ON LEARNING AND OPTIMIZATION OF THE KERNEL FUNCTIONS IN SCARCITY OF LABELED DATA

M. EHSAN ABBASNEJAD

UNIVERSITI SAINS MALAYSIA

2010

ON LEARNING AND OPTIMIZATION OF THE KERNEL FUNCTIONS IN SCARCITY OF LABELED DATA

by

M. EHSAN ABBASNEJAD

Thesis submitted in fulfilment of the requirements
for the degree of
Master of Science

March 2010

ACKNOWLEDGEMENTS

Many thanks to Prof. Donald Knuth for giving us TEX, and Leslie Lamport for LATEX.

Dr. Dhanesh's effort in officially raising awareness about LATEX at USM during NaCSPC'05 was incredibly refreshing and much welcome. Dr. Dhanesh also followed this up by suggesting that I give an introductory workshop on LATEX during CSPC'07, and to LATEX the programme book cum abstract collections (a first!). Many, many thanks to him and Dr. Azman, who was *very* supportive during the last minute changes and glitches. Also, thanks to Nur Hussein, Adib, Seng Soon and Anusha for happily trying out early versions of my LATEX thesis templates, and for their feedbacks, and to everyone who attended my talk, requested tutorials, downloaded from my website (http://liantze.googlepages.com/latextypesetting), etc, etc.

Hope everyone graduates quickly then!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PLATES

Page

LIST OF ABBREVIATIONS

IPS Institut Pengajian Siswazah

PPSK Pusat Pengajian Sains Komputer

USM Universiti Sains Malaysia

UTMK Unit Terjemahan Melalui Komputer

LIST OF SYMBOLS

$\lim$ limit

θ angle in radians

PENULISAN TESIS DENGAN LATEX

ABSTRAK

Ini merupakan abstrak Melayu untuk tesis USM. Ianya disediakan dengan sistem penyediaan dokumen LaTeX.

ON LEARNING AND OPTIMIZATION OF THE KERNEL FUNCTIONS IN SCARCITY OF LABELED DATA

ABSTRACT

This is the English abstract of a USM thesis. It was prepared with the LaTeX document typesetting system.

CHAPTER 1

INTRODUCTION

This chapter describes the motivation and reasons for the growing interest in methods for learning from data and introduces informally some relevant terminology. Additionally, the problem details and contributions made in this thesis is presented.

1.1 Notation

Throughout this thesis I will use the following notations: vectors will be represented as lowercase letters x and may also be indexed x_i. Sets are represented by calligraphic uppercase letters $\mathcal{X}$, $\mathcal{Y}$. The matrices are denoted by the uppercase letters like A. The symbols $\mathbb{R}$ and $\mathbb{R}^d$ denote the set of reals, and the d-dimensional real vector space respectively. Further, the small Greek alphabets are used to denote the real values unless otherwise stated. We use $\log$ to represent the natural logarithm.

1.2 Introduction to Machine Learning

Machine learning is an attempt to mimic human-beings ability to learn, predict and generalize the previous experiences. Computes are able to observe the events and with the help of machine learning we are hoping to use the observations to predict the future ones. The observations are used to formulate a hypothesis about a specific incident. Along with the developments made in machine learning computers showed a significant level of learning ability. Currently, the most important applications of machine learning are in computer vision (e.g. computers are trained

to recognize objects in a picture), information retrieval (e.g. identifying the titles related to one specific news artile), natural language processing (e.g. computers are taught to recognize part of speech in a document) and many new applications emerging like finding genes in DNA sequences, mining the social networks and etc.

Machine learning algorithms (or learning algorithms in short) are using the observations–or *training examples (samples)* in the machine learning parlance–as previous experience to learn to improve their behavior in future. The process of using the data to learn the prediction model is called *training* phase. In the training phase, it is believed that the nature of the underlying problem can be learnt by investigating the instances of data. However, in many cases data has misleading instances that cannot be trusted due to the imperfection in the current measurement devises. This is one of the reasons that opting a specific model for a problem at hand is highly challenging.

The training examples are commonly represented in the form of a list of all the values extracted from the training object called *feature vector*[1]. This representation paves the way to deal with data as points or vectors and perform well-established mathematical and physical analysis.

Machine learning algorithms are usually organized into several categories based on the type of training examples[2]

- **Supervised learning** the most traditional category of learning algorithms that deals with the *labeled data*. By labeled data we mean each entry in a dataset is recognized as being

[1] there are other representations like graphs or strings, but in this thesis our emphasis is on the commonly known vector representation.

[2] Apart from the presented categories, other type of machine learning algorithms like reinforcement learning is also considered which is beyond the scope of this thesis.

assigned to a specific class. In these types of problems, we are given a set

$$\mathcal{X} = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d \tag{1.1}$$

of n training examples each of them is assigned a label $\xi = \{\pm 1\}^n$. In the supervised case, we are interested in finding a function (or hypothesis) h as

$$R = \sum_{i=1}^{n} Q(h(x_i), \xi_i)), \quad h : \mathbb{R}^n \to \pm 1 \tag{1.2}$$

where Q is called the loss-function and assess the quality of the hypothesis h. In general, it is rational to find a hypothesis that minimizes the risk function. It should be noted that a function that set the risk function to zero is unlikely to be a good hypothesis as it will fit the training examples with a small probability to be able to predict unseen examples accurately–the problem commonly known as *over-fitting* in contradictory point with *under-fitting* where hypothesis is not good enough for the testing examples. One of the well-known examples of supervised algorithms in support vector machine (SVM) as we will discuss in subsequent chapters.

- **Unsupervised learning** in the case where no labeled data is available, the problem is known as unsupervised learning. Two common classes of unsupervised algorithms are dimensionality reduction and clustering. In dimensionality reduction one intends to find a space in which data represents with a lower dimensionality as its original space. In these algorithms, the goal is to preserve some aspects of data in the lower dimensions. A popular dimensionality reduction technique is known as kernel principle component analysis (kernel-PCA) which we will describe in the following chapters. In clustering the goal is to put the data with similar attributes in one category. As it is easily inferred the most important problem faced in clustering is that what aspects of these attributes should

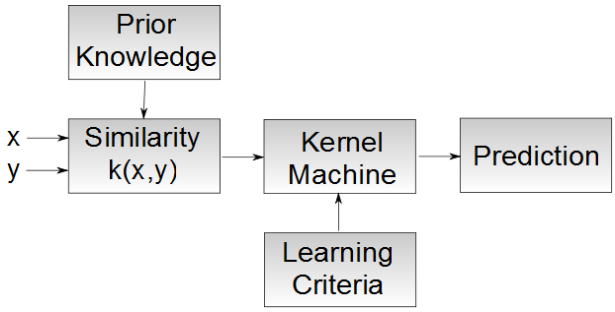

Figure 1.1: A general view of the kernel machines framework

be considered and how to assess the similarity between them. as an example consider classifying three birds ostrich, crow and swan in two categories, should we assign the same class to ostrich and crow as they commonly have similar colors or in contrast crow and swan belong to same categories as they both can fly.

- **Semi-supervised learning** a new class of learning algorithms that attempts to bridge the gap between supervised and unsupervised methods. In other words, with semi-supervised learning it is intended to use few labeled examples to assist build a more robust algorithms that can recognize unlabeled examples.

A large class of learning algorithms that has been studied lately in the community is called *kernel(-based) methods* or *kernel machines*. There are supervised, unsupervised and semi-supervised kernel-based algorithms. These algorithms are the potential solutions to many problems because of their lower error rate compared to other methods, relatively fast training time and elegant compatibility with high dimensional data. In short, these algorithms work based on the notation of *similarity* provided with the kernel functions.

The field of machine learning is closely related to the study in *optimization* as the final objective of the learning algorithms are typically formed as an optimization problem. Hence, the success of machine learning algorithms greatly depends on the performance of the computational aspects of the optimization problem. Additionally, there are sets of problems that

can be solved using the state of the art optimization techniques. Therefore, this is particularly important for the machine learning community to obtain the solution to their problems such that they are solvable.

1.3 Motivation and problem statement

Many of the algorithms in machine learning are dependent on an appropriate selection of a measure of similarity or dissimilarity (e.g. clustering we discussed earlier). In case of kernel methods, this measure is the kernel function. There have been various kernels[3] proposed which tend to exhibit diverse characteristics. Moreover, many of the well-known kernels have parameters to selectŮcalled hyper-parametersŮthat must be finely tuned. Choosing an appropriate kernel and possibly its hyper-parameters is a challenging task which needs a lot of experience as well as several hours of experiments to overcome.

↪ examples from the paper noting the problem

In this thesis, we are interesting in exploring possible answers for the following questions

1. Is it possible to learn the kernel function itself from the data and use the *optimal* kernel obtained in the kernel machine?
2. What if there are no labeled examples or the number of labeled examples is scarce, *i.e.* they are not sufficient to correctly predict as expected?

Although the naïve method of iterative training and testing may seem to be the easy answer to the cases where enough labeled examples are available, we will argue that in many cases the results produced are not well enough. In order to answer the abovementioned questions,

[3] we use 'kernel' to refer to the kernel function

we believe the structure of the dataset should be investigated. Intuitively, it means we have to consider a group to be able to recognize the relationship between a pair in it. However this solution poses a new question

3. How is it possible to extract the intrinsic structure of a dataset?

Motivated by the abovementioned questions, we propose and investigate to methods to learn the kernel function from a dataset.

