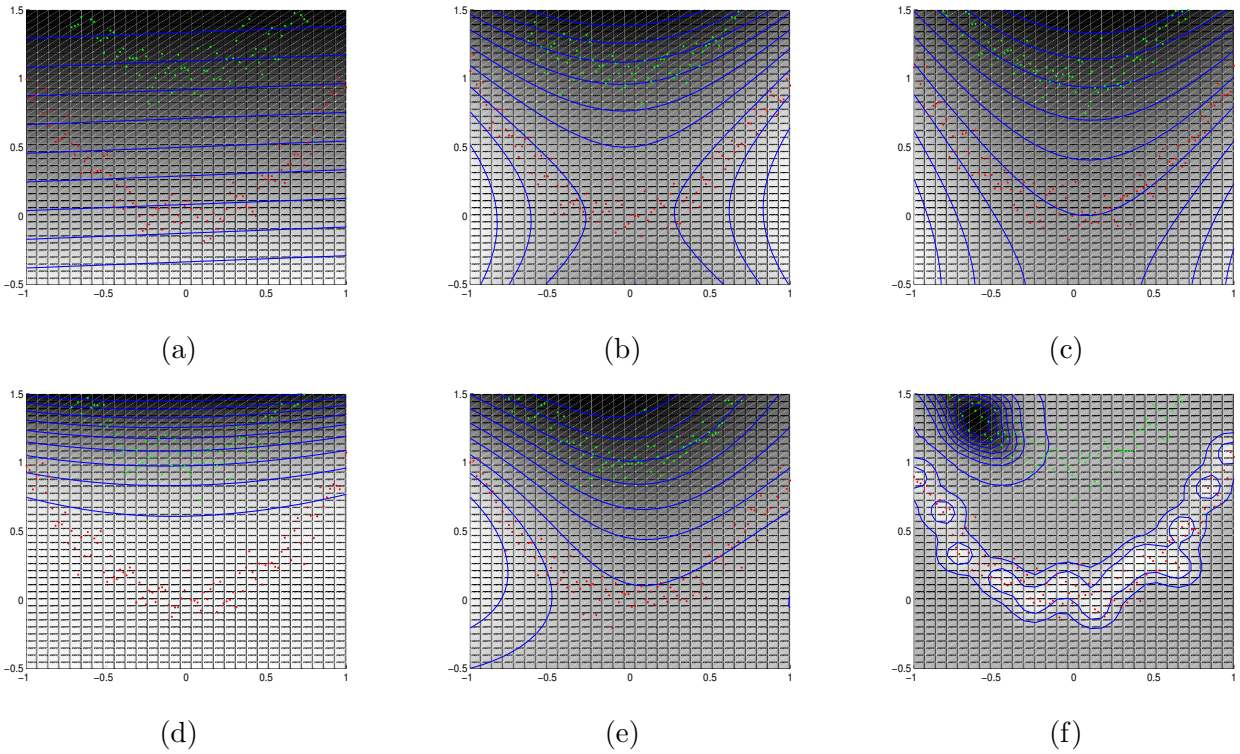


Figure 2.3: The kernel perceptron detects nonlinear relationships. The kernel perceptron algorithm is used to learn a classification function to separate two data sets (one in green and the other in red) each distributed in parabolic shapes. The blue lines in each image are isocontours i.e., contours along which the classification function's value is constant. The background color at each point indicates the value of function at each point (higher values are shown brighter). Thus, each blue line corresponds to a boundary between the class at certain threshold. (a) shows the result obtained using a linear kernel $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ which fails to find a satisfactory solution. (b) and (c) show results with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$ and $\kappa(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$ respectively. The exact parabolic boundary that separates the two distributions are found. (d) shows the result with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^3$. Although the boundary is nonlinear, it is not useful for classifying unseen examples. (e) and (f) show the results with $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1, 0.1$ respectively. The solution in the first case is expected to be more stable on unseen examples.

From the discussion above, it can be seen that the functional form $\kappa(.,.)$ is of greater use, in practice, than the mapping $\phi()$ since the mapping is never used in the computation. Although the mapping $\phi()$ is introduced first in the current discussion, in general, the choice of $\kappa(.,.)$ implicitly defines the mapping $\phi(.,.)$. As seen earlier, there may be several mappings that correspond to the inner product $\kappa(,)$. There are several functions which are valid kernel functions such as the gaussian radial basis kernel [17]

$$\kappa(\mathbf{x},\mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}) \tag{2.15}$$

and the sigmoid kernel [17]

$$\kappa(\mathbf{x},\mathbf{y}) = tanh(\kappa(\langle\mathbf{x},\mathbf{y}\rangle) + \Theta) \tag{2.16}$$

However, not all functional forms $\kappa(,)$ correspond to inner product in some feature space. The question of what functions $\kappa(,)$ satisfy this criterion is connected to Reproducing Kernel Hilbert Spaces and is discussed widely in literature [17, 15]. The theory of kernel function is briefly reviewed in the following section.

2.3 The theory of kernel functions

The main theme of the kernel trick is to operate in a feature space via a kernel function $\kappa(,)$. The feature space is accessed indirectly via pairwise inner products. To understand how this indirect access to a different feature space enables the implementation of linear algorithms, we first review some properties of linear functions and the inner product.

2.3.1 Inner product and Hilbert Spaces

Linear Function Given a vector space X over $\mathcal{R}$, a function $f : X \mapsto \mathcal{R}$ is said to be *linear* if i) $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$ and ii) $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$ for all $\mathbf{x},\mathbf{y} \in X$ and $\alpha \in \mathcal{R}$.

Inner Product Space A vector space X over $\mathcal{R}$ is known as an *inner product space* over $\mathcal{R}$ if there exists a real symmetric bilinear map $\langle .,. \rangle : X \times X \mapsto \mathcal{R}$ such that $\langle \mathbf{x},\mathbf{x}\rangle \geq 0$ for all $\mathbf{x} \in X$. The map $\langle .,. \rangle$ is known as the inner product or dot product. The inner product is said to be *strict* if $\langle \mathbf{x},\mathbf{x}\rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$. The *norm* on a strict inner product space and the associated distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ are defined as

$$||\mathbf{x}|| = \sqrt{\langle \mathbf{x},\mathbf{x}\rangle}$$

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||$$

Cauchy-Schwartz Inequality For all $\mathbf{x},\mathbf{y}$ in an inner product space X over $\mathcal{R}$, the following inequality holds

$$\langle \mathbf{x},\mathbf{y}\rangle \leq ||\mathbf{x}||||\mathbf{y}|| \tag{2.17}$$

where, in a strict inner product space, the equality holds if and only if $\mathbf{x}$ and $\mathbf{y}$ are linearly dependent.

Proof Consider the following inequality where $\alpha \in \mathcal{R}$

$$\langle \alpha\mathbf{x}+\mathbf{y}, \alpha\mathbf{x}+\mathbf{y}\rangle = \alpha^2||\mathbf{x}||^2 + 2\alpha\langle \mathbf{x},\mathbf{y}\rangle + ||\mathbf{y}||^2 \geq 0 \tag{2.18}$$

The above inequality holds for all α. Hence, considering the equality as a quadratic equation the determinant is non-positive since there is at most one root

$$4\langle \mathbf{x}, \mathbf{y}\rangle^2 - 4||\mathbf{x}||^2||\mathbf{y}||^2 \leq 0$$

$$\Rightarrow \langle \mathbf{x}, \mathbf{y}\rangle \leq ||\mathbf{x}||||\mathbf{y}|| \tag{2.19}$$

The equality, in a strict inner product space, implies $\alpha\mathbf{x}+\mathbf{y} = 0$ which shows $\mathbf{x}$ and $\mathbf{y}$ are rescalings of the same vector.

The set of vectors $X_0 = \{\mathbf{x} \in X \ni ||\mathbf{x}|| = 0\}$ forms a linear subspace of X. This is so since if $\mathbf{x}, \mathbf{y} \in X_0$, then for all $\alpha, \beta \in \mathcal{R}$ we have

$$||\alpha\mathbf{x} + \beta\mathbf{y}||^2 = \langle \alpha\mathbf{x} + \beta\mathbf{y}, \alpha\mathbf{x} + \beta\mathbf{y}\rangle = \alpha^2||\mathbf{x}||^2 + 2\alpha\beta\langle \mathbf{x}, \mathbf{y}\rangle + \beta^2||\mathbf{y}||^2 = 0$$

since $\langle \mathbf{x}, \mathbf{y}\rangle^2 \leq ||\mathbf{x}||^2||\mathbf{y}||^2 = 0$ by the Cauchy-Schwartz inequality. Hence $\alpha\mathbf{x} + \beta\mathbf{y} \in X_0$. This means that any non-strict inner product space can be converted to a strict inner product by taking the quotient space with respect to this subspace.

Separability and Completeness A Cauchy sequence is a sequence of elements $\{h_n\}_{n\geq 1}$ from an inner product space X with the following property

$$\lim_{n\to\infty} \sup_{m>n} ||h_n - h_m|| \to 0 \tag{2.20}$$

If every Cauchy sequence $\{h_n\}_{n\geq 1}$ of elements of X converges to an element $h \in X$, then X is said to be *complete*. If for all $\epsilon > 0$ there exists a finite set of elements $h_1, \cdots, h_n$, such that for all $h \in X$,

$$\min_i ||h_i - h|| < \epsilon, \tag{2.21}$$

then X is said to be *separable* .

Hilbert Space A separable and complete inner product space is known as a Hilbert Space. One particular Hilbert space of interest to us is the space L_2, which is the set of all countable sequences of real numbers $\mathbf{x} = (x_1, \cdots, x_n, \cdots)$ satisfying

$$\sum_i^\infty x_i^2 < \infty$$

where the inner product is defined by

$$\langle \mathbf{x}, \mathbf{y}\rangle = \sum_{i=1}^\infty x_i y_i$$

Similarly, the space of square integrable functions on a compact subset X of $\mathcal{R}^n$,

$$\{f : \int_X f(x)^2 dx < \infty\}$$

with the inner product defined by

$$\langle f, g\rangle = \int_X f(x)g(x)dx$$

is a Hilbert space.A Hilbert space is isomorphic to either $\mathcal{R}^n$ for some finite n or to the space L_2. This implies that a coordinate system can be given to such spaces. Since kernel functions need to

map input data to a space where the mapped data can be viewed as vectors, the feature space needs to be a Hilbert space. Although the feature vectors are never explicitly accessed, this condition is crucial for the kernel function to be a valid inner product.

Linear algorithms search for a weight vector in the feature space. However this weight vector is also point the feature space and each point $\mathbf{w}$ defines a linear function via the inner product : $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$. Thus, estimating the linear function is equivalent to find the element $\mathbf{w}$ of the feature space. The key property that allows the dual form of a linear algorithm to be used is that this element $\mathbf{w}$ is a linear combination of the feature vectors of the training samples.

Dimensionality and Rank Given two vectors $\mathbf{x}$ and $\mathbf{y}$ in a strict inner product space X, the angle between them is defined by

$$\theta = \arccos \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{||\mathbf{x}||||\mathbf{y}||} \tag{2.22}$$

The vectors are said to be *orthogonal* if $\theta = 0$. A set of vectors $B = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ is called *orthonormal* if

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \delta_{ij} \tag{2.23}$$

Given an orthonormal set B, and a vector $\mathbf{w}$ the following expansion of the vector in terms of the elements of B is called it *Fourier Series* of the vector :

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{w} \rangle \mathbf{x}_i \tag{2.24}$$

If $\hat{\mathbf{w}} = \mathbf{w}$, for all $\mathbf{w}$, then B is a *basis* of X and the number of elements in B is the *dimensionality* of X. An alternate representation of a set of vectors, given their coordinates in some basis, is the *data matrix* representation. Given m vectors, $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ from an n-dimensional space, the *data matrix*, $\mathbf{X}$, of size $n \times m$, is formed by arranging the vectors as the columns of the matrix i.e.,

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{bmatrix}$$

Hereafter, the coordinates of a data sample will always be taken as a column vector and the ith column of the matrix $\mathbf{X}$ will be denoted by $\mathbf{X}^i$. The dimensionality of the space spanned by columns of $\mathbf{X}$ (also known as the *column space*) is called the *rank* of the matrix. Thus if the rank is r the there exist r linearly independent vectors $\mathbf{r}_1, \cdots, \mathbf{r}_r$ which form an $n \times r$ data matrix $\mathbf{R}$. The coordinates describing $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ in this basis form an $r \times m$ matrix $\mathbf{S}$. Thus $\mathbf{X}$ can be factorized as

$$\mathbf{X} = \mathbf{RS} \tag{2.25}$$

The rank of $\mathbf{X}^t$ is equal to the rank of $\mathbf{X}$. The matrix $\mathbf{X}$ is said to be *full rank* matrix if $rank(\mathbf{X}) = \min(m, n)$.

Eigen Values and Eigen Vectors An $n \times n$ matrix $\mathbf{A}$ is said to be singular if it is not *full rank*. This implies that the columns of $\mathbf{A}$ are linearly dependent. Thus there exists coefficients $x_1, \cdots, x_n$ such that $\sum_{i=1}^{n} x_i \mathbf{A}^i = \mathbf{0}$. Thus with $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^t$ we have

$$\mathbf{Ax} = \mathbf{0} = 0\mathbf{x} \tag{2.26}$$

The columns of an $n \times n$ non-singular matrix span a space of dimension n. Thus all the n unit vectors $\mathbf{e}_1, \cdots, \mathbf{e}_n$ can all be expressed as linear combination of the columns. The $n \times n$ matrix $\mathbf{B}$ with the corresponding coefficients $\mathbf{b}_1, \cdots, \mathbf{b}_n$ as columns is the multiplicative inverse of $\mathbf{A}$ since

$$\mathbf{Ab}_i = \mathbf{e}_i \tag{2.27}$$

$$\mathbf{AB} = \mathbf{I} \tag{2.28}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. If there exists a real number λ and a vector $\mathbf{x}$ such that, for a square matrix $\mathbf{A}$

$$\mathbf{Ax} = \lambda\mathbf{x} \tag{2.29}$$

then, λ is called an eigen value of the matrix $\mathbf{A}$ and $\mathbf{x}$ the corresponding eigen vector. Thus, for a singular matrix 0 is always an eigenvalue. Conversely, if 0 is an eigenvalue, then the matrix is singular. Eigen values and eigen vectors of matrices are important in solving optimization problems that appear in a number of kernelized linear algorithms. The following optimization problem (involving a quadratic term) appears frequently:

$$\max_{\mathbf{x}} \frac{\mathbf{x}^t\mathbf{Ax}}{\mathbf{x}^t\mathbf{x}} \tag{2.30}$$

Since the solution is invariant to scale, an additional constraint on the norm of the $\mathbf{x}$, $||\mathbf{x}|| = \mathbf{x}^t\mathbf{x} = 1$ is imposed. The equation can now be solved using Lagrangian multiplier λ. The optimization problem can be posed as :

$$\max_{\mathbf{x}} \mathbf{x}^t\mathbf{Ax} - \lambda(\mathbf{x}^t\mathbf{x} - 1) \tag{2.31}$$

Differentiating with respect to $\mathbf{x}$ and setting to 0 we have,

$$\mathbf{Ax} = \lambda\mathbf{x} \tag{2.32}$$

This implies that the solution is an eigen vector of the matrix $\mathbf{A}$. Further, the value of the expression itself is

$$\frac{\mathbf{x}^t\mathbf{Ax}}{\mathbf{x}^t\mathbf{x}} = \frac{\mathbf{x}^t\lambda\mathbf{x}}{\mathbf{x}^t\mathbf{x}} = \lambda \tag{2.33}$$

which is maximum when the eigen vector corresponds to the maximum eigen value. This result is used in a number of linear algorithms. A symmetric matrix of size $n \times n$ has at most n non-zero eigen values. In this case, The eigen vectors corresponding to distinct eigen values are orthogonal. Usually, the eigen vectors are assumed to be normalized such that their norm is 1. Given eigen vectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$and corresponding eigen values $\lambda_1, \cdots, \lambda_n$ the matrix $\mathbf{V}$ whose columns are the eigen vectors and a diagonal matrix Λ, with $\Lambda_{ii} = \lambda_i$ satisfy :

$$\mathbf{AV} = \mathbf{V}\Lambda \Rightarrow \mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^t \tag{2.34}$$

This is known as the eigen decomposition of the matrix $\mathbf{A}$. An interesting consequence of this decomposition is the following factorization for non-singular matrices:

$$\mathbf{A}^{-1} = \mathbf{V}\Lambda^{-1}\mathbf{V}^t \tag{2.35}$$

Further for singular matrices, the only non-zero eigen values and eigen vectors need to be collected in to the matrices $\mathbf{V}$ and Λ the factorization of $\mathbf{A}$ to hold.

Positive semi-definite matrices A symmetric matrix $\mathbf{A}$ is said to be positive semi-definite if all its eigen values are non-negative. It can be shown that this condition holds, if and only if, for all vectors $\mathbf{x}$

$$\mathbf{x}^t\mathbf{Ax} \geq 0 \tag{2.36}$$

The matrix is said to be positive definite if, for $\mathbf{x} \neq 0$,

$$\mathbf{x}^t\mathbf{Ax} > 0$$

An important property of positive semi-definite matrices is that for every such matrix $\mathbf{A}$ there exists a real matrix $\mathbf{B}$ such that :

$$\mathbf{A} = \mathbf{B}^t\mathbf{B} \tag{2.37}$$

The eigen decomposition of the matrix $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^t$ can be used to make a choice of $\mathbf{B}$ as $\mathbf{B} = \sqrt{\Lambda}\mathbf{V}^t$. The choice of $\mathbf{B}$, however, is not unique. It is easy to see that the converse of the above statement is also true since any matrix that can factorized in the above manner satisfies, for all vectors $\mathbf{x}$:

$$\mathbf{x}^t\mathbf{A}\mathbf{x} = \mathbf{x}^t\mathbf{B}^t\mathbf{B}\mathbf{x} = (\mathbf{B}\mathbf{x})^t(\mathbf{B}\mathbf{x}) = ||\mathbf{B}\mathbf{x}||^2 \geq 0 \tag{2.38}$$

Positive definiteness plays an important role in establishing the properties of a valid kernel function. These properties and other desirable qualities of a kernel function are discussed in section 2.3.2.

2.3.2 What functions constitute valid kernel functions?

As seen in section 2.2 the kernel function can be used to access the feature space only via a finite set of samples. Thus, the kernelized gram matrix i.e. the kernel matrix is central to the kernelization of algorithms. An important property of these matrices is that they are positive semi-definite. For a kernel matrix on the set $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, we have $\mathbf{K}_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$ and hence for any vector $\mathbf{x}$,

$$\begin{aligned}\mathbf{x}^t\mathbf{K}\mathbf{x} &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \mathbf{K_{ij}} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle \\ &= \langle\sum_{i=1}^{n} x_i\phi(\mathbf{x}_i), \sum_{j=1}^{n} x_j\phi(\mathbf{x}_j)\rangle = ||\sum_{j=1}^{n} x_j\phi(\mathbf{x}_j)||^2 \geq 0\end{aligned} \tag{2.39}$$

Thus, the kernel matrix (and its special case, the Gram matrix) are positive semi-definite. This property of matrices can be used to characterize the functions that are valid kernels. The following property is essential for a function to be a kernel function.

Finitely positive semi-definite functions Given an input space (not necessarily a vector space) $\mathcal{X}$, a symmetric function

$$\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$$

is said to be finitely positive semi-definite if the matrix formed by evaluating the function on all pairs of elements of any finite subset of $\mathcal{X}$ is positive semi-definite.

A function $\kappa(.,.) : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ that is continuous or has a finite domain can be decomposed in to inner product of the images of its arguments under a feature mapping in to a Hilbert space $\mathcal{F}$,$\phi(.) : \mathcal{X} \mapsto \mathcal{F}$ i.e.

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle \tag{2.40}$$

if and only if $\kappa(.,.)$ is finitely positive semi-definite on $\mathcal{X}$. The necessary condition is proved by the fact that the kernel matrix is always positive semi-definite. The sufficient condition is proved by explicit construction of the feature space $\mathcal{F}$and the map $\phi(.)$. The feature space is the following space of functions

$$\mathcal{F} = \{\sum_{i=1}^{n} a_i\kappa(\mathbf{x}_i, .)|n \in \mathcal{N}, \mathbf{x}_i \in \mathcal{X}\} \tag{2.41}$$

It can be shown that $\mathcal{F}$ is a Hilbert space over $\mathcal{R}$ with the following definition for the inner product

$$\langle\sum_{i=1}^{n} a_i\kappa(\mathbf{x}_i, .), \sum_{j=1}^{m} b_j\kappa(\mathbf{y}_j, .)\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m} a_i b_j \kappa(\mathbf{x}_i, \mathbf{y}_j) \tag{2.42}$$

It is easy to see that the inner product so defined is real-valued, since $\kappa(.,.)$ is real-valued. Let the arguments to the inner product be $f = \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .)$ and $g = \sum_{j=1}^{m} b_j \kappa(\mathbf{y}_j, .)$. The linearity and nonegativity of the inner product in each of its arguments can be seen from the following equations

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j \kappa(\mathbf{x}_i, \mathbf{y}_j) = \sum_{j=1}^{m} b_j f(\mathbf{y}_j) = \sum_{i=1}^{n} a_i g(\mathbf{x}_i) \tag{2.43}$$

$$\langle f, f \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^t \mathbf{K} \mathbf{a} \geq 0 \tag{2.44}$$

where $\mathbf{a} = [a_1 \quad \cdots \quad a_n]^t$ and $\mathbf{K}$ is the kernel matrix. We have used the fact kernel matrices are positive semi-definite in the second equation. The proofs for completeness and separability of $\mathcal{F}$ are more involved and are omitted for brevity. The specification of the map $\phi : \mathcal{X} \mapsto \mathcal{F}$ completes the decomposition of $\kappa(.,.)$ in to inner product of elements in $\mathcal{F}$. Since the requisite property of this map is $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y})$ it can be seen that the map

$$\phi(\mathbf{x}) = \kappa(\mathbf{x}, .) \tag{2.45}$$

satisfies the requirement. Further it can be shown that this inner product enables us access to an the element $f = \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .)$ of $\mathcal{F}$ since

$$\langle f, \phi(\mathbf{x}) \rangle = \langle \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .), \kappa(\mathbf{x}, .) \rangle = \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x}) \tag{2.46}$$

Thus a function f in the function space $\mathcal{F}$ can be represented as linear function via the inner product (with itself) defined above. This is the so called *reproducing property* of the kernel function $\kappa(.,.)$. For this reason, the space $\mathcal{F}$ corresponding to the function $\kappa(.,.)$ is called the Reproducing Kernel Hilbert Space (RKHS). This property is a key characteristic of kernel function since any function $\kappa(.,.)$ that satisfies the reproducing property in a Hilbert Space of functions $\mathcal{F}$ satisfies the positive semi-definiteness property. Since $\langle f, \phi(\mathbf{x}) \rangle = f(\mathbf{x})$ the function $\kappa(.,.)$ applied to an input pair $\mathbf{x}_i$ and $\mathbf{x}_j$ can be decomposed as follows:

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \kappa(\mathbf{x}, .), \kappa(\mathbf{y}, .) \rangle \tag{2.47}$$

Thus, for the kernel matrix $\mathbf{K}$ any vector $\mathbf{a} = [a_1 \quad \cdots \quad a_n]^t$ the following result holds

$$\begin{aligned}
\mathbf{a}^t \mathbf{K} \mathbf{a} &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \langle \kappa(\mathbf{x}_i, .), \kappa(\mathbf{x}_j, .) \rangle \\
&= \langle \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .), \sum_{j=1}^{n} a_j \kappa(\mathbf{x}_j, .) \rangle \\
&= || \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .) ||^2 \geq 0
\end{aligned}$$

Since the above relation holds for any finite subset, the function $\kappa(,)$ is finitely positive semi-definite. When the correspondence between the kernel function and associated RKHS needs emphasis, the notation $\mathcal{F}_\kappa$ for the RKHS and $\langle ., . \rangle_{F_\kappa}$ for the inner product in $\mathcal{F}_k$ will be used.

Mercer Theorem Although the positive semi-definiteness characterizes kernel functions, the feature space that is used is a function space which is different from the traditional representation of the data i.e. feature vectors. The Mercer theorem enables the construction of a feature space whose elements are feature vectors and not functions. The theorem is stated below with out proof : Let $\mathcal{X}$ be a compact subset of $\mathcal{R}^n$. Let a continuous symmetric function κ be such that the integral operator

$$I_\kappa : L_2(\mathcal{X}) \mapsto L_2(\mathcal{X})$$

$$I_\kappa f = \int_{\mathcal{X}} \kappa(,\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$$

is always positive for all $f \in L_2(\mathcal{X})$,

$$\int_{\mathcal{X}\times\mathcal{X}} \kappa(\mathbf{x},\mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \tag{2.48}$$

Then $\kappa(\mathbf{x},\mathbf{y})$ can be expanded in to a uniformly convergent series in terms of orthonormal function ψ_i,

$$\kappa(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{y}) \tag{2.49}$$

The above expansion allows the construction of the map ϕ in an explicit feature vector form:

$$\phi(\mathbf{x}) = \begin{bmatrix} \sqrt{\lambda_1}\psi_1(\mathbf{x}) & \cdots & \sqrt{\lambda_i}\psi_i(\mathbf{x}) & \cdots \end{bmatrix}^t \tag{2.50}$$

Other interpretations of the kernel function such as covariance function determined by a probabilistic function can be found in standard literature [1, 92, 94].

2.3.3 Kernel Design

One of the major challenges related to kernel algorithms is the choice of the kernel function. New kernel functions can be constructed from known kernel functions by performing certain operations on them. For instance, given two kernel function $\kappa_1(.,.)$ and $\kappa_2(.,.)$ the following functions are all valid kernel functions

$$\kappa(\mathbf{x},\mathbf{y}) = a\kappa_1(\mathbf{x},\mathbf{y}), a \in \mathcal{R}^+$$

$$\kappa(\mathbf{x},\mathbf{y}) = \kappa_1(\mathbf{x},\mathbf{y}) + \kappa_2(\mathbf{x},\mathbf{y})$$

$$\kappa(\mathbf{x},\mathbf{y}) = \kappa_1(\mathbf{x},\mathbf{y})\kappa_2(\mathbf{x},\mathbf{y})$$

$$\kappa(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} a_i \kappa_1(\mathbf{x},\mathbf{y})^i, n \in \mathcal{N}, a_i \geq 0, a_n \neq 0$$

$$\kappa(\mathbf{x},\mathbf{y}) = \exp(\kappa_1(\mathbf{x},\mathbf{y}))$$

However, it is not straightforward to incorporate prior knowledge about a problem in to kernel functions. In case of supervised problems, it is possible to define an *ideal* similarity matrix and try to align a kernel matrix to it. Given a two class problem with the target labels $\{\pm 1\}$, let $\mathbf{y}$ be the vector of labels if the input samples. The matrix $\mathbf{y}\mathbf{y}^t$ defines an ideal similarity matrix since all pairs of examples that have the same label give rise to a similarity score of 1 while those that have different labels given a score of -1. Using the *frobenius inner product*,

$$\langle \mathbf{A}, \mathbf{B} \rangle = trace(\mathbf{A}^t\mathbf{B})$$

the *alignment* or similarity between two matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ can be defined as

$$\frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle}{||\mathbf{K}_1||||\mathbf{K}_2||} \tag{2.51}$$

as a similarity measure between two matrices. For $\mathbf{K}_1 = \mathbf{K}$ and $\mathbf{K}_2 = \mathbf{y}\mathbf{y}^t$ this measure becomes

$$\frac{\mathbf{y}^t \mathbf{K} \mathbf{y}}{n||\mathbf{K}||} \tag{2.52}$$

where n is the number of samples. The similarity can be used to determine if a kernel matrix suits a task well. Alternate methods for learning the kernel matrix for tasks such as transduction and nonlinear component analysis have been proposed recently [95, 96, 97]. However, choosing the kernel function or learning the kernel matrix that is optimal for a general task is still an open problem.

2.4 Kernelization of Algorithms

2.4.1 The Data Matrix Representation

Although properties of kernel function are important, often the kernel matrix plays more important role, in practice, than the kernel function. Since kernel function allows access to the feature space only via the input samples, the pairwise inner products between elements of a finite input set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ are the only information available on the geometry of the feature space. This information is embedded in the kernel matrix $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Most kernel algorithms often work with this matrix rather than the kernel function itself. The representation of sets of feature vectors as data matrices greatly simplifies the process of kernelization and to easily *see* kernel matrices or elements of it in equations. Let the images of input samples under the map ϕ form the columns of the matrix $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix}$$

The key relationship to note is that $\mathbf{K} = \mathbf{X}^t\mathbf{X}$. Several operations such as summation of the columns, selection of columns etc. can be represented as matrix multiplications. For instance, the solution vector $\mathbf{w}$ in section 2.2 can be expressed as:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) = \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \mathbf{X}\boldsymbol{\alpha} \tag{2.53}$$

where $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix}^t$. To see how data matrix representation makes kernelization easy, consider centering of the data in the feature space. The challenge is to remove the sample mean $\boldsymbol{\phi}_C = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)$ from each column of the data matrix and then work with the resulting data. The sample mean can be written in terms of the data matrix form by setting $\alpha_i = \frac{1}{n}$ in the above equation. Since there is no way of accessing the sample mean, we must construct an equivalent centered kernel $\mathbf{K}_C$. It turns out that this can be done by operations on the original $\mathbf{K}$ alone. First, the centered data matrix $\mathbf{X}_C$ can be expressed as

$$\begin{aligned} \mathbf{X}_C &= \begin{bmatrix} \phi(\mathbf{x}_1) - \boldsymbol{\phi}_C & \phi(\mathbf{x}_2) - \boldsymbol{\phi}_C & \cdots & \phi(\mathbf{x}_n) - \boldsymbol{\phi}_C \end{bmatrix} \\ &= \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix} - \begin{bmatrix} \boldsymbol{\phi}_C & \cdots & \boldsymbol{\phi}_C \end{bmatrix} \\ &= \mathbf{X} - \begin{bmatrix} \boldsymbol{\phi}_C \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \\ &= \mathbf{X} - \mathbf{X} \begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}^t \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} = \mathbf{X} - \mathbf{X}\mathbf{1}_{\frac{1}{n}} \end{aligned} \tag{2.54}$$

where $\mathbf{1}_a$ is a matrix with all elements equal to a (we do not explicitly show the size in the notation since it is evident from the context. Whenever the size needs emphasis, such matrices will be distinguished explicitly). Thus, the centered kernel matrix can be written as

$$\begin{aligned} \mathbf{K}_C &= \mathbf{X}_C^t\mathbf{X}_C = (\mathbf{X} - \mathbf{X}\mathbf{1}_{\frac{1}{n}})^t(\mathbf{X} - \mathbf{X}\mathbf{1}_{\frac{1}{n}}) \\ &= \mathbf{X}^t\mathbf{X} - \mathbf{X}^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} - \mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X} + \mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} \\ \Rightarrow \mathbf{K}_C &= \mathbf{K} - \mathbf{K}\mathbf{1}_{\frac{1}{n}} - \mathbf{1}_{\frac{1}{n}}\mathbf{K} + \mathbf{1}_{\frac{1}{n}}\mathbf{K}\mathbf{1}_{\frac{1}{n}} \end{aligned} \tag{2.55}$$

which can be computed from the original kernel matrix $\mathbf{K}$ alone. To use relationships that are inferred by a kernelized algorithm on unseen data, $\mathcal{X}_t = \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \cdots, \hat{\mathbf{x}}_m\}$, we need the pairwise inner product information between the unseen samples and the samples from which the algorithm inferred the relationship. This is encoded is an $m \times n$ matrix $\hat{\mathbf{K}}_{ij} = \langle\phi(\hat{\mathbf{x}}_i), \phi(\mathbf{x}_j)\rangle = \kappa(\hat{\mathbf{x}}_i, \mathbf{x}_j)$. Once again, the data matrix notation simplifies equations :

$$\hat{\mathbf{K}} = \begin{bmatrix}\phi(\hat{\mathbf{x}}_1) & \phi(\hat{\mathbf{x}}_2) & \cdots & \phi(\hat{\mathbf{x}}_n)\end{bmatrix}^t \mathbf{X} = \hat{\mathbf{X}}^t\mathbf{X} \tag{2.56}$$

It is easy to show that to center the $\hat{\mathbf{X}}$ with respect to the sample mean ϕ_C one needs to modify the kernel matrix $\hat{\mathbf{K}}$ as follows

$$\hat{\mathbf{K}}_C = \hat{\mathbf{K}} - \hat{\mathbf{K}}\mathbf{1}_{\frac{1}{n}} - \mathbf{1}'_{\frac{1}{n}}\mathbf{K} + \mathbf{1}'_{\frac{1}{n}}\mathbf{K}\mathbf{1}_{\frac{1}{n}} \tag{2.57}$$

where $\mathbf{1}'_{\frac{1}{n}}$ is an $m \times n$ matrix with all entries equal to $\frac{1}{n}$. Another important advantage with the data matrix notation is the connection to the sample covariance matrix which encodes the second order statistics of the input set of samples. The covariance matrix $\mathbf{C}$ in the feature space is $N \times N$ matrix with $\mathbf{C}_{ij} = \sum_{k=1}^{n}(\phi(\mathbf{x}_k)_i - \boldsymbol{\phi}_{Ci})(\phi(\mathbf{x}_k)_j - \boldsymbol{\phi}_{Cj})$, where N is dimension of the feature space. Using the data matrix notation the sample covariance matrix can be represented as :

$$\mathbf{C} = \mathbf{X}_C\mathbf{X}_C^t \tag{2.58}$$

The eigen vectors of the sample covariance matrix are of great practical importance for many tasks involving compression, dimensionality reduction etc. However, it is not possible to explicitly compute the matrix $\mathbf{C}$ since $\phi()$ typically maps the input samples to a very high dimensional space. Even if $\phi()$ were the identity map the size of $\mathbf{C}$ is $N \times N$, where N is the dimensionality of the data, making it impossible to work with such matrices. The following property gives a workaround to this problem :

$$\begin{aligned} \mathbf{A}\mathbf{A}^t\mathbf{x} &= \lambda\mathbf{x} \\ \Rightarrow \mathbf{A}^t\mathbf{A}\mathbf{A}^t\mathbf{x} &= \mathbf{A}^t\lambda\mathbf{x} \\ \Rightarrow (\mathbf{A}^t\mathbf{A})(\mathbf{A}^t\mathbf{x}) &= \lambda(\mathbf{A}^t\mathbf{x}) \end{aligned} \tag{2.59}$$

The above equations show that the eigen values of $\mathbf{A}\mathbf{A}^t$ and $\mathbf{A}^t\mathbf{A}$ are same. Moreover, $\mathbf{v}$ is an eigen vector of $\mathbf{A}\mathbf{A}^t$ only if $\mathbf{A}^t\mathbf{v}$ is an eigen vector $\mathbf{A}^t\mathbf{A}$. Thus, the eigen values of the matrix $\mathbf{K}_C$ are same as that of the matrix $\mathbf{C}$. However, the eigen vectors are still not accessible since $\mathbf{X}_C^t$ is not directly accessible. The dual representation helps in overcoming this problem. If each eigen vector $\mathbf{v}$ of the matrix $\mathbf{C}$ lies in span of columns of $\mathbf{X}_C$ i.e $\mathbf{v} = \mathbf{X}_C\boldsymbol{\alpha}$, then $\mathbf{X}_C^t\mathbf{v} = \mathbf{X}_C^t\mathbf{X}_C\boldsymbol{\alpha} = \mathbf{K}_C\boldsymbol{\alpha}$ is an eigen vector of $\mathbf{K}_C$. This fact is the foundation for kernel principal component analysis(KPCA) [3] which is popular for nonlinear component analysis. In the remainder of this section examples of kernelized versions of several linear algorithms are presented. In many cases the data matrix representation simplifies the task and is useful when kernelizing new algorithms.