The possible solution to the abovementioned questions, are possible solution to the following problems too:

1. **Transfer learning and multi-task learning** as two very closely related fields of research seek to use the assistance of other data sources to formulate a more accurate learning algorithm. The optimal kernel learned from a dataset can be transferred to the other dataset to enhance the prediction accuracy.

2. **Distance learning** is very closely related to the objective of this thesis. As we will note in subsequent chapters, distance metrics can be obtained from a kernel function, therefore finding a suitable kernel function solves the problem for algorithms that require a distance metric to operate.

3. **New family of kernel-based learning algorithms** can be devised if an algorithm can compute the data-dependent kernel. In this case, the only thing a user required to do is to select few parameters and the algorithm can work with a higher accuracy compared to the contemporary algorithms.

1.4 Objectives

In this thesis our objectives are as follows

1. The notation of kernel functions are recently being used in the machine learning community and the study of different aspects of kernel functions is of a great use. Furthermore, the use of these functions in two of the popular kernel-based algorithms is investigated.

2. The supervised kernel-based methods, specifically SVM, found a variety of applications in the real-world problems. However, supervised algorithms are producing acceptable results only if the number of labeled examples is sufficient[4]. Additionally, the choice of the kernel function is of a great importance in quarantining the efficiency of supervised kernel-based algorithms. We will investigate the case of insufficient labeled data in training SVM and the influence of optimizing kernels in the accuracy of the predictions. In order to improve the leaning process, in which the kernel is optimized, we *transfer* the structure of unlabeled examples from a similar problem.

3. In many of the current methods to learn the kernel, the algorithm is specifically designed to operate in a supervised manner. Thus, kernel selection in unsupervised case is still tedious. We attempt to overcome this problem in an unsupervised case. It makes the algorithm general purpose that can easily be applied to supervised cases. Furthermore, as there are no labeled examples, the structure of the unlabeled examples should be more closely scrutinized; hence, the method to capture the intrinsic characteristics of the dataset is even more challenging to design. In order to evaluate the results of the algorithm we pick two of the most popular kernel-based methods, a supervised and an unsupervised, to test the results of the dataset.

[4] as we will discuss specifically in the case of SVM, if the labeled examples are among the support vectors the accuracy of predictions are acceptable. However, in general this assumption cannot be made.

1.5 Scope

We will define the scope of the presented work as

1. The results of the proposed algorithms in this thesis are examined by the benchmark or synthetic datasets available openly for the researchers. Although all the benchmark datasets are created from real-world problems, but it is not intended to look into providing solution for a specific problem in this thesis. Furthermore, in spite of reporting the short summery on the benchmark datasets, the quality in which the dataset has been produced is beyond the scope of this work.

2. Although the proposed approaches are generic in nature, the reported results are only obtained from the evaluation of the respective algorithm on the mentioned kernel-based algorithm.

3. The result of the second algorithm is obtained from solving a convex optimization problem. The convexity and some examples of the convex programs will be explained but the detailed description on the methods to solve any specific convex problem is beyond the scope of this thesis.

1.6 Overview of methodology

As mentioned earlier, we are proposing two methods of learning the kernel functions with special probe on the influence of the structure of the dataset. In cases where sufficient number of labeled examples is not available the structure of the dataset is the only hint on the criteria defined on the quality of the optimal kernel obtained.

In the first method, limited labeled examples are available which should be utilized. As it is shown in Fig. (1.2), the algorithm requires three inputs: $\mathcal{X}$ as the labeled training example,

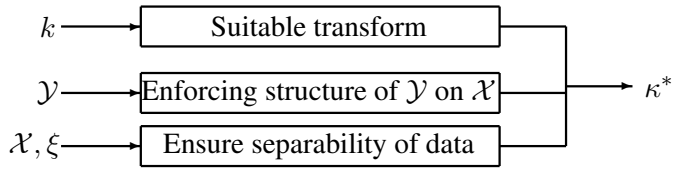

Figure 1.2: $\mathcal{X}$ denotes the set with labels in ξ and $\mathcal{Y}$ is set of unlabeled examples from the similar problem. Satisfaction of the criteria mentioned in this figure causes the optimal kernel κ^* to be obtained

$\mathcal{Y}$ as the training examples obtained from a similar problem and k as the initial kernel function. We are looking for a suitable transformation of k that satisfies the objective criteria defined on both $\mathcal{X}$ and $\mathcal{Y}$. There are two sets of criteria considered, firstly on the labeled examples that the similarity and the dissimilarity between them should be preserved (and often magnified as it will be discussed). Secondly, the structure of the dataset $\mathcal{X}$ represented by the transformed kernel κ^* should have the highest resemblance with the dataset $\mathcal{Y}$ obtained from the similar problem. This is because the number of labeled training examples are not sufficient to make an informed decision on them therefore an attempt to use the aid of the additional data is made. Ultimately, the algorithm is simplified as optimization problem that can be easily solved using available optimization techniques.

In the second approach as illustrated in Fig. (1.3), unlike the first one, we define a parametric kernel based on the set of initial kernel functions. The objective is to find the weightage of the parameters and consequently obtain the optimal kernel such that the similarity of each instance is best measured with respect to the influence values defined on the on the structure of the dataset. The process that we use to infer the structural aspects of the dataset is of a great importance. In the second approach unlike the first one the structural aspect is more closely examined as there are no labeled examples available. This approach is proposed in two settings which eventually form two optimization problems.

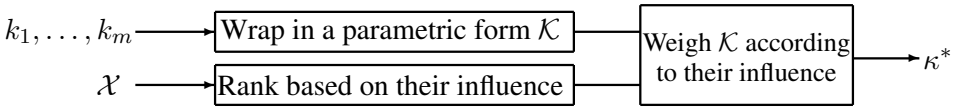

Figure 1.3: given the dataset $\mathcal{X}$ the optimal kernel function is selected from the prametric representation of the set of kernel functions.

1.7 Contributions

This thesis presents two approaches to lean the kernel. In order to learn the kernel two methods to capture the structure of the dataset is also considered. Finally, to solve the optimization problems obtained, convexity is examined and applied to the machine learning. Specifically the contributions of this thesis can be summarizes as

1. A supervised approach to learn the optimal kernel in cases where labeled data are scarce
2. An unsupervised approach which captures the intrinsic characteristics of the dataset and consequently learns the kernel using two convex optimization methods
3. The application of two well-known convex optimization techniques in machine learning and kernel methods

1.8 Outline of thesis

The organization of this thesis is as follows: chapter one (current chapter) provides an overview of the content and the directions for the rest of the thesis. Chapter 2 is dedicated to theoretical background on kernel functions, the kernel-based algorithms used and the convex optimization. In chapter 3 the existing related works on learning the kernel is presented. In chapter 4 the supervised approach and the experimental results are presented. Chapter 5 continues with the unsupervised method and the empirical evolutions. In the final chapter we will conclude the thesis and present the future directions of this research.

CHAPTER 2

THEORETICAL BACKGROUND

2.1 Kernels

Kernels have two important properties from computational point of view: (a) enable access to high-dimensional feature space at low computational cost both in time and space, (b) most of the algorithms could virtually be presented as convex optimization that does not suffer from local optima.

For an eigenvalue, eigenvector pair λ, v of matrix A, the *Rayleigh quotient* is defined as:

$$\frac{v^\top A v}{v^\top v} = \lambda \tag{2.1}$$

For an eigenvalue, eigenvector pair λ, v of matrix A, *deflation* is defined as the following transformation:

$$A \mapsto \tilde{A}, \quad \tilde{A} = A - \lambda v v^\top \tag{2.2}$$

Theorem 1 (Courant-Fisher) *If $A \in \mathbb{R}^{n \times n}$ is symmetric, then for $k = 1, \ldots, n$, the kth eigenvalue $\lambda_k(A)$ of matrix A satisfies*

$$\lambda_k(A) = \textit{maximize}_{dim(T)=k} \textit{minimize}_{0 \neq v \in T} \frac{v^\top A v}{v^\top v} = \textit{minimize}_{dim(T)=n-k+1} \textit{maximize}_{0 \neq v \in T} \frac{v^\top A v}{v^\top v} \tag{2.3}$$

with the extrema achieved by the corresponding eigenvector.

Matrix A is positive semi-definite if and only if $A = B^\top B$ for some real valued function B (which is not unique).

Matrix A is positive (semi-)definite if and only if all the principle minors are positive (semi-)definite. From this proposition, one can infer that the diagonal entries of matrix A must be non-negative.

If a matrix is considered as a transformation, the ratio of the volume of the image of any object to its pre-image is equal to the absolute value of the determinant.

A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ satisfies the finitely positive semi-definite property if it is a symmetric function for which the matrices formed by restriction to any finite subset of the space $\mathcal{X}$ are positive semi-definite.

Definition 1 (Hilbert Space) *A Hilbert space $\mathcal{H}$ is an inner product space with the additional properties that is separable and complete.*

Theorem 2 (kernel functions) *A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, which is either continues or has a finite domain, can be decomposed $k(x, y) = \langle \phi(x), \phi(y) \rangle$ into a feature map ϕ into a Hilbert space $\mathcal{H}$ applied to both its arguments followed by the evaluation of the inner product in $\mathcal{H}$ if and only if it satisfies the finitely positive semi-definite property.*

The feature space $\mathcal{H}$ can be written as a set of functions that we will use in the learning problem:

$$\mathcal{H} = \{\sum_{i=1}^{\ell} \alpha_i k(x_i, .) | \alpha_i \in \mathbb{R}, \ell \in \mathbb{N}\} \tag{2.4}$$

Each point in this feature space is in fact a function (referred to as a function space) which

is closed under multiplication by a scalar and addition of functions. Let f and g be two of the functions in this feature space $\mathcal{H}$ defined as:

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x), \;\; g(x) = \sum_{i=1}^{m} \beta_i k(y_i, x) \tag{2.5}$$

then the product of these functions is defined as:

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, y_j) = \sum_{i=1}^{n} \alpha_i g(x_i) = \sum_{i=1}^{m} \beta_i f(y_i) \tag{2.6}$$

It is easy to show that the inner product is real-valued, symmetric, bilinear and positive semi-definite. Replacing g in Eq. (2.6) with $k(x, .)$, we have:

$$\langle f, k(x, .) \rangle = \sum_{i}^{n} \alpha_i k(x_i, x) = f(x) \tag{2.7}$$

This fact is referred to as the reproducing property of kernels and the space of the positive semi-definite function k denoted by $\mathcal{H}_k$ is called the *Reproducing Kernel Hilbert Space (RKHS)*.

Theorem 3 (Mercer Theorem) *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^n$. Suppose k is a continuous symmetric function such that the integral operator $T_k : L_2(\mathcal{X}) \rightarrow L_2(\mathcal{X})$ is:*

$$(T_k f)(.) = \int_{\mathcal{X}} k(., x) f(x) dx, \tag{2.8}$$

is positive, that is

$$\int_{\mathcal{X} \times \mathcal{X}} k(x, y) f(x) f(y) dx dy \geq 0, \;\; \forall f \in L_2(\mathcal{X}). \tag{2.9}$$

Then we can expand $k(x, y)$ in a uniformly convergent series on $\mathcal{X} \times \mathcal{X}$ in terms of functions

ϕ_i, satisfies $\langle \phi_i, \phi_j \rangle = \delta_{ij}$:

$$K(x, y) = \sum_{i=1}^{\infty} \phi_i(x)\phi_i(y). \tag{2.10}$$

Furthermore, the series $\sum_{i=1}^{\infty} \|\phi_i\|^2_{L_2(\mathcal{X})}$ is convergent.

Provided a kernel matrix is positive semi-definite, there is no need to consider the feature space constructed from the feature map. In this case, kernel functions can be used as similarity measure rather than constructing the corresponding feature space. If normalized kernels are used this similarity measure can be used as the a priori probability of the inputs being in the same class minus the a priori probability of being in different classes. Therefore, a kernel function is integrated into a learning algorithm as an interface between data input and the learning algorithm.

It is now clear why kernel functions are the most essential component in the theory of kernel methods. As it is mentioned in any learning algorithm the sole information about input data is obtained through kernel functions. Therefore, kernel functions are the essential data structure in implementation of learning algorithms. This means, the data input is not constrained to be real-vectors; it can be strings, graphs, images or time series.

The kernel matrix can be decomposed as:

$$K = \sum_{i=1}^{m} \lambda_i v_i v_i^{\top} \tag{2.11}$$

where v_i are eigenvectors and λ_i are the corresponding eigenvalues. Extending this to the eigenfunctions of the underlying integral operator:

$$f(.) \mapsto \int_{\mathcal{X}} k(., x) f(x) dx \tag{2.12}$$

Therefore a kernel function can be thought of as a prior over the eigenfunctions of the kernel operator.

In a supervised learning with target values $\{-1, +1\}$ values y, matrix $H_{ij} = y_i y_j K_{ij}$ called Hessian which can be thought of as the Schur product (enrtywise multiplication) of matrix $yy^\top$ and K.

Frobenius inner product between two matrices with identical dimensions is defined as:

$$\langle M, N \rangle = M.N = \sum_{i,j=1}^{n} M_{ij} N_{ij} = tr(M^\top N) \tag{2.13}$$

with the corresponding norm known as *Frobenius norm*. Based on this definition the kernel alignment between two kernel matrices can be defined.

Definition 2 *The alignment $A(K_1, K_2)$ between two kernel matrices K_1 and K_2 is given by:*

$$A(K_1, K_2) = \frac{\langle K_1, K_2 \rangle}{\sqrt{\langle K_1, K_1 \rangle \langle K_2, K_2 \rangle}} \tag{2.14}$$

the alignment between a kernel K and a target y is simply defined as $A(K, yy^\top)$. This is because $yy^\top$ known as the ideal kernel defines the target similarity values for a specific dataset. For $y \in -1, +1^m$ produces:

$$A(K, yy^\top) = \frac{y^\top K y}{m \|K\|} \tag{2.15}$$

The alignment measure can be thought of as the cosine between two matrices viewed as m^2-dimensional vectors, it satisfies $-1 \leq A(K_1, K_2) \leq +1$. As K_1 and K_2 are positive semi-definite, the lower bound on alignment is in fact 0.

The alignment can be considered *Pearson correlation coefficient* between the random vari-

ables $K_1(x, y)$ and $K_2(x, y)$ generated with uniform distribution over the pairs (x_i, y_i). It is related to the distance between normalized kernel matrices in the Frobenius norm:

$$\left\| \frac{K_1}{\|K_1\|} - \frac{K_2}{\|K_2\|} \right\| = 2 - A(K_1, K_2) \tag{2.16}$$

2.1.1 operations on kernels

Kernel functions are closed under following operations allowing the new kernels to be introduced. Let k_1 and k_2 be kernels over $\mathcal{X} \times \mathcal{X}, \mathcal{X} \subseteq \mathbb{R}^n, a \in \mathbb{R}^+, f(.)$ a real-valued function on $\mathcal{X}$, $\phi :\mapsto \mathbb{R}^N$ with k_3 a kernel over $\mathbb{R}^N \times \mathbb{R}^N$, and B a symmetric positive semi-definite $n \times n$ matrix. In addition, assume $p(x)$ be a polynomial with positive coefficients. The followings are hold:

1. $k(x, y) = k_1(x, y) + k_2(x, y)$
2. $k(x, y) = ak_1(x, y)$
3. $k(x, y) = k_1(x, y)k_2(x, y)$
4. $k(x, y) = f(x)f(y)$
5. $k(x, y) = k_3(\phi(x), \phi(y))$
6. $k(x, y) = x^\top By$
7. $k(x, y) = p(k_1(x, y))$
8. $k(x, y) = \exp(k_1(x, y))$
9. $k(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$

It should be noted that by Taylor expansion of the exponential function is a polynomial with infinite degree:

$$\exp(x) = \sum_{i=0}^{\infty} \frac{1}{i!} x^i \tag{2.17}$$

Therefore, one can observe that the Gaussian kernel is in fact corresponds to a feature space with infinite dimensions.

2.2 Detecting Patterns

Definition 3 (Pattern Analysis) *A pattern analysis algorithm takes as input a finite set $\mathcal{D}$ of n data items generated i.i.d. according to a fixed distribution $\mathcal{D}$ and a confidence parameter $\delta \in (0, 1)$. Its output is either an indication that no patterns were detectable, or* pattern function f *that with probability $1 - \delta$ satisfies*

$$\mathbb{E}_{\mathcal{D}} f(x) \approx 0 \tag{2.18}$$

The value of the expectation is known as the generalization error *of the pattern function f.*

For example, assuming c is the true value to be estimated by $h(x)$, $f(x) = |c - h(x)|$ could be a possible candidate of a correct pattern function.

Theorem 4 (McDiarmid) *Let $x_1, x_2, \ldots, x_n$ be independent variables taking values in a set A, and assume that $f : A^n \to \mathbb{R}$ satisfies*

$$\sup_{x_1, x_2, \ldots, x_n,\, \hat{x_i} \in A} |f(x_1, \ldots, x_n) - f(x_1, \ldots, \hat{x_i}, x_{i+1}, \ldots, x_n)| \leq c_i,\ 1 \leq i \leq n. \tag{2.19}$$

Then for all $\epsilon > 0$*:*

$$P\{f(x_1, \ldots, x_n) - \mathbb{E}f(x_1, \ldots, x_n) \geq \epsilon\} \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right) \tag{2.20}$$

Using McDiarmid?s theorem one can find a bound on the accuracy of finding the center of projected data points in the feature space. Assuming $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, the expectation of the image of $\mathcal{X}$ in feature space is:

$$\mathbb{E}_x[\phi(x)] = \int_{\mathcal{X}} \phi(x) dP(x). \tag{2.21}$$

The center of mass of training examples is calculated as:

$$\phi_{\mathcal{X}} = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i). \tag{2.22}$$

The measure of accuracy in this case can therefore be calculated as:

$$h(\mathcal{X}) = \|\phi_X - \mathbb{E}_x[\phi(x)]\| \tag{2.23}$$

Applying McDiarmid?s theorem and denoting $\hat{\mathcal{X}}$ as the dataset with x_i replaced with $\hat{x_i}$, we have:

$$\begin{aligned} |h(\mathcal{X}) - h(\hat{\mathcal{X}})| &= |\|\phi_{\mathcal{X}} - \mathbb{E}_x[\phi(x)]\| - \|\phi_{\hat{\mathcal{X}}} - \mathbb{E}_x[\phi(x)]\|| \\ &\leq \|\phi_{\mathcal{X}} - \phi_{\hat{\mathcal{X}}}\| = \frac{1}{n}\|\phi(x_i) - \phi(\hat{x_i})\| \leq \frac{2R}{n}, \end{aligned} \tag{2.24}$$

where $R = \sup_{x \in \mathcal{X}} \|\phi(x)\|$. Hence, $c_i = \frac{2R}{n}$ then we obtain:

$$P\left\{h(\mathcal{X}) - \mathbb{E}_{\mathcal{X}}[h(\mathcal{X})] \geq \epsilon\right\} \leq \exp\left(-\frac{2n\epsilon^2}{4R^2}\right) \tag{2.25}$$

$\hookrightarrow$ it should be completed for multidimensional cases.