2.4.2 Basic Algorithms : Distance, Variance and Normalization

Distances and Normalization Distance between two points in the feature space is one of the simplest measures used for a variety of tasks. Distance is defined in terms of inner product and hence the kernelized version of distance-finding is straightforward :

$$\begin{aligned} d(\phi(\mathbf{x}),\phi(\mathbf{y})) &= \sqrt{||\phi(\mathbf{x})-\phi(\mathbf{y})||^2} \\ &= [\langle\phi(\mathbf{x})-\phi(\mathbf{y}),\phi(\mathbf{x})-\phi(\mathbf{y})\rangle]^{\frac{1}{2}} \\ &= [\langle\phi(\mathbf{x}),\phi(\mathbf{x})\rangle+\langle\phi(\mathbf{y}),\phi(\mathbf{y})\rangle-2\langle\phi(\mathbf{x}),\phi(\mathbf{y})\rangle]^{\frac{1}{2}} \\ &= \kappa(\mathbf{x},\mathbf{x})+\kappa(\mathbf{y},\mathbf{y})-2\kappa(\mathbf{x},\mathbf{y}) \end{aligned} \tag{2.60}$$

The ability to compute distance enables us to kernelize one of the simplest algorithms for anomaly detection i.e. finding if a new sample $\mathbf{x}_t$ belongs to the same distribution as the training set $\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n\}$. A simple (albeit naive) way of doing this is to approximate the distribution with a normal distribution with unit covariance (i.e., spherical distribution), $\mathcal{N}(\mu,\mathbf{I})$, where μ is the sample mean, $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$. The distance from the μ then becomes the criterion to decide if $\mathbf{x}_t$ is an anomaly or not. The equations are all again simplified using the data matrix notation. The sample mean is $\mathbf{X}\mathbf{1}_{\frac{1}{n}}$ and the distance d from $\phi(\mathbf{x}_t)$ to the sample mean is given by

$$\begin{aligned} d^2 &= (\phi(\mathbf{x}_t)-\mathbf{X}\mathbf{1}_{\frac{1}{n}})^t(\phi(\mathbf{x}_t)-\mathbf{X}\mathbf{1}_{\frac{1}{n}}) \\ &= \phi(\mathbf{x}_t)^t\phi(\mathbf{x}_t)-2\phi(\mathbf{x}_t)^t\mathbf{X}\mathbf{1}_{\frac{1}{n}}+\mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} \\ &= \kappa(\mathbf{x}_t,\mathbf{x}_t)-2\mathbf{k}_t^t\mathbf{1}_{\frac{1}{n}}+\mathbf{1}_{\frac{1}{n}}^t\mathbf{K}\mathbf{1}_{\frac{1}{n}} \end{aligned} \tag{2.61}$$

where $\mathbf{k}_t=\begin{bmatrix}\kappa(\mathbf{x}_1,\mathbf{x}_t) & \kappa(\mathbf{x}_2,\mathbf{x}_t) & \cdots & \kappa(\mathbf{x}_n,\mathbf{x}_t)\end{bmatrix}^t$ is the vector containing the inner products of $\phi(\mathbf{x}_t)$ with the images of training samples under $\phi(.)$. It should be noted that matrices of the form $\mathbf{A}^t\mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are data matrices, are accessible in the feature space. Figure 2.4 show how the distance in the feature space with various kernels looks like in the input space.

Like centering of the data, normalization of data can also be done by operations on the kernel matrix alone. Given a point $\phi(\mathbf{x})$ in the feature space, the normalized vector is

$$\hat{\phi}(\mathbf{x})=\frac{\phi(\mathbf{x})}{||\phi(\mathbf{x})||}$$

Thus the new inner product in the feature space becomes

$$\begin{aligned} \hat{\kappa}(\mathbf{x},\mathbf{y}) &= \langle\hat{\phi}(\mathbf{x}),\hat{\phi}(\mathbf{y})\rangle \\ &= \langle\frac{\phi(\mathbf{x})}{||\phi(\mathbf{x})||},\frac{\phi(\mathbf{y})}{||\phi(\mathbf{y})||}\rangle \\ &= \frac{\langle\phi(\mathbf{x}),\phi(\mathbf{y})\rangle}{||\phi(\mathbf{x})||||\phi(\mathbf{y})||} \\ &= \frac{\kappa(\mathbf{x},\mathbf{y})}{\sqrt{\kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{y},\mathbf{y})}} \end{aligned} \tag{2.62}$$

The original kernel matrix $\mathbf{K}$ can modified to obtain a new kernel matrix $\hat{\mathbf{K}}$ with the above inner product. If $\mathbf{D}$ is a diagonal matrix such that $\mathbf{D}_{ii}=\frac{1}{\sqrt{\mathbf{K}_{ii}}}$ then $\hat{\mathbf{K}}$ can be expressed as a simple matrix multiplication :

$$\hat{\mathbf{K}}=\mathbf{DKD} \tag{2.63}$$

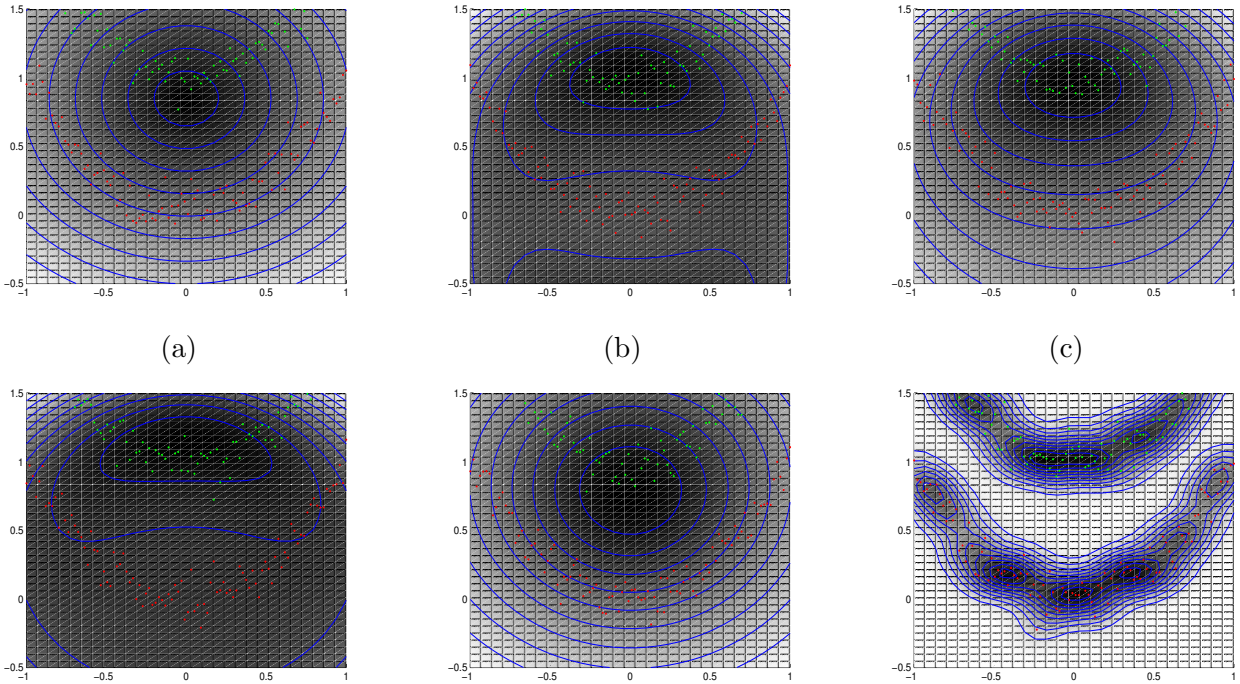


Figure 2.4: Visualizing distance in the feature space gives an idea of the the geometry of the features. The distance finding algorithm is used to calculate the distance of the points in the input space and the sample mean of two data sets (one in green and the other in red) each distributed in parabolic shapes. The contours are curves along which the distance to the mean is constant Results : (a) linear kernel, (b) and (c) with $\kappa(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle^2$ and $\kappa(\mathbf{x},\mathbf{y}) = (1+\langle \mathbf{x},\mathbf{y}\rangle)^2$ respectively, (d) with $\kappa(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle^3$. (e) and (f) with $\kappa(\mathbf{x},\mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1, 0.1$ respectively. It can be seen that the isosurfaces in each feature space define complex curves in the input space. Also, in (f), note the complexity of the feature space corresponding to Gaussian kernel function where points from both the distributions are close to the mean.

Variance Apart from distance between two points and norms of vectors, the projection of a vector $\mathbf{x}$ on to a line $\mathbf{w}$ and variance of a collection of points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ along the line $\mathbf{w}$ are some of the important quantities in analysis of data. These quantities can be computed in the feature space, provided the line in the feature space lies in the span of images of a known set of points under the mapping ϕ. Thus, using the data matrix notation the line is $\mathbf{w} = \mathbf{X}\alpha$ and the sample mean is $\mu = \mathbf{X}\mathbf{1}_{\frac{1}{n}}$. The projection of a vector $\mathbf{x}$ on to $\mathbf{w}$ is given by :

$$P_{\mathbf{w}}(\mathbf{x}) = \frac{\langle \mathbf{x}, \mathbf{w} \rangle}{||\mathbf{w}||}\mathbf{w} \tag{2.64}$$

Kernelizing this using the data matrix notation, the projection of $\phi(\mathbf{x}_t)$ on to $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$ is:

$$\frac{\langle \mathbf{x}_t, \mathbf{w} \rangle}{||\mathbf{w}||}\mathbf{w} = \frac{\phi(\mathbf{x}_t)^t \mathbf{X}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t \mathbf{X}^t \mathbf{X}\boldsymbol{\alpha}} = \frac{\mathbf{k}_t^t \boldsymbol{\alpha}}{\boldsymbol{\alpha}^t \mathbf{K}\boldsymbol{\alpha}} \tag{2.65}$$

Many linear algorithms search for a direction along which the variance of input samples meets certain criterion. The variance of the input set along the line $\mathbf{w}$ can be computed by combining the results above. For simplicity, we assume $\mathbf{w}$ is normalized in feature space such that $||\mathbf{w}|| = \sqrt{\boldsymbol{\alpha}^t \mathbf{K}\boldsymbol{\alpha}} = 1$. The variance can be expressed as below :

$$\begin{aligned} \sigma_{\mathbf{w}}^2 &= \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t \mathbf{w} - \boldsymbol{\phi}_C^t \mathbf{w})^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t \mathbf{X}\boldsymbol{\alpha} - \mathbf{1}_{\frac{1}{n}}^t \mathbf{X}^t \mathbf{X}\boldsymbol{\alpha})^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{k}_i^t \boldsymbol{\alpha} - \mathbf{1}_{\frac{1}{n}}^t \mathbf{K}\boldsymbol{\alpha})^2 \end{aligned} \tag{2.66}$$

An alternate form that is more suitable for solving optimization problems can be obtained by noting the following identities

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t \mathbf{w} - \boldsymbol{\phi}_C^t \mathbf{w})^2 &= \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t \mathbf{w})^2 - (\boldsymbol{\phi}_C^t \mathbf{w})^2 \\ \sum_{i=1}^{n}(\mathbf{k}_i^t \boldsymbol{\alpha})^2 &= \boldsymbol{\alpha}^t \mathbf{K}^2 \boldsymbol{\alpha} \end{aligned} \tag{2.67}$$

Using these identities the variance can be expressed more compactly as

$$\sigma_{\mathbf{w}}^2 = \frac{1}{n}\boldsymbol{\alpha}^t \mathbf{K}^2 \boldsymbol{\alpha} - (\mathbf{1}_{\frac{1}{n}}^t \mathbf{K}\boldsymbol{\alpha})^2 \tag{2.68}$$

Orthonormalization Many pattern analysis algorithms, given an input set $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, construct an orthonormal basis $\mathcal{Q} = \{\mathbf{q}_1, \cdots, \mathbf{q}_d\}$ that spans the input set. For simplicity we assume that the given input samples are linearly independent so that $d = n$. The popular Gram-Schmidt orthonormalization procedure builds the basis inductively as follows:

- The first basis vector is obtained by normalizing $\mathbf{x}_1$ i.e.,

$$\mathbf{q}_1 = \frac{\mathbf{x}_1}{||\mathbf{x}_1||}$$

- The ith basis vector $\mathbf{q}_i$ is obtained from $\mathbf{x}_i$ by first subtracting the projections of $\mathbf{x}_i$ on to $\mathbf{q}_1, \cdots, \mathbf{q}_{i-1}$ and normalizing the residual i.e.,

$$\mathbf{q}_i = \frac{\mathbf{x}_i - \sum_{j=1}^{i-1} \langle \mathbf{q}_j, \mathbf{x}_j \rangle \mathbf{q}_j}{||\mathbf{x}_i - \sum_{j=1}^{i-1} \langle \mathbf{q}_j, \mathbf{x}_j \rangle \mathbf{q}_j||}$$

Building an orthonormal basis in the feature space using the above algorithm is not possible since the images $\phi(\mathbf{x}_i)$ are not accessible directly. However, since the basis set is in the span of the input set of samples each basis vector $\mathbf{q}_i$ in the feature space can be expressed as $\mathbf{q}_i = \mathbf{X}\boldsymbol{\alpha}^i$. The problem of basis construction now can reposed as estimation of the vectors $\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^n$. Since $\phi(\mathbf{x}_i)$ can be represented as $\mathbf{X}\mathbf{e}_i$ where $\mathbf{e}_i$ is the ith canonical vector we have the following procedure for orthonormal basis construction:

- The first basis vector can be written as

$$\mathbf{q}_1 = \frac{\mathbf{X}\mathbf{e}_1}{\sqrt{\mathbf{e}_1^t \mathbf{X}^t \mathbf{X} \mathbf{e}_1}} = \mathbf{X} \frac{\mathbf{e}_1}{\sqrt{\mathbf{e}_1^t \mathbf{K} \mathbf{e}_1}}$$

  and hence the dual coefficients of the first basis vector are given by

$$\boldsymbol{\alpha}^1 = \frac{\mathbf{e}_1}{\sqrt{\mathbf{e}_1^t \mathbf{K} \mathbf{e}_1}} \tag{2.69}$$

- The unnormalized ith basis vector is given by

$$\begin{aligned} \hat{\mathbf{q}}_i &= \mathbf{X}\mathbf{e}_i - \sum_{j=1}^{i-1} (\mathbf{X}\mathbf{e}_i)^t (\mathbf{X}\boldsymbol{\alpha}^j) \mathbf{X}\boldsymbol{\alpha}^j \\ &= \mathbf{X}\mathbf{e}_i - \sum_{j=1}^{i-1} (\mathbf{e}_i^t \mathbf{X}^t \mathbf{X}\boldsymbol{\alpha}^j) \mathbf{X}\boldsymbol{\alpha}^j \\ &= \mathbf{X}\mathbf{e}_i - \mathbf{X} \sum_{j=1}^{i-1} (\mathbf{e}_i^t \mathbf{K}\boldsymbol{\alpha}^j) \boldsymbol{\alpha}^j \end{aligned} \tag{2.70}$$

  Thus the dual coefficients of the unnormalized ith basis vector are given by

$$\hat{\boldsymbol{\alpha}}^i = \mathbf{e}_i - \sum_{j=1}^{i-1} (\mathbf{e}_i^t \mathbf{K}\boldsymbol{\alpha}^j) \boldsymbol{\alpha}^j \tag{2.71}$$

  The normalized coefficients $\boldsymbol{\alpha}^i$ can be done by setting $||\hat{\mathbf{q}}_i|| = \sqrt{\hat{\boldsymbol{\alpha}}^{it} \mathbf{X}^t \mathbf{X} \hat{\boldsymbol{\alpha}}^i} = 1$ resulting in

$$\boldsymbol{\alpha}^i = \frac{1}{\sqrt{\hat{\boldsymbol{\alpha}}^{it} \mathbf{X}^t \mathbf{X} \hat{\boldsymbol{\alpha}}^i}} \hat{\boldsymbol{\alpha}}^i \tag{2.72}$$

The above procedure gives the coefficients of the basis vectors in terms of the original. To use the the basis for computation, we should be able to project a new sample $\phi(\mathbf{x}_t)$ on to the basis vectors. This is straightforward since $\langle \phi(\mathbf{x}_t), \mathbf{q}_i \rangle = \mathbf{k}_t^t \boldsymbol{\alpha}^i$. The methods describe above help us in analysis of the data in the feature space, using measures directly related to inner product such as distances and projections. However, deeper analysis of data is required for tasks such as compression of data without losing much of the inherent structure (compression and dimensionality reduction) and selecting features that best suit a task e.g classification (feature selection). Examples of kernelization of algorithms which perform such tasks are presented in the following sections.

2.5 Feature Extraction, Feature Selection and Modeling

2.5.1 Principal Component Analysis

The goal of principal component analysis (PCA) is to find the directions along which the variance is maximum. The reason for seeking these directions is that the directions with low variance do not provide any information about the data. Such directions may be discarded by projecting the data on the more informative directions (called principal components). It can be shown that these directions are the eigen vectors corresponding to larger eigen values of the covariance matrix. The kernelization of this method was proposed by Scholkopf *et.al* [3] As shown in section 2.4, the dual coefficients $\boldsymbol{\alpha}$ of these eigen vectors(in the feature space) satisfy the property that $\mathbf{K}_C\boldsymbol{\alpha}$ are eigen vectors of $\boldsymbol{\alpha}$. Thus, we have,

$$\begin{aligned} \mathbf{K}_C(\mathbf{K}_C\boldsymbol{\alpha}) &= \lambda\mathbf{K}_C\boldsymbol{\alpha} \\ \mathbf{K}_C\boldsymbol{\alpha} &= \lambda\boldsymbol{\alpha} \end{aligned} \tag{2.73}$$

The above equation indicates that the eigen vectors corresponding to the maximum eigen values of the centered kernel matrix give the dual coefficients of the principal components in the feature space. This result can be obtained by directly optimizing the variance along a line $\mathbf{w} = \mathbf{X}_C\boldsymbol{\alpha}$. Assuming that the data is centered, the variance, as shown in section 2.4, can be expressed as

$$\sigma^2_{\boldsymbol{\alpha}} = \frac{1}{n}\boldsymbol{\alpha}^t\mathbf{K}_C^2\boldsymbol{\alpha} \tag{2.74}$$

Maximizing the above equation subject to the constraint $||\mathbf{w}||^2 = \boldsymbol{\alpha}^t\mathbf{K}_C\boldsymbol{\alpha} = 1$ using lagrangian multiplier λ_1 results in:

$$\begin{aligned} \frac{\partial(\sigma^2_{\boldsymbol{\alpha}} - \lambda_1(\boldsymbol{\alpha}^t\mathbf{K}_C\boldsymbol{\alpha} - 1))}{\partial\boldsymbol{\alpha}} &= \frac{2}{n}\mathbf{K}_C^2\boldsymbol{\alpha} - 2\lambda_1\mathbf{K}_C\boldsymbol{\alpha} = 0 \\ \Rightarrow \mathbf{K}_C\boldsymbol{\alpha} &= n\lambda_1\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha} \end{aligned} \tag{2.75}$$

Since the eigen vectors need to be normalized in the feature space, the eigen vectors corresponding to the top k eigen values $\lambda_1, \cdots, \lambda_k$ of $\mathbf{K}_C$, $\{\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^k\}$, need to be normalized such that $||\mathbf{X}\boldsymbol{\alpha}^i||^2 = 1$. Since

$$||\mathbf{X}\boldsymbol{\alpha}^i||^2 = \boldsymbol{\alpha}^{it}\mathbf{K}_C\boldsymbol{\alpha}^i = \boldsymbol{\alpha}^{it}\lambda_i\boldsymbol{\alpha}^i = \lambda_i||\boldsymbol{\alpha}||^2 = \lambda_i$$

the normalized vectors $\{\lambda_1^{-1/2}\boldsymbol{\alpha}^1, \cdots, \lambda_k^{-1/2}\boldsymbol{\alpha}^k\}$ give the final dual coefficient of the principal components in the feature space (called kernel principal components). Kernel principal component analysis has been extensively used for extraction of nonlinear feature descriptors [35, 98]. Figure 2.5 demonstrates how kernel principal components aid in description of nonlinear relationships among data.

2.5.2 Linear Discriminant Analysis

Principal Component Analysis aids in extracting description that described the data well. However, not all such features are useful for every task. For instance, in the case of face recognition, given three persons A, B and C, the features that aid in distinguishing between A and B are, in general, different from those that discriminate between B and C. Hence, the selection of features that are

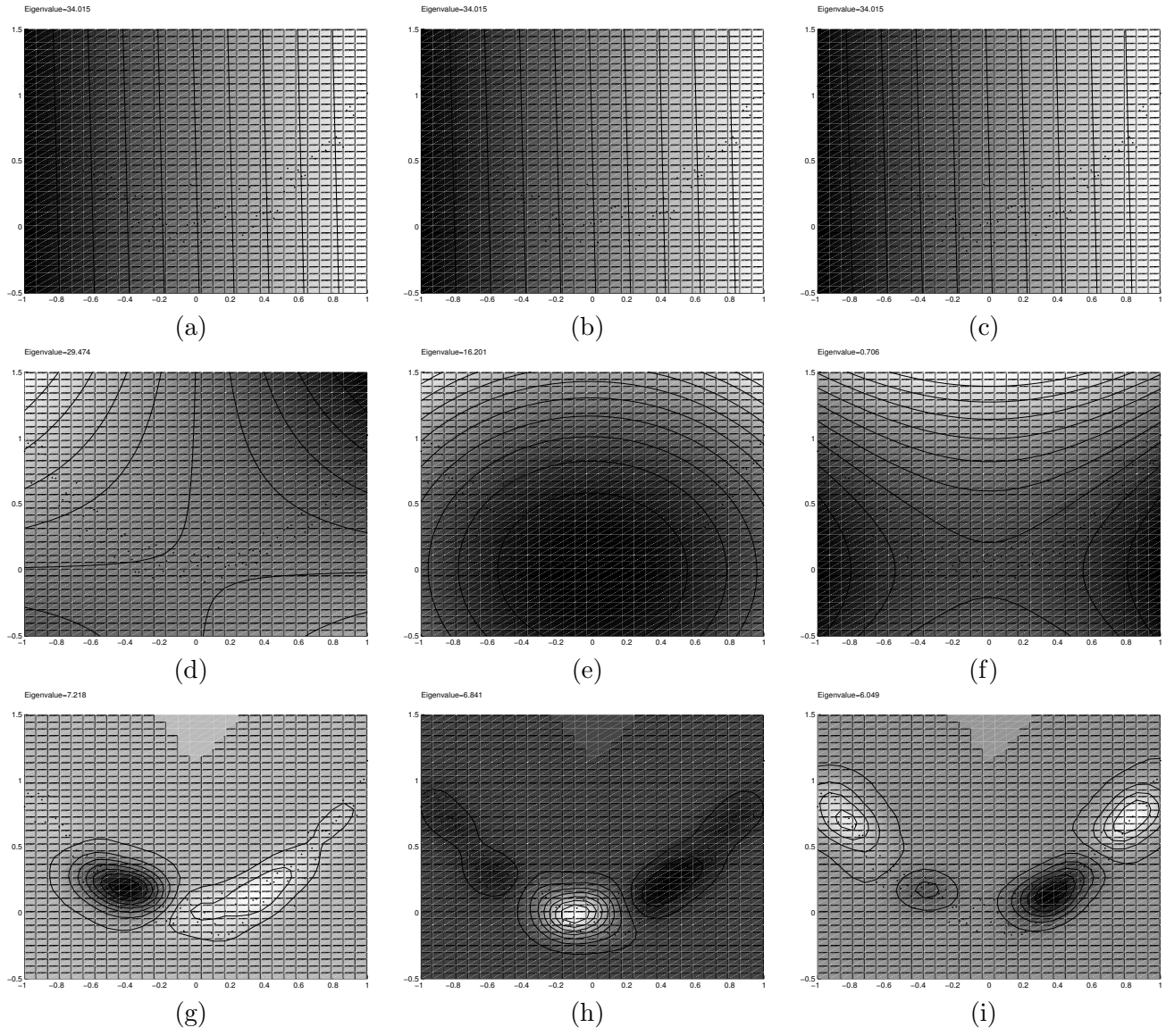


Figure 2.5: Nonlinear component analysis using kernel functions. The top three eigen vectors of the kernel matrix were used to derive the components shown with various kernels. (a),(b),(c) : linear kernel, (d),(e),(f) : quadratic kernel, (g),(h),(i) : Gaussian radial basis kernel with $\sigma = 0.1$. In all cases the third component has very low significance compared to the other two. This is expected since the intrinsic dimension of the input set it 2. However, the nonlinear components describe the data better.

optimal for a given task is an important problem in pattern analysis. Linear Discriminant Analysis is a method to find the best directions that aid in distinguishing between different classes. The Fisher Discriminant Analysis is one such method, the kernel variant of which was proposed by Mika et.al [4]. The current discussion addresses the case of two classes. The key idea is that the sample mean and variance of each class describe the distribution of that class. Let $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ be the sample means of the two classes. Then, the quantity $\langle \boldsymbol{\mu}_+, \mathbf{w} \rangle - \langle \boldsymbol{\mu}_-, \mathbf{w} \rangle = \mu_{\mathbf{w}}^+ - \mu_{\mathbf{w}}^-$ is a measure of how much the distributions differ along the direction $\mathbf{w}$. Discriminating between the classes also would require the samples of each class to be tightly clustered around the sample mean. The variance along $\mathbf{w}$, for each class, can be used as measure for this fit. There are many objective functions involving these terms, which can be optimized to obtain the optimal direction $\mathbf{w}$. A popular objective function used frequently is

$$J(\mathbf{w}) = \frac{(\mu_{\mathbf{w}}^+ - \mu_{\mathbf{w}}^-)^2}{{\sigma_{\mathbf{w}}^+}^2 + {\sigma_{\mathbf{w}}^-}^2} \tag{2.76}$$

To express the above equation in the dual form using the data matrix notation, the following additional symbols are defined. Let n be the number of samples and $\mathbf{X}$ be the data matrix. The number of samples belonging to the two classes be n^+ and n^-. The vector $\mathbf{j}_+$ is n-vector with $j_{+i} = \frac{1}{n^+}$ if ith column of $\mathbf{X}$ is in the positive class and $j_{+i} = 0$ otherwise. The vector $\mathbf{j}_-$ is defined similarly. Thus the sample means of the positive and negative classes are :

$$\boldsymbol{\mu}_C^+ = \mathbf{X}\mathbf{j}_+$$
$$\boldsymbol{\mu}_C^- = \mathbf{X}\mathbf{j}_-$$

The positive and negative data matrices $\mathbf{X}_+$ and $\mathbf{X}_-$ are formed by taking the input samples belonging to the respective class. The matrix $\mathbf{I}_+$ is a selector matrix whose ith column is the ith canonical vector $\mathbf{e}_i$ if the ith sample belongs to the positive class and $\mathbf{0}$ otherwise. Thus the data matrices are related by :

$$\mathbf{X}_+ = \mathbf{X}\mathbf{I}_+$$
$$\mathbf{X}_- = \mathbf{X}\mathbf{I}_-$$

Similar the class specific kernel matrices also can be defined as $\mathbf{K}_+ = \mathbf{X}_+^t\mathbf{X}_+$ and $\mathbf{K}_- = \mathbf{X}_-^t\mathbf{X}_-$. Thus the dual objective function can be written as

$$\begin{aligned} J(\boldsymbol{\alpha}) &= \frac{(\boldsymbol{\alpha}^t\mathbf{X}^t\mathbf{X}\mathbf{j}_+ - \boldsymbol{\alpha}^t\mathbf{X}^t\mathbf{X}\mathbf{j}_-)^2}{\frac{1}{n^+}\boldsymbol{\alpha}^t\mathbf{K}_+^2\boldsymbol{\alpha} - (\mathbf{1}^t_{\frac{1}{n^+}}\mathbf{K}_+\boldsymbol{\alpha})^2 + \frac{1}{n^-}\boldsymbol{\alpha}^t\mathbf{K}_-^2\boldsymbol{\alpha} - (\mathbf{1}^t_{\frac{1}{n^-}}\mathbf{K}_-\boldsymbol{\alpha})^2} \\ &= \frac{\boldsymbol{\alpha}^t(\mathbf{K}\mathbf{j}_+\mathbf{j}_+^t\mathbf{K} + \mathbf{K}\mathbf{j}_-\mathbf{j}_-^t\mathbf{K} - 2\mathbf{K}\mathbf{j}_+\mathbf{j}_-^t\mathbf{K})\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t(\frac{1}{n^+}\mathbf{K}_+^2 + \frac{1}{n^-}\mathbf{K}_-^2 - \mathbf{K}_+\mathbf{1}_{\frac{1}{n^+}}\mathbf{1}^t_{\frac{1}{n^+}}\mathbf{K}_+ - \mathbf{K}_-\mathbf{1}_{\frac{1}{n^-}}\mathbf{1}^t_{\frac{1}{n^-}}\mathbf{K}_-)\boldsymbol{\alpha}} \\ &= \frac{\boldsymbol{\alpha}^t\mathbf{A}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t\mathbf{B}\boldsymbol{\alpha}} \end{aligned} \tag{2.77}$$

where $\mathbf{A}$ and $\mathbf{B}$ can be computed from the kernel matrix alone. Optimizing this function subject to the constraint $||\mathbf{w}||^2 = \boldsymbol{\alpha}^t\mathbf{K}\boldsymbol{\alpha} = 1$ leads to the following generalized eigen vector problem

$$\mathbf{A}\boldsymbol{\alpha} = \lambda(\mathbf{B} + \mathbf{K})\boldsymbol{\alpha} \tag{2.78}$$

Thus, the most discriminative directions can be found by finding the generalized eigen vectors corresponding the maximum generalized eigen values of the matrices above. The projection of test sample and other computations are similar to the case of kernel PCA and kernel perceptron. Figure 2.6 shows how kernel fisher discriminants extract nonlinear relationships in the input space.