Definition 4 (Lipschitz Condition) *A loss function $A : \mathbb{R} \to [0, 1]$ is Lipschitz with constant L if it satisfies*

$$|A(a) - A(\dot{a})| \leq L|a - \dot{a}|, \forall a, \dot{a} \in \mathbb{R} \tag{2.26}$$

Definition 5 (Margin) *For a function $h : \mathcal{X} \to \mathbb{R}$,* margin *is defined on an example is defined as (x, y) to be $yh(x)$. The* functional margin *of the training set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, is defined as:*

$$m(\mathcal{D}, h) = \textit{minimize}_{1<i<n} y_i g(x_i). \tag{2.27}$$

Theorem 5 *Fix γ and let $\mathcal{F}$ be the class of functions mapping from $Z = X \times Y$ to $\mathbb{R}$ given by $f(x, y) = -yh(x)$, where h is a linear function in a kernel-defined feature space with norm at most 1. Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be drawn according to probability distribution $\mathcal{D}$ and fix $\delta \in (0, 1)$. Then with probability at least $1 - \delta$ over samples of size n we have:*

$$\begin{aligned} P_{\mathcal{D}}(y \neq \textit{sgn}(h(x))) &= \mathbb{E}_{\mathcal{D}}[H(-yh(x)] \\ &\leq \frac{1}{n\gamma} \sum_{i=1}^{n} \xi_i + \frac{4}{n\gamma} \sqrt{tr(K)} + 3\sqrt{\frac{\ln \frac{2}{\delta}}{2n}} \end{aligned} \tag{2.28}$$

where K is the kernel matrix for the training set, H is the Heaviside function that returns 1 for values greater than 0 and zero otherwise. In this equation, $\xi_i = \xi((x_i, y_i), \gamma, h)$ amounts the value function h fails to achieve desired margin γ for the example (x_i, y_i).

2.3 Algorithms in the Feature Space

2.3.1 Means and Distances

Given a finite set $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ of an input space $\mathcal{X}$, a kernel $k(x, y)$ and a feature map ϕ into a feature space $\mathcal{H}$. Therefore $\phi(\mathcal{D})$ is a subset of the inner product space $\mathcal{H}$.

Norm of feature vectors the norm of the feature vector in the feature space is given by:

$$\|\phi(x)\|_2 = \sqrt{\langle \phi(x), \phi(x) \rangle} = \sqrt{k(x, x)} \tag{2.29}$$

Note 1 *Using the norm of feature vectors we can obtain the* normalization *transformation of feature vectors in the feature space:*

$$\hat{\phi}(x) = \frac{\phi(x)}{\|\phi(x)\|} \tag{2.30}$$

Therefore for two data points x and y we have:

$$\hat{k}(x, y) = \left\langle \hat{\phi}(x), \hat{\phi}(y) \right\rangle = \frac{\langle \phi(x), \phi(y) \rangle}{\|\phi(x)\| \|\phi(y)\|} = \frac{k(x, y)}{\sqrt{k(x, x) k(y, y)}} \tag{2.31}$$

Distances between feature vectors a special case of the norm is the length of line connecting two points $\phi(x)$ and $\phi(y)$.

$$\|\phi(x) - \phi(y)\| = \langle \phi(x) - \phi(y), \phi(x) - \phi(y) \rangle = k(x, x) - 2k(x, y) + k(y, y). \tag{2.32}$$

Distances between centers of mass the center of mass is given as

$$\bar{\phi}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i). \tag{2.33}$$

Despite inaccessibility to direct feature maps, the center of mass in the feature space can be calculated as

$$\left\|\bar{\phi}(\mathcal{D})\right\|^2 = \langle\bar{\phi}(\mathcal{D}), \bar{\phi}(\mathcal{D})\rangle = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}k(x_i, x_j). \tag{2.34}$$

From this equation, one can easily observe that in case the center of mass is the origin in the feature space, the sum of entries in the kernel matrix is zero.

Note 2 *The center of mass $\bar{\phi}(\mathcal{D})$ of a set of points $\phi(\mathcal{D})$ solves the following optimization problem:*

$$\textit{minimize}_{\mu}\frac{1}{n}\sum_{i=1}^{n}\|\phi(x_i) - \mu\|^2 \tag{2.35}$$

This corresponds to the fact that center of mass is the point that has the smallest distance to the rest of points in the set

Centering Data a mapping function to the feature space can be found that the feature space formed is centered. In order to do that, the center of mass can be moved to the origin. Hence, the points in the feature space are measured with respect to the center of the mass:

$$\begin{aligned}
\kappa(x, y) &= \left\langle \phi(x) - \frac{1}{n}\sum_{i=1}^{n}\phi(x_i), \phi(y) - \frac{1}{n}\sum_{i=1}^{n}\phi(y_i)\right\rangle \\
&= k(x, y) - \frac{1}{n}\sum_{i=1}^{n}k(x, x_i) - \frac{1}{n}\sum_{i=1}^{n}k(y, x_i) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}k(x_i, x_j) \\
\therefore\ \mathcal{K} &= K - \frac{1}{n}\mathbf{1}\mathbf{1}^\top K - \frac{1}{n}K\mathbf{1}\mathbf{1}^\top + \frac{1}{n^2}(\mathbf{1}^\top K\mathbf{1})\mathbf{1}\mathbf{1}^\top
\end{aligned} \tag{2.36}$$

2.3.2 Gram-Schmidt orthonormalization

Given a sequence of linearly independent vectors the method creates the basis by orthonormalizing each vector to all of the earlier vectors. The first basis vector is given by

$$q_1 = \frac{\phi(x_1)}{\|\phi(x_1)\|} \tag{2.37}$$

The ith vector is then obtained by subtracting from $\phi(x_i)$ multiples of $q_1, \ldots, q_{i-1}$ in order to ensure it becomes orthogonal to each of them

$$\phi(x_i) \to \phi(x_i) - \sum_{j=1}^{i-1} \langle q_j, \phi(x_i) \rangle q_j = (I - Q_{i-1} Q_{i-1}^\top)\phi(x_i) \tag{2.38}$$

where Q_i is the matrix whose i columns are the first i vectors $q_1, \ldots, q_i$. If we let $\nu_i = \|(I - Q_{i-1}Q_{i-1}^\top)\phi(x_i)\|$, by normalizing the projecting we obtain $q_i = \nu_i^{-1}(I - Q_{i-1}Q_{i-1}^\top)\phi(x_i)$.

$\hookrightarrow$ It follows that . . .

2.3.3 Spread of Data

It is possible to induce about the spread of data through the *covariance* defined to be the expectation of the products of the variables assuming the mean of respective variable is zero, *i.e.*

$$\text{cov}(x, y) = \mathbb{E}_{xy}[xy] \tag{2.39}$$

In case we are working in the feature space, a set of mapped points in the feature space denoted by $\Phi = \{\phi(x_1), \ldots, \phi(x_n)\}$ is used to construct the covariance matrix Σ as follows:

$$\begin{aligned} \Sigma &= \left(\frac{1}{n} \sum_{i=1}^{n} \phi(x_i)_s \phi(x_i)_t \right)_{s,t=1}^{d} \\ \therefore \quad n\Sigma &= (\Phi^\top \Phi)_{s,t=1}^{d} \end{aligned} \tag{2.40}$$

If we consider a unit vector $v \in \mathbb{R}^d$, the norm of the projection of each point in the feature space on v is $\|P_v(\phi(x_i)\| = v^\top \phi(x_i)/(v^\top v) = v^\top \phi(x_i)$. Therefore we have:

$$\mu_v = \mathbb{E}[\|P_v(\Phi)\|] = \mathbb{E}[v^\top \Phi] = v^\top \mathbb{E}[\Phi] = 0 \tag{2.41}$$

This is because we assumed data is centered in the feature space. Consequently the variance of the norms of the projections can be calculated:

$$\sigma^2 = \mathbb{E}[(\|P_v(\Phi)\| - \mu_v)^2] = \frac{1}{n}\sum_{i=1}^{n} \|P_v(\phi(x_i))\|^2 = \frac{1}{n} v^\top \Phi^\top \Phi v \tag{2.42}$$

in which final result is obtained due to $\|P_v(\phi(x_i)\| = v^\top \phi(x_i)$ as discussed earlier. In cases where mean is not zero, we have:

$$\sigma^2 = \mathbb{E}[(\|P_v(\Phi)\| - \mu_v)^2] = \mathbb{E}[\|P_v(\phi(x_i))\|^2] - \mu_v^2 = \frac{1}{n} v^\top \Phi^\top \Phi v - (\frac{1}{n} v^\top \Phi^\top \mathbf{1})^2 \tag{2.43}$$

In order to compute the variance in the feature space, we select vector v as a linear combination of the training points, *i.e.*

$$v = \sum_{i=1}^{n} \alpha_i \phi(x_i) = \Phi^\top \alpha \tag{2.44}$$

Replacing the value of v in Eq.(2.43) we have:

$$\sigma^2 = \frac{1}{n} \alpha^\top \Phi \Phi^\top \Phi \Phi^\top \alpha - (\frac{1}{n} \alpha^\top \Phi \Phi^\top \mathbf{1})^2 = \frac{1}{n} \alpha^\top K^\top K \alpha - (\frac{1}{n} \alpha^\top K \mathbf{1})^2 \tag{2.45}$$

which is easily computable based on the kernel matrix K.

2.4 Pattern Analysis using eigen-decomposition

2.4.1 Principle Component Analysis

If we assume data is centered in the feature space, we can compute the variance of the projection in the direction of the normalized vector w as

$$\frac{1}{n}(P_w(\phi(x_i)))^2 = \mathbb{E}[w^\top \phi(x_i)\phi(x_i)^t opw] = w^\top \Sigma w \tag{2.46}$$

Therefore, in order to maximize the variance in the desired direction one can maximize the following criteria obtained from Raleigh quotient introduced earlier:

$$\text{maximize}_w \ \frac{w^\top \Sigma w}{w^\top w} \tag{2.47}$$

The solution for such an optimization is given by the eigenvector corresponding to the largest eigenvalue. Similarly a consequent direction where variance is maximized is computed by the eigenvectors corresponding to the sorted eigenvalues of the covariance matrix. $*$ Rescaling a matrix does not alter the eigenvectors, it just rescales the corresponding eigenvalues. In addition, by Courant-Fisher theorem we have:

$$\lambda_1(n\Sigma) = \lambda(\Phi^\top \Phi) = \text{maximize}_{\dim(T)=1}\text{minimize}_{0\neq u\in T}\frac{u^\top \Phi^\top \Phi u}{u^\top u} \tag{2.48}$$

$\hookrightarrow$ continue . . .

Having a set of data and project it into the space spanned by its eigenvectors. Taking the first $k < n$ eigenvectors sorted by their eigenvalues of the covariance matrix, the projection of data in the subspace is achieved that has the larger variance in the respective axes. The new coordinates are called *Principle Coordinates* and the algorithm is known as *Principle Component Analysis*(PCA) which is a famous dimensionality reduction algorithm. In cases where

eigenvalues beyond the kth are small, data has a little variance in these directions. Therefore, data can be projected in first k eigenvalues. The small eigenvalues then may be regarded as noise that is eliminated during this process.

The ordered eigenvalues of the covariance matrix corresponds to the directions of the variance of data and the eigenvalues equals to the variance of the data being captured in eigenvectors directions. This can be regarded as identifying pattern in training data.

Kernel PCA is the representation of the PCA algorithm in the kernel-defined feature space. In kernel-PCA, the mapped points in the feature space are used to compute the variance:

$$P_{u_j}(\phi(x)) \tag{2.49}$$

$\hookrightarrow$ to be completed from the paper: *Nonlinear Component Analysis as a kernel Eigenvalue problem* if not fine, go to page 449 of the book learning with kernels or the paper itself.

2.5 Pattern analysis using convex optimization

2.5.1 Support Vector Machines

If we have a set S of data, there exists a norm 1 linear function

$$h(x) = \langle w, \phi(x_i) \rangle + b \tag{2.50}$$

determined by the weight vector w and threshold b, then there exists $\gamma > 0$, such that

$$\xi_i = (\gamma - y_i g(x_i))_+ = 0 \quad 1 \leq i \leq n. \tag{2.51}$$

This implies that the training data is linearly separable by a hyperplane γ. As $\|w\| = 1$, the expression $\langle w, \phi(x_i)\rangle$ measures the length of the perpendicular projection of the point $\phi(x_i)$ onto the ray determined by w. the functional margin of a linear function with norm 1 as the *geometric margin* of the associated classifier. In this sense, the maximum margin classifier is found by maximizing the geometric margin. The classifier formed by this method is often referred to as *maximal margin hyperplane* or *hard margin support vector machine*. The solution of the following optimization is deciding the aforementioned hyperplane:

$$\begin{aligned} \text{minimize}_{w,b,\gamma} \quad & \gamma \\ \text{subject to} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq \gamma, \quad i = 1, \ldots, n \\ & \|w\| = 1 \end{aligned} \tag{2.52}$$

Although enlarging the hyperplane ensures the availability of the result but, the hyperplane found this way is not robust in the sense that adding a new training point can reduce the margin or even rendering training data non-separable.

1-norm soft margin support vector machines Taking the product of vectors corresponds to taking logical ?and? operation on the vector entries. For example, in a two-dimensional vector we can define the feature map as follows:

$$\phi : \mathcal{X} = \mathbb{R}^2 \mapsto \mathcal{H} = \mathbb{R}^3$$

$$([x]_1, [x]_2) \mapsto ([x]_1^2, [x]_2^2, [x]_1[x]_2) \tag{2.53}$$

In this case the dimensions of the space obtained is

$$\begin{pmatrix} d+N-1 \\ d \end{pmatrix} \tag{2.54}$$

The other terms used in the literature to refer to positive definite kernel are reproducing kernel, Mercer kernel, admissible kernel, Support Vector kernel, nonnegative definite kernel, and covariance function. Positive definiteness implies positivity on the diagonal and symmetry.

Proposition 1 (Cauchy-Schwarz Inequality for kernels) *If k is a positive definite kernel and $x_1, x_2 \in \mathcal{X}$, then*

$$|k(x_1, x_2)|^2 \leq k(x_1, x_1)k(x_2, x_2) \tag{2.55}$$

We denote $\mathbb{R}^{\mathcal{X}} : \{f : \mathcal{X} \mapsto \mathbb{R}\}$ as a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ to $\mathbb{R}$ as

$$\phi : \mathcal{X} \mapsto \mathbb{R}^{\mathcal{X}} \iff x \mapsto k(..x). \tag{2.56}$$

Then, $\phi(x)$ is the function that assigns the value $k(x, x')$ to $x' \in \mathcal{X}$, *i.e.* , $\phi(x)(.) = k(x, .)$. This function represents the similarity to all other points. The kernel also represents the feature space of the dot product associated with ϕ using the following steps:

- the image of ϕ is converted to the vector space
- defining the dot product by a positive definite bilinear form
- the dot product satisfies $k(x, x') = \langle \phi(x), \phi(x') \rangle$

In order to define the vector space, we take a linear combination of the following form:

$$f(.) = \sum_{i=1}^{m} \alpha_i k(., x_i). \tag{2.57}$$

Then the dot product of f and another function

$$g(.) = \sum_{j=1}^{n} \beta_j k(., x_j). \tag{2.58}$$

is defined as

$$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j k(x_i, x_j). \tag{2.59}$$

This dot product amounts to $k(x, x') = \langle \phi(x), \phi(x') \rangle$.

Proposition 2 (Kernel Trick) *Given an algorithm which is formulated in terms of a positive define kernel k, one can construct an alternative algorithm by replacing k by another positive definite kernel $\tilde{k}$.*

It has been mid 1990s that it became apparent that any algorithm that only depends on the dot product can be kernelized. On the other hand, the use of kernel has been extended beyond the limits of dot products to the general sets such as text and graphs.

2.5.2 Reproducing kernel Hilbert spaces

RKHS can be defined as a Hilbert space of functions f on set $\mathcal{X}$ such that evaluation functionals (the maps $f \mapsto f(x')$ where $x' \in \mathcal{X}$) are continuous. In this case the *Reisz* representation theorem, for each $x' \in \mathcal{X}$ there exists a unique function of x, called

$$f(x, x') = \langle f, k(., x') \rangle. \tag{2.60}$$

For each x, the Mercer theorem defines the feature maps as the eigenfunctions of the kernel function:

$$\phi : x \mapsto (\sqrt{\lambda_i}\psi_i(x))_{i=1,\ldots,\mathcal{H}_N} \tag{2.61}$$

Proposition 3 (Mercer Kernel Map) *If k is a kernel satisfying Mercer's theorem, we can construct a mapping ϕ into a space where k acts as a dot product, i.e. ,*

$$\langle \phi(x), \phi(x') \rangle = k(x, x') \tag{2.62}$$

for almost all $x, x' \in \mathcal{X}$.

Proposition 4 (Mercer kernels are positive definite) *Let $\mathcal{X} = [a, b]$ be a compact interval and let $k : [a, b] \times [a, b] \to \mathcal{C}$ be continues. Then k is a positive definite kernel if and only if*

$$\int_a^b \int_a^b k(x, x') f(x) f(x') dx dx' \geq 0 \tag{2.63}$$

for each continues function $f : \mathcal{X} \to \mathcal{C}$

$$f(x) = \sum_{i=1}^{\infty} \alpha_i k(x, x_i) = \sum_{i=1}^{\infty} \alpha_i \sum_{j=1}^{N_{\mathcal{H}}} \lambda_j \psi_j(x) \psi_j(x_i) \tag{2.64}$$

$\hookrightarrow$ complete Mercer part page(59)

Provided everything we do with the kernels reduces to the dot products, the RKHS and the Mercer's kernel is the same. For instance, if ϕ_1 and ϕ_2 are mappings to the $\mathcal{H}_1$ and $\mathcal{H}_2$ feature spaces associated with kernel k, *i.e.* ,

$$k(x, x') = \langle \phi_i(x), \phi_i(x') \rangle_{\mathcal{H}_i}, \quad i = 1, 2 \tag{2.65}$$

Then it will usually does *not* mean $\phi_1 = \phi_2$. However, we have $\langle \phi_1(x), \phi_1(x') \rangle_{\mathcal{H}_1} = \langle \phi_2(x), \phi_2(x') \rangle_{\mathcal{H}_2}$.