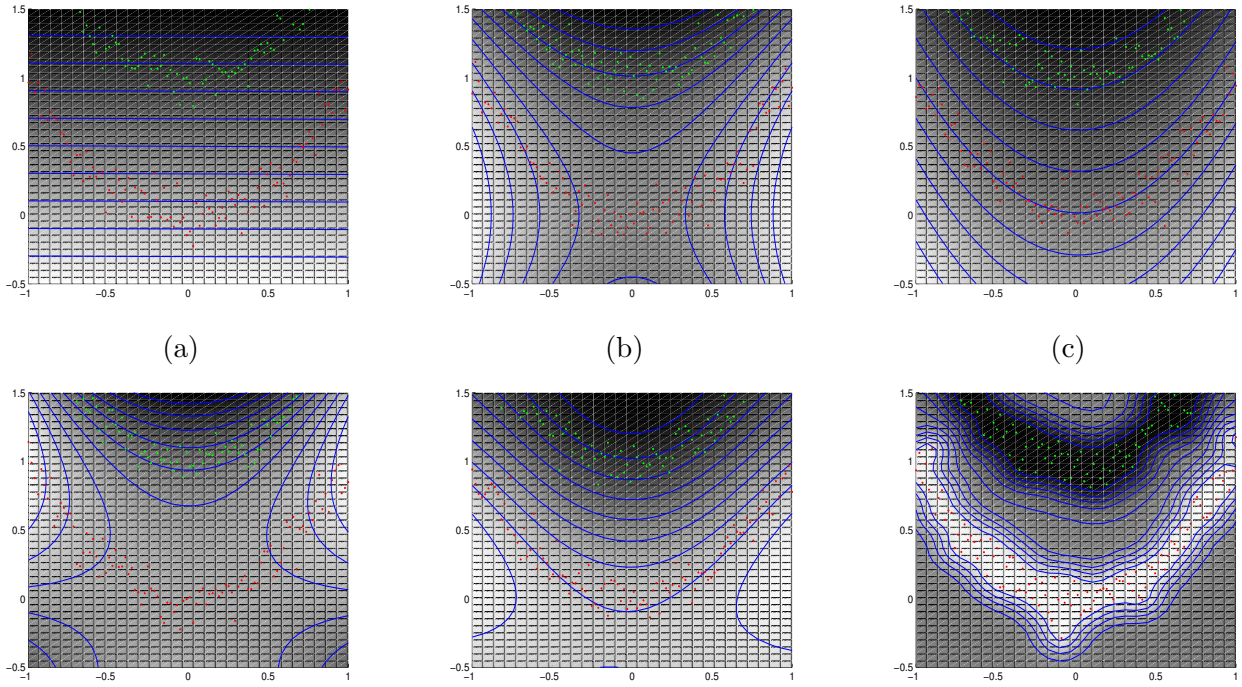


Figure 2.6: Kernel fisher discriminant analysis finds the most discriminative directions in the feature space. Here, the algorithm is used to learn a classification function to separate two data sets (one in green and the other in red) each distributed in parabolic shapes. (a) shows the result using a linear kernel $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ which fails to find a satisfactory solution. (b) and (c) show results with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$ and $\kappa(\mathbf{x}, \mathbf{y}) = (1+\langle \mathbf{x}, \mathbf{y} \rangle)^2$ respectively. (d) shows the result with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^3$. (e) and (f) show the results with $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1, 0.1$ respectively. Note the difference between these boundaries and the ones obtained with the kernel perceptron in figure 2.3
.

2.6 Classification and Support Vector Machine

2.6.1 Optimal Separating Hyperplanes

While feature extraction and feature selection help us in extracting meaningful and relevant descriptions of data, classification is an altogether different problem, where the goal is to learn boundaries that separate different classes of data. This is a more challenging problem since such classification rules, which are inferred from a finite set of data, are expected to classify unseen examples with high accuracy. The input set from which the classification function is learned is called the *training set*. Since this set is finite, it is not guaranteed that a given classification function would commit errors with the same frequency on unseen examples as it did on this set. However, statistical learning theory [15] allows us to relate the error rate on the training set and the true error rate i.e. the expected error on unseen examples. This relation is the foundation for support vector algorithm. A complete review of learning theory is beyond the scope of this thesis and the additional details can be found in Vapnik [15]. The central result that aids in understanding the development of the support vector algorithm is reviewed below.

Consider a input set $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where $y_i \in \{\pm 1\}$, with each example drawn independently and distributed identically (*iid*) according to a distribution $P(\mathbf{x}, y)$. Let f be a function parametrized by α, the task of which is to learn the relationship between $\mathbf{x}_i$ and y_i. Thus, expected error rate of f is :

$$R(\alpha) = \int \frac{1}{2}|f_\alpha(\mathbf{x}) - y| dP(\mathbf{x}, y) \tag{2.79}$$

However $R(\alpha)$ is not computable as we have only finite amount of data. What can be computed is the empirical error i.e the error rate of f on the training set :

$$R_{emp}(\alpha) = \sum_{i=1}^{n} \frac{1}{2n}|f_\alpha(\mathbf{x}_i) - y_i| \tag{2.80}$$

As the number of input samples increases i.e., $n \to \infty$, the empirical risk $R_{emp}(\alpha)$ approaches $R(\alpha)$. However, the rate at which the two quantities converge and the exact relationship between them is not apparent. The central result of learning theory is that, with a probability of $1 - \eta$ where $0 \leq \eta \leq 1$, the following inequality holds

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h \log(2n/h) + h - \log(\eta/4)}{n}} \tag{2.81}$$

where h is a measure of descriptive power of f, called the Vapnik Chervonenkis(VC) dimension. The VC dimension is a property of a class of functions $\{f_\alpha\}$. For a two-class classification problem, there exists a straightforward characterization of VC dimension. Given a set n input samples, there exist 2^n different labellings possible since each sample can be assigned a label $+1$ or -1. If for each such labeling, a member $f_\alpha^* \in \{f_\alpha\}$ can be found such $R_{emp}(\alpha^*) = 0$, then the input set is said to be *shattered* by the the function class $\{f_\alpha\}$. The VC dimension of a function class $\{f_\alpha\}$ is the maximum number of points that can be shattered by $\{f_\alpha\}$. For instance, for lines in $\mathcal{R}^2$ the VC dimension is 3 as proven by figure 2.7.

The second result that is used in the development of SVM is the upper bound on the VC dimension of class of hyperplanes in $\mathcal{R}^n$ described by parameters $\alpha = (\mathbf{w}, b)$:

$$f_{\mathbf{w},b} = sign(\langle \mathbf{w}, \mathbf{x} \rangle + b) \tag{2.82}$$

To make this function class more precise we need to obtain a canonical form to describe each member of $(\mathbf{w}, b)$ in the class $\{f_{\mathbf{w},b}$ since for any $k \neq 0$, $(k\mathbf{w}, kb)$ also describes the same member. This

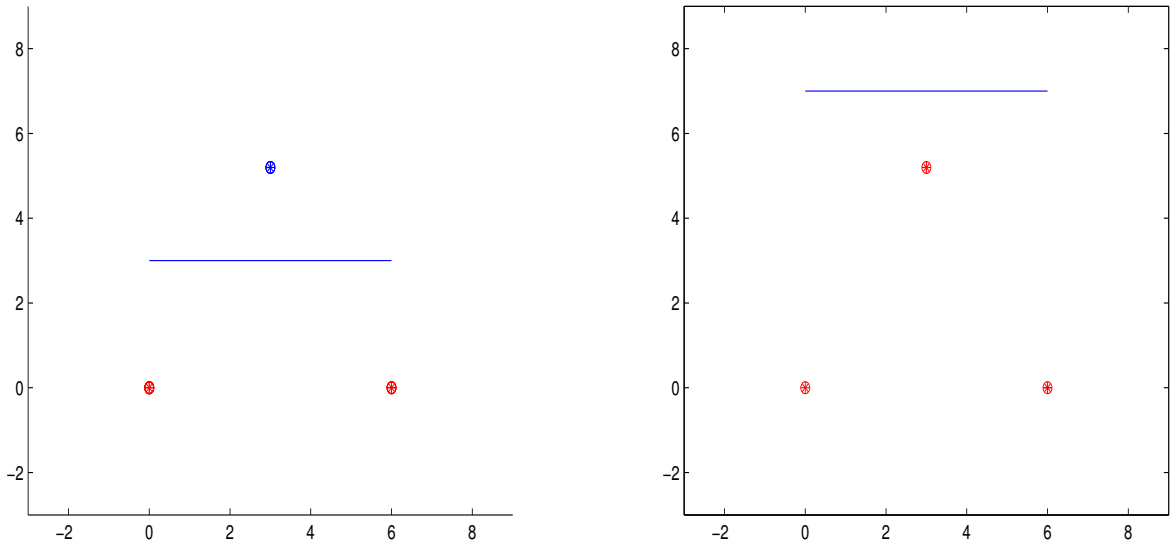


Figure 2.7: VC dimensions of lines in $\mathcal{R}^2$ is 3. Lines separating two labellings of vertices of an equilateral triangle. By symmetry, the remaining labellings also can be separated.

canonical form can be obtained by enforcing the constraint that the closest points to the hyperplane should be at unit distance from the plane. Figure 2.8 shows how this is achieved. Although this constraint does not distinguish between the pairs $(\mathbf{w}, b)$ and $(-\mathbf{w}, -b)$, the class labels allows us to do this. Formally, this constraint can be expressed as

$$\min_{i=1\cdots n} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1 \tag{2.83}$$

Vapnik [15] shows that the following upper bound for VC dimension of the above function class holds :
Given an input set $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, if R is the radius of the smallest hypersphere enclosing these points, then the VC dimension h of the function class $\{f_{\mathbf{w},b} : ||\mathbf{w}|| \leq A\}$, where

$$f_{\mathbf{w},b} = sign(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

satisfies the following inequality

$$h < R^2 A^2 + 1 \tag{2.84}$$

The above two results aid in choosing the plane, among the planes that classify the data correctly, that has the least expected risk. Since $R(\alpha)$ increases with h and h is bounded by A the optimal canonical hyperplane is obtained by minimizing $||\mathbf{w}||$.

2.6.2 Finding the optimal separating hyperplane

Assuming that there exist hyperplanes that classify the data correctly, the optimization problem can be posed formally as follows. The function to be minimized is

$$J(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 \tag{2.85}$$

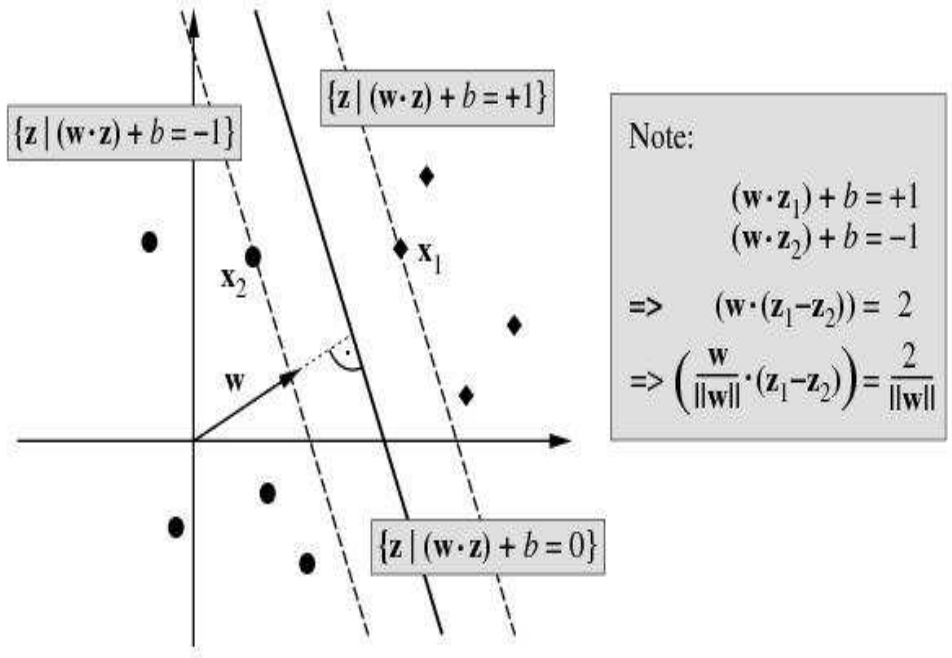


Figure 2.8: Obtaining a canonical representation of a hyperplane described by $(\mathbf{w}, b)$. Image taken from [99].

subject to the following n constraints

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \tag{2.86}$$

Using the lagrangian multipliers $\alpha_1, \cdots, \alpha_n$ the Lagrangian becomes

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) \tag{2.87}$$

Setting the partial derivatives $\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{b}$ and $\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\mathbf{w}}$ to 0 we obtain

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.88}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{2.89}$$

A key property of the above optimization (a result of Kuhn-Tucker theorem) is that only those α_i are non-zero for which the input samples lies exactly at unit distance from the canonical plane i.e $(\mathbf{x_i}, y_i)$ satisfies the following constraint

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 = 0 \tag{2.90}$$

Such vectors are called *support vectors* and are the most important subset of the input samples. In fact, removing the other samples does not affect the final solution. The dual optimization function W($\boldsymbol{\alpha}$) by using this fact and by substituting the expression for $\mathbf{w}$ in $L(.,.,.)$. The function to maximize is :

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{2.91}$$

subject to the constraints

$$\begin{aligned} \alpha_i \geq 0, i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned} \tag{2.92}$$

Solving the above optimization problem gives the dual coefficients of the vector $\mathbf{w}$. Note that $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ and depends on the sign the of the label assigned to $\mathbf{x}_i$ unlike the previous cases where $\mathbf{w}$ was a linear combination of the input samples. A big drawback of the above formulation is the assumption that the data is separable by a hyperplane. However, even when this is not the case the plane that separates a good fraction of examples is still useful. To allows for examples that violate the constraint in equation(2.86) slack variables $\xi_i \geq 0$ for $i = 1, \cdots n$ are introduced with the following relaxed separation constraints

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \tag{2.93}$$

Minimizing the bound on the expected error leads to the following optimization : minimize

$$j(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 + \gamma \sum_{i=1}^{n} \xi_i \tag{2.94}$$

where γ is a constant that controls the empirical risk i.e the number of mis-classifications on the training set. Setting up the Lagrangian as in the separable case the dual optimization problem can be found to be : maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{2.95}$$

subject to the constraints

$$\begin{aligned} \gamma \geq \alpha_i \geq 0, i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned} \tag{2.96}$$

The final step is to introduce nonlinearity by kernelizing the SV algorithm. This is rather straightforward as equation(2.95) accesses the input samples only via the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ which in the feature space is $\kappa(\mathbf{x}_i, \mathbf{x}_j)$. Since the constraints do not involve the input samples the optimization problem in the feature space can be posed as : maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{2.97}$$

subject to the constraints

$$\begin{aligned} \gamma \geq \alpha_i \geq 0, i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned} \tag{2.98}$$

Projecting a test sample on to the hyperplane is done in a manner similar to the previous cases. The support vector machine and its many variants are widely studied in literature. Ever since their discovery, SVMs have been widely used for a number of applications involving classification and recognition. Many approaches to problems such as parameter selection, parallelization and other complexity issues have been proposed earlier. The reader is referred to [2, 93] for more information on these techniques and applications. Figure 2.9 shows how SVM learns optimal separating boundaries with various kernels.

2.7 Further Challenges in Kernel Methods

The above examples demonstrate various ways in which the kernel paradigm can be used to make algorithms more descriptive while retaining their statistical properties. Kernel methods possess many advantages other than nonlinearity such as modularity, ability to work with heterogeneous descriptions of data, incorporation of prior knowledge etc. However, a major issue in all the kernel methods is the choice of kernel function. The kernel function defines the geometry of the space in which an algorithm operates and this is crucial for the performance of the algorithm in that space. Although many methods have been developed for selection of kernel function optimal for a specific task [96, 95], in general, there is no way of choosing or constructing a kernel that is optimal for a given problem.

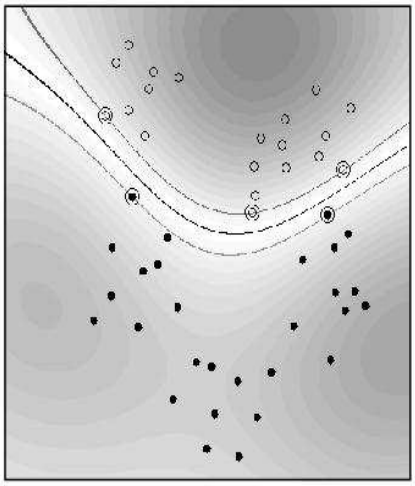


Figure 2.9: Boundary learned by an SVM with an Gaussian radial basis kernel

Another important limitation is the complexity of kernel algorithms. Kernel methods access the feature space via the input samples and hence kernel algorithms need to store all the relevant input samples. For instance, in the case of SVMs all the support vectors need to stored so that they can be used to project a test sample on the the separating hyperplane. Similarly, while inferring relationships from large number of samples, the size of the kernel matrices grows quadratically with the number of samples. Methods such as the reduced set SVM [100], approximate factorization of kernel matrix [101] attempt to reduce the space and time complexity in the case of SVMs. Despite these recent developments scalability of kernel methods still is a challenging problem with scope for further research.

Apart from kernel selection and scalability, the design of kernels, incorporating priors knowledge and invariances in to kernel functions are some of the challenges in kernel methods. Despite these shortcomings the numerous advantages of kernel method over traditional linear algorithms make them the preferred choice for practical applications. With a number of problems still open, kernel methods is an active area of research with promising future prospects.

Chapter 3

Kernel Algorithms For Feature Selection and Modeling

3.1 Introduction

Feature selection and Modeling are two of the important problems in analysis of data. In a number of practical scenarios, data is described using a large number of features for each sample. However not all of these may be relevant when performing a particular task. Selection of the relevant features helps us in i) dealing with less bulky descriptions of data and ii) discarding irrelevant information that could potentially make the task much more difficult. For instance, detection of sign boards in an image requires features that describe the overall appearance of a sign board (large vertical edges, circular region at the top etc.) while recognizing the sign itself would need much finer details (such as the direction of dominant edges on the board etc.). The problem of selection of features that are optimal for a given task is known as feature selection. Modeling is the process of constructing a model, given some finite set of data samples, that has a statistical behavior (exactly or approximately) similar to the source process that generates the data. Constructing such a model helps us in a number of tasks such as compression, generation of synthetic data, testing whether a new sample is generated by the source etc. Methods that select derived features that depend linearly on the input set of features can also be viewed feature selectors [60, 102] although they do not *select* features directly from the input set of features. As in the case of linear algorithms for feature extraction and classification, the power of such methods can be enhanced by introducing nonlinearity via kernel functions. Similarly modeling schemes where the model estimation requires only the pairwise inner product information can be kernelized to enable construction of richer models. In this chapter, we kernelize two such algorithms Multiple Exemplar Biased Discriminant Analysis(*MEDBA*) and Linear Predictive Coding(*LPC*) [103] for feature selection and modeling respectively. The methods and their applications are informally described below. The mathematical details and kernelization are discussed in the following sections.

Retrieval of data samples that are similar to a given set of examples and verifying whether a new data sample belongs to the same distribution as a given set of examples are important problems in pattern analysis. Popular applications include the content based image retrieval systems where images *similar* to a given a image are retrieved. However, since the interpretation of images by humans is highly subjective a fixed similarity measure would not retrieve satisfactory results consistently. Features that capture a particular user's notion of similarity need to be extracted [104, 105] for such tasks. Thus the problem boils to down to selection of features that best characterize a class of interest(positive class) and aid in rejection of those examples that do not belong to this

class(negative classes). A closely related problem of great practical importance is verifying if a sample belongs to the class of interest or not. Authentication using biometric traits such as facial features, hand-geometry fall is a popular application. Again, features that are biased towards the positive class are best suited for this task. The biased discriminant analysis(BDA) [38] extracts such discriminants and was applied to both the problems described above with significant improvement in performance [39, 106]. However, BDA computes statistics of the data distribution using a single sample, i.e., the sample mean, as a prototype which is suboptimal in many real world scenarios as the underlying distribution are multimodal. Using multiple examples to compute second order statistics overcomes this limitation as was shown in the case of multiple exemplar discriminant analysis [71]. The multiple exemplar biased discriminant combines these two methods to obtain a richer set of discriminants. The kernel variant of the method is used to enhance the accuracy of a hand-geometry based authentication system. An empirical technique for selecting the optimal kernel function is used to further enhance the performance of the system. In addition to these, a hierarchical classification framework is introduced to reduce the errors caused due to presence of highly similar classes in the input set of classes.

Sequence data comprises of samples with an associated *time* variable that defines an ordering on the samples. Such representation of data is commonly used in a number of applications where the data samples are generated in a sequential order (as in the case of speech signals) or where the order of samples is important for analysis of data (as in the case of protein sequences). Modeling the relationship between successive samples of such sequences aids in tasks such as compression, prediction and synthesis. Linear Predictive Coding(LPC) [103] is a popular modeling technique widely used to solve a number of practical problems such as speech recognition, handwriting recognition and planar shape recognition [75, 77, 107]. The simplicity of the model and the ease of estimation of model parameters have made it popular for these applications. However, when the signal is complex the linear model cannot encode the relationships effectively. The kernel trick enables us to overcome this problem by introducing nonlinearity without significant increase in complexity. Kernel Linear Predictive coding (KLPC) is the kernel variant of LPC. Although the technique cannot be used for tasks such as prediction, owing to the inaccessibility of the feature space, the model parameters can be used for tasks such as recognition. The method is used to recognize hand-written characters and planar shapes (from their contours). In both the experiments the method resulted in significant improvement in performance compared to the linear variant.

Sections 3.2 and 3.3 describes kernel multiple exemplar biased discriminant analysis and its application to biometric authentication. Sections 3.4, 3.5 and 3.6 described kernel linear predictive coding and its application to handwriting and planar shape recognition. Section 3.7 discusses possible extensions to the methods.

3.2 Authentication using Kernel MEBDA

In many practical applications an input sample comes with a claimed label (such as a biometric trait along with claimed identity of a person). Authentication of such a claim poses challenges that are markedly different from those involved in problems like recognition and detection. While traditional methods used for recognition can be used for authentication, they are often suboptimal. The issues that arise in building authentication systems that motivated the currently proposed solution are

- In presence of multiple classes (as is the case in practice) it is impractical to extract discriminants that separate each pair of input classes. Hence a feature set that discriminates between the class of interest and the rest needs to be extracted.

- The number of samples available for each class in practice is often very small. Since the sample mean is taken as an ideal prototype representing each class, small numbers of samples renders the estimation of discriminating directions unstable. Hence all the examples available for each class should be used while estimating the discriminating directions [71].

- Traditional feature selection schemes based on linear discriminant analysis(MDA) that identify a global set of features that aid in discriminating between a set of classes are conventionally used for biometric based recognition [108]. However, these methods extract features that are linear in the input set of features and hence are not descriptive enough to represent complex class boundaries.

- The presence of highly similar classes in the input set of classes is a source of errors in two ways : i) Errors that occur due to the inherent similarity of the two classes ii) Errors caused to erroneous estimation of discriminants. The erroneous estimation is a result of treating two highly similar classes as separate classes.

Many methods in literature separately address the problems described above [38, 71, 40, 72]. The solution proposed in this chapter combines the advantages of these solutions together to solve the problem of authentication effectively even with a relatively less discriminative feature set. The steps that lead to the final algorithm are described below.

3.2.1 Biased Discriminant Analysis

Selection of class specific features that discriminate each class in the input set from the rest is straightforward when the features are highly discriminative (as in the case of *strong* biometrics such as finger prints [68]). However when the features distributions of different classes overlap substantially the discriminant extraction must be biased towards the positive class. The biased discriminant analysis [38] does this in the following manner :

- Given the input set of samples $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and a class of interest P, $\mathcal{X}$ is partitioned in to two sets $\mathcal{X}_P = \{\mathbf{x}_i, \mathbf{x}_i \in P, i = 1, \cdots, n\}$ and $\mathcal{X}_N = \{\mathbf{x}_i, \mathbf{x}_i \notin P, i = 1, \cdots, n\}$ which represent the positive class and negative class respectively.

- The sample mean of the positive class $\boldsymbol{\mu} = \frac{1}{|\mathcal{X}_P|} \sum_{\mathbf{x}_i \in \mathcal{X}_P} \mathbf{x}_i$ is treated as the prototype of the positive class and the following scatter matrices are computed

$$\begin{aligned} \mathbf{S}_p &= \sum_{\mathbf{x}_i \in \mathcal{X}_P} (\mathbf{x}_i - \boldsymbol{\mu}\ )(\mathbf{x}_i - \boldsymbol{\mu}\ )^t \\ \mathbf{S}_n &= \sum_{\mathbf{x}_i \in \mathcal{X}_N} (\mathbf{x}_i - \boldsymbol{\mu}\ )(\mathbf{x}_i - \boldsymbol{\mu}\ )^t \end{aligned} \tag{3.1}$$

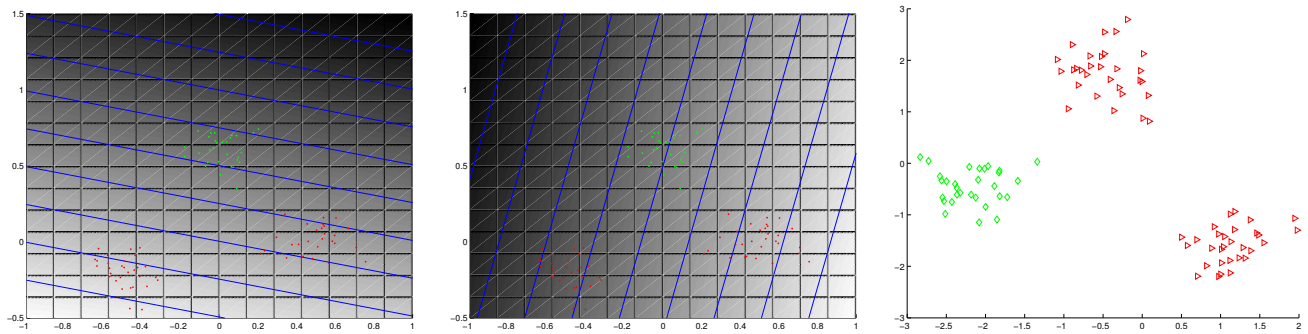
  These matrices measure the scatter of the positive class and the negative class around the sample mean of the positive class.

- The objective of BDA is to find a direction $\mathbf{w}$ such the positive scatter is small and the negative scatter is high. Thus the problem can be posed as the maximization of the following joint optimization function :

$$J(\mathbf{w}) = \frac{\mathbf{w}^t \mathbf{S}_n \mathbf{w}}{\mathbf{w}^t \mathbf{S}_p \mathbf{w}} \tag{3.2}$$

As shown in the previous chapter, the above function attains a maximum when $\mathbf{w}$ is the generalized eigen vector corresponding to the generalized eigen values of the matrix $\mathbf{S}_p^{-1}\mathbf{S}_n$. However, when the number of positive samples is less the matrix $\mathbf{S}_p$ becomes singular. This is usually overcome by adding a regularization term $\lambda\mathbf{I}$ at the cost of a suboptimal solution. Alternate methods to overcome singularity and small-sample issues arise by using the method proposed for MDA [109] which uses generalized singular values and corresponding vectors to solve the problem.

Figure 3.1 shows the method extracts discriminants that aid in separating the positive class from the negative class. The single class framework described above is gaining increasing attention for problems involving authentication and retrieval. Class specific feature selection using BDA is applied to hand-geometry based authentication in [106] with significant improvement in the false acceptance and false rejection rates (*FAR* and *FRR* respectively).



a b c

Figure 3.1: The direction selected using the biased discriminant analysis aid in separating the positive class from the negative class. The positive class is shown in green and two negative classes in red (all following a unimodal distribution). (a) and (b) show the first two biased discriminants and (c) shows the transformed data.

An alternative way to handle the small-sample size problem is motivated by the multiple exemplar discriminant analysis [71]. The central idea behind the multiple exemplar method is to treat all the available samples of the positive class as representatives of the class and compute the scatter matrices as follows

$$\begin{aligned}\mathbf{S}_p &= \sum_{\mathbf{x}_i \in \mathcal{X}_P} \sum_{\mathbf{x}_j \in \mathcal{X}_P} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^t \\ \mathbf{S}_n &= \sum_{\mathbf{x}_i \in \mathcal{X}_P} \sum_{\mathbf{x}_j \in \mathcal{X}_N} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^t \end{aligned} \tag{3.3}$$

The optimization problem is solved using generalized eigen vectors of the above matrices as above. As argued in [71] this methods aids in estimation of better discriminants when the number of available positive samples is small and also in cases where the distribution of the positive class in multimodal. Figure 3.2 demonstrates that this method extracts better discriminants in the case multimodal distributions.

Selection of features using BDA and Multiple Exemplar BDA address the first two issues mentioned at the beginning of this section. However both the methods extract discriminants that linear

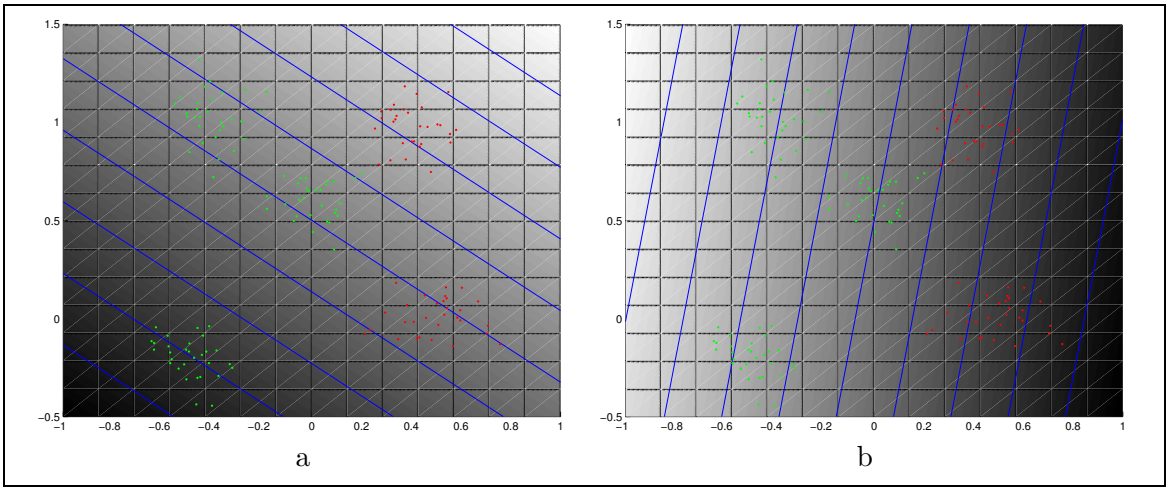


Figure 3.2: The directions selected using the multiple exemplar biased discriminant analysis are better suited for authentication in case of multimodal distributions. The positive class (shown in green) follows a multimodal distribution. (a) shows the first direction selected by BDA and (b) shows the direction chosen by multiple exemplar BDA.

in the input set of features. The limited descriptive power of linear discriminants precludes the application of these methods to problems involving complex data distributions. The use of the kernel trick greatly enhances the utility of these algorithms by selecting nonlinear features. The kernelization of these algorithms is described below.