2.5.3 the shape of data in the feature space

By the Mercer's theorem one can think of the feature map as a map into a high or possibly infinite dimensional feature space. The range of the data in the feature space decreases as the dimensionality increase with a rate equal to the eigenvalue.

Proposition 5 (Continuity of the feature map) *If $\mathcal{X}$ is a topological space and k is a continues positive definite kernel function on $\mathcal{X} \times \mathcal{X}$, then there exists a Hilbert space $\mathcal{H}$ and a continues map $\phi : \mathcal{X} \to \mathcal{H}$ such that for all $x, x' \in \mathcal{X}$, we have $k(x, x') = \langle \phi(x), \phi(x') \rangle$.*

Proposition 6 (Empirical kernel map) *For a given set $\{z_1, z_2, \ldots, z_n\} \subset \mathcal{X}$, we call*

$$\phi_n : \mathbb{R}^N \to \mathbb{R}^n \; where \; x \mapsto k(., x)|_{\{z_1, z_2, \ldots, z_n\}} = (k(z_1, x), k(z_2, x), \ldots, k(z_n, x))^\top \quad (2.66)$$

the empirical kernel map.

For a given dataset it can be proved that for a kernel k such that the Gram matrix $K_{ij} = k(x_i, x_j)$ is positive definite, then it is possible to define a map ϕ into an m-dimensional feature space $\mathcal{H}$ such that

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \quad (2.67)$$

Conversely, having a mapping function ϕ into the feature space $\mathcal{H}$, the matrix $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$ is positive definite.

2.5.4 Regularization

The objective of a learning algorithm can be defined as finding a function f from a set of functions such that the empirical risk is minimized. The following regularized risk could be defined as one objective function whose minimization we seek:

$$R_{reg}[f] = R_{emp}[f] + \lambda\Omega[f] \tag{2.68}$$

In this equation, R_{emp} is the empirical risk function and $\Omega[f]$ is the regularization term which leads to a better conditioning of the problem, $\lambda > 0$ is the regularization parameter that controls the tradeoff between regularization term and the empirical risk. As an examples, maximum margin classifier can be rewritten as minimization of the following regularization function, specifying $\frac{1}{2}\|w\|^2$ as regularization term

$$R_{reg}[f] = R_{emp}[f] + \frac{\lambda}{2}\|w\|^2 \tag{2.69}$$

The geometrical interpretation of minimizing $\frac{1}{2}\|w\|^2$ is to find the flattest function with sufficient approximation quality. Furthermore, we can think of the target feature space as the reproducing kernel Hilbert space. In that case, the regularization in terms of the RKHS is written as

$$R_{reg}[f] = R_{emp}[f] + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}} \tag{2.70}$$

Theorem 6 (Representer Theorem) *Denoted by $\Omega : [0, \infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by $c : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the regularized risk*

$$c((x_1, y_1, f(x_1)), \ldots, (x_n, y_n, f(x_n)) + \Omega(\|f\|_{\mathcal{H}}) \tag{2.71}$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x) \tag{2.72}$$

The significance of this theorem is that although we might try to solve an optimization problem in an infinite dimensional space $\mathcal{H}$, containing linear combinations of kernels centered on arbitrary points on $\mathcal{X}$, it states that the solution lies in the span of the training points. In case of the support vector classification, the regularization function is written as

$$c((x_i, y_i, f(x_i))_i) = \frac{1}{n} \sum_{i=1}^{n} \text{maximize}(0, 1 - y_i f(x_i)) \tag{2.73}$$

The kernel-PCA is also corresponds to the case of

$$c((x_i, y_i, f(x_i))_i) = \begin{cases} 0 & if \frac{1}{n} \sum_i (f(x_i) - \frac{1}{n} \sum_j f(x_j))^2 = 1 \\ \infty & otherwise \end{cases} \tag{2.74}$$

2.5.5 Gradient Descent

$\hookrightarrow$ to be completed . . .

2.6 Support Vector Machine

2.6.1 separating hyperplane

If we are given a dot product space $\mathcal{H}$ and a set of points $\{x_1, \ldots, x_n\}$, any hyperplane can be written as

$$\{\langle w, x\rangle + b = 0\}, \;\; b \in \mathbb{R}, w \in \mathcal{H} \tag{2.75}$$

where w is a vector orthogonal to the hyperplane. For the purpose of classification, the aforementioned hyperlane can be used in the following form:

$$\begin{aligned} f_{w,b}: \qquad & \mathcal{H} \to \{-1, +1\} \\ & x \mapsto f_{w,b}(x) = \text{sgn}(\langle w, x\rangle + b) \end{aligned} \tag{2.76}$$

We attempt to find the solution $f_{w,b}$ such that the maximum number of training examples satisfies $f_{w,b}(x_i) = y_i$. The training set in which all the examples are correctly classified is called *separable*. With this logic, we intend to w such that the following is satisified

$$y_i(\langle w, x\rangle + b) \geq 1 \tag{2.77}$$

In order to maximize the distance between hyperplanes on positive and negative side, the following optimization problem needs to be solved

$$\begin{aligned} \text{minimize}_{w\in\mathcal{H}, b\in\mathbb{R}} \qquad & \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(\langle w, x\rangle + b) \geq 1 \qquad i = 1, \ldots, n \end{aligned} \tag{2.78}$$

This is called the *primal* optimization problem. To solve this optimization we can form the

dual which can be showed to have similar results. The Lagrangian is introduced as follows

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle w, x \rangle + b) - 1) \tag{2.79}$$

which we seek to maximize with respect to α and minimize with respect to w and b. Consequently, taking derivatives corresponds to:

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial w} L(w, b, \alpha) = 0 \tag{2.80}$$

which leads to

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.81}$$

In these expressions, only non-zero α_i correspond to constraints. The patterns that has the $\alpha_i > 0$ are called *support vectors*. Replacing the values obtained from the dual form in the initial optimization forms the following problem

$$\begin{aligned} \text{maximize}_{\alpha \in \mathbb{R}^n} \quad & \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, n \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{2.82}$$

This leads to the decision function

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \langle x, x_i \rangle + b \right) \tag{2.83}$$

In this formulation the value of x could be regarded as the output of a function that maps data to another space as $\phi(x)$. The mapping could be to a higher-dimensional space. Therefore the dot product can be rewritten as $\langle \phi(x_i), \phi(x_j) \rangle$. This dot product could be easily substituted to

the kernel function:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{2.84}$$

By this substitution the dot product to the equivalent value obtained from the kernel function the computational cost is significantly reduced. Consequently, the decision function is

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i k(x, x_i) + b \right) \tag{2.85}$$

This formulation is the *nonlinear* or kernelized representation of support vector machines. The threshold b is computed from the training examples as

$$\begin{aligned} f(x_j) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i k(x_j, x_i) + b \right) = y_j \\ \rightarrow b = y_j - \sum_{i=1}^{n} y_i \alpha_i k(x_j, x_i) \end{aligned} \tag{2.86}$$

This approach is occasionally called *hard margin classifier*.

The classifier introduced earlier is not achievable in practice as the separating hyperplane is difficult to find. Even though the separating hyperplane is found it may not be the best solution to the classifier. In order to give the examples freedom to violate the hard margin classifier, Cortes and Vapnik proposed a new method that is known as the *soft margin* classifier. In this formulation, the slack variable ξ is introduced. Therefore, the separation hyperplane is relaxed as

$$y_i(\langle w, x \rangle + b) \geq 1 - \xi_i, \;\; i = 1, \ldots, n \tag{2.87}$$

Choosing $\xi_i \geq 0$ large enough would give the desirable results. The formulation known as *C-SVM* is

$$\text{minimize}_{w \in \mathcal{H}, b \in \mathbb{R}} \quad \tfrac{1}{2}\|w\|^2 + \frac{C}{n} \textstyle\sum_{i=1}^{n} \xi_i \tag{2.88}$$

where $C \geq 0$ is the parameter to be determined by the user that controls the tradeoff between penalization and the margin. The solution, similar to the previous case, is as

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.89}$$

The following quadratic programming problem is obtained from this formulation:

$$\begin{aligned} \text{maximize}_{\alpha \in \mathbb{R}^n} \quad & \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{C}{n}, \;\; i = 1, \ldots, n \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{2.90}$$

2.6.2 constructing kernels

The choice of the kernels should reflect the prior knowledge about how data was generatedk.

1. $k(x, x') = \lim_{n \to \infty} k_n(x, x')$

2. the kernel function can be obtained from a mapping to the function space. Consider $x \mapsto f_x$ where f_x is a real-valued function. Let the dot product of the function space as

$$\langle f_x, f_{x'} \rangle = \int f_x(u) f_{x'}(u) du \tag{2.91}$$

   Then the kernels from this dot product can be obtained.