3.2.2 Kernel BDA and Multiple Exemplar KBDA

The data matrix notation described in the previous chapter simplifies the expression for the scatter matrices and makes the kernelization easier. Let $\phi(.)$ be the feature embedding corresponding to the kernel function $\kappa(.,.)$. The data matrix in the feature space $\mathbf{X} = \begin{bmatrix}\phi(\mathbf{x}_1) & \cdots & \phi(\mathbf{x}_n)\end{bmatrix}$. The data matrices $\mathbf{X}_p$ and $\mathbf{X}_n$ consist of the positive samples and negative samples respectively. Let p be the number of positive samples. Using the notation of the previous chapter the sample mean of the positive samples can be written as $\mathbf{X}_p\mathbf{1}_{\frac{1}{p}}$. Hence the scatter matrices (in the feature space) can be written as follows (to indicate the size of the matrix $\mathbf{1}_{\frac{1}{p}}$ the notation $\mathbf{J}_{m\times n}$ is used for a matrix of size $m \times n$ with all entries equal to $\frac{1}{p}$):

$$\begin{aligned}\mathbf{S}_p &= (\mathbf{X}_p - \mathbf{X}_p\mathbf{J}_{p\times p})(\mathbf{X}_p - \mathbf{X}_p\mathbf{J}_{p\times p})^t \\ \mathbf{S}_n &= (\mathbf{X}_n - \mathbf{X}_p\mathbf{J}_{p\times n})(\mathbf{X}_n - \mathbf{X}_p\mathbf{J}_{p\times n})^t\end{aligned} \tag{3.4}$$

The optimization problem can now be expressed as estimation of the dual coefficients $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1, \cdots, \alpha_n\end{bmatrix}$ where $\mathbf{X}\boldsymbol{\alpha}$ is the desired discriminant direction in the feature space. Thus the objective function in terms of $\boldsymbol{\alpha}$ can be expressed as

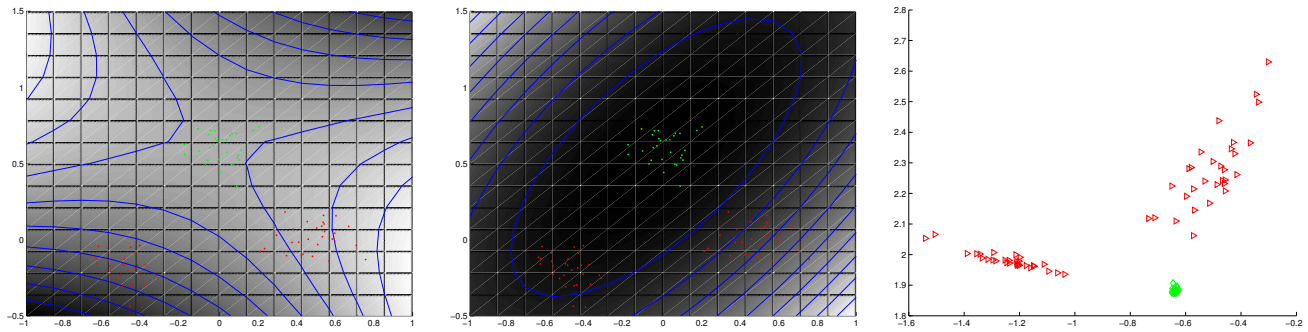
$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}\ {}^t\mathbf{X}^t(\mathbf{X}_n - \mathbf{X}_p\mathbf{J}_{p\times n})(\mathbf{X}_n - \mathbf{X}_p\mathbf{J}_{p\times n})^t\mathbf{X}\boldsymbol{\alpha}}{\boldsymbol{\alpha}\ {}^t\mathbf{X}^t(\mathbf{X}_p - \mathbf{X}_p\mathbf{J}_{p\times p})(\mathbf{X}_p - \mathbf{X}_p\mathbf{J}_{p\times p})^t\mathbf{X}\boldsymbol{\alpha}} \tag{3.5}$$

To complete the kernelization it is sufficient to show that the matrices in the above equation can be computed using the kernel matrix alone. It is evident that the matrices $\mathbf{K}_p = \mathbf{X}^t\mathbf{X}_p$ and

$\mathbf{K}_n = \mathbf{X}^t\mathbf{X}_n$ can be formed by selecting the appropriate columns of the kernel matrix $\mathbf{K} = \mathbf{X}^t\mathbf{X}$. Using the matrices the objective function can be expressed as

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^t(\mathbf{K}_n - \mathbf{K}_p\mathbf{J}_{p\times n})(\mathbf{K}_n - \mathbf{K}_p\mathbf{J}_{p\times n})^t\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t(\mathbf{K}_p - \mathbf{K}_p\mathbf{J}_{p\times p})(\mathbf{K}_p - \mathbf{K}_p\mathbf{J}_{p\times p})^t\boldsymbol{\alpha}} \tag{3.6}$$

It can be seen that matrices in the numerator and denominator can be computed from the kernel matrix alone. The dual coefficients $\boldsymbol{\alpha}$ can be found from the generalized eigenvectors of these matrices. Figure 3.3 shows how kernel BDA extracts nonlinear discriminants that aid in separation of positive class distributed in a complex manner. Although the kernel biased discriminant analysis



a b c

Figure 3.3: The directions selected using kernel biased discriminant analysis using $\kappa(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$. The discriminants are nonlinear in the input feature space and are better suited for authentication in case complex distributions. (a) and (b) show the first two direction mapped to input space. (c) shows the transformed data.

extracts nonlinear discriminants that are more effective than linear discriminants, it still suffers from the small-sample size problem. The kernel multiple exemplar discriminant analysis overcomes this problem by making use of all the available positive samples. In this case the scatter matrices in the feature space are computed using as below

$$\begin{aligned}
\mathbf{S}_p &= \sum_{i=1}^{p}(\mathbf{X}_p - \phi(\mathbf{x}_i^p)\mathbf{J}_{1\times p})(\mathbf{X}_p - \phi(\mathbf{x}_i^p)\mathbf{J}_{1\times p})^t \\
\mathbf{S}_n &= \sum_{i=1}^{p}(\mathbf{X}_n - \phi(\mathbf{x}_i^p)\mathbf{J}_{1\times n})(\mathbf{X}_n - \phi(\mathbf{x}_i^p)\mathbf{J}_{1\times n})^t
\end{aligned} \tag{3.7}$$

where $\mathbf{x}_i^p$ is the i^{th} positive sample. Since the term $\mathbf{X}^t\phi(\mathbf{x}_i^p)$ is a column (denoted by $\mathbf{k}_i^p$) of the kernel matrix $\mathbf{K}$, the objective function can be expressed in terms of $\boldsymbol{\alpha}$ and matrices computed from the kernel matrix alone :

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^t \sum_{i=1}^{p}(\mathbf{K}_n - \mathbf{k}_i^p\mathbf{J}_{1\times n})(\mathbf{K}_n - \mathbf{k}_i^p\mathbf{J}_{1\times n})^t\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t \sum_{i=1}^{p}(\mathbf{K}_p - \mathbf{k}_i^p\mathbf{J}_{1\times p})(\mathbf{K}_p - \mathbf{k}_i^p\mathbf{J}_{1\times p})^t\boldsymbol{\alpha}}$$

The dual coefficients can be found in a manner similar to kernel BDA. This method combines the advantages of kernels with the use of multiple exemplars to obtain discriminants that are better than those obtained with kernel BDA. Figure 3.4 shows the discriminants obtained using multiple exemplar kernel BDA.

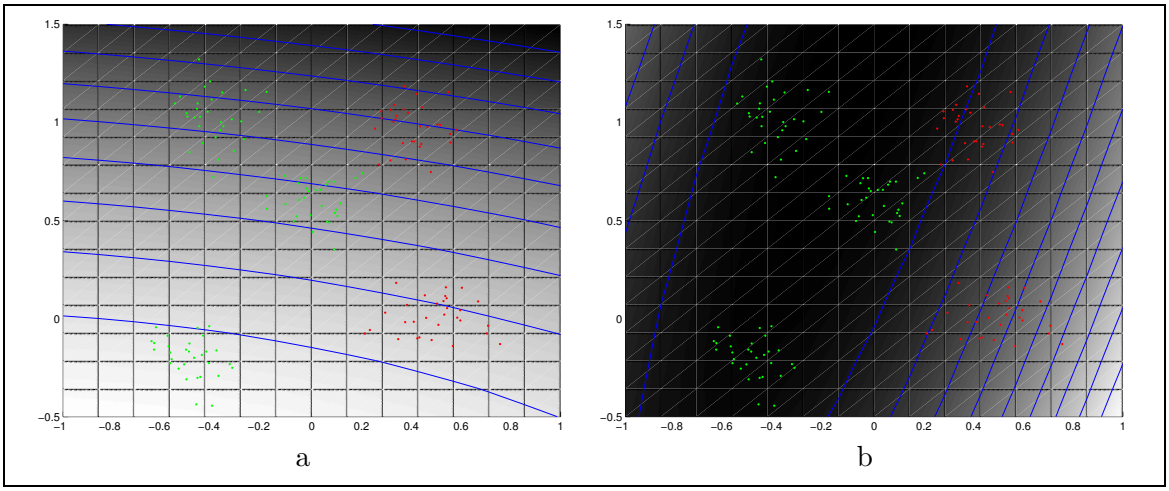


Figure 3.4: The directions selected using multiple exemplar kernel biased discriminant analysis. The discriminants are nonlinear in the input feature space and are capable of handling multimodal distributions as well. The positive class (shown in green) follows a multimodal distribution. (a) shows the first direction selected by kernel BDA and (b) shows the direction chosen by multiple exemplar kernel BDA.

Although multiple exemplar kernel BDA combines the advantages of several methods to extract powerful discriminants, it may not always be possible to construct good class boundaries for all the classes in the input set. This is due to the presence of classes that are inherently very similar. This problem becomes more pronounced when the feature set chosen is *weak* (i.e. is not highly discriminative). It is futile to search for an optimal set of discriminants that separate such classes while treating them separately. This observation leads to the following hierarchical authentication scheme that handles such difficult cases more efficiently.

3.2.3 Hierarchical Authentication

The presence of highly similar classes in the data affects the performance in two ways. Since the methods described above treat all the classes other than the positive class as a single class, the presence of data distributed similarly as the positive class results in higher number of false acceptances and additionally corrupts the computed discriminants resulting in increase in errors on other classes. The reason for this is that the small number of features that help in discriminating between the two similar classes are overlooked in the presence of a large number of negative classes. A direct way to overcome this is to introduce *hybrid* classes that consist of classes that are difficult to separate . The authentication is done in two stages after forming the hybrid classes. In the first stage, one of the above discriminant analysis methods is used to verify whether the sample belongs to the right hybrid class (which might be homogeneous if it is well separated from other classes). In the second stage, if the sample belongs to a hybrid class, the sample is *recognized* by combining multiple dichotomizers that use discriminative features for separating two classes. In the current scheme, these features are extracted using kernel fisher discriminant analysis [110]. The second stage requires the more discriminative framework owing to the high similarity between classes within the hybrid classes.This hierarchical framework is similar, in spirit, to the one used for classification [111]. Figure 3.5 shows a schematic diagram of the proposed authentication framework. The resulting algorithm provides an increased performance particularly in the case of similar classes.

Complexity The time taken to authenticate a test sample, using the KBDA algorithm, is dom-

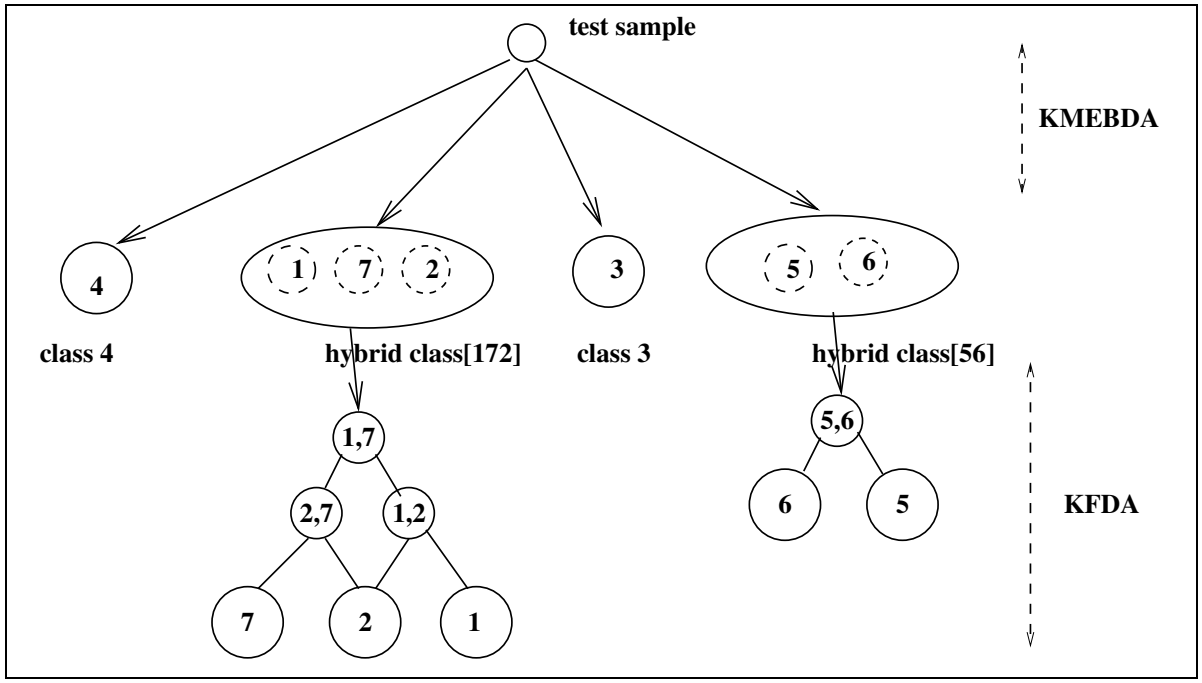


Figure 3.5: The two stage authentication framework proposed to handle highly similar classes in the input set.

inated by the computation of the kernel function with all the samples in the training set. Hence the complexity is $O(NC)$ where N is the number of training samples and C is the cost of kernel function evaluation. This is higher than the constant time that the linear BDA algorithm. Since the kernel function is the only way to access the feature space, this step cannot be avoided. The hierarchical authentication thus has very little additional computational complexity compared to the KBDA algorithm since the complexity is $O(NC + nC')$ where n is the number of samples in the hybrid class and C' is the cost of evaluation of the kernel function at the second level (the kernels at the two stages are different).

3.3 Application to Hand-Geometry based Authentication

3.3.1 Kernel Selection for Authentication

The choice of the kernel function plays an important role in kernel-based algorithms as it encodes the similarity between the images of the samples in a different feature space. Selecting the appropriate kernel for a specific problem is a challenging and widely researched issue [1]. In the case of single-class problems the objective of kernel selection differs from other problems for the following reasons : i) The goal in the case of authentication systems is to choose a kernel that maximizes the similarity between positive samples and the dissimilarity between the positive and negative samples while the (dis)similarities between negative samples bear no significance. Using a different kernel for each class implies a different set of class-specific features are extracted for each class which is desirable for verification algorithms. ii) The performance of authentication systems is characterized two quantities called the false acceptance rate (FAR) and the false rejection rate (FRR). FAR is the expected probability that a negative sample will be accepted as a positive sample by the authentication system. FRR is the probability that a positive sample will be rejected by the

system. In general, it is difficult to build systems that have both a very low FAR and a very low FRR. Thus the desirable values of FAR and FRR vary with the application. This flexibility must be built in to the kernel selection technique.

The non-parametric nature of kernel-based algorithms makes kernel selection an important step while using them. Solutions obtained using non-parametric methods tend to over-fit the training data and thus run the risk of poor generalization. For instance in the case of BDA, by using a Gaussian-rbf kernel of sufficiently small width σ it is possible to obtain an arbitrarily small FAR while the FRR might increase. To address these problems we select the optimal kernel function for each class by performing a search over a set of kernel functions (the parameters of which are varied smoothly during the search) such that the quantity (involving both the FAR and FRR)

$$J(\kappa) = (FAR + \eta FRR)^2$$

over a validation dataset is minimized. The parameter η allows us to change the relative importance of FAR and FRR in determining the feature set. The quantities FAR and FRR are dependent on the data set used and hence it must be ensured that they reflect the true values. This is done by partitioning the data set randomly into train and validation sets several times and acquiring the mean values of FAR and FRR. The objective function used above was found to vary smoothly as the parameters of the kernel function are varied. Figure 3.6 shows the variations of FAR and FRR as the kernel parameters are varied and the optimal FAR and FRR rates as the value of η is varied. It can be seen that by selecting an appropriate kernel and η it is possible to obtain acceptable FAR and FRR.

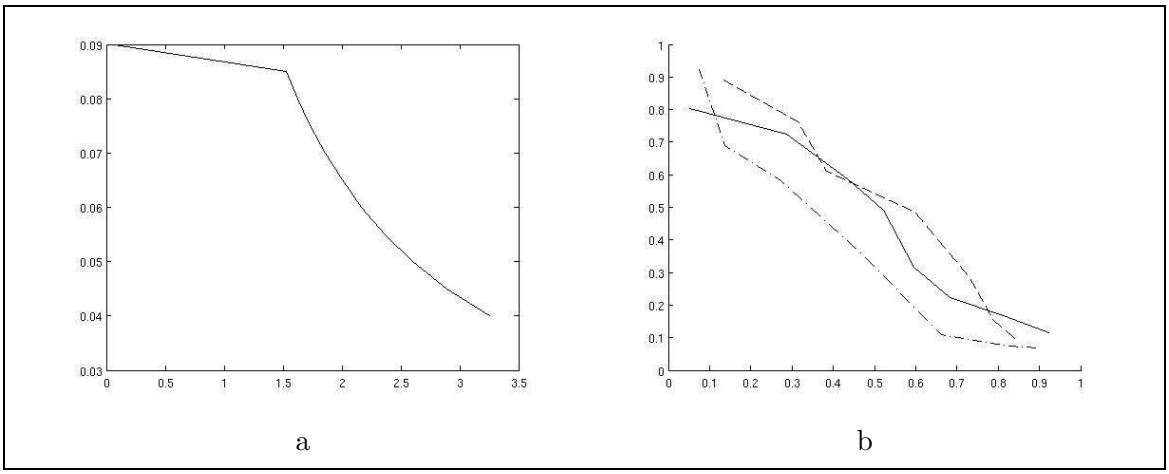


Figure 3.6: (a) The variation of the FAR and FRR rates using the KBDA algorithm as the width of the Gaussian kernel is varied (b) The FAR and FRR rates at the optimal value of $J(\kappa)$ obtained for different values of η using the linear(–), polynomial(-) and Gaussian(-.) kernels. It can bee seen that the polynomial and Gaussian kernels perform better in general.

3.3.2 Experimental Results

The proposed scheme is used to perform biometric authentication using hand-geometry features used in [106]. Figure 3.7 shows the image acquired and the contour extracted. The hand geometry data is collected from 40 subjects over a duration of 2 months. Top view image of the hand placed

on a translucent flat surface(illuminated by a light source beneath) is captured using a camera. Five rigid pegs are fixed on the flat surface to help the user position the hand properly. The image thus acquired is binarized and the contour of the hand is extracted. From the contour a set of 24 features that includes lengths of fingers, widths at various points is extracted.

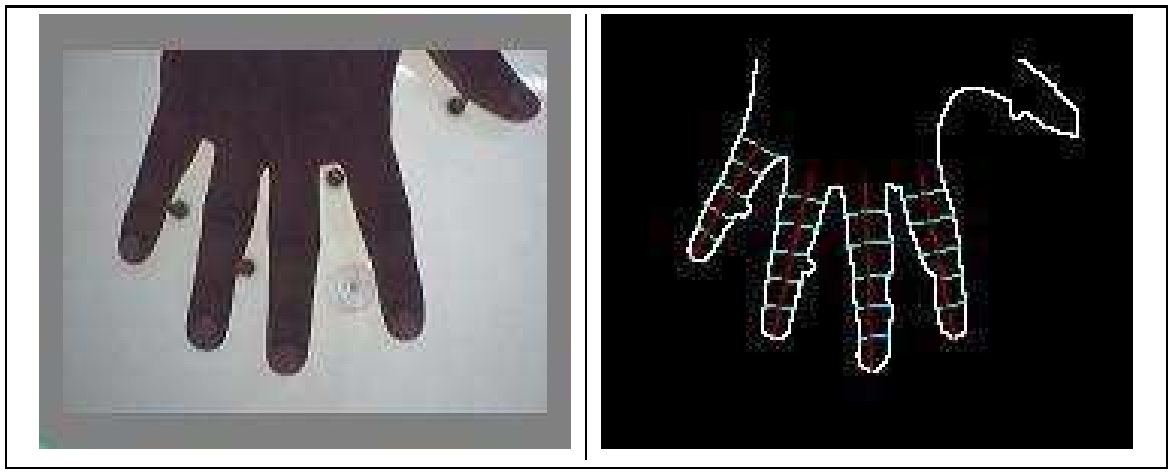


Figure 3.7: The hand image acquired and the raw hand-geometry features extracted from these images

The raw features describing the hand-geometry are not rich enough to discriminate between the subjects accurately. Alternate set of features were extracted using linear and nonlinear variants of BDA. Figure 3.8 depicts the resulting feature distributions of a subset of the classes. The authentication experiments were performed by using the features extracted by projecting on to the top 15 discriminants. The complete dataset is randomly split into two sets : the test and train datasets. The biased discriminants for each class are learned using the train set. Then, for each class, the test samples are all claimed to have the label of that class and are authenticated by projecting on the discriminant space of that class. The mean FRR and FAR values over all the classes is taken as the FRR and FAR values for the dataset.

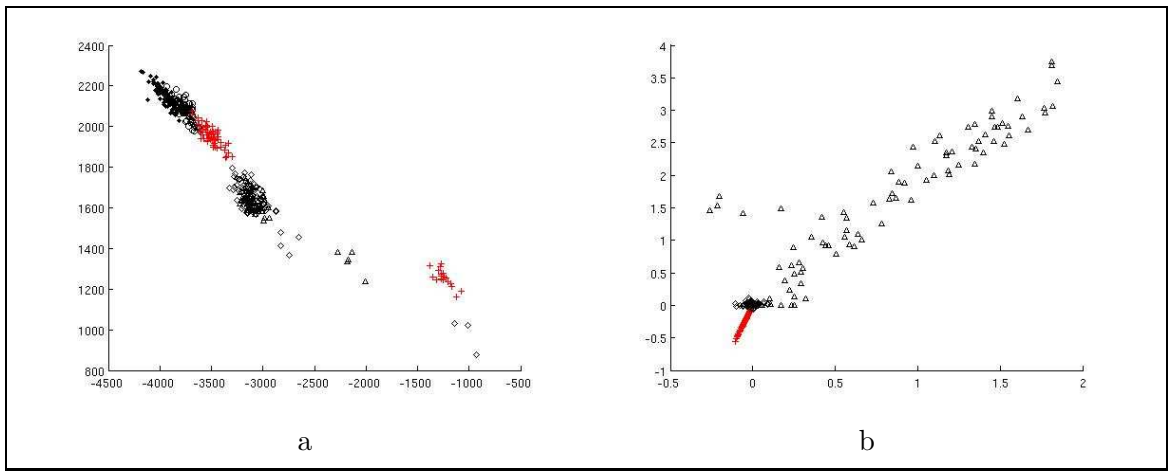


Figure 3.8: Five classes from the hand-geometry data on to the top two directions found by the linear (a) and kernel variants of BDA (b). A Gaussian-rbf kernel with $\sigma = 2$ was used. It can be seen the positive class(shown in red) is better separated in the second case.

	Raw	BDA	$(1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$	RBF $\sigma = 0.5$
FRR	3.3 %	0.8 %	1.4%	1.9 %
FAR	15 %	8.3%	2.9%	1.4 %

Table 3.1: FAR and FRR rates using the raw features, BDA and kernel BDA using two different kernels. The FAR is significantly less for the kernel variants.

	Linear	$(1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$	RBF $\sigma = 0.5$
FRR	0.8 %	1.4%	1.9 %
FAR	8.3%	2.9%	1.4 %
FRR	0.5 %	0.7%	0.8 %
FAR	4.3%	1.9%	1.4 %

Table 3.2: FAR and FRR rates using KBDA and Kernel MEBDA using different kernels. The first two rows show the rates with KBDA and the next two rows show the rates with kernel MEBDA

The average values of FRR and FAR over 100 such trials using the raw, BDA-transformed and kernel BDA-transformed features are shown in Table 3.1. A second degree polynomial kernel and Gaussian-rbf kernel of width 0.5 were used. Note the increase in FRR and and decrease in FAR with kernel BDA. This is expected as the estimated distribution is more closer to the distribution of train samples. Table 3.2 shows a comparison of kernel BDA and multiple exemplar kernel BDA on the same dataset. The kernel selection experiments were done in a similar manner and the kernel was selected to minimize $(FAR + \eta FRR)^2$ and the results using the optimal kernel selected are shown in Table 3.3. Observe that with the kernel selection both the FAR and FRR reach an acceptable value while it may not be so in the general case. Further the parameter η allows us to weigh the importance of these two rates. A value $\eta = 0.5$ was used for the experiments. This means that the FAR is given more importance than the FRR. The results obtained reflect this fact. When the two-stage hierarchical authentication scheme was used the results further improved. The 40 classes resulted in 24 hybrid classes with all the hybrid classes containing less then 3 homogeneous classes. The kernels for the second level (in KFDA) were chose using cross validation. The resulting FAR and FRR are shown in Table 3.3. Observe the increase in both the FRR and FAR over the normal scheme.

A comparison of the results demonstrates that the while the kernel variants of the original biased discriminant analysis algorithm perform better than their linear counterparts they do suffer from problems such as small sample size, overfitting and presence of similar classes. The proposed techniques to overcome these problems using multiple exemplars and hierarchical authentication lead to significant increase in the performance of the authentication system.

	RBF $\sigma = 0.5$	Optimal	Hierarchical
FRR	1.9	0.96%	0.62%
FAR	1.4	0.28%	0.13%

Table 3.3: Comparison of results using KBDA, KBDA with kernel selection and the hierarchical scheme.

3.4 Autoregressive Modeling of Time Series

The second kernel algorithm we develop addressed the issue of nonlinear modeling of time series data. Nonlinear modeling of time series data has been studied extensively in the past [112]. A well-known statistical approach to the study of nonlinear relationships is first to transform the variables such that the relationship is linearized, and thereafter to use linear modeling techniques on the transformed data for developing a compact representation of the signal. This is precisely the technique employed by kernel-function based algorithms. Here we attempt nonlinear modeling of signals by the application of an autoregressive model on the data mapped to a feature space defined by a kernel function. This extends the advantages of the autoregressive modeling techniques to the characterization of nonlinear signals. A restriction imposed by this approach is that, in all but a few cases, the signal can not be reconstructed in the input space. However the model parameters thus estimated can be used as features that characterize the signal which in turn can be used for tasks like recognition and classification.

Many of the traditional nonlinear modeling techniques require iterative optimization of objective functions. These iterative optimization procedures are computationally intensive and often numerically unstable. They could also fail to converge to the desired solution [113]. In contrast, linear models and algorithms are computationally efficient. They could also have closed-form solutions. Moreover linear algorithms [114] are numerically more stable and statistical inferences made using them are more reliable [115]. Due to the efficiency and reliability of linear methods they have been used widely for the tasks of modeling and recognition, even in cases where there is no adequate justification for the linearity assumption. The motivation for the current method is to retain the advantages of the linear modeling schemes to solve the modeling problems, which need nonlinearity. The use of kernel functions is a natural way to achieve this. We use autoregressive model along with kernel functions to perform linear modeling in a feature space defined by the kernel function. The work of Chakrabartty et.al [116] that uses the kernel trick to capture higher order correlation between speech samples, in our knowledge, is the closest to the current one. They use growth transformation with kernel regression for extraction of robust speech features with empirical evidence on robustness of the kernel-based features. We show that the parameters of an autoregressive model, applied to the data in a feature space defined by the kernel function, can be computed from the kernel matrix alone by minimizing the prediction error. The resulting model parameters are employed to perform planar-shape recognition and hand-written character recognition. Results of our experiments demonstrate that the kernel variant of autoregressive coefficients yields model parameters that are better suited for such tasks.

3.4.1 Autoregressive modeling of time series

Autoregressive (AR) model is a popular linear model employed for the modeling time series data in applications like speech processing, image compression, redundancy removal, on-line handwriting recognition etc. An AR model explains the univariate time series data by expressing the signal at instant i as

$$x_i = \sum_{j=1}^{p} \alpha_{p-j+1} x_{i-j} + e_i$$

where the process mean is assumed to be zero and e_i is white noise. The value of p is known as the order of the AR model. This is a linear regression of the sample at instant i against previous p samples. The parameters of the model can be estimated by minimizing the squared prediction

error

$$\xi = \sum_{i=p+1}^{l} (x_i - \hat{x}_i)^2 = \sum_{i=p+1}^{l} (x_i - \sum_{j=1}^{p} \alpha_{p-j+1} x_{i-j})^2$$

where l is the total number of samples.The solution can be obtained by setting $\frac{\partial \xi}{\partial \alpha} = 0$ and solving the resulting set of linear equations. Autocorrelation, Levinson-Durbin recursion are some of the popular methods for numerical computation. The modeling scheme can be extended to vector valued sequence $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l$ in a straightforward manner with the definitions

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{p} \alpha_{p-j+1} \mathbf{x}_{i-j}$$

$$\xi = \sum_{j=p+1}^{l} \|\mathbf{x}_j - \hat{\mathbf{x}}_j\|^2$$

The estimation of the model parameters can be done efficiently and the estimation procedures are. in general, numerically stable. This aspect of linear predictive coding led to its use in a number of applications involving time series. It has been employed for a number of tasks such as speech recognition [107], recognition of planar shapes from their silhouette representation [77] and recognition of handwritten characters [75]. However the linear model cannot capture variations in complex signals. Kernelizing the method introduces nonlinearity while preserving the statistical and computational advantages described above. The next section describes the kernelization of autoregressive modeling.

3.5 Autoregressive model in the feature space

Once again the data matrix notation is used to simplify the process of kernelization. We use the notation

$$\mathbf{X} = \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_l) \end{bmatrix}$$

for the data matrix and $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \end{bmatrix}^t$ for the vector of prediction coefficients. Writing the prediction equation in terms of the data matrix needs a running window that selects p previous consecutive samples. This is done using a selector matrix which *selects* p consecutive samples. The matrix that selects the samples $\phi(\mathbf{x}_{i-1}), \cdots, \phi(\mathbf{x}_{i-p})$ can be expressed as :

$$\mathbf{J}_i = \begin{bmatrix} \mathbf{0}_{(i-p-1)\times p} \\ \mathbf{I}_p \\ \mathbf{0}_{(l-i+1)\times p} \end{bmatrix} \tag{3.8}$$

where $\mathbf{0}_{m\times n}$ is an $m \times n$ matrix with all entries equal to 0 and $\mathbf{I}_m$ is an $m \times m$ identity matrix. Using this notation the prediction equation in the feature space which is

$$\widehat{\phi(\mathbf{x}_i)} = \sum_{j=1}^{p} \alpha_{p-j+1} \phi(\mathbf{x}_{i-j})$$

can be rewritten compactly as

$$\widehat{\phi(\mathbf{x}_i)} = \mathbf{X}\mathbf{J}_i\boldsymbol{\alpha}$$

Note that the matrix $\mathbf{X}$ is of dimension $N \times l$ where N is the dimension of the feature space which is typically very high. It is infeasible to compute the map $\phi(.)$ and hence this matrix is inaccessible.

However as in the case of dual formulations of linear algorithms seen in the last chapter, $\boldsymbol{\alpha}$ can be estimated without the knowledge of $\mathbf{X}$ as shown below. The prediction error between the real and predicted samples in the feature space can be written as

$$\xi(\boldsymbol{\alpha}) = \sum_{i=p+1}^{l} \|\phi(\mathbf{x}_i) - \widehat{\phi(\mathbf{x}_i)}\|^2$$

Noting that $\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{x}^t\mathbf{x}$, substituting the expression for the predicted sample and using the bilinearity of inner product we have

$$\xi(\boldsymbol{\alpha}) = \sum_{i=p+1}^{l} \left(\phi^t(\mathbf{x}_i)\phi(\mathbf{x}_i) - 2\phi^t(\mathbf{x}_i)\mathbf{X}\mathbf{J}_i\boldsymbol{\alpha} + \boldsymbol{\alpha}\, {}^t\mathbf{J}_i{}^t\mathbf{X}^t\mathbf{X}\mathbf{J}_i\boldsymbol{\alpha} \right)$$

Setting $\frac{\partial \xi}{\partial \boldsymbol{\alpha}} = 0$ and solving for $\boldsymbol{\alpha}$ we have

$$\boldsymbol{\alpha} = \left(\sum_{i=p+1}^{l} \mathbf{J}_i{}^t\mathbf{X}^t\mathbf{X}\mathbf{J}_i \right)^{-1} \left(\sum_{i=p+1}^{l} \mathbf{J}_i{}^t\mathbf{X}^t\phi(\mathbf{x}_i) \right)$$

Note that $\mathbf{X}^t\mathbf{X}$ is precisely the Kernel matrix $\mathbf{K}$ defined by $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ which is of dimension $l \times l$ independent of N. The vector $\mathbf{X}^t\phi(\mathbf{x}_i)$ is the ith column $\mathbf{K}^i$of the Kernel matrix. Substituting these in to the equation

$$\boldsymbol{\alpha} = \left(\sum_{i=p+1}^{l} \mathbf{J}_i{}^t\mathbf{K}\mathbf{J}_i \right)^{-1} \left(\sum_{i=p+1}^{l} \mathbf{J}_i{}^t\mathbf{K}^i \right)$$

It can be easily seen that the final computation requires only the Kernel Matrix $\mathbf{K}$ which can be computed using the kernel function κ. Thus the model-parameter estimation can be done efficiently irrespective of the dimensionality of the feature space. The zero mean assumption can be handled by centering the kernel matrix in the feature space : $\mathbf{K_{ij}} = \langle \phi(\mathbf{x_i}) - \frac{1}{l}\sum_{k=1}^{l}\phi(\mathbf{x_k}), \phi(\mathbf{x_j}) - \frac{1}{l}\sum_{k=1}^{l}\phi(\mathbf{x_k}) \rangle$. This can be done by modifying the original kernel matrix as

$$\mathbf{K} = \mathbf{K} - \frac{1}{l}\mathbf{1}\mathbf{K} - \frac{1}{l}\mathbf{K}\mathbf{1} + \frac{1}{l^2}\mathbf{1}\mathbf{K}\mathbf{1}$$

where $\mathbf{1}$ is an $l \times l$ matrix with all entries equal to one. Note that the residual can also be measured by using the kernel matrix alone.

Note that the matrix $\sum_{i=p+1}^{l} \mathbf{J}_i{}^t\mathbf{K}\mathbf{J}_i$ is a sum of kernel matrices and is positive semi-definite. It is not guaranteed to be invertible always. In such cases numerical techniques like adding an additional term $\lambda\mathbf{I}_p$ can be used. This is equivalent to adding a penalty term controlling the norm of $\boldsymbol{\alpha}$. Algorithm 4 summarizes the entire procedure. Since the order of the model is typically much smaller compared to the number of samples, the algorithm runs in time $O(l^2c)$ where c is the cost of evaluation the kernel function on a pair of data points. An interesting observation is that the complete $l \times l$ kernel matrix is accessed only during the centering operation. For the remaining computation only a $p \times p$ kernel (sub-) matrix moving along the diagonal of the full kernel matrix is accessed. This can be exploited in improving the computational complexity of the algorithm.

We have shown that autoregressive modeling of data in a transformed space can be done efficiently using the kernel trick. A drawback of using the kernel trick is the lack of explicit control of the

Input : $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l\}, p, \kappa(.,.)$
Output : $\boldsymbol{\alpha}$, ξ
Compute the Kernel Matrix , $\mathbf{K}_{ij} \leftarrow \kappa(\mathbf{x}_i, \mathbf{x}_j)$
$\mathbf{K} \leftarrow \mathbf{K} - \frac{1}{l}\mathbf{1K} - \frac{1}{l}\mathbf{K1} + \frac{1}{l^2}\mathbf{1K1}$
$\mathbf{B} \leftarrow \mathbf{0}_{p\times p}$
$\mathbf{y} \leftarrow \mathbf{0}_{p\times 1}$
for $i = p+1 \cdots l$ **do**
  $\mathbf{B} \leftarrow \mathbf{B} + \mathbf{K}(i-p:i-1, i-p:i-1)$
  $\mathbf{y} \leftarrow \mathbf{y} + \mathbf{K}(i-p:i-1, i)$
end for
$\boldsymbol{\alpha} \leftarrow (\mathbf{B} + \lambda \mathbf{I}_p)^{-1}\mathbf{y}$
$\xi \leftarrow 0$
for $i = p+1 \cdots l$ **do**
  $\xi \leftarrow \xi + \mathbf{K}_{ii} - 2 * \mathbf{K}(i-p:i-1, i)^t \boldsymbol{\alpha}$
  $+\boldsymbol{\alpha}\ ^t\mathbf{K}(i-p:i-1, i-p:i-1)\boldsymbol{\alpha}$
end for
return $\boldsymbol{\alpha}$, ξ

Algorithm 4: Kernel AR model

transformed space and inaccessibility to the samples in it. If the type of the nonlinearity is known apriori, an appropriate kernel can be employed for obtaining the accurate model. When there is no information about the type of nonlinearity, selection of kernel becomes more of an empirical procedure. While selection of kernel functions appropriate for a given task is still an open problem, empirical methods are shown to be effective for the selection kernels. The model parameters estimated using the proposed method can be used as a representation of the sequence for use with other algorithms for classification and recognition.

The algorithm presented uses the efficiency of the linear model with the nonlinearity introduced by the kernel trick to give an efficient nonlinear modeling scheme. Interpretation of AR modeling in the feature space is rather difficult as the nonlinear map involved $\phi(.)$ can distort the spatial ordering and continuity in which case application of an AR model may not have justification. This problem is more pronounced in the multidimensional case where the geometry of signal in the feature space can be quite complex. Despite this apparent lack of physical justification the efficiency and simplicity of the method make it very promising for nonlinear modeling.

3.6 Applications of Kernelized AR Model

3.6.1 Experiments on Synthetic Data

The kernel version of the AR modeling scheme proposed in the previous section, models a signal in the feature space. However, the predicted signal can not always be mapped to the input space. This is because the points $\phi(\mathbf{x}_i)$ are difficult to compute. Even if $\phi(.)$ is computationally feasible, an inverse mapping may not be defined for all the points. Thus it is difficult to test the goodness of the fit in the input space. However, when the input space is one dimensional, and kernel function is polynomial, i.e., $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^t\mathbf{y})^d$, it is possible to compute and invert the nonlinear map $\phi(x) = x^d$. For the experimentation, we use 1-d signals synthesized using known and unknown generative models. We then model the signal with the proposed algorithm using polynomial kernels

of various degrees. Some of the results are described and discussed in the rest of this section.

A signal is synthesized using the following generative model $x_n^7 = x_{n-1}^7 - 3x_{n-2}^7 + 3x_{n-3}^7$. It may be noted that this signal will have a linear structure in the feature space defined by the polynomial kernel of degree seven. We model the signal with polynomial kernels of varying degree. Performance of the modeling is analyzed with the help of prediction error, which is defined as the sum of squared errors between the predicted(modeled) and actual samples. This computation is done in the input space, so that the performance of all the kernels can be compared and analyzed in the same framework. Figure 3.9 shows the relative values of the average prediction error in the input space plotted against the degree of the kernel. Note that the computations are done only for the integer values of the kernel degree. Prediction error is computed as the mean value of the prediction error over 1000 trials and for different of the order of prediction.

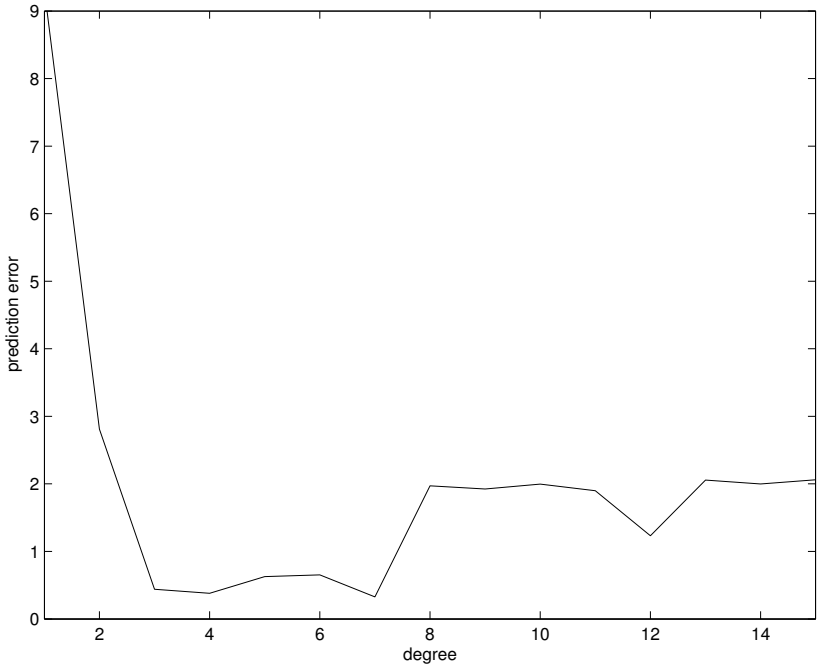


Figure 3.9: Prediction error in the input space on a signal generated by $x_n^7 = x_{n-1}^7 - 3x_{n-2}^7 + 3x_{n-3}^7$. Note the minima at $d = 7$.

The minimum error was obtained for kernel with degree seven. It may be noticed that the higher degree kernels perform much better the linear kernels. However, the error associated with various higher order kernels need not be monotonically decreasing. This observation highlights the fact that selection of appropriate kernel is possible for the accurate modeling. Also the experiment proves that the method does not overfit the data as in the case of many nonlinear modeling schemes. Similar results were obtained for many other generating models. The model parameters obtained in each case were the same as the parameters of the generating model . This experiment demonstrates that the algorithm performs as expected, and can model the signal with very high accuracy, provided the kernel can approximate the nonlinearity in the signal.

The second experiment is performed to study the sensitivity of the algorithm to the presence of the noise in the input data. When noise is present in the data, the modeling scheme can perform inferior in the feature space. The sensitivity of the algorithm may depend on the map $\phi(.)$ and hence the kernel function. We conducted experiments with various signal to noise ratios of a

$\frac{SNR\rightarrow}{d\downarrow}$	**30dB**	**24dB**	**20dB**	**18dB**	**16dB**
1	0.68	1.53	2.79	4.22	6.06
2	0.78	1.64	2.90	4.30	6.11
3	0.98	1.86	3.06	4.45	6.20
4	1.16	2.04	3.15	4.50	6.12

Table 3.4: Table showing the percentage change in prediction error as the noise percentage is varied for kernels of various degrees.

synthetic signal and the percentage deterioration in the input space is studied. Table 3.6.1 depicts the percentage increase in the prediction error as the noise increases in the input space. It can be seen that the sensitivity of the (higher degree) kernel AR modeling scheme is very similar to the conventional AR modeling (i.e., linear kernels) procedure. As the noise increases, all kernels monotonically deteriorate in performance. As the noise increases, very often the linear methods break faster compared to the proposed kernel scheme, with higher order polynomial kernels.

The third experiment was done to test the effectiveness of the algorithm on signals whose generative model is unknown. Once again it was observed that the kernels of higher degree model the signal better. The proposed algorithm was also compared with nonlinear autoregressive modeling methods like the Local linear estimate. Preliminary results show that the method proposed here performs comparatively and is more stable since it does not require any iterative optimization.

3.6.2 Application to Shape and Handwriting Recognition

Autoregressive models have been employed for shape recognition [77] and handwritten character recognition [75]. The contours of planar shapes can be described using a time series which can be obtained by sampling the points along the perimeter as described by Kartikeyan and Sarkar [77]. Using a similar representation, we applied the kernel autoregressive modeling technique for recognition of shapes. The shapes were taken from the SIID shape database [117]. Sample shapes from the databased are shown in figure 3.10. The contours of these shapes are extracted and are represented using a time series. The time series' corresponding to a training set of shapes are modeled using autoregressive models in a kernel defined feature space. Model parameters for a test sample are similarly extracted and the euclidean distance between the model parameters is used to assign labels to test samples. Table 3.5 gives the results of the recognition for various kernels. It can be seen that the nonlinear kernels result in higher performance.

	LPC	$(1+\langle \mathbf{x},\mathbf{y})\rangle^2$	RBF $\sigma = 0.5$
Shape	74.1%	81.3 %	83.4%
Handwriting	76.8%	82.5%	86.1%

Table 3.5: Comparison of performance (recognition accuracy in percentage) of shape recognition from silhouettes and handwritten character recognition using linear prediction coefficients and kernel linear prediction coefficients with two different kernels. The first column shows results using LPC while the second and third column show results using KLPC with the given kernels.

A similar experiment is performed to recognize online handwritten characters from their time

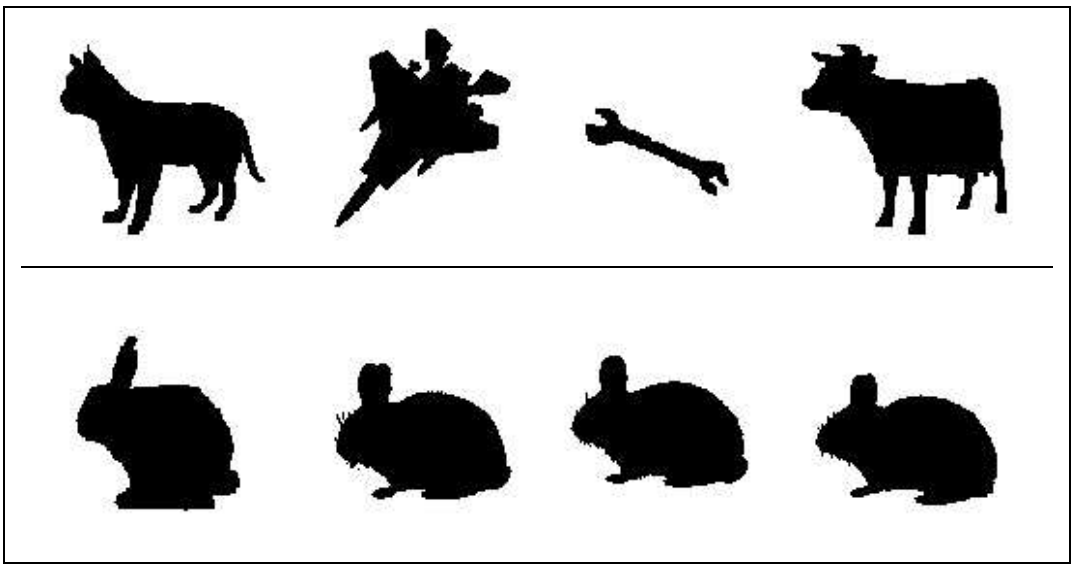


Figure 3.10: Sample images from the shape databased used for the shape recognition experiment. The top row shows the different classes. The second row shows how the shapes vary within a class.

series representations. Online handwriting defines a natural time series representation strokes in the characters. For the sake of simplicity recognition of characters made of a single stroke is considered in the current work. The character pairs used for our experiments are those described in [118] which consist of similar partial strokes. Sample images of the characters are shown in figure 3.11. The characters were broken in to substrokes and the kernel LPC algorithm was used to extract the features from each substroke. A euclidean distance between based classification rule was used to recognize unknown samples. The recognition accuracies for pairs of characters for different kernels are show in table 3.5. The improvement in results in both the experiments demonstrates that the introduction of nonlinearity via kernel function gives rise to more descriptive features that capable of distinguishing between different signals better.

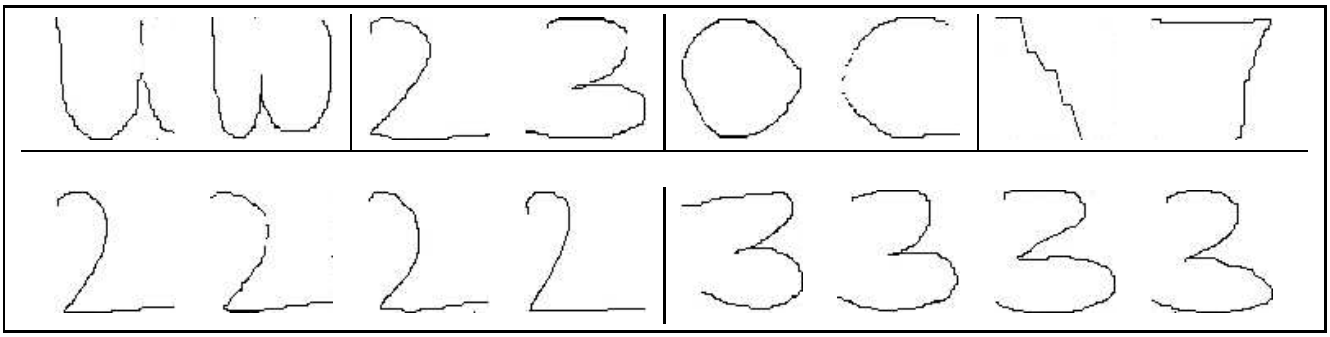


Figure 3.11: Sample images of the character pairs used for the handwritten character recognition experiment. The first rows shows some of the different pairs used. The second row shows the different samples from one pair of characters (showing the intra-class and inter-calss variation).

3.7 Summary and Conclusions

In this chapter we demonstrated how kernelization aids in enhancing the performance at two important tasks, namely feature selection and modeling. The performance of a verification systems is greatly improved by using a single class framework and nonlinear features using the kernel trick. Nonlinear biased discriminants based on the kernel trick are used for authentication. The feature selection problem ,thus, is posed as a kernel selection problem. A tunable objective function using FAR and FRR rates is used to perform the selection. The full set of available samples is used to describe the distribution of the positive class. A hierarchical authentication framework is introduced to reduce the errors caused by highly similar classes. The increase in accuracy is primarily due to the use of nonlinear discriminants which is enabled by the kernelization. Efficient search techniques for kernel selection and for learning a class specific kernel matrix are promising directions for future research.

For the task of modeling, we presented a novel method for modeling nonlinear relations in time series by application of a linear model in a kernel-defined feature space. Preliminary experiments on synthetic one dimensional signals, recognition of shapes from their silhouettes and character recognition demonstrate that nonlinear prediction coefficients are better suited for these tasks. The method is promising feature extraction method for multidimensional sequence recognition tasks like video-event recognition and handwriting recognition to achieve better performance and robustness.

Chapter 4

Face Video Alteration Using Tensorial Factorization

4.1 Introduction

Automated analysis of images or videos of human faces is an important area of research in computer vision and pattern recognition [70, 119, 120, 121]. Detection and recognition of faces in still images and videos plays pivotal role in a number of practical applications such as surveillance, broadcast news video processing, security and access control, image and video retreival, content management etc. [122, 123, 124]. Methods for detection and recognition of faces received wide attention owing to the large number of immediate applications [119, 125]. However, there exist other challenging and relevant problems in face video analysis with powerful practical applications that received comparatively less attention. Three such problems are addressed in this chapter using factorization of appearance models:

- *Facial Expression Transfer* is pictorially explained in Figure 4.1. Given a video of a person's face with an expression, the task is to synthesize the video of a second person (whose *neutral* face is given). It has several applications in interactive systems, gaming and virtual worlds. Examples include player look-alike characters in computer games which show different expressions as the game progresses and personalized *emoticons* that use a person's face rather than generic smileys which enhance user experience.

- *Face morphing* is pictorially explained in Figure 4.2. Given two images of faces of different persons, the task is to synthesize a sequence of face-images that depicts a visually smooth transformation from once face to the other. It is popular in the movie and entertainment industry for generation of visual effects. Recent techniques [126, 127] that address several variants of the problem such as morphing between different views, different views of different persons etc. indicate the growing interest in the problem.

- *Expression recognition* is the task of learning the appearance of an expression from a set of facial expression videos and then classifying the expression in a previously unseen video. It is a desirable feature of the next generation human-computer interfaces. Computers that can recognize facial expressions and respond to the emotions of humans accordingly enable better human-machine communication.

In addition to the applications mentioned above, analysis of facial expressions helps in preprocessing face videos, which helps in improving the accuracy of expression independent face recogni-

Figure 4.1: The facial expression transfer problem : Given the videos of a person with different expressions the goal is to synthesize the videos of a second person with the same expressions (marked by ?'s in (a)). (b) shows the results using our method.

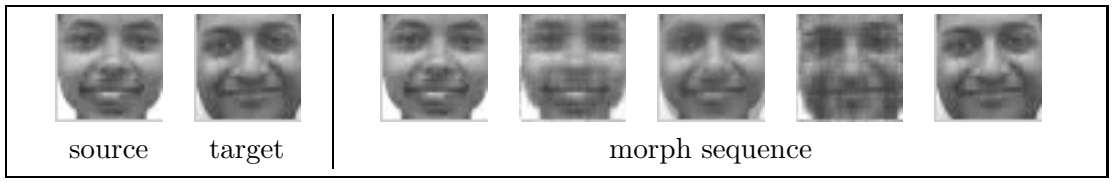


Figure 4.2: The face morphing problem : Given the source and target face images the goal is to generate the intermediate images of an image sequence that depicts a visually smooth transformation between the two faces. Our result is shown on the right.

tion [128]. The appearance of facial expressions is a result of a complex physical phenomenon which involves a large number of factors such as the face geometry, complexion and muscular motion. A comprehensive analysis of facial expressions requires knowlege of all these factors and the way they interact i.e. the underlying generative process needs to be modelled and estimated. Similarly face morphing requires the interpolation of different types of features of faces such as shape, color and texture. Past approaches to these problems are indicative of this paradigm. Previous methods for expression transfer [79, 80, 81, 82, 27], morphing [80, 87] and expression recognition [27, 83, 86] relied heavily on the information such as geometry, texture, shape, models of muscular motion, feature correspondences and complex modeling schemes. While this information characterizes the underlying physical processes well, it is not available in all cases and is often difficult to compute from images or videos alone. Many of the previous methods, which use 3D models, need precise correspondending feature points. However, perceptual systems recognize a person's face independent of the expression and vice versa from a single stimulus i.e., the appearance. This suggests that appearance alone provides significant clues for recognizing a face or an expression. The motivation for the current work is the fact that methods that can use appearance information alone can solve these problems. In addition to being applicable to scenarios where rich information is not available (or is difficult to obtain), such methods are complementary to other methods which use richer information.

In this chapter, we propose methods that differ from previous methods in their approach. Appearance information alone is used to solve the above problems. The key to our methods is the tensor representation of a face video which enables the proposed solution. The face video is represented as a tensor which is then decomposed into positive low-rank factors. Such a positive factorization results in the local-parts decomposition of the face, which may also be considered as a generative model for the image patches. By changing the factors corresponding to these regions, the appearance of the face in the video can be affected. Given a facial expression video, decomposing

it and transferring the appearance of the regions of interest to another face video is a simple way to generate the effect of an expression transfer. Similarly, smoothly changing the appearance of individual regions of a face to match that of another face results in a morph sequence. In addition, the appearance of these regions can be used for recognition of facial expressions. The key requirement for these methods is to establish the correspondence between the factors and the regions of the face. Correspondence between factors across videos also needs to be established so that factors can be transferred across videos. The following sections describe the methods proposed and related techniques in detail. Section 4.2 reviews the earlier techniques for each of the face video applications considered in this chapter. In Section 4.3, we propose the tensor representation of face videos suitable for a variety of tasks and the NTF algorithm [9, 10] for face video factorization. Section 4.4 describes methods to analyze the factors of the face video and determine the factors that affect the appearance of specific regions in the video. Section 4.5 describes the algorithms to perform expression transfer, morphing and expression recognition. Section 4.6 presents the experimental results and discussionis and Section 4.7 provides concluding remarks. Parts of the current work appeared separately in [129, 130].

4.2 Related Work

The face video applications considered in this chapter have been studied extensively in computer vision. The state of the art for each problem is reviewed in Section 4.2.1. Factorization methods and tensor representation of image data have been used to solve a variety of problems in computer vision. Popular factorization and tensor methods in computer vision are reviewed in Section 4.2.2.

4.2.1 Facial Expression Analysis and Morphing

Facial expression transfer and facial expression recongition received relatively less attention in research [131] compared to the more popular problems of face recognition and detection. Morphing has been extensively studied in Computer Vision as well as Graphics owing to its popularity in the movie and entertainment industry. A brief overview of the prominent approaches to each problem is given below.

The appearance of face in a facial expression video is influenced by a number of factors such as geometry, texture and muscular motion. Another popular interpretation is to view the appearance as a result of interaction between two orthogonal factors, such as style and content [23] where the expression is the style and the underlying face is the content or vice-versa. Past approaches for expression transfer and synthesis can be categorized into two classes : i) Methods that simulate the underlying physics by making use of the geometry, texture and muscular motion and transfer or synthesize expressions [79, 80, 81]. ii) Methods that explicitly untangle the expression and face factors and then use these factors to perform expression transfer [82, 27, 8]. Traditional approaches to facial expression transfer based on warping [79] or morphing [80, 81] fall in to the former class. Noh and Neumann [79] transfer vertex motion vectors from a known source face model to the target model to synthesize expressions. Pighin *et. al* [81] recover the geometry of the face from multiple views. They use 3D shape morphing between the face models and blend the textures to generate realistic facial expressions. However, warping based methods ignore texture variations while the morph based methods cannot be used to transfer expressions across subjects. Thus, they can be used for expression *synthesis* but not *transfer*. Seitz and Dyer [80] use both morphing and warping to synthesize changes both in the image structure and viewpoint. Liu *et.al* [132] use *expression ratio images* together with warping to capture variations in illumination and synthesize more expressive expressions. Recent methods [82, 27, 8] for expression transfer, which follow the

paradigm of separation of underlying expression and face factors, fall in to the second class of algorithms. In this case, not only the factors need to be untangled but also the way they interact to generate the appearance needs to be learned. Existing factor models have been found to be ineffective in capturing such interactions [23]. Du and Lin [82] attempt to learn a linear mapping between parameters representing expression and appearance. Wang and Ahuja [27] use a multi-factor analysis method and decompose a collection of face expression images into two separate expression and person subspaces and use them to map an expression on to a new person's face. Vlasic *et.al* [8] use the same approach with an extended set of factors that includes geometry, identity, expression and viseme.

Automatic recognition of facial expressions is challenging since the appearance of different faces with the same facial expression varies widely. Recognizing expression with high accuracy requires effective encoding of the expressions. Sebe *et. al* [83] use the FACS [133] in conjunction with a complex classifier to recognize expressions. However, traditional encoding schemes such as FACS do not incorporate temporal information since they are designed for static expression images and are primarily intended for human use. Inputs that incorporate temporal information, such as muscular motion, which characterize the underlying generative process better and aid in better recognition. Essa and Pentland [84] use Dynamic Models and Motion Energy templates to recognize expressions. Chibelushi and Low [85] use local facial dynamics to recognize expression in occluded scenarios. Cohen *et. al* [86] use a multilevel HMM to recognize expressions in a video using temporal cues.

Morphing or image metamorphosis is the animated transformation from one digital image to the other. Generation of a morph sequence between two images requires smooth interpolation of various characteristics such as shape, texture and illumination [134]. Morphing between two face images has an additional constraint that the intermediate images must retain face-like characteristics. This is more difficult than general image metamorphosis where intermediate images that do not map to semantic concepts (such as face) are tolerable. Past approaches to face morphing required the specification of feature points and the correspondence between them across the source and target images [80, 87, 88]. The shape and texture characteristics are then smoothly interpolated to generate the intermediate images. A Bayesian framework for generating a morph field by distorting the brightness and geometry of the source image is proposed in [87]. The method proposed performs morphing between faces of different subjects from different viewpoints without the knowledge of 3D information. Seitz and Dyer [80] use warping to generate morph sequences between two faces in different views. Benson [88] uses warping and blending along with user specified landmark points to obtain high quality morph sequence between faces. User specified control points play a crucial role in such methods. The success of industrial systems such as the Morf [135], a computer-graphics program allowing the fluid, onscreen transformation of one object to another demonstrate the potential of Morphing techniques.

A common feature of all the methods reviwed above, for expression analysis and morphing, is that they use information such as geometry of the face and semantic ratings along with the appearance of the images. All of these methods require accurate tracking of feature points across images. Many of them require more complex information such as 3D mesh models of the faces and muscular motion. Such information may not be available always or may be difficult to obtain. Perceptual systems, on the otherhand, derive the information required for recognition of expressions independent of the person's identity and *vice versa* from a single stimulus, namely the appearance. While information such as geometry, motion models etc. characterize the underlying generative process, the appearance is the observable output of the process captured in images/videos. Thus, they complete the description of the generative process and are complementary in nature. Ideally, all of this information needs to be taken into account to solve the problems listed above. However, the ability of perceptual systems to work with appearance information alone suggests that it is

feasible to design methods that can perform these tasks using the appearance information alone. Such alternative approaches that work with appearance information alone are useful in two ways : i) They are applicable to cases where rich information is not available and ii) They complement approaches that use more complex information. The approach described in this chapter falls into this class. Our results demonstrate the efficacy of this approach and the power of this new paradigm.

4.2.2 Factorization and Tensor Methods in Computer Vision

Many important clues to the analysis of facial expressions are provided by differences in the appearance of the face with an expression and a *neutral* face. These differences are more pronounced in some regions of the face and less in the others. For instance, for the expression *smile*, the appearance changes most near the mouth region while in the other regions it remains nearly the same. Similarly, in morph sequence generation it is necessary to smoothly interpolate the characteristics of corresponding regions of the faces. To be able to do this, a generative model for the image patches needs to be estimated. The tensor representation of face videos enables one such model. Tensor representation of image collections and the use of multilinear algebra on such tensors are gaining wide attention in computer vision [7, 64, 26]. The tensor representation of image collections and videos is more natural as it preserves the spatial coherency by retaining the 2-dimensional structure of individual images. Popular methods for dimensionality reduction and recognition, such as Eigenfaces [61], are based on matrix representations of image collections (*data matrices*). When the images belong to a single object such as face, non-negative factorization of such matrices results in factors that closely correspond to local parts of the object [63]. This observation motivated methods for similar factorization of a tensor into positive factors [9, 26]. The non-negative tensor factorization(*NTF*) proposed in [9] uses a positive-preserving gradient descent procedure to factorize the tensors. The factors obtained result in local parts decomposition as in the case of matrix factorization, with an additional advantage that the invariant parts do not get distributed over multiple factors. The local parts decomposition obtained by NTF of videos has been used for sparse encoding of images [10]. The factors obtained can also be considered as appearance models for various regions in the image. This observation is the basis for the approach presented in this chapter.

Factorization methods are popular in computer vision [23, 63, 5]. Popular applications of factorization methods include recovery of structure from motion(*SfM*) [5], separation of style and content [23], and local parts decomposition [63]. The use of factorization techniques differs in aspects like the way factors are extracted, interpreted and modeled. Tomasi and Kanade [5] recover the scene structure and the camera motion from a sequence of images taken with a moving camera. They factorize a measurement matrix into shape and motion factors using Singular Value Decomposition(*SVD*). Tenenbaum and Freeman [23] use a bilinear model to model the interaction between style and content parameters. They use *SVD* and Expectation Maximization(*EM*) algorithms to carry out the tasks of classification, extrapolation and translation which require inferring the model, and one or both of the factors. The technique is used to separate typographic content from its font, perform content classification, render the content in novel fonts etc. which is analogous to expression transfer in typography. Both these algorithms fall into the class of algorithms where the factors are fixed in number, interpretable and the factor model corresponds to the underlying generative model. Another class of algorithms attempt to identify sets of latent variables that aid in elimination of redundancies in the data. The small set of variables along with the factor model explains the structure in the data and results in compact representation. Ghahramani and Hinton [136] model the covariance structure of data assuming a simple factor model that is linear in the factors. A mixture of such models has been used for simultaneous dimensionality reduction and

clustering. Techniques like Principal Component Analysis and Positive Tensor Factorization [26] extract factors that enable dimensionality reduction or compression. Tensorial factorization methods are gaining wide attention in computer vision [7, 64] as tensor representation is more suitable for image collections or videos. The observation that positive factorization of matrices results in local parts decomposition of objects [63] resulted in similar factorization methods for tensors [9, 26]. Such factorization results in sparse encoding [10] of the data. The sparse and separable factors obtained using positive factorization are used for a variety of tasks like image encoding, model selection and face recognition [9]. The factors obtained by this latter class of techniques usually lack meaningful interpretations that correspond to the properties of the generative process, but they aid in reducing the bulkiness of data.