3. the *iterated kernel* is obtained as:

$$k^{(2)}(x, x') = \int k(x, x'') k(x', x'') dx'' \tag{2.92}$$

4. $(k_1 \otimes k_2)(x_1, x_2, x'_1, x'_2) = k_1(x_1, x_2) k_2(x'_1, x'_2)$ (Tensor product)

5. $(k_1 \oplus k_2)(x_1, x_2, x'_1, x'_2) = k_1(x_1, x_2) + k_2(x'_1, x'_2)$ (Direct sum)

2.7 Geometrical interpretation of kernel functions

In order for a set to be called *manifold* of dimension d, it must be (1) Hausdorff, (2) locally Euclidian of dimension d and (3) have a countable basis of open sets. The mapped points in the feature space form a Hilbert space which is a metric space and hence it is Hausdorff. If $\mathcal{H}$ is separable then the Hilbert space has countable open sets, therefore the third condition is met. The second condition is satisfied whenever the mapping function ϕ is of the fixed rank d everywhere. In order for the mapped points to form a differentiable manifold, all the derivatives of ϕ exist and are continuous.

2.7.1 Measures of distances in the feature space

There are three cases of distances considered in the feature space:

1. Intrinsic distance measured along the surface itself. The distance is generated by the Riemannian metric in the feature space.

2. The Euclidian distance between points is considered which measures the straight line joining two points.

3. The affine distance function which measures the distance of the projections of the points along some vector $w \in \mathcal{H}$

It should be noted that a metric is a function $d(x, x')$ which satisfies

$$d(x, x') = d(x', x) \geq 0$$

$$d(x, x') = 0 \Leftrightarrow x = x'$$

$$d(x, x') + d(x', '') \geq d(x'', x) \tag{2.93}$$

A Riemannian metric is the symmetric, positive definite bilinear function defined on a Riemannian manifold. This metric is defined such that the distance between two points $x(t_0), x(t_1)$ along a path lying in the feature space parameterized by $t \in [t_0, t_1]$ is given by the following path integral

$$\rho(t_0, t_1) = \int_{x(t_0)}^{x(t_1)} \left(g_{\mu\nu} \frac{\partial x^\mu}{\partial t} \frac{\partial x^\nu}{\partial t} \right)^{\frac{1}{2}} dt \tag{2.94}$$

$\hookrightarrow$ the Riemannian tensor metric should be written

CHAPTER 3

LITRETURE REVIEW: LEARNING THE KERNELS

3.1 Data dependant kernel functions

3.1.1 Natural kernels

Assuming data has been generated from a probability distribution $p(x|\theta)$, where θ is the set of parameters for this distribution. Let $l(x,\theta) = \ln p(x|\theta)$, the derivative map of $l(x,\theta)$ called the score $V_\theta : \mathcal{X} \to \mathbb{R}^r$ as

$$V_\theta(x) = (\partial_{\theta^1} l(x,\theta), \partial_{\theta^2} l(x,\theta), \ldots, \partial_{\theta^r} l(x,\theta)) = \nabla_\theta l(x,\theta) \tag{3.1}$$

which is considered to be the natural basis for the tangent vectors of the Riemannian surface constructed from the probability distribution[1]. In this case, the metric defined in the tangent space is given by the inverse of the *Fisher information metric*:

$$\mathcal{I} = E_p\left[V_\theta(x)V_\theta(x)^\top\right], \textit{i.e.}\ , \mathcal{I} = E_p\left[\partial_{\theta^i} \ln p(x|\theta)\partial_{\theta^j} \ln p(x|\theta)\right] \tag{3.2}$$

In this expression, E_p denotes the expectation with respect to density p.

With the above introduction, the natural kernel is defined as

$$k(x,x') = V_\theta(x)^\top M^{-1} V_\theta(x') = \nabla_\theta l(x,\theta)^\top M^{-1} \nabla_\theta l(x',\theta) \tag{3.3}$$

[1]This field of study known as *information geometry* **?** and focused on converging statistics, linear algebra, geometry, differential equations and etc.

In case $M = \mathcal{I}$ we obtain the *Fisher kernel* and if $M = \mathbf{1}$ the kernel obtained is called the *plain kernel*.

3.1.2 Marginalization kernels

Marginalization kernels are inspired by Hidden markov Model (HMM) as one of the most important graphical models. It is assumed that data is generated with a set of hidden parameters $h \in H$. Additionally, it is assumed that data generation is independent, *i.e.* $p(x, x'|h) = p(x|h)p(x'|h)$. If we have a prior probability of $p(h)$, the marginalized probability is as follows

$$k(x, x') = \sum_{h \in H} p(x|h)p(x'|h)p(h) \tag{3.4}$$

This kernel is specifically useful in cases that data is generated in a sequence, for example a set of strings of characters or extended to the graphs in a more general case.

3.1.3 Probability kernels

In case we need to map data points to probability distribution $\mathcal{P}_\theta$ parameterized by θ, the *probability product kernel* is proposed as **?** defined as

$$k(x, x') = k_{\mathcal{P}}(p^\rho_{\hat{\theta}}(x), p^\rho_{\hat{\theta}}(x')) \tag{3.5}$$

In this expression, $\rho > 0$ is a constant and p^ρ is not infinity. In case $\rho = \frac{1}{2}$ the kernel obtained is the famous *Bhattacharyya kernel*, *i.e.*

$$k(x, x') = \int \sqrt{p_{\hat{\theta}}(x)}\sqrt{p_{\hat{\theta}}(x'))}dx \tag{3.6}$$

The obvious map of the f the function that maps data to the probability space $f : \mathcal{X} \to \mathcal{P}_\theta$

is the one maximized the maximum likelihood, *i.e.* ,

$$x \rightarrow p_{\hat{\theta}}(x), \;\; \hat{\theta}(x) = \text{argmaximize}_{\theta \in \Theta} p_{\theta}(x) \tag{3.7}$$

3.1.4 Kullback-Laibler kernel

Similar to the probability kernel, another kernel defined on a probabilistic manner is proposed in **?** which uses the popular Kullback-Leibler divergence to define the similarity of two points. This kernel is defined as

$$k(x, y) = k_{\mathcal{P}}(p_x, p_y) = \exp(-\alpha \tilde{D}(p_x \| p_y) + \beta) \tag{3.8}$$

In this definition, $\tilde{D}(p_x \| p_y) = D(p_x \| p_y) + D(p_y \| p_x)$ is the symmetric version of KL-divergence and α, β are constant values.

3.1.5 Heat kernel

Heat or diffusion kernel is defined over the statistical manifold of data with a strong physical interpretation.

3.1.6 Laplacian graph kernel

3.2 Model Selection

3.2.1 Empirical Error Criterion for model selection

In a method proposed by Adankon *et al.* **?** Optimizing Resourcesin Model Selection for Support Vector Machines is working based on the concept of empirical error criterion. The

decision function defined earlier in SVM is used:

$$f_i = \sum_{j=1}^{SV} \alpha_j y_j k(x_j, x_i) + b \tag{3.9}$$

Let $t_i = \dfrac{(y_i + 1)}{2}$ therefore empirical error is defined as

$$E_i = |t_i - \hat{p_i}| \tag{3.10}$$

where $\hat{p_i}$ is the posterior probability corresponding to data point x_i. The probability used in this paper is the sigmoid funciton used in Platt's **?** probabilistic SVM as:

$$\hat{p_i} = \frac{1}{1 + \exp(A.f_i + B)} \tag{3.11}$$

Assuming the kernel function is parameterized by $\theta = [\theta_1, \ldots, \theta_n]$. The objective of the algorithm is formulated in terms of a gradient descent for minimization of the empirical error. In order to formulate the gradient descent, the derivative of the objective function is taken as

$$\frac{\partial E}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\frac{1}{n}\sum_{i=1}^{n} E_i\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial E_i}{\partial \theta} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial E_i}{\partial f_i}.\frac{\partial f_i}{\partial \theta} \tag{3.12}$$

In this expression, the value of this derivative then computed as

$$\frac{\partial E_i}{\partial f_i} = A y_i \hat{p_i}(1 - \hat{p_i}), \tag{3.13}$$

and the second part is approximated as

$$\frac{\partial f_i}{\partial \theta} = \sum_{j=1}^{SV} y_j \alpha_j \frac{\partial k(x_j, x_i)}{\partial \theta}. \tag{3.14}$$

Therefore an iterative approach is proposed that takes a subset of the dataset at the initial stage and attempts to incrementally add the next set to the training set such that the points remains in the set at the end of each iteration is as close as possible to the margin. In this method, the parameter α and θ are jointly optimized. The iterations are as follows:

1. train the SVM with the training set
2. estimate parameters A and B of the sigmoid function
3. compute gradient of the empirical error
4. update kernel parameters
5. subtract examples not close to the margin
6. add another subset of the remaining examples

3.2.2 DC-programming for learning a kernel

Unlike previously mentioned approaches that proposed to use convex optimization to learn a suitable kernel, a method proposed by Argyriou *et al.* **?** that uses DC (difference of convex) programming as an optimization approach. This supervised method works based on the regularization functional

$$Q(f) = \sum_{j} q(y_i, f(x_i)) + \lambda \|f\|_K^2 \tag{3.15}$$

In this method, a class of continuously parameterized set of kernels considered as

$$\mathcal{K} = \left\{ \int_{\Omega} G(\omega) dp(\omega) : p \in \mathcal{P}(\Omega) \right\} \tag{3.16}$$

where ω is the set of parameters and $\mathcal{P}(\Omega)$ is a set of probability measures on Ω and $G(.)(x, x')$ is the continues base kernel with parameter ω for $x, x' \in \mathcal{X}$. As an example Gaussian kernel can be considered, *i.e.* , $G(\omega)(x, x') = \exp(-\omega\|x - x'\|^2)$. The remarkable point of consideration about this formulation is the infiniteness of the base kernels that poses difficulty on the optimization too.The rational objective of the algorithm is considered to be the minimization of the regularization error as

$$E(K) = \text{minimize}\{Q(f) : f \in \mathcal{H}_K\} \tag{3.17}$$

The algorithm is devised as a saddle point problem of the following form:

$$\text{maximize}\{\text{minimize}\{R(\alpha, K) : c \in \mathbb{R}^n\} : K \in \mathcal{K}\} \tag{3.18}$$

where R is introduced as

$$R(\alpha, K) = \frac{1}{4\lambda}\langle\alpha, K\alpha\rangle + \sum_{i=1}^{n} q^*(y, \alpha_i) \tag{3.19}$$

and q^* is the conjugate function of q

$$q^*(y, v) = \sup\{wv - q(y, w) : w \in \mathbb{R}\} \tag{3.20}$$

The notation of $R(\alpha, K)$ is obtained from the von Neumann minimax theorem. In order to maximize the outer part of the Eqn. (3.18), the DC-programming is devised. The DC-function is formulated as

$$f : \Omega \to \mathbb{R}, f(\omega) = g(\omega) - h(\omega), \omega \in \Omega \tag{3.21}$$

where ω is a closed convex subset of $\mathbb{R}^d$ and g, h are two convex funciions. Consequently, the DC-program is defined as

$$\inf\{f(\omega) : \omega \in \Omega, f_i(\omega) \leq 0\} \tag{3.22}$$

$\hookrightarrow$ check ''DC-Programming overview'', ''learning the kernel function via regularization''

3.2.3 Information geometric approach to improve SVM

The Riemannian metric used in the feature space is obtained from

$$g_{ij} = \frac{\partial}{\partial x_i} \frac{\partial}{\partial x'_j} k(x, x')|_{x'=x} \tag{3.23}$$

The volume of the Riemannian space is defined as

$$dV = \sqrt{\det |g_{ij}(x)|}\ dx_1 \dots dx_n \tag{3.24}$$

The factor $\sqrt{\det |g_{ij}(x)|}$ represents how a local area is magnified in the feature space under mapping ϕ. The Riemannian distance is defined as

$$ds^2 = \sum_{i,j=1}^{n} g_{ij}\, dx_i\, dx_j \tag{3.25}$$

By increasing the value of metric g_{ij}, the distance between points around the decision boundary is increased. Thus, the nonlinear mapping ϕ is modified such that $\sqrt{g(x)}$ is enlarged around the boundaries. The boundaries are selected to be the support vectors in the SVM. A conformal transformation $\tilde{g}_{ij}(x) = \Omega(x) g_{ij}(x)$ is proposed to solve the problem where the conformal map $\Omega(x)$ has a large value around the boundaries. The conformal transformation won't change the angle between points and therefore the spatial characteristics remains unchanged. Since finding $\tilde{\phi}$ that realizes this transformation is difficult, a modification of the kernel is used. The

following definition is used to modify the kernel.

Definition 6 (Conformal transformation of a kernel) *The conformal transformation of a kernel function with factor $c(x)$ is defined as*

$$k(x, x') = c(x)c(x')k(x, x') \tag{3.26}$$

The choice of $c(x)$ is made using the Gaussian kernel as

$$\sum_{\ell=1}^{|\mathcal{S}|} \alpha_\ell e^{-\frac{|| \mathbf{x} - s_\ell ||^2}{2\sigma}} \tag{3.27}$$

where s_ℓ denotes each support vector in the set of support vectors $\mathcal{S}$. Finally, the kernel selection algorithm is proposed that follows two steps:

1. the SVM is trained using the kernel function k and obtain the support vectors
2. use the conformal transformation of kernels according to Eqns. (3.26) and (3.27)
3. train the SVM using the modified kernel

3.2.4 kernel alignment measure for kernel selection

Kernel alignment measure proposed by Cristianini *et al.* **?** used by them to derive a semi-supervised kernel selection algorithm. Using the eigendecomposition of the kernel matrix given as

$$K = \sum_i \alpha_i v_i v_i^\top \tag{3.28}$$

Consequently the kernel alignment measure could be modified as

$$A(y) = \frac{\langle K, yy^\top \rangle}{n\sqrt{\sum_{ij} \alpha_i \alpha_j \langle v_i v_i^\top, v_j v_j^\top \rangle}} = \frac{\sum_i \alpha_i \langle v_i, y \rangle^2}{\sqrt{\langle yy^\top, yy^\top \rangle}\sqrt{\sum_i \alpha_i^2}} \tag{3.29}$$

Therefore, the optimization objective, considering Laplacian multiplier λ, can be formulated as

$$\text{maximize} \ \sum_i \alpha_i \langle v_i, y \rangle^2 - \lambda(\sum_i \alpha_i^2 - 1) \tag{3.30}$$

Setting the derivative of the objective function to zero yields$\alpha_i \propto \langle v_i, y \rangle^2$. Replacing value of α in Eqn. (3.29) we have

$$A(y) = \frac{\sqrt{\sum_i \langle v_i, y \rangle^4}}{n} \tag{3.31}$$

Finally, the optimal kernel is obtained from an iterative approach that modifies a kernel matrix by decomposing it to its rank one eigenvectors. In each iteration, the coefficient α is updated by measuring the alignment between v_i and y.

3.2.5 Boosting for kernel design

3.2.6 Linear programming for kernel design using relative similarity

In the method proposed by Fung *et al.* **?** which is based on the relative relation between objects defined by a comparative value stating object a is more similar to object b rather than c. The relative relation is denoted by $\mathcal{T} = \{(i, j, k) | \kappa(x_i, x_j) > \kappa(x_i, x_k)\}$ as the constraints provided by the user and the objective of the algorithm is to find the parameterized kernel $\kappa = \sum_{i=1}^{m} \alpha_i k_i(x, y)$ (and the kernel matrix $\mathcal{K}$ from κ) similar to Lanckriet's formulation.

Consequently, the following optimization problem is achieved

$$\begin{aligned} \text{minimize}_{\alpha,\epsilon} \quad & \sum_t \epsilon_t + \gamma h(\mathcal{K}) \\ \text{subject to} \quad & \mathcal{K}_{i,j} + \epsilon_t > \mathcal{K}_{i,k} \\ & \epsilon \geq 0 \\ & \mathcal{K} \succeq 0 \end{aligned} \tag{3.32}$$

In this optimization problem, ϵ_t denotes the slack variables and t indexes $\mathcal{T}$, $h(\mathcal{K})$ is a regularizer on $\mathcal{K}$ and γ controls the tradeoff between constraints and the regularization strength. Although this formulation can be solved using semidefinite programming but the following theorem simplifies the computation and reduces the problem to a linear programming problem.

Theorem 7 (Diagonal dominance theorem) *Given a matrix $M \in \mathbb{R}^{d\times d}$ is symmetric and that for each $i = 1, \ldots, n$, we have*

$$M_{ii} \geq \sum_{j\neq i} |M_{ij}| \tag{3.33}$$

then M is positive semidefinite. If the inequalities above are all strict, then M is positive definite.

Using the abovementioned theorem, a linear programming problem is formulated as

$$\begin{aligned}
\text{minimize}_{\alpha,\epsilon,s,r} \quad & \sum_t \epsilon_t + \gamma_1 \sum_i s_i + \gamma_2 \sum_i \mathcal{K}_{ii} \\
\text{subject to} \quad & \mathcal{K}_{ij} + \epsilon_t - \mathcal{K}_{ik} \geq 1 \\
& -r_{ij} \leq \mathcal{K}_{ij} \leq r_{ij} \\
& \mathcal{K}_{ii} - \sum_{i \neq j} r_{ij} \geq 0 \\
& -s_i \leq \alpha_j \leq s_j \\
& \epsilon \geq 0
\end{aligned}$$

3.2.7 Learning kernel matrix using Bregman divergence

The Bregman divergence for a strictly convex function ψ over a convex set s is defined as

$$D_\psi(x, y) = \psi(x) - \psi(y) - (x - y)^\top \nabla\psi(y) \tag{3.34}$$

This definition is extended to the Bregman matrix divergence as

$$D_\psi(X, Y) = \psi(X) - \psi(Y) - \text{tr}((\nabla\psi(Y))^\top (X - Y)) \tag{3.35}$$

Given the Bregman divergence formulation, the method proposed by Kulis *et al.* **??** as

$$\begin{aligned}
\text{minimize} \quad & D_\psi(\mathcal{K}, K) \\
\text{subject to} \quad & \text{tr}(\mathcal{K}A_i) \leq b_i, \quad 1 \leq i \leq c \\
& \text{rank}(\mathcal{K}) \leq r \\
& \mathcal{K} \succeq 0
\end{aligned} \tag{3.36}$$

in which K is the input kernel matrix(e.g. graph Laplacian kernel). The rank of the matrix is limited to be r as the authors intended to use the kernel obtained from this algorithm in dimensionality reduction. The small rank of the matrix allows the small eigenvalues to be easily omitted. If the rank of matrix K is less than r then the resulting optimization is convex because the two cases for the update criteria considered are maintains the range space of the input kernel. Considering two special cases where ψ computes the negative entropy of the eigenvalues of a positive semidefinite matrix and Burg entropy, the von Neumann and Burg divergence is achieved. Using these two cases, the update criteria in the optimization is formulated. The importance of these technical is that, if someone does not consider limiting the rank of the desired