Orthogonal to the requirements of compact representations and interpretable factors, is the need for identification of factors (interpretable or otherwise) that cause the data to exhibit certain properties of interest. The performance of computer vision algorithms at tasks such as object classification, object detection is significantly enhanced by using a set of features that are relevant to the task, rather than a full set of features representing the data. For instance, detecting a face needs features that characterize the holistic appearance of face, while discriminating between two face classes would require filters capturing finer variations in the appearance of the faces. Similarly, identification of factors that give rise to properties of interest in the observed data has interesting applications such as data synthesis and recognition. For instance, given two sets of data with different desirable properties, identifying the relevant constituent factors and the factor model enables the synthesis of a third collection with both of these properties. In this chapter, we also propose methods to identify such factors from a pool of factors obtained by decomposing a video represented as a tensor. The factors, so selected, enable efficient appearance based solutions for two challenging tasks: facial expression transfer and facial expression recognition.

The approach described in this chapter consists of the following steps :

- The video is represented as a tensor. In case of expression videos, temporal segmentation, temporal alignment and a minimal spatial alignment are assumed.
- The tensor is factorized using the the NTF algorithm. The various factors obtained represent appearance models for the image regions.
- Factors relevant for expression transfer and expression recognition are identifed. (For morphing this is not necessary since all the factors are relevant).
- For expression transfer, the relevant factors, identified as above, are transferred to the target video (represented as a tensor and decomposed in to factors). Expression Recognition is performed using the relevant factors to extract disriminatory information. Morphing is done by gradually transfering the factors from one video to the other.

The details of each of the above steps are described in the following sections.

4.3 Representation and Factorization of Face Videos

Representation of image data as matrices is popular in computer vision [61, 63]. In this representation a 2D image is reshaped in to a one dimensional vector that forms one column or row of the data matrix. This representation is motivated by the representation of information as a feature vector, which is popular in pattern recognition [137]. Factorization of such matrices resulted in a number of algorithms, such as prinicipal component analysis, with powerful applications in dimensionality

reduction and face recognition [61] . However, this representation is not always optimal for analysis of image data as it disrupts the inherent 2D structure of images which precludes algorithms from exploiting the spatial coherency present therein. The tensor representation of image collections which retains the 2D structure is more suitable for image data. An N-valent tensor is an N-dimensional array of numbers. A video can naturally be viewed as a 3-valent tensor with two spatial dimensions and one temporal dimension. This tensor can be decomposed in to its constituent low rank factors using techniques such as HOSVD [25], Positive Tensor Factorization [26] or Non-negative Tensor Factorization [9]. When the constituent factors are positive, the decomposition results in factors that affect the appearance in various small regions in each frame of the video. Thus it is analogous to generative models for the image pathces.

Non-negative Tensor Factorization: Let G be an N-valent tensor of dimensions $d_1 \times d_2 \cdots \times d_N$. The rank of G can be defined in a manner similar to the 2D (matrix) case. The tensor G is of rank k if and only if k is the smallest number such that G can be expressed as a sum of k rank-1 tensors, i.e a sum of N-fold outer products:

$$G = \sum_{j=1}^{k} \otimes_{i=1}^{N} \mathbf{u}_i^j \tag{4.1}$$

where $\mathbf{u}_i^j \in R^{d_i}$. The decomposition of the tensor involves finding its rank-k approximation. While the notion of rank extends quite naturally to tensors, finding the rank of a tensor or decomposing a tensor is more difficult than in the matrix case. For matrices, the rank-k approximation can be reduced to repeated rank-1 approximations while for tensors, repeated deflation by dominant rank-1 tensors need not be a converging process. However, the factorization of a tensor is usually unique unlike matrix factorization [138]. Recently Hazan and Shashua [9] proposed a non-negative Tensor Factorization (NTF) method. Given a N-valent tensor G the method approximates G with a non-negative rank-k tensor $\sum_{j=1}^{k} \otimes_{i=1}^{N} \mathbf{u}_i^j$ described by Nk vectors $\mathbf{u}_i^j$ such that the reconstruction error:

$$\frac{1}{2} \left\| G - \sum_{j=1}^{k} \otimes_{i=1}^{N} \mathbf{u}_i^j \right\|^2 \tag{4.2}$$

is minimized subject to the condition $\mathbf{u}_i^j \geq 0$. An iterative scheme, where the vectors $\mathbf{u}_i^j$ are updated using a positive-preserving gradient descent rule, is used for estimation of $\mathbf{u}_i^j$. Thus, given positive initial estimates the method results in non-negative rank-k approximation of the G. If G is 3-valent tensor and of dimensions $d_1 \times d_2 \times d_3$ indexed by the indices i_1, i_2, i_3 with $1 \leq i_j \leq d_j$ for $j = 1, 2, 3$ then, given initial estimates of $\mathbf{u}_i^j$, the update rules are as follows [10]:

$$\begin{aligned}
u_{1,i}^j &\leftarrow \frac{u_{1,i}^j \sum_{s,t} G_{i,s,t} u_{2,s}^j u_{3,t}^j}{\sum_{m=1}^{k} u_{1,i}^m \langle \mathbf{u}_2^m, \mathbf{u}_2^j \rangle \langle \mathbf{u}_3^m, \mathbf{u}_3^j \rangle} \\
u_{2,i}^j &\leftarrow \frac{u_{2,i}^j \sum_{r,t} G_{r,i,t} u_{1,r}^j u_{3,t}^j}{\sum_{m=1}^{k} u_{2,i}^m \langle \mathbf{u}_1^m, \mathbf{u}_1^j \rangle \langle \mathbf{u}_3^m, \mathbf{u}_3^j \rangle} \\
u_{3,i}^j &\leftarrow \frac{u_{3,i}^j \sum_{r,s} G_{r,s,i} u_{1,r}^j u_{2,s}^j}{\sum_{m=1}^{k} u_{3,i}^m \langle \mathbf{u}_1^m, \mathbf{u}_1^j \rangle \langle \mathbf{u}_2^m, \mathbf{u}_2^j \rangle}
\end{aligned} \tag{4.3}$$

Factorization of Face Videos: Given a face video with n frames, each of width w and height h, it can be represented as a $h \times w \times n$ tensor where every frame of the video forms a slice of the tensor. This tensor can be decomposed into $3k$ vectors $\mathbf{u}_i^j$ to obtain a rank-k approximation of the video. An alternative interpretation of such a decomposition is pictorially shown in Figure 4.3. The t-th frame of the approximation of the tensor is given by $G_t = \sum_{j=1}^{k} u_{3,t}^j (\mathbf{u}_1^j \otimes \mathbf{u}_2^j)$, which is a linear combination of the matrices $\mathbf{u}_1^j \otimes \mathbf{u}_2^j$ weighted by the t-th coefficients of $\mathbf{u}_3^j$. The matrices $\mathbf{u}_1^j \otimes \mathbf{u}_2^j$ can be viewed as basis images which contribute to the appearance of the face. The positivity of the factors implies that these basis images can make only *additive* contributions to each frame of the video. The sparseness implies that the appearance of the object is each frame is not distributed over many of these basis images and can be represented using only a few of them. Figure 4.4 shows frames from face video and the basis images corresponding to the factors obtained upon factorization of its tensor representation. It can be seen that the energy in the basis images is located near the regions corresponding to different regions of the face.

An important implication of these facts is that the invariant content in the video is encoded in few basis images which contribute uniformly to all the frames. In contrast, the appearance of the dynamic content in the video is encoded in basis images whose contribution to different frames in the video varies widely. These facts essentially imply that the basis images correspond to various parts of the face, both static as well as dynamic. Tensorial factorization of videos, thus, also can viewed as a tool for separation of appearance and dynamics in the video. These observations help us in determining factors that affect the appearance of a particular region in the face video. The factors affecting the dynamic content are most relevant for the analysis of facial expression videos. These facts will be used in identification of relevant factors for specific tasks. Figure 4.5 shows frames from the reconstructed video and the variation of the reconstruction error with the number of factors i.e., the rank k. Usually, the low rank approximation results in reduction of size by an order of magnitude, while maintaining a satisfactory visual quality.

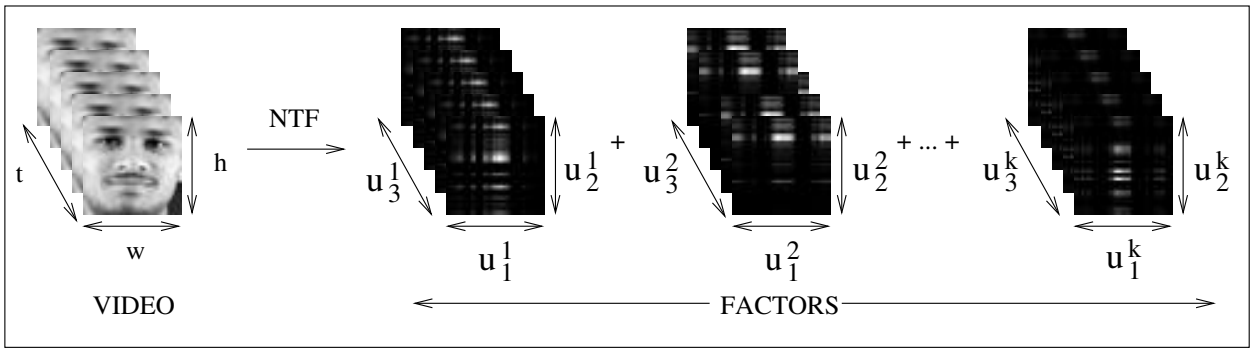


Figure 4.3: Tensor Factorization of a video. Each frame of the video is sum of the corresponding frames from the low rank factors. Each low rank factor is formed by stacking weighted versions of a basis image.

4.4 Identification of Relevant Factors

The tasks such as expression transfer require identification of factors that affect the appearance of certain regions of face in the video. Once such factors are identified, changing those factors enables us to manipulate the appearance of those regions in the face. For instance, expression transfer or expression recognition requires identification of factors that best represent the expression. Two

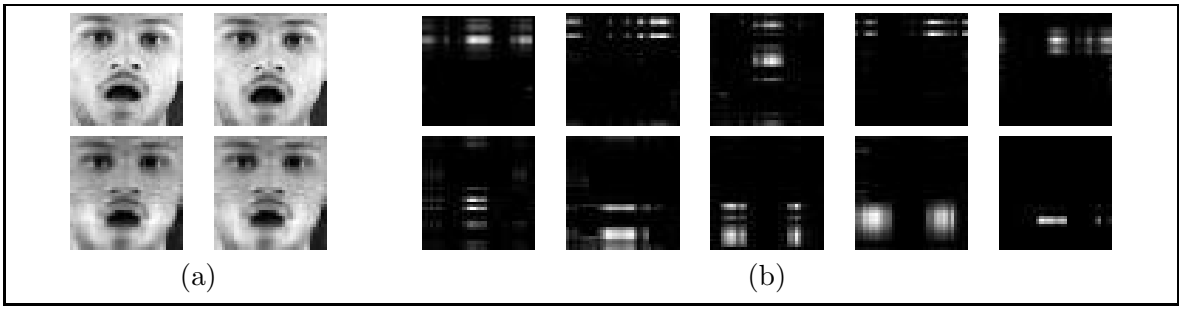


Figure 4.4: Basis obtained by tensorial factorization of a face video (with the expression *surprise*). (a) shows the representative frames of the original video (top row) and the reconstructed video (second row). (b) shows a subset of the basis-image set. It can be seen that the energy in these images is concentrated near the regions which correspond to location of parts like cheeks, mouth, nose *etc.*

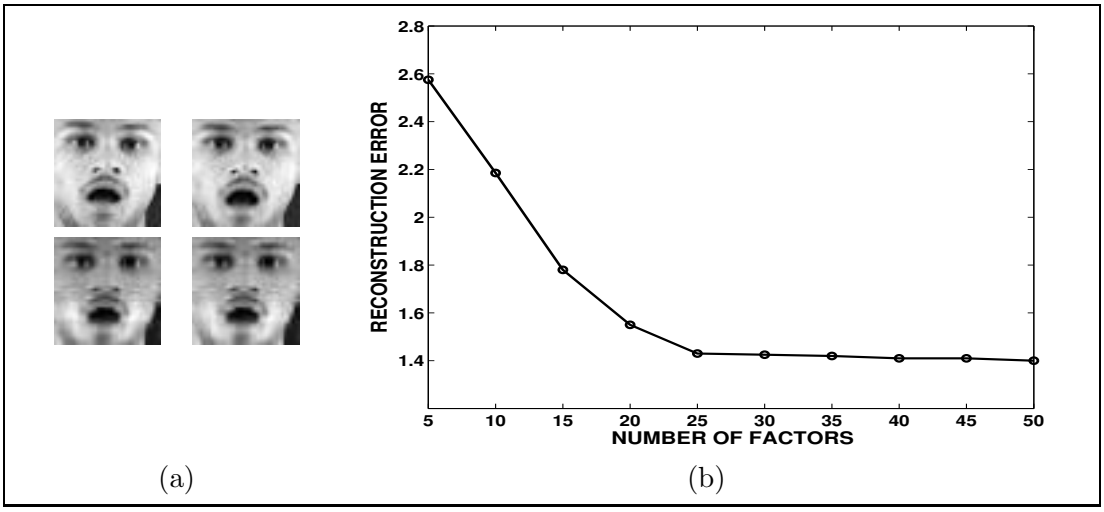


Figure 4.5: (a) The original and reconstructed frames of a face video with k=20 and k=50. (b) The reconstruction error plotted against the rank k

simple methods to identify the factors relevant to a task are described below.

4.4.1 Basis-Image based method

The identification of factors that best represent an expression requires the knowledge of the appearance of a neutral face. A neutral face video, of the same subject as in the expression video, is used for this purpose. Identification of relevant factors is done by factorizing both the expression video and the neutral face video. Let V_E and V_N be the expression and neutral face videos and let $\mathbf{U}_E = \{\mathbf{u}_j^i\}$ and $\mathbf{U}_N = \{\mathbf{v}_j^i\}$ be the factors obtained by decomposing the tensor representation of V_E and V_N. To facilitate selection of relevant factors, the factors must be aligned i.e. factors causing the appearance of same region in the frames of the video must be identified. Let $\mathbf{B}_E$ and $\mathbf{B}_N$ be the basis image sets corresponding to the factors $\mathbf{U}_E$ and $\mathbf{U}_N$ respectively. The correspondence between the factors can be established using a greedy algorithm that uses the similarity score between elements of the basis image sets $\mathbf{B}_E$ and $\mathbf{B}_N$. Since the basis images are all positive and

sparse with local patches, rudimentary metrics like sum of squared difference capture the similarity well. For the current work, a similarity score based on the distance between the centroids of the local patches in the two images and the distribution of pixels around the centroid is used. The correspondence between the factors belonging to the two sets can be established by selecting the best matching pair of factors, eliminating them and then repeating the process for the remaining set of factors. Algorithm 5 gives the complete description of alignment of factors.

1: Build basis image sets : $\mathbf{B}_E \leftarrow \mathbf{U}_E$, $\mathbf{B}_N \leftarrow \mathbf{U}_N$
2: Compute similarity scores : $S_{ij} \leftarrow similarity(\mathbf{B}_E^i, \mathbf{B}_N^j)$
3: $I \leftarrow \phi$
4: **for** $i = 1$ to k **do**
5: Find p,q such that S_{pq} is maximum where $\mathbf{B}_E^p \in \mathbf{B}_E$, $\mathbf{B}_N^q \in \mathbf{B}_N$
6: $I = I \cup \{(p, q)\}$
7: $\mathbf{B}_E \leftarrow \mathbf{B}_E - \{\mathbf{B}_E^p\}$, $\mathbf{B}_N \leftarrow \mathbf{B}_N - \{\mathbf{B}_N^q\}$
8: **end for**
9: **return** I

Algorithm 5: AlignFactors($\mathbf{U}_E, \mathbf{U}_N, k$)

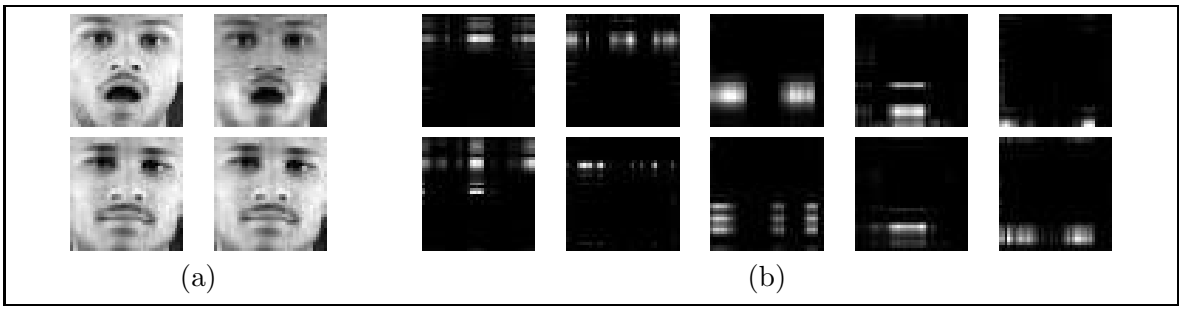


Figure 4.6: Alignment of factors corresponding to a facial expression video and a neutral face video. (a) shows frames from both the videos and their reconstructions. (b) shows the aligned factors. The factors of the first video are shown in the top row and the corresponding factors are shown in the next row.

4.4.2 Factorization of the difference tensor

An alternative scheme for identification of relevant factors, that are useful for recognition, arises from the Fisher-like criterion for discriminant analysis. Given two videos V_1 and V_2, represented as tensors $\mathbf{G}_1$ and $\mathbf{G}_2$, the objective is to find a rank-k tensor $\mathbf{W} = \sum_{j=1}^{k} \otimes_{i=1}^{3} \mathbf{w}_i^j$ such that projections of $\mathbf{G}_1$ and $\mathbf{G}_2$ on to this tensor differ the most. The objective is to optimize the quantity

$$\langle \sum_{j=1}^{k} \otimes_{i=1}^{3} \mathbf{w}_i^j, \mathbf{G}_1 \rangle - \langle \sum_{j=1}^{k} \otimes_{i=1}^{3} \mathbf{w}_i^j, \mathbf{G}_2 \rangle \tag{4.4}$$

which is equivalent to optimizing $\langle \sum_{j=1}^{k} \otimes_{i=1}^{3} \mathbf{w}_i^j, \Delta\mathbf{G} \rangle$, where $\Delta\mathbf{G} = \mathbf{G}_1 - \mathbf{G}_2$ captures the changes in the appearance of the video. Although the tensor $\Delta\mathbf{G}$ is not guaranteed to be positive, it can be normalized such that the elements are all non-negative and $\mathbf{W}$ is estimated as the rank-k

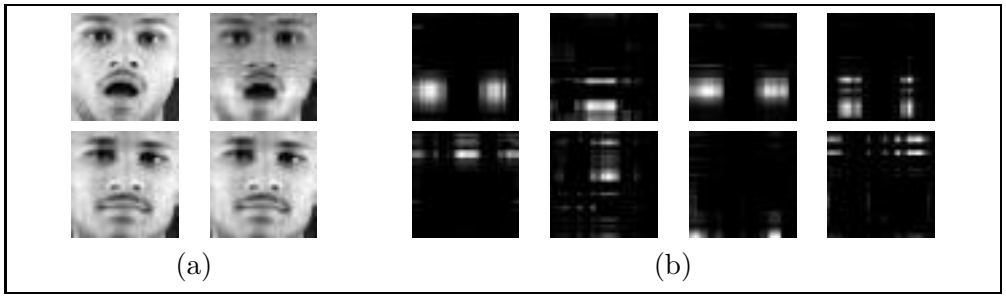


Figure 4.7: Identification of expression-specific factors using basis images. (a) shows representative frames from two videos and their reconstructions from the factors : a *surprise* expression video (top row) and a neutral face video of the same subject (second row) . (b) (top row) shows the basis images chosen corresponding to the factors chosen by the method. Note that the energy in these images is centered around the mouth region where the appearance differs significantly. The result of transferring these factors to the neutral video is shown in Figure 4.9. The second row shows the basis that are least preferred. These images resemble the invariant parts like nose and eyes.

approximation of $\Delta\mathbf{G}$. The factors $\mathbf{w}_i^j$ are not useful for the problem of expression transfer, but when the basis images corresponding to these factors are used as filters, the frames of the two videos give to rise markedly different responses to those filters. Thus, the factors provide a bank of highly discriminative filters that can be used effectively for classification tasks. Figure 4.8 shows the factors obtained by factorizing the difference tensor. It can be seen that the basis images corresponding to the resulting factors are useful for discriminating between the two expressions (*neutral* and *surprise*).

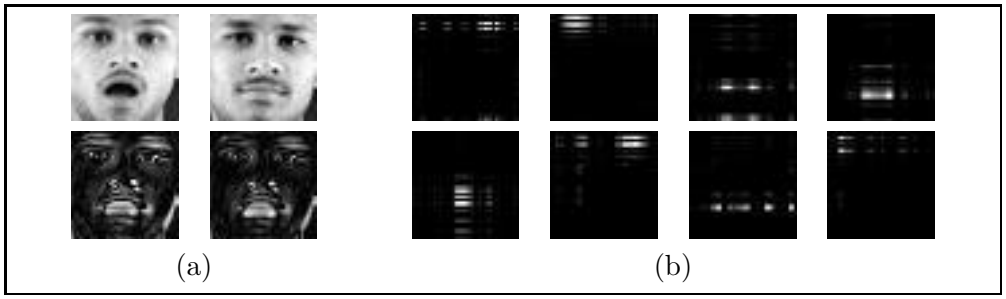


Figure 4.8: Identification of discriminative filters by factorizing the difference tensor. (a) (top row) shows representative frames from both the videos. The next row shows frames from the difference tensor and reconstructions. (b) shows the basis images corresponding to the factors chosen by the algorithm obtained by factorizing the difference tensor. The energy in the images is centered around the mouth and the eyebrows where the appearance differs from the neutral face.

4.5 Face Video Processing using NTF

4.5.1 Expression Transfer

The problem of expression transfer has the flavor of style and content separation problem described in [23], where the expression can be thought of as style and the underlying face as content. The

appearance is a result of interaction between these two factors. An alternative pair of factors that cause the appearance of a face are the shape and texture characteristics of the face. However, the interaction between such factors may not be modeled effectively by simple models like linear or bi-linear models. As a simple and useful alternative, the appearance factors obtained by decomposition of the facial expression video are used to transfer the expression. As observed above, the knowledge of appearance of the neutral face is a prerequisite for identification of the expression-specific factors, as well as, for transfer of the expression. The expression transfer problem considered in the current work is posed as follows: Given a video V_{1E} of a subject P_1 with certain expression E, the neutral face videos of the subject P_1 and another subject P_2, synthesize the video V_{2E} of subject P_2 with the same expression E.

1: $\mathbf{U}_{1E} \leftarrow NTF(V_{1E})$; $\mathbf{U}_{1N} \leftarrow NTF(V_{1N})$; $\mathbf{U}_{2N} \leftarrow NTF(V_{2N})$
2: $I = AlignFactors(\mathbf{U}_{1E}, \mathbf{U}_{1N})$
3: $J = AlignFactors(\mathbf{U}_{1N}, \mathbf{U}_{2N})$
4: Choose $\mathbf{U}^*_{1E} \subset \mathbf{U}_{1E}$ such that elements of $\mathbf{U}^*_{1E}$ are maximally dissimilar to corresponding factors in $\mathbf{U}_{1N}$, $\mathbf{U}^*_{1N}$ be the corresponding factors in $\mathbf{U}_{1N}$
5: Find $\mathbf{U}^*_{2N} \subset \mathbf{U}_{2N}$ the factors corresponding to $\mathbf{U}^*_{1E}$ using I, J and $\mathbf{U}^*_{1N}$
6: $\mathbf{U}_{2E} \leftarrow \mathbf{U}_{2N} \cup \mathbf{U}^*_{1E} - \mathbf{U}^*_{2N}$
7: $V_{2E} \leftarrow NTFReconstruct(\mathbf{U}_{2N})$

Algorithm 6: TransferExpression($V_{1E}, V_{1N}, V_{2N}, k$)

The expression specific factors can be identified by using the algorithm described in Section 4.4. Once the factors are aligned and the relevant factors are identified, the transfer is achieved by directly transferring the expression-specific factors in the source video to the target neutral video. Figure 4.9 shows the reconstructed video after the expression-specific factors are transfered to the neutral faces of the same subject and of a second subject. Algorithm 6 summarizes the algorithm for expression transfer. The appearance based solution presented here transfers the appearance factors alone and might appear like a *cut and paste* method that results in discontinuities in the frames of synthesized videos. However, experiments have shown that the synthesized videos are visually satisfactory with little or no discontinuities. The quality of synthesized video does depend on the source video i.e how close the shape of the source face is to the target face, differences in facial features like complexion, how well the expression is articulated in the source video etc.

4.5.2 Expression Recognition

The second problem of interest is the recognition of facial expressions. Recognition of facial expressions is a challenging task as the appearance of expression varies drastically for different subjects. We explore the possibility of solving this problem using appearance based features alone, by using the factors obtained by tensorial factorization. We used the neutral face video, during training, for identification of expression-specific factors. The recognition is done by comparing the constituent factors of the test video with the expression specific factors of the samples in the training set. For each expression-specific factor set in the training data, a maximally similar subset of the factors of the test video is found. The matching score is computed as the mean similarity score between the matched factors. The test video is assigned the label of the training sample that gives the maximum matching score.

A second method for recognition of expressions uses the factors obtained by decomposition of the difference tensor. First a classifier that can discriminate between two expressions is built

and the DDAG architecture [139] is used to extend it to multiple expressions. The basis images, corresponding to the factors of the difference tensors, are used as filters and the mean response to each filter over the entire video is taken as a feature. Feature vectors, that are built in this manner, are compared to training samples using euclidean distance as the metric.

4.5.3 Morphing

Image metamorphosis, also known as morphing, consists of a fluid transformation from a source image to target image. The technique has several applications in generating visual effects and recognition of faces. Factorizing the video provides a natural way to generate such a morph sequence, by replacing the factors of source video with the factors of the target video successively. Given the source and target images, two tensors are built using these images. The first tensor consists of the source image stacked repeatedly and the second tensor is built similarly from the target image. Factorizing both these tensors results in a local-parts decomposition of the two face images. Transferring the appearance of these parts is a natural way of generating a morph sequence. The factors are aligned using the algorithm described in Section 4.4 and the factors in the source tensor are successively replaced with corresponding factors from the target tensors. The tensors reconstructed from the intermediate set of factors consist of replications of a single image that represents one image of the morph sequence. Algorithm 7 gives the complete description of the procedure from generating a morph sequence M between two images I_1 and I_2.

1: $\mathbf{V}_1 \leftarrow imageToTensor(I_1)$, $\mathbf{V}_2 \leftarrow imageToTensor(I_2)$
2: $\mathbf{U}_1 \leftarrow NTF(V_1)$; $\mathbf{U}_2 \leftarrow NTF(V_2)$;
3: $C = AlignFactors(\mathbf{U}_1, \mathbf{U}_2)$
4: $M_0 = I_1$
5: $\mathbf{U}_t = U_1$
6: **for** $i = 1$ to k **do**
7: Find j such that $(i, j) \in C$ $\mathbf{U}_t = \mathbf{U}_t - U_1^i \cup U_2^j$
8: $V_t \leftarrow NTFReconstruct(\mathbf{U}_t)$
9: $M_i = V_{t,1}$
10: **end for**
11: **return** M

Algorithm 7: MorphSequence(I_1, I_2)

4.6 Results and Discussion

Some results of our experiments with the above algorithms were presented in Section 5.1. More results, with further discussion and insights into the results, are presented below. All the experiments were conducted on a dataset that was collected in-house. An OLYMPUS C7000 camera was used to capture the videos, at 30fps, under controlled settings with minimal illumination or pose variation. The videos were preprocessed to segment the faces and scale them to a fixed size. The algorithms proposed here work best in presence of a good degree of alignment of faces in the videos. The dataset consists of 13 subjects in 8 different expressions including a neutral face video for each subject. The frame count in the videos was equalized (to 60), for the ease of implementation, by deleting frames where there was no change in appearance. The frames were all scaled to fixed dimensions (of 100×100 pixels). Figure 4.9 shows representative results of the experiments on this

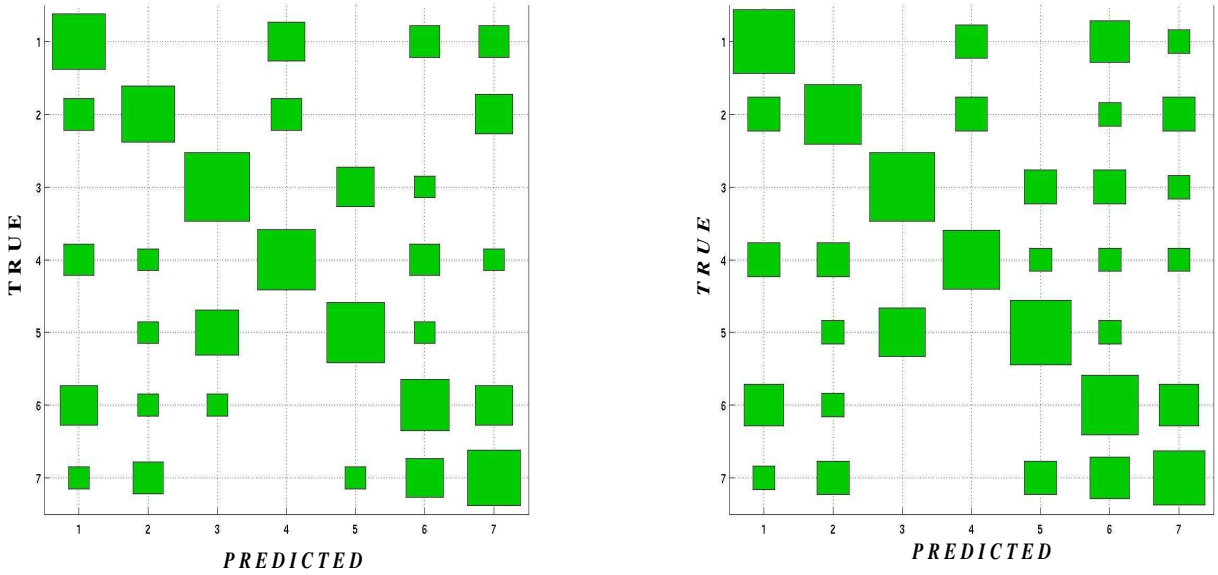


Table 4.1: The Hinton diagram of the confusion matrix for the expression recognition task. The squares along the diagonal represent fraction of correctly recognized samples. It can be seen that the accuracy is reasonable despite the absence the of feature correspondences or complex modeling schemes.

dataset. It can be seen that the results are visually satisfactory despite appearance information alone being made use of. The minor discontinuities and artifacts are due to the erroneous transfer of some rank-1 factors which are manifested as horizontal/vertical lines. Further, large variations in complexion and face shape also result in discontinuities. Low-pass filtering and contrast enhancement were applied on the frames of the synthesized video as a post-processing step to enhance the quality.

The dataset used for expression recognition is same as the dataset used for the expression transfer experiments. Excluding the neutral video, there were 7 expressions of 13 subjects. Since the sample size was small, leave-one out mode testing was used for testing both the algorithms. The Hinton diagrams corresponding to the confusion matrices obtained using both the methods are shown in table 4.1. The leading diagonal elements show that the recognition accuracy is quite satisfactory, despite the use of appearance information alone. The accuracy improved when the images of the faces of different persons are further aligned, by manually selecting the control points such as the centers of eyes and tip of the nose. The overall accuracy is around 51%, comparable to the state of the art methods which use feature correspondences and muscular motion models [83]. The size of the dataset that is used for the experiments precludes any definite conclusion. However, as the nature of information used by the current technique differs from that of existing ones, they are complementary and development of a hybrid solution with improved recognition rate is a promising direction.

Figure 4.11 shows the result of applying the face morphing method presented above on a pair of faces. It can be seen that the transition is visually smooth. Thus, the algorithm provides a simple yet effective way to perform face morphing.

We have proposed methods for expressions transfer, expression recogntion and face morphing

using tensorial factorization. The factors obtained by factorizing the tensor representation of videos were used to perform these tasks. We have proposed heuristical methods to align the factors and identify the relevant subset of factors, which are then used to perform the above tasks.

The methods proposed above for the three tasks differ significantly from the previous methods, since appearance information alone is employed. The results of expression transfer are visually pleasing. However, it must be noted that proposed method is far from the ideal solution which could model the underlying physical phenomenon completely. Such a solution would require richer information such as geometry of the faces, muscular motion, texture properties like complexion etc. Using appearance information alone to achieve the effect of expression transfer has inherent drawbacks. This is due to a variety of facts that prevent appearance alone to completely characterize facial expression : appearance of the same expression varies widely across persons, the appearance also depends on the geometry of the face, the differences in skin complexion, texture properties etc. The proposed method, therefore, fails to produce satisfactory results in the cases where the faces in the videos lack proper alignment and faces differ widely in complexion and shape. Also, the results depend on the way expression is articulated in the source video: transferring the smile of two different persons to the same person produces different results. Further issues like lack of alignment, illumination and shadows caused by light sources etc. have not been addressed in this work. Transferring such surface detail completely requires more complex information. Similarly, A visually smooth morph sequence requires a smooth interpolation of many characteristics apart from appearance and hence the quality of morph sequence depends on the nature of information being interpolated. When the faces are close in shape, our algorithm produces visually satisfactory results while large disparity in shape or complexion, results in in a jerky morph sequence. Despite this limitation, the method can be successfully employed to obtain morph sequences in cases where information such as mesh models of the face is not available.

In general, untangling all the factors involved in a complex phenomenon like articulation of facial expressions and modeling their interaction is a difficult task. However, attempting to capture the essence of the interaction between all these factors from the appearance alone is infeasible. Despite this fact, in a large number of cases the appearance alone provides enough clues for recognition. The experimental results show us that the methods provide a simple and efficient alternative for cases where the geometric information and feature correspondences are difficult to obtain. Moreover, since they work with lesser information they only complement the methods that use more complex information. These traits, along with computational efficiency, make the proposed methods attractive alternatives to the more complex solutions.

4.7 Summary and Conclusions

In summary, we have used NTF technique to perform various tasks on face videos using appearance information alone. To the best of our knowledge, a purely appearance based approach to expression transfer and expression recognition has not been attempted so far.The methods based on tensor representation of videos are more natural, simpler and insightful. The contributions of the current work are:

- Simple and efficient methods for selection of task-specific factors from the set of factors obtained by tensorial factorization of videos.
- An efficient technique to perform facial expression transfer without requiring feature correspondences or muscular motion models.

- A simple NTF based technique to perform perform face-image metamorphosis by changing the appearance using the factors.
- A novel technique complementary to existing methods for recognition of expressions using expression-specific factor in facial expression videos.

Multilinear techniques offer new insights into analysis and processing of video information. The factors obtained by factorization of videos can be used for a number of other purposes like dynamic event analysis and background modeling. The appearance and dynamics separation achieved by tensorial factorization provides valuable cues for analysis of dynamic events in videos and we are actively pursuing this problem. In summary, task-specific factor selection makes it possible to solve a wide range of problems using tensorial factorization of videos/image-cubes and is a promising direction for future research.

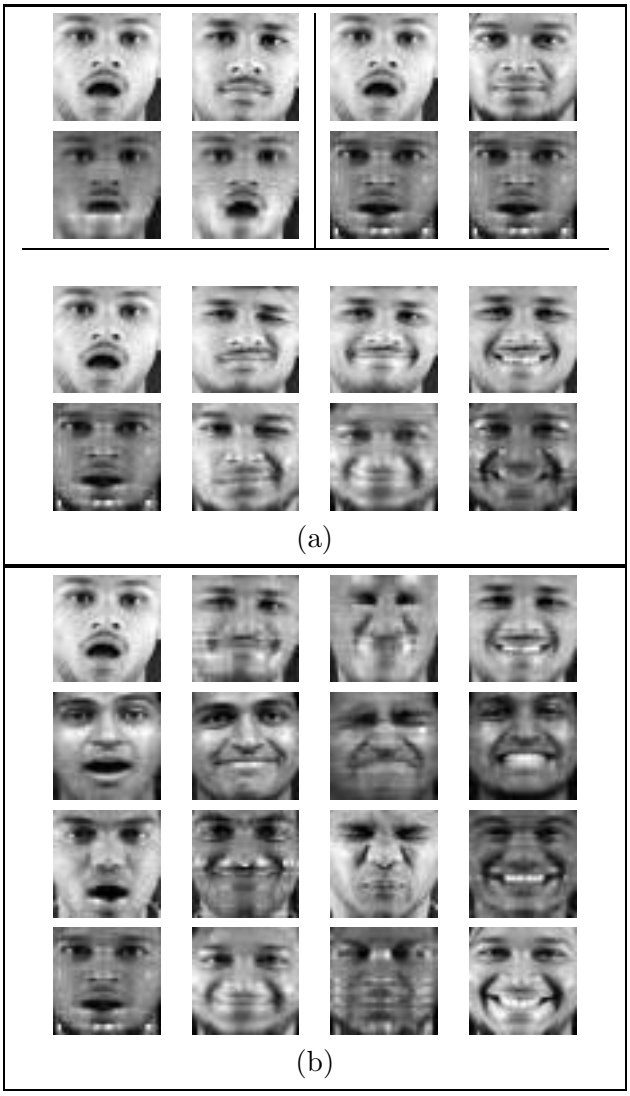


Figure 4.9: Results of expression transfer using the algorithm described in section 4.5. (a) : The top row shows frames from the facial expression video and the neutral face video(The first neutral face corresponds to the same subject). The second row shows the frames of the synthesized video. The third and fourth rows show four expressions (surprise, wink, smile, giggle) transferred to the neutral face of another subject. (b) : results on a set of four subjects where the expression in the videos along the diagonal were transfered to other subjects (the columns show the same expression for different subjects and the rows show the same subject in different expressions). Only the diagonal videos were originally available.

Figure 4.10: Expressions used for the expression recognition experiment :*Smile*, *Left wink*, *Surprise*, *Giggle*, *Mock*, *Right wink*, *Disgust*

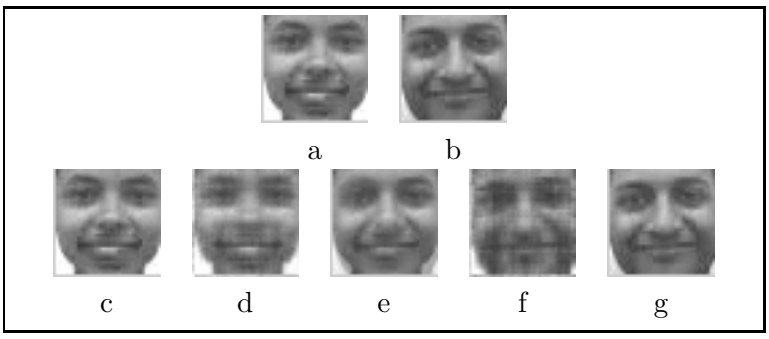


Figure 4.11: Face Morphing. Upper row: (a), (b) the two input images. Lower row: (c), (g) the input images, (d) - (f) the transition images.

Chapter 5

Data Analysis using Kernel Functions and Factorization

5.1 Introduction

The previous chapters have demonstrated the power of Kernelization and Factorization for the analysis of data. The two methods seemingly process data in orthogonal manner. Kernel methods attempt to simplify the relations in the data by recoding the data. The kernel function allows implicit recoding of the data, thus, ensuring that the resulting methods are efficient. Although the feature space is never explicitly accessed, it must be noted that, for most kernel functions, the corresponding feature space has dimensionality much larger than the input space. In effect, kernel functions are used to derive new features from existing ones, which possibly render the regularities in the data simpler and easily detectable. Factorization, on the other hand, is primarily used to detect and exploit the redundancies in the data. For instance, principal component analysis and discriminant analysis extract directions that are most relevant for a particular task by ignoring the irrelevant features. In cases such as style and content separation [23], factorization learns the generative process and the variables associated with the process, thus, arriving at compact representations of data. Kernelization and Factorization, thus, perform two opposite tasks. This suggests that the methods can be used together to perform a comprehensive analysis of data: The use of kernel functions, first, allows us to work in a feature space with large number of features where the patterns to be detected become simpler. Factorization, then, helps in ignoring the irrelevant features from the large set and arrive at a set of features that is small and yet powerful enough to capture any complex regularities in the data.

This chapter gives insights into how the two important techniques presented in the previous two chapters can be used together to enable a number of useful tasks. The advantages of mapping data to a different space, nonlinearly related to the input space, can only be realized when the data can be analyzed in that space. However, since the kernel functions map data only implicitly, the analysis of data in kernel-defined feature spaces is not straightforward and must be done via operations on the kernel matrix. Factorization is a useful technique to perform a number of useful tasks such as component analysis. In addition, computing a low-rank approximation of such matrices reduces the complexity of kernel algorithms in practical scenarios where the size of input is huge. In this chapter existing methods for component analysis and modeling in kernel defined feature spaces are reviewed. We argue that factorization of the kernel matrix is the common thread that runs through these various methods. We also show how factorization and kernelization can be used together to perform tasks such as style and content separation in the feature space. Kernelization allows us

to work with data (implicitly) mapped to a space where the relations of interest become simpler. Factorization allows us to learn such simple relations in an efficient manner. A combination of these tools enables efficient solutions to several problems.

Section 5.2 reviews the literature related to the use of factorization methods together with kernel functions. Section 5.3 shows how the use of covariance matrix, the gram matrix and their factors are used in various component analysis algorithms. Section 5.4 uses the kernel trick with the asymmetric bilinear model for interaction of style and content proposed in [23]. The results are demonstrated on typographic content classification. The superior performance of the kernel variant demostrates the power of the joint use of kernel functions and factorization.

5.2 Background

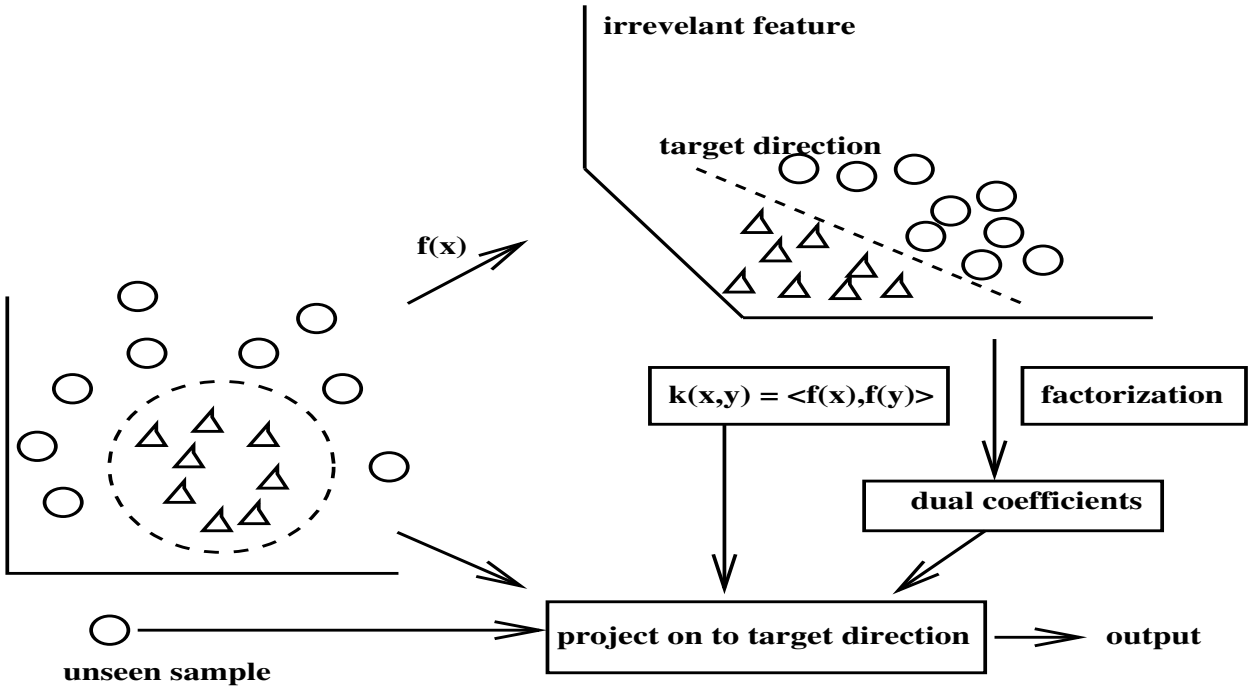


Figure 5.1: The use of kernel function with factorization to extract useful information from data. Kernel functions generate potentially useful features. Factorization refines them by finding the most relevant directions. The dual coefficients, the input samples and the kernel function are all used to analyze the corresponding information in a new sample.

The representation of nonlinearities using kernel functions allows linear algorithms to be implemented in the corresponding feature space. Most linear algorithms analyze the second order statistics of data, such as variance along a direction, and attempt to find solutions that are optimal with respect to them. The key property of the kernel trick is that the inner product information allows such second order statistics to be measured indirectly. The covariance and scatter matrices are used extensively in a number of feature extraction algorithms such as PCA [6] and discriminant analysis [60]. These matrices, in the feature space corresponding to a kernel function, cannot be directly computed without knowledge of feature embedding. However, the information that is

extracted by principal component analysis i.e., the eigen vectors corresponding to the top eigen values uses eigen decomposition of the covariance matrix. This decomposition is closely related to the eigen decomposition of the Gram matrix (which is the kernel matrix in the feature space). Thus, the information can indirectly be accessed in the feature space. This is the key to the kernel principal component analysis algorithm [3], also discussed in chapter 2. This connection between factorization of the kernel matrix and covariance matrices is used extensively for kernelization of new algorithms and increasing the efficiency of existing ones. Mei *et.al* [40] use Generalized Singular Value Decomposition for extraction of kernel biased discriminants in the feature space. Xiong *et.al* [140] use QR decomposition to increase the efficiency of kernel discriminant analysis. Non-negative factorization of kernel matrix [141] is used for extraction of nonlinear features for recognition. The technique of deriving a large number of features using kernels and then using factorization to pick the most useful (for a task) of them is used in most kernelized algorithms [1, 92]. Figure 5.1 gives an overview of the technique.

The separation of style and content using bilinear models [23] uses factorization to fit the model to observed data and learn the style and content parameters. However, the linear factor model is quite restrictive for complex phenomenon like human gait and facial expressions. Elgammal and Lee separate style and content on a nonlinear manifold [142]. They use the method for recognition of human gait. Wang *et.al* [143] use multifactor gaussian process to model the interaction of style and content. They use kernel functions with each factor separately, thus, allowing nonlinearity besides using multiple factors. The method proposed in the current work maps the input observations to the feature space, instead, and uses eigen decomposition of a matrix, computed using the kernel function alone. The content parameters can be computed directly and the style parameters can be accessed indirectly for computation of quantities such as squared norms. These computations are used to classify known typographic content rendered in unknown fonts (styles). The content classification problem is pictorially depicted in figure 5.2.

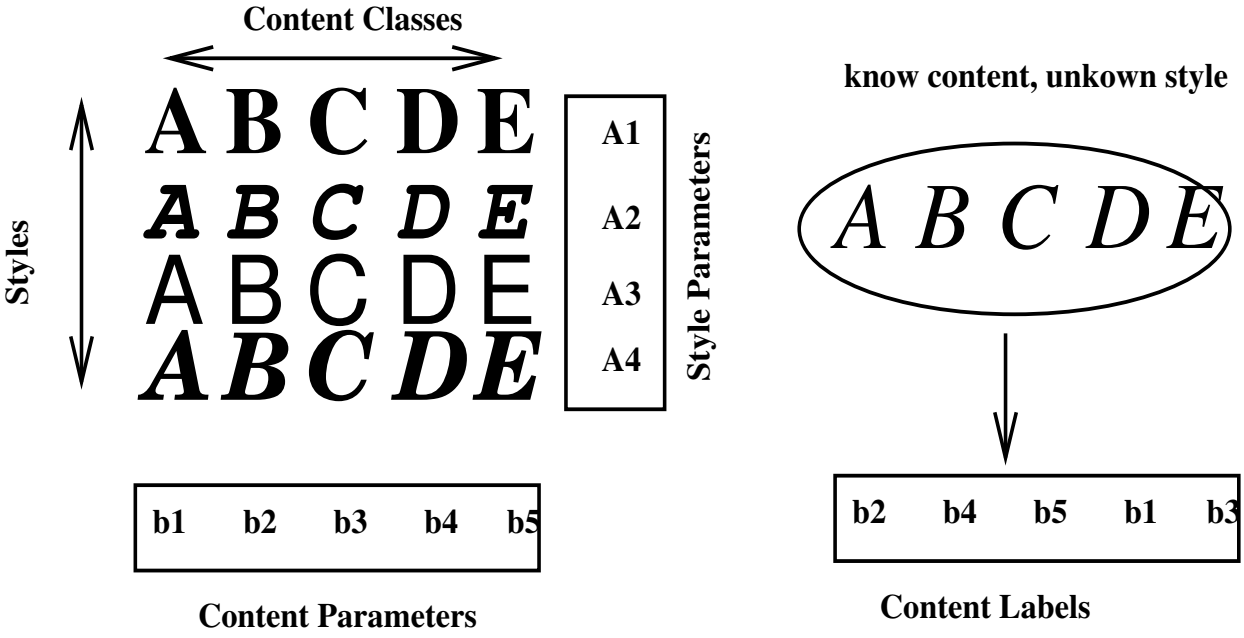


Figure 5.2: The typographic content classification problem. First, the content and style parameters are learnt from an input set of samples. When the known content is rendered in an unknown style, the task is to assign content labels to the new observations.

5.3 Complementary nature of Kernel Functions and Factorization

One of the important advantages of using kernel functions is the ability to work with large number of nonlinear features. The space and time complexity of computing such features, explicitly, is large. Moreover, in most cases, there is no prior knowledge of the class of features that is optimal for solving the problem. Kernel functions help us in tackling these problems by implicit transformation of data. However, this efficiency comes at a cost: kernel functions do not facilitate a precise control over the resultant feature space i.e. there is no straightforward way to ignore a subset of features in the feature space. This is also the case with rescaling of individual features to increase their relative importance in an algorithm. The only way to manipulate features in this manner is via the input samples: finding necessary features by projecting on to vectors which lie, in the feature space, in the span of images of the input samples. In the input space, factorization is the popular tool for such analysis of features. Various feature extraction algorithms use factorization, directly or indirectly, to identify the most relevant set of features. Due to the connection between sample covariance matrix and the kernel matrix it is possible to analyze features, in the feature space, using factorization of the kernel matrix. Further, the low rank approximation of the kernel matrix and the covariance matrix in the feature space are useful for tasks such as mixture modeling in feature space and increasing efficiency of kernel algorithms (such as SVM). Examples of the use of kernel functions followed by factorization are given below.

5.3.1 Analysis of the Feature Space

We show the use of kernel functions followed by factorization for detection of rich and useful features with the example of kernel principal component analysis [3]. Principal component analysis [6] is used to find the directions that best represent the data. Thus, given an input data set $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, the direction $\mathbf{w}$ that best represents the data is the principal component. Given that the best first order representation of the data is the mean $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$, the direction $\mathbf{w}$ is found by maximizing the residual along the direction $\mathbf{w}$:

$$\sigma^2(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\langle \mathbf{x}_i - \boldsymbol{\mu}, \mathbf{w}\rangle^2 = \sum_{i=1}^{n}\langle \mathbf{w}, \mathbf{x}_i - \boldsymbol{\mu}\rangle\langle \mathbf{x}_i - \boldsymbol{\mu}, \mathbf{w}\rangle \tag{5.1}$$

The statistics of the data used in the above equation are embedded in the sample covariance matrix $\mathbf{C}$ which capture the covariance between all possible pairs of features :

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^t \tag{5.2}$$

The direction $\mathbf{w}$ can now be found by maximizing the following criterion subject to the constraint $||\mathbf{w}||^2 = 1$

$$\xi(\mathbf{w}) = \mathbf{w}^t\mathbf{C}\mathbf{w} \tag{5.3}$$

Using a lagrangian multiplier, it can be found that $\mathbf{w}$ is the eigen vector corresponding to the top eigen value of $\mathbf{C}$. Thus, eigen decomposition of $\mathbf{C}$ gives the directions that best represent the data. Performing such a direct analysis in the feature space is not possible since the individual features, and hence the covariance between them cannot be accessed. However, since the solution vector can only be in the span of the input vectors, the dual representation can be used to access the residual along a direction determined by the dual coefficients $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_n\end{bmatrix}^t$,i.e., along

the direction $\sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$. As shown in chapter 2, the variance along this direction, in the feature space, can be measured using the centered kernel matrix $\mathbf{K}_C$ as

$$\sigma^2(\boldsymbol{\alpha}) = \frac{1}{n}\boldsymbol{\alpha}^t \mathbf{K}_C^2 \boldsymbol{\alpha} \tag{5.4}$$

and the constraint $||\mathbf{w}||^2 = 1$ is equivalent to the constraint $\boldsymbol{\alpha}^t \mathbf{K}_C \boldsymbol{\alpha} = 1$. Once again solving the optimization problem using lagrangian multipliers yields the following solution :

$$\mathbf{K}_C \boldsymbol{\alpha} = \lambda \boldsymbol{\alpha} \tag{5.5}$$

i.e., the eigen vector corresponding to the maximum eigen value of the kernel matrix gives the optimal dual coefficients. This example demonstrates the analysis of features in the feature space using factorization. In both the input space and the feature space the eigen decomposition enables identification of relevant features. Kernel functions capture higher order statistics by generating large number of nonlinear features and factorization helps in extracting useful information out of them. The recent use of kernel methods for face recognition [144] as an alternative higher order statistics(HOS) [145] demonstrates the utility of this technique.

5.3.2 Low-rank approximation of Covariance and Kernel Matrices

Apart from analysis of features, factorization of the kernel matrix has been used for several other tasks as well. One of the popular usages is in modeling a data set using a gaussian distributions in the feature space. The covariance matrix Σ which is central to the gaussian distribution cannot be accessed in the feature space. However, the covariance matrix can be approximated using the top k eigen value - eigen vector pairs in the feature space :

$$\Sigma = \sum_{i=1}^{k} \lambda_i \mathbf{v}_i \mathbf{v}_i^t \tag{5.6}$$

Since the dual coefficients corresponding to the eigen vectors can be computed in the feature space, this approximation allows computation of terms such as

$$\phi(\mathbf{x})^t \Sigma^{-1} \phi(\mathbf{x}) = \phi(\mathbf{x})^t \left(\sum_{i=1}^{k} \lambda_i^{-1} \mathbf{v}_i \mathbf{v}_i^t \right) \phi(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i^{-1} \langle \phi(\mathbf{x}), \mathbf{v}_i \rangle^2 \tag{5.7}$$

The inner product in the above equation is the projection of $\phi(\mathbf{x})$ on to an eigen vector of the covariance matrix. As shown in chapter 2, given the dual coefficients, this projection can be computed using the kernel function alone. This workaround to access the covariance matrix is used in modeling in feature space [33] and kernel design [146].

Factorization of the kernel matrix is one of the popular ways to solve the optimization problem in support vector machine(SVM). However, in domains such as datamining where the number of samples is much larger than the dimensionality of the data the size of kernel matrices becomes very large (as it grows quadratically with the number of samples). Operating on the full data, in such cases, is not only expensive but also unnecessary since the intrinsic dimensionality of the data is usually much lower. Wu *et.al* [147] use a low rank approximation of the kernel matrix (and other matrices involved) and incrementally update the factors as new data arrives to increase the efficiency. Thus, factorization and low rank approximation of kernel matrices are powerful techniques that aid in tasks such as kernelization of new algorithms and increasing efficiency of method involving these matrices.

5.4 Nonlinear Style and Content Separation using Kernels

This section uses the kernel trick to introduce nonlinearity into the algorithm for separation of style and content proposed by Freeman and Tanenbaum [23] using asymmetric bilinear models. The asymmetric case is simpler for model fitting and for kernelization. However, it does not capture the interaction that is independent of style and content. Thus, the model can not be used for the translation task.

5.4.1 Asymmetric Bilinear Factor Model

In the asymmetric model for interaction of style and content, the parameters for style s and content c are denoted by $\mathbf{a}^s \in \mathcal{R}^I$ and $\mathbf{b}^c \in \mathcal{R}^J$ respectively. Let $\mathbf{y}^{sc} \in \mathcal{R}^H$ be the observation vector corresponding to the content c rendered in style s. The asymmetric model of interaction models the generation of each component $\mathbf{y}_h^{sc}$ of the observation vector as bilinear combination of the style and content parameters, where the coefficients (interaction parameters) are specific to the style, as follows :

$$\mathbf{y}_h^{sc} = \sum_{ij} w_{ijh}^s a_i^s b_j^c \tag{5.8}$$

Here, the coefficients w_{ijh}^s are specific to the style. However, representing style using two different sets of variables is unnecessary and the variables can be absorbed into a single representation at the cost of increase in number of parameters. The style specific terms corresponding to style s can be combined as

$$a_{jh}^s = \sum_i w_{ijh}^s a_i^s \tag{5.9}$$

The number of parameters representing style thus increases from I to JH. The new expression for the observation becomes

$$\mathbf{y}_h^{sc} = \sum_j a_{jh}^s b_j^c \tag{5.10}$$

The above equation can be expressed more compactly using matrix notation. Let $\mathbf{A}^s$ be the matrix corresponding to style s, of size $H \times J$ with entries a_{jh}^s. Using this matrix the expression for the observation vector becomes

$$\mathbf{y}^{sc} = \mathbf{A}^s \mathbf{b}^c \tag{5.11}$$

A consequence of this model is that there are no parameters that are independent of both style and content. Thus, it is not possible to learn any structure in the observation vectors that is independent of these two factors. However, if the model parameters are estimated accurately, they can be used for classification of known content rendered in new styles. Alternate interpretations of the above equation and analogies to representation such as eigenfaces [61] can be found in [23].

5.4.2 Model Fitting

In the case of style and content classification, model estimation needs to be done both during the training as testing phases. First, a model is learnt from the training data and then is adapted to fit to test data that belongs to one of the known style or content classes. The model parameters during training are estimated so that the least squared error

$$E = \sum_{t=1}^{T} \sum_{s=1}^{S} \sum_{c=1}^{C} \delta^{sc}(t) ||\mathbf{y}(t) - \mathbf{A}^s b^c||^2 \tag{5.12}$$

where $\delta^{sc}(t)$ are indicator variables i.e., $\delta^{sc}(t) = 1$, if $\mathbf{y}(t)$ is in content class c and style s and 0 otherwise. If the number of samples present in each of the $S \times C$ content-style pairs is same, then a closed form solution using Singular Value Decomposition exists. Consider the mean observation vectors from each style-content pair :

$$\tilde{\mathbf{y}}^{sc} = \frac{\sum_t \delta^{sc}(t)\mathbf{y}(t)}{\sum_t \delta^{sc}(t)} \tag{5.13}$$

It can be seen that the $SH \times C$ data matrix formed by stacking all these mean observations has the following factorization :

$$\mathbf{Y} = \begin{bmatrix} \tilde{\mathbf{y}}^{11} & \cdots & \tilde{\mathbf{y}}^{1C} \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{y}}^{S1} & \cdots & \tilde{\mathbf{y}}^{SC} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^1\mathbf{b}^1 & \cdots & \mathbf{A}^1\mathbf{b}^C \\ \vdots & \ddots & \vdots \\ \mathbf{A}^S\mathbf{b}^1 & \cdots & \mathbf{A}^S\mathbf{b}^C \end{bmatrix} = \begin{bmatrix} \mathbf{A}^1 \\ \vdots \\ \mathbf{A}^S \end{bmatrix} \begin{bmatrix} \mathbf{b}^1 & \cdots & \mathbf{b}^C \end{bmatrix} = \mathbf{AB} \tag{5.14}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ have been used to represent the style and content factors. Estimating $\mathbf{A}$ and $\mathbf{B}$ can be done using the singular value decomposition of $\mathbf{Y}$. Since SVD yields $\mathbf{Y} = \mathbf{UDV}^t$, $\mathbf{A}$ can be defined as the first J columns of the matrix $\mathbf{UD}$ while $\mathbf{B}$ is defined as the first J rows of $\mathbf{V}^t$. Details concerning the selection of model parameter J, balancing the matrix $\mathbf{Y}$ in case of skewed distributions can be found in [23]. The use of these parameters for style and content classification, extrapolation and translation requires adaptation of the learnt model to fit the test data. However, in the current work, we consider only content classification which does not require the algorithms proposed therein.

5.4.3 Extracting content parameters in feature space

The above model fitting can be done in the feature space corresponding to a kernel function. To demonstrate this, we begin the observation vectors mapped in to feature space via the feature map $\phi(.)$ corresponding to the kernel function $\kappa(.,.) : \mathbf{y}(t) \mapsto \phi(\mathbf{y}(t))$. Thus, the elements of the matrix $\mathbf{Y}$ are of the form

$$\tilde{\mathbf{y}_\phi}^{sc} = \frac{\sum_t \delta^{sc}(t)\phi(\mathbf{y}(t))}{\sum_t \delta^{sc}(t)} \tag{5.15}$$

These elements cannot be accessed in the feature space, but the elements of the $C \times C$ matrix $\mathbf{Y}^t\mathbf{Y}$ can be accessed. This is so, because this matrix is

$$\mathbf{Y}^t\mathbf{Y} = \begin{bmatrix} \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{s1}, \tilde{\mathbf{y}_\phi}^{s1} \rangle & \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{s1}, \tilde{\mathbf{y}_\phi}^{s2} \rangle & \cdots & \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{s1}, \tilde{\mathbf{y}_\phi}^{sC} \rangle \\ \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{s2}, \tilde{\mathbf{y}_\phi}^{s1} \rangle & \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{s2}, \tilde{\mathbf{y}_\phi}^{s2} \rangle & \cdots & \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{s2}, \tilde{\mathbf{y}_\phi}^{sC} \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{sC}, \tilde{\mathbf{y}_\phi}^{s1} \rangle & \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{sC}, \tilde{\mathbf{y}_\phi}^{s2} \rangle & \cdots & \sum_{s=1}^S \langle \tilde{\mathbf{y}_\phi}^{sC}, \tilde{\mathbf{y}_\phi}^{sC} \rangle \end{bmatrix} \tag{5.16}$$

The inner products in each of the elements above are of the form $\langle \tilde{\mathbf{y}_\phi}^{si}, \tilde{\mathbf{y}_\phi}^{sj} \rangle$, where s is a style class and i and j are two different content classes. These inner products can be computed using the kernel function alone as follows :

$$\langle \tilde{\mathbf{y}_\phi}^{si}, \tilde{\mathbf{y}_\phi}^{sj} \rangle = \langle \frac{\sum_p \delta^{si}(p)\phi(\mathbf{y}(p))}{\sum_p \delta^{si}(p)}, \frac{\sum_t \delta^{sj}(q)\phi(\mathbf{y}(q))}{\sum_q \delta^{sj}(q)} \rangle = \frac{\sum_p \sum_q \delta^{si}(p)\delta^{sj}(q)\kappa(\mathbf{y}(p), \mathbf{y}(q))}{\sum_p \delta^{sj}(p) \sum_q \delta^{sj}(q)} \tag{5.17}$$

Thus, the matrix $\mathbf{Y}^t\mathbf{Y}$ can be computed in the feature space. Since $\mathbf{Y} = \mathbf{UDV}^t$ it can be seen that $\mathbf{V}$ can be obtained by eigen decomposition of $\mathbf{Y}^t\mathbf{Y}$ since,

$$\mathbf{Y}^t\mathbf{YV} = (\mathbf{VD}^t\mathbf{U}^t)(\mathbf{UDV}^t)\mathbf{V} = \mathbf{VD}^t(\mathbf{U}^t\mathbf{U})\mathbf{D}(\mathbf{V}^t\mathbf{V}) = \mathbf{VD}^t\mathbf{D} = \mathbf{V}\Lambda \tag{5.18}$$

Thus, the content parameters can be estimated using the eigen decompostion of the matrix. The style parameters are related to the matrix $\mathbf{Y}$ in a more direct manner as

$$\mathbf{YV} = \mathbf{UDV}^t\mathbf{V} = \mathbf{UD} \tag{5.19}$$

Although $\mathbf{V}$ can be computed, $\mathbf{Y}$ cannot be and therefore there is no direct access to the style parameters. However, the indirect form is still useful for the task of classification : Given known content rendered in a new style, assigning content labels to the observation.

5.4.4 Content classification in the feature space

The content classification problem can be described as follows : Given observations in known content and unknown style, label the observations with their content. Let the new set of observations belonging to the style $\tilde{s}$ be $\mathbf{y}_1^{\tilde{s}}, \mathbf{y}_2^{\tilde{s}}, \cdots, \mathbf{y}_N^{\tilde{s}}$. The mean observation in the feature space be $\tilde{\mathbf{y}}^{\tilde{s}}$, i.e,

$$\tilde{\mathbf{y}}^{\tilde{s}} = \sum_{i=1}^{N} \phi(\mathbf{y}_i^{\tilde{s}}) \tag{5.20}$$

Following the asymmetric model of interaction of style and content, for each new observation there exist an unknown style parameter matrix $\mathbf{A}^{\tilde{s}}$ and a known content parameter vector $\mathbf{b}^c$ such that

$$\tilde{\mathbf{y}}^{\tilde{s}} = \mathbf{A}^{\tilde{s}}\mathbf{b}^c \tag{5.21}$$

Assuming (as in the case of [23]) that the probability that the observation vector $\mathbf{y}$ belongs to thecontent class c follows a gaussian distribution with variance σ^2

$$p(\mathbf{y}|\tilde{s}, c) \propto exp(-||\mathbf{y} - \mathbf{A}^{\tilde{s}}\mathbf{b}^c||^2/\sigma^2) \tag{5.22}$$

the content parameter that maximizes this probability can be found, if the style parameters are known $\mathbf{A}^{\tilde{s}}$. Although the style parameters cannot be computed explicitly in the feature space, the probability can be computed using the fact that $\tilde{\mathbf{y}}^t\mathbf{b}^{c^t} = \mathbf{A}^{\tilde{s}}$

$$||\mathbf{y} - \mathbf{A}^{\tilde{s}}\mathbf{b}^c||^2 = \langle \mathbf{y}, \mathbf{y} \rangle - 2\mathbf{b}^{c^t}\mathbf{A}^{\tilde{s}^t}\mathbf{y} + \mathbf{b}^{c^t}\mathbf{A}^{\tilde{s}^t}\mathbf{A}^{\tilde{s}}\mathbf{b}^c \tag{5.23}$$

Both the terms $\mathbf{A}^{\tilde{s}^t}\mathbf{y}$ and $\mathbf{A}^{\tilde{s}^t}\mathbf{A}^{\tilde{s}}$ involve only inner products between elements in the feature spaces and can be computed using the kernel function alone :

$$\mathbf{A}^{\tilde{s}^t}\mathbf{y} = \mathbf{b}^{c^t}(\tilde{\mathbf{y}}^{\tilde{s}^t}\mathbf{y}) \tag{5.24}$$

$$\mathbf{A}^{\tilde{s}^t}\mathbf{A}^{\tilde{s}} = \mathbf{b}^{c^t}(\tilde{\mathbf{y}}^{\tilde{s}^t}\tilde{\mathbf{y}}^{\tilde{s}})\mathbf{b}^c \tag{5.25}$$

Using the above probability measure, the content parameters that maximize the probability $p(\mathbf{y}|\tilde{s}, c)$ across all observations and content classes is chosen as the content class of the new observation set. The use of kernels introduces nonlinearity in to the framework described originally in [23]. Further, the kernelized algorithm uses factorization of a matrix, like the primal algorithm, retaining the efficiency. The performance of the kernel variant (with nonlinear kernels) is superior to the linear variant as demonstrated by the results below.

5.4.5 Results

We performed experiments on typographic content rendered in different fonts. Each character image was rescaled to fixed dimension of 100×100 pixels. The observation vectors corresponding $\mathbf{y}^{sc}$ were formed by stacking all the pixels. The kernelized asymmetric bilinear model was used with several kernels using various values of σ for the gaussian probability distribution. Table 5.1 shows the best accuracy for five different content classes and 5 different styles with different kernels. For each style, the algorithm was trained with characters from the remaining four styles and the content parameters were used to classify the characters. The accuracy shown is the average accuracy across styles. It can be seen that the method performs better using non-linear kernels compared to the linear kernel. This demonstrates that the adding nonlinearity to the model via kernel functions allows it to adapt to a complex phenomenon better.

	'A'	'B'	'C'	'D'	'E'
$\langle \mathbf{x}, \mathbf{y} \rangle$	62.1%	58.6%	61.4%	59.3%	67.1%
$(1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$	64.2%	59.3%	63.2%	62.0%	67.8%
$(1 + \langle \mathbf{x}, \mathbf{y} \rangle)^3$	65.1%	61.0%	64.7%	63.1%	68.4%
$exp(-\frac{\|\|\mathbf{x}-\mathbf{y}\|\|^2}{\sigma^2}); \sigma = 1$	61.0%	55.3%	59.2%	58.7%	65.1%
$exp(-\frac{\|\|\mathbf{x}-\mathbf{y}\|\|^2}{\sigma^2}); \sigma = 0.1$	71.2%	64.6%	68.3%	63.7%	71.8%

Table 5.1: Results of the content classification on five character types with various kernels (kernel function shown in the leftmost column). The accuracy is the mean across five styles.

5.5 Summary and Conclusions

In this chapter, we investigate the complementary nature of kernel methods and factorization. The rich set of features that are generated by using kernel functions can be exploited only when the irrelevant information is discarded. Due to the inaccessibility of the feature space, this cannot be done directly. Factorization is the tool that enables such an analysis of features in the feature space. It aids in discovery of latent variables that help in various tasks such as explaining the structure in the data, discrimination between classes of data. A combination of these tools leads to powerful algorithms that are capable of detecting, learning and making use of complex patterns in input the data. Examples of existing kernel algorithms which use this paradigm, directly or indirectly, are reviewed. We also demonstrate the power of these tools by kernelizing a bilinear model of interaction between style and content. The model is used to classify typographic content rendered in unknown fonts. The results demonstrate that the kernel variant is superior to the linear variant.

Kernel function and factorization form a powerful set of tools to perform a number of tasks such as feature extraction and selection, modeling and reducing computational complexity. The expressive power, statistical and numerical stability of the resultant algorithms makes them ideal for dealing with complex patterns in data. Problems in Image and Video Analysis fall in to this category. Thus, the combination of kernelization and factorization is a promising solution to problems in this domain.

Chapter 6

Conclusions

6.1 Summary and Contributions

The domain of image and video analysis poses a number of challenging problems that require extensive analysis of data. Emulating perceptual systems which perform higher level tasks, such as recognition of objects from images, requires detection of complex regularities in the data. Such regularities are not directly evident in the data. Alternate representations of the data make it feasible to identify and learn such relationships. Often, modeling the generative processes that generate such data aid in discovering the relationships. This involves discovery of latent variables that characterize the source process and the manner in which they interact to generate the data. The motivation for the work in this thesis is the fact that Kernel functions and Factorization are efficient and elegant tools to perform these two orthogonal tasks. We use these tools to develop novel techniques to solve problems in image and video analysis.

The representation of nonlinearities using kernel function leads to algorithms that are capable of detecting and learning complex nonlinear relationships in the input. At the same time, computational efficiency and statistical stability of the algorithms is not sacrificed for the expressive power. Two new kernel-based algorithms are proposed in the thesis for two separate tasks : i) Kernel Multiple Exemplar Biased Discriminant Analysis for selection of most discriminative features from a pool of derived features (nonlinear in the input features) and ii) Kernel Linear Predictive Coding for nonlinear modeling of time series. These methods have been used to solve important problems in image analysis. The nonlinear features generated by the kernels and the feature selection scheme that picks the most discriminative ones among them, together, enhance the performance of hand-geometry based biometric authentication system. Similarly, the nonlinear modeling technique increases the accuracy of model-based recognition methods used for recognition of objects (from their silhouettes) and handwritten characters (from the stroke information).

Analysis of generative processes using factorization helps in performing tasks such as compression, generation of synthetic data with properties of interest, model-based recognition of observed data etc. We use tensor representation of video data and factorization of tensors to perform challenging tasks in face video analysis. The use of tensor factorization enables appearance based solutions to expression transfer, expression recognition and morphing. The methods not only provide simple and efficient alternatives to the existing solutions to these problems (which use richer information), but also are complementary to them.

Kernel Functions and Factorization perform orthogonal functions in the analysis of data. Thus, they form a complementary set of tools for detection and inference of complex regularities in the data. Existing kernel algorithms use a combination of these tools for a number of tasks such as

feature extraction, feature selection, modeling and classification. We investigate the complementary nature of the two methods and their use in popular kernel-based algorithms. We also demonstrate the utility of these tools by applying a combination of them to kernelize an asymmetric bilinear factor model used to model the interaction of style and content. We use the kernelize model to classify typographical content (using the images of characters).

In summary, we use representation of nonlinearities via kernel functions and factor analysis using factorization for solving problems in image and video analysis. Effective manipulation of dimensionality of representation of visual data using these techniques leads to simple and efficient algorithms to solve various challenging problems. The applications considered, in this thesis, are of practical importance and the results obtained demonstrate that kernelization and factorization are promising tools for analysis of visual data.

6.2 Future Work

While kernelization and factorization are powerful tools for data analysis, the unsolved challenges in these domains have significant implications in practice. The choice of kernel function for a given task is widely researched issue. Theoretical and empirical tools to choose or design a kernel function optimal for a given task would greatly enhance the utility of kernel methods. Further, very few kernel functions are specifically designed for analysis of visual data. Design of new kernel functions that incorporate domain specific knowledge, such as geometric invariances, will improve performance of kernel methods on visual data. Further, the use of kernel methods in computer vision has been chiefly with the traditional feature vector representations. Using a richer representation such as the tensor representation of image collections and designing kernels for such representations is a promising direction for further research. In addition, selection of optimal kernel function for a given task is the most actively pursued research issue with many practical implications. Recent developments point towards existence of tractable solutions to this problem [95, 44] which makes it a very promising area of research.

Tensor representation and multilinear techniques offer new insights into video analysis. Factorization of tensors separates the dynamic and appearance aspects of the video. Such a separation is a valuable cue for analysis of dynamic events in the video. The use of kernel functions and tensorial methods together is an unexplored area. Factorization of the kernel matrix extracts useful information in the feature space. Similarly, factorization of a tensor composed of multiple kernel matrices can result in useful information spanning multiple feature spaces corresponding to the different kernel matrices. The joint use of kernel functions and factorization still remains largely unexplored and our investigations indicate that the methods can be used for many more useful tasks. Thus, the use of kernel functions, tensor representation and factorization for image and video understanding is a promising research direction that brings advances in multiple fields to solve problems of practical importance.

Appendix A

Related Publications

- S. Manikandan, Ranjeeth Kumar and C. V. Jawahar
  Tensorial Factorization Methods for Manipulation of Face Videos
  The 3rd International Conference on Visual Information Engineering (VIE)
  pp. 476-481, September 2006, Bangalore, India.

- Ranjeeth Kumar, S. Manikandan and C. V. Jawahar
  Task Specific Factors for Video Characterization
  The 5th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)
  LNCS 4338 pp. 376-387, December 2006, Madurai, India.

- Ranjeeth Kumar and C. V. Jawahar
  Class-Specific Kernel Selection for Verification Problems
  The 6th International Conference on Advances in Pattern Recognition (ICAPR)
  January 2007, Kolkata, India.

- Ranjeeth Kumar and C. V. Jawahar
  Kernel Approach to Autoregressive Modeling
  The 13th National Conference on Communications (NCC)
  January 2007, Kanpur, India.

- (*submitted*)
  Ranjeeth Kumar, S. Manikandan and C. V. Jawahar
  Face Video Alteration Using Tensorial Methods
  Pattern Recognition, Journal of Pattern Recognition Society

Appendix B

Notation

The notation used in the thesis follows that of popular books on kernel methods [92, 1]. Vectors are denoted using bold symbols. The kernel function is denoted using $\kappa(,)$ and the corresponding feature embedding using $\phi(.)$. The dual coefficients in feature space are denoted using $\boldsymbol{\alpha}$. $\mathbf{K}$ denotes the kernel matrix and subscripted or superscripted $\mathbf{K}$ denotes the quantities related to the kernel matrix. Data matrices are usually indicated using the letter $\mathbf{X}$. $\mathbf{x}$ denotes a single data sample and y its label. Factors of a matrix resulting from methods such as SVD, QR decomposition etc. are denoted using the standard symbols such as $\mathbf{U}, \mathbf{D}, \mathbf{V}, \Lambda, \mathbf{Q}, \mathbf{R}$. The symbols $\otimes$ denotes the outer product of vectors. The thesis uses indicator and selection matrices such as $\mathbf{I}_+$, which select columns of data matrix satisfying certain properties, extensively in the derivation. Vectors and Matrices whose entries are equal to a constant element a are denoted by $\mathbf{1}_a$, used to represent summations using matrix multiplications. Although, this might appear to be introducing needless additional symbols in to the equations, it greatly simplifies the kernelization procedure by making the presence of kernel matrix or its derivatives self-evident.

Bibliography

[1] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[2] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[3] B. Scholkopf, A. Smola, and K. uller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[4] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller, "Fisher discriminant analysis with kernels," in *Proceedings of IEEE Neural Networks for Signal Processing*, pp. 41–48, 1999.

[5] Carlo Tomasi and Takeo Kanade, "Shape and motion without depth," in *Proc. of the Third IEEE International Conf. on Computer Vision*, pp. 91–95, 1990.

[6] I. T. Jolliffe, *Principal Component Analysis*. Springer, 1986.

[7] M. Alex, O. Vasilescu, and Demetri Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces.," in *Proc. of the Seventh European Conf. on Computer Vision*, vol. 1, pp. 447–460, 2002.

[8] D. Vlasic, M. Brand, H. Pfister, and J. Popovic, "Face transfer with multilinear models," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 426–433, 2005.

[9] Amnon Shashua and Tamir Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," *Proc. of the International Conf. on Machine Learning*, pp. 792–799, 2005.

[10] Tamir Hazan, Simon Polak, and Amnon Shashua, "Sparse image coding using a 3d non-negative tensor factorization," *Proc. of the Tenth IEEE International Conf. on Computer Vision*, pp. 50–57, 2005.

[11] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[13] J. Quinlan, "C4.5 programs for machine learning," 1993.

[14] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[15] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[16] A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, (New York, NY, USA), pp. 144–152, ACM Press, 1992.

[18] S.-W. Lee and A. Verri, eds., *Pattern Recognition with Support Vector Machines, First International Workshop, SVM 2002, Niagara Falls, Canada, August 10, 2002, Proceedings*, vol. 2388 of *Lecture Notes in Computer Science*, Springer, 2002.

[19] M. Pontil and A. Verri, "Support vector machines for 3d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 637–646, 1998.

[20] G. Guo, S. Li, and K. Chan, "Face recognition by support vector machines," in *Proc. of the International Conferences on Automatic Face and Gesture Recognition*, pp. 196–201, 2000.

[21] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2003.

[22] Z. Ghahramani and G. E. Hinton, "Hierarchical non-linear factor analysis and topographic maps," in *Proc. Conf. Advances in Neural Information Processing Systems, NIPS* (M. I. Jordan, M. J. Kearns, and S. A. Solla, eds.), vol. 10, MIT Press, 1997.

[23] Joshua B. Tenenbaum and William T. Freeman, "Separating style and content with bilinear models," *Neural Computation*, vol. 12, pp. 1247–1283, 2000.

[24] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[25] De Lathauwer L., De Moor B., and Vandewalle J., "A multilinear singular value decomposition," *Matrix Analysis and Applications*, vol. 21, pp. 1253–1278, 2000.

[26] Max Welling and Markus Weber, "Positive tensor factorization," *Pattern Recognition Letters*, vol. 22, pp. 1255–1261, 2001.

[27] Hongcheng Wang and Narendra Ahuja, "Facial expression decomposition," in *Proc. of the Ninth IEEE International Conf. on Computer Vision*, pp. 958–965, 2003.

[28] D. Hilbert, "Grundzge einer allgemeinen theorie der linearen integralgleichungen," in *Nachrichten von der Konigl. Gesellschaft der Wissenschaften zu Gottingen, Mathematisch-physikalische Klasse*, pp. 49–91, 1904(1).

[29] J. Mercer, "Functions of positive and negative type, and their connection with the theory of integral equations," *Philosophical Transactions of the Royal Society, London*, vol. 209, pp. 415–446, 1909.

[30] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," tech. rep., Cambridge, MA, USA, 1989.

[31] G. Wahba, "Spline models for observational data," in *CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia*, vol. 59, 1990.

[32] S. Saitoh, *Theory of reproducing kernels and its applications*. Longman Scientific and Technical, Essex, England, 1988.

[33] J. Wang, J. Lee, and C. Zhang, "Kernel trick embedded gaussian mixture model.," in *Algorithmic Learning Theory, 14th International Conference, ALT 2003, Sapporo, Japan, October 17-19, 2003, Proceedings*, vol. 2842, pp. 159–174, 2003.

[34] S. Avidan, "Support vector tracking.," *IEEE Transcations on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.

[35] M. Yang, N. Ahuja, and D. Kriegman in *Proceedings of the International Conference on Image Processing*, pp. 37–40, 2000.

[36] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, no. 1398, pp. 137–142, Springer Verlag, Heidelberg, DE, 1998.

[37] J. Wang, J. Lee, and C. Zhang, "Kernel gmm and its application to image binarization," in *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, (Washington, DC, USA), pp. 533–536, IEEE Computer Society, 2003.

[38] P. J. D. Pillo, "Biased discriminant analysis: Evaluation of the optimum probability of misclassification," *Communications in Statistics-Theory and Methods*, vol. A8, pp. 1447–1457, 1979.

[39] X. S. Zhou and T. S. Huang, "Small Sample Learning during Multimedia Retrieval using BiasMap," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11–17, 2001.

[40] L. Mei, G. Brunner, L. Setia, and H. Burkhardt, "Kernel Biased Discriminant Analysis using Histogram Intersection Kernel for Content-Based Image Retrieval," in *Sixth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'05)*, no. 3578, pp. 63–70, 2005.

[41] S. V. Vishwanathan, A. J. Smola, and R. Vidal, "Binet-cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes," *International Jouranl of Computer Vision*, vol. 73, no. 1, pp. 95–119, 2007.

[42] G. R. G. Lanckriet, N. Christianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," in *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, (San Francisco, CA, USA), pp. 323–330, Morgan Kaufmann Publishers Inc., 2002.

[43] T. Jebara, "Multi-task feature and kernel selection for svms," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, (New York, NY, USA), p. 55, ACM, 2004.

[44] S.-J. Kim, A. Magnani, and S. Boyd, "Optimal kernel selection in kernel fisher discriminant analysis," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, (New York, NY, USA), pp. 465–472, ACM, 2006.

[45] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems 11*, 1999.

[46] C. Watkins, "Dynamic alignment kernels," in *Advances in Large Margin Classifiers* (A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, eds.), (Cambridge, MA), pp. 39–50, MIT Press, 2000.

[47] E. C.Leslie and W.S.Noble, "The spectrum kernel: a string kernel for protein classification," in *Pacific symposium on biocomputing*, 2002.

[48] D. Haussler, "Convolution kernels on discrete structures," tech. rep., 1999.

[49] R. I. Kondor and J. D. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, (San Francisco, CA, USA), pp. 315–322, Morgan Kaufmann Publishers Inc., 2002.

[50] J.-P. Vert, "A tree kernel to analyze phylogenetic profiles," *Bioinformatics*, vol. 18, pp. S276–S284, 2002.

[51] E. Takimoto and M. K. Warmuth, "Path kernels and multiplicative updates," *Jornal of Machine Learning Research*, vol. 4, pp. 773–818, 2003.

[52] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *Jornal of Machine Learning Research*, vol. 5, pp. 819–844, 2004.

[53] A. Barla, F. Odone, and A. Verri, "Hausdorff kernel for 3d object acquisition and detection," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, (London, UK), pp. 20–33, Springer-Verlag, 2002.

[54] S. Boughorbel, J.-P. Tarel, and N. Boujemaa, "Generalized histogram intersection kernel for image recognition," in *Proceedings of IEEE International Conference on Image Processing (ICIP'05)*, vol. III, (Genova, Italy), pp. 161 – 164, 2005.

[55] C. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 206–218, March 1997.

[56] B. Triggs, "Factorization methods for projective structure and motion," in *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, (Washington, DC, USA), p. 845, IEEE Computer Society, 1996.

[57] J. Costeira and T. Kanade, "A multibody factorization method for independently moving-objects," *International Journal of Computer Vision*, vol. 29, pp. 159–179, September 1998.

[58] A. Heyden, R. Berthilsson, and G. Sparr, "An iterative factorization method for projective structure and motion from image sequences," *Image and Vision Computing*, vol. 17, pp. 981–991, November 1999.

[59] W. Tang and Y. Hung, "A subspace method for projective reconstruction from multiple images with missing data," *Image and Vision Computing*, vol. 24, pp. 515–524, May 2006.

[60] R. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[61] Matthew Turk and Alex Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neurosccience*, vol. 3(1), pp. 71–86, 1991.

[62] Peter N. Belhumeur, Joao Hespanha, and David J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[63] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[64] A. Shashua and A. Levin, "Linear image coding for regression and classification using the tensor-rank principle," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 42–49, 2001.

[65] G. McLachlan, "Mixtures of factor analyzers," in *Proc. 17th International Conf. on Machine Learning*, pp. 599–606, Morgan Kaufmann, San Francisco, CA, 2000.

[66] M. Tipping and C. Bishop, "Probabilistic principal component analysis," tech. rep., 1997.

[67] K. Alahari and C. Jawahar, "Dynamic events as mixtures of spatial and temporal features," in *Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 540–551, 2006.

[68] A. K. Jain and D. Maltoni, *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[69] J. Daugman, "The importance of being random: statistical principles of iris recognition," *Pattern Recognition*, vol. 36, pp. 279–291, February 2003.

[70] Stan Z. Li and Anil K. Jain, eds., *Handbook of Face Recognition*. New York, USA: Springer, 2005.

[71] S. K. Zhou and R. Chellappa., "Multiple-Exemplar Discriminant Analysis for Face Recognition.," in *Internationl Conference on Pattern Recognition*, vol. 4, pp. 191–194, 2004.

[72] D. L. Swets and J. Weng, "Hierarchical Discriminant Analysis for Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 386–401, 1999.

[73] L. Gupta and M. D. Srinath, "Invariant planar shape recognition using dynamic alignment," *Pattern Recognition*, vol. 21, no. 3, pp. 235–239, 1988.

[74] S. Kuthirummal, C. V. Jawahar, and P. J. Narayanan, "Planar shape recognition across multiple views," in *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 1*, (Washington, DC, USA), p. 10456, IEEE Computer Society, 2002.

[75] A. Z. Wu and I. C. Jou and S. Y. Lee, "On-Line Signature Verification using LPC Cepstrum and Neural Networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 27, no. 1, pp. 148–153, 1997.

[76] C. Tappert, C. Suen, and T. Wakahara, "On-line handwriting recognition: A survey," in *International Conference on Pattern Recognition*, pp. II: 1123–1132, 1988.

[77] Kartikeyan, B. and Sarkar, A., "Shape Description by Time Series," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 9, pp. 977–984, 1989.

[78] I. Sekita, T. Kurita, and N. Otsu, "Complex autoregressive model for shape recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 4, pp. 489–496, 1992.

[79] Jun-yong Noh and Ulrich Neumann, "Expression cloning," in *Proc. of the Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 277–288, 2001.

[80] Steven M. Seitz and Charles R. Dyer, "View morphing," in *Proc. of the Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 21–30, 1996.

[81] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, "Synthesizing realistic facial expressions from photographs," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 75–84, ACM Press, 1998.

[82] Du Y. and Lin X., "Mapping emotional status to facial expressions," in *Proc. of the International Conf. on Pattern Recognition*, vol. II, pp. 524–527, 2002.

[83] Nicu Sebe, Michael S. Lew, Ira Cohen, Ashutosh Garg, and Thomas S. Huang, "Emotion recognition using a cauchy naive bayes classifier," in *Proc. of the 16th International Conf. on Pattern Recognition*, vol. 1, pp. 17–20, 2002.

[84] Irfan A. Essa and Alex Pentland, "Facial expression recognition using a dynamic model and motion energy," in *ICCV*, pp. 360–367, 1995.

[85] Fabrice Bourel, Claude C. Chibelushi, and Adrian A. Low, "Robust facial expression recognition using a state-based model of spatially-localised facial dynamics.," in *5th IEEE International Conference on Automatic Face and Gesture Recognition (2002)*, pp. 113–118, 2002.

[86] Ira Cohen, Ashutosh Garg, and Thomas Huang, "Emotion recognition from facial expressions using multilevel hmm," in *NIPS Workshop on Affective Computing*, 2000.

[87] Martin Bichsel, "Automatic interpolation and recognition of face images by morphing," *Proc. of the Second International Conf. on Automatic Face and Gesture Recognition*, pp. 128–135, 1996.

[88] P. J. Benson, "Morph transformation of the facial image.," *Image Vision Computing*, vol. 12, no. 10, pp. 691–696, 1994.

[89] A. Sun, E. Lim, and W. Ng, "Web classification using support vector machine," in *Proceedings of the fourth international workshop on Web information and data management*, pp. 96–99, 2002.

[90] J. Makhoul, "Linear Prediction : A Tutorial Review," in *Proc. IEEE, 63(4)*, pp. 561–580, 1975.

[91] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, 1950.

[92] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA, USA: MIT Press, 2001.

[93] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, 2000.

[94] B. Scholkopf and A. J. Smola, *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[95] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming.," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[96] T. Graepel, "Kernel matrix completion by semidefinite programming.," in *Artificial Neural Networks - ICANN 2002, International Conference, Madrid, Spain, August 28-30, 2002, Proceedings*, vol. 2415, pp. 694–699, 2002.

[97] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, (New York, NY, USA), p. 106, ACM Press, 2004.

[98] C. J. Twining and C. J. Taylor, "Kernel principal component analysis and the construction of non-linear active shape models.," in *Proceedings of the British Machine Vision Conference 2001, BMVC 2001, Manchester, UK, 10-13 September 2001*, 2001.

[99] B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., *Advances in kernel methods: support vector learning*. Cambridge, MA, USA: MIT Press, 1999.

[100] C. J. C. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector machines," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, p. 375, The MIT Press, 1997.

[101] G. Wu, E. Chang, Y. K. Chen, and C. Hughes, "Incremental approximate matrix factorization for speeding up support vector machines," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 760–766, ACM Press, 2006.

[102] K. Fukunaga, *Introduction to statistical pattern recognition (2nd ed.)*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[103] J. Makhoul, "Linear Prediction : A Tutorial Review," in *Proc. IEEE, 63(4)*, pp. 561–580, 1975.

[104] Nuno Vasconcelos, "From Pixels to Semantic Spaces: Advances in Content-Based Image Retrieval," *Computer*, vol. 40, no. 7, pp. 20–26, 2007.

[105] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content–Based Image Retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 644–655, 1998. (Special Issue on Segmentation, Description, and Retrieval of Video Content).

[106] V. Roy and C. V. Jawahar, "Hand-Geometry Based Person Authentication Using Incremental Biased Discriminant Analysis," in *Proceedings of the National Conference on Communication(NCC 2006)*, pp. 261–265, 2006.

[107] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.

[108] C. S. Avilla and R. S. Rello, "Biometric identification through hand geometry measurements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1168–1171, 1168–1171.

[109] C. H. Park and H. Park, "Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition," *SIAM Journal of Matrix Analysis and Applications*, vol. 27, no. 1, pp. 87–102, 2005.

[110] J. W. B. S. A. S. S. Mika, G. Ratsch and K.-R. Muller, "Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 623–633, 2003.

[111] M. N. S. S. K. P. Kumar and C. V. Jawahar, "Configurable hybrid architectures for character recognition applications," in *International Conference on Document Analysis and Recognition*, pp. 1199–1205, 2005.

[112] J. K. Suykens and J. Vandewalle, *Nonlinear Modeling*. Springer, 1998.

[113] George A. F. Seber and C. J. Wild, *Nonlinear Regression*. Wiley-IEEE, 2003.

[114] Ronald Christensen, *Advanced Linear Modeling*. Springer, 2001.

[115] Vladimir Naumovich Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1999.

[116] Shantanu Chakrabartty and Yunbin Deng and Gert Cauwenberghs, "Robust Speech Feature Extraction by Growth Transformation in Reproducing Kernel Hilbert Space," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 133–136, 2004.

[117] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia, "Symmetry-based indexing of image databases," *Journal of Visual Communication and Image Representation*, vol. 9, pp. 366–380, December 1998.

[118] Alahari, K. and Putrevu, S.L. and Jawahar, C.V., "Discriminant substrokes for online handwriting recognition," in *International Conference on Document Analysis and Recognition*, vol. 1, pp. 499–503, 2005.

[119] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 399–458, 2003.

[120] R. Gross, J. Shi, and J. Cohn, "Quo vadis face recognition? - the current state of the art in face recognition," tech. rep., Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.

[121] Ming-Hsuan Yang and David J. Kriegman and Narendra Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, 2002.

[122] L. Torres and J. Vila , "Automatic face recognition for video indexing applications," *Pattern Recognition*, vol. 35, no. 3, pp. 615–625, 2001.

[123] http://www.biometrics.gov/docs/facerec.pdf.

[124] Everingham, M., Sivic, J., and Zisserman, A., "Hello! my name is... buffy – automatic naming of characters in tv video," in *Proceedings of the British Machine Vision Conference*, vol. 3, pp. 899–908, 2006.

[125] Paul A. Viola and Michael J. Jones, "Robust real-time face detection.," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[126] G. Wolberg, "Recent advances in image morphing," in *CGI '96: Proceedings of the 1996 Conference on Computer Graphics International*, p. 64, 1996.

[127] T. Ezzat and T. Poggio, "Miketalk: A talking facial display based on morphing visemes," in *IEEE Computer Animation*, pp. 96–102, 1998.

[128] A. Lanitis, C. J. Taylor, and T. F. Cootes, "Toward automatic simulation of aging effects on face images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 442–455, 2002.

[129] Manikandan, S., Ranjeeth Kumar, and C. V. Jawahar, "Tensorial factorization methods for manipulation of face videos," in *Proc. of the 3rd International Conference on Visual Information Engineering*, (Bangalore, India), pp. 476–481, September 2006.

[130] Ranjeeth Kumar, Manikandan, S., and C. V. Jawahar, "Task specific factors for video characterization," in *Proc. of the 5th Indian Conference on Computer Vision, Graphics and Image Processing*, (Madurai, India), pp. 376–387, December 2006.

[131] B. Fasel and J. Luettin, "Automatic facial expression analysis: A survey," *Pattern Recognition*, vol. 36, pp. 259–275, 2003.

[132] Zicheng Liu, Ying Shan, and Zhengyou Zhang, "Expressive expression mapping with ratio images," in *Proc. of the 28th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 271–276, 2001.

[133] P. Ekman and W. V. Friesen, *Facial Action Coding System : Investigator's Guide*. Consulting Pshycologists Press, Palo Alto, CA, 1978.

[134] G. Wolberg, "Image morphing: a survey," *The Visual Computer*, vol. 14, no. 8/9, pp. 360–372, 1998.

[135] http://www.ilm.com/awards-tech.html.

[136] Zoubin Ghahramani and Geoffrey E. Hinton, "The EM algorithm for mixtures of factor analyzers," Tech. Rep. CRG-TR-96-1, 21 1996.

[137] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[138] J. Kruskal, "Three way arrays: Rank and uniqueness of trinlinear decomposition with applications to arithmetic complexity and statistics," *Linear Algebra and its Applications*, vol. 18, pp. 95–138, 1977.

[139] John Platt, Nello Cristianini, and John Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems 12* (S. Solla, T. Leen, and K.-R. Mueller, eds.), pp. 547–553, 2000.

[140] T. Xiong, J. Ye, Q. Li, R. Janardan, and V. Cherkassky, "Efficient kernel discriminant analysis via qr decomposition," in *Neural Information Processing Systems*, 2004.

[141] D. Zhang, Z.-H. Zhou, and S. Chen, "Non-negative matrix factorization on kernels," in *Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence (PRICAI'06)*, p. LNAI 4099, 2006.

[142] A. M. Elgammal and C.-S. Lee, "Separating style and content on a nonlinear manifold," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 478–485, 2004.

[143] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Multifactor gaussian process models for style-content separation," in *ICML '07: Proceedings of the 24th international conference on Machine learning*, (New York, NY, USA), pp. 975–982, ACM, 2007.

[144] M.-H. Yang, "Face recognition using kernel methods," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS], Vancouver, Canada*, pp. 1457–1464, 2001.

[145] A. N. Rajagopalan, P. Burlina, and R. Chellappa, "Higher order statistical learning for vehicle detection in images," in *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, (Washington, DC, USA), p. 1204, IEEE Computer Society, 1999.

[146] R. I. Kondor and T. Jebara, "A kernel between sets of vectors," in *International Conference on Machine Learning*, pp. 361–368, 2003.

[147] G. Wu, E. Chang, Y. K. Chen, and C. Hughes, "Incremental approximate matrix factorization for speeding up support vector machines," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 760–766, ACM, 2006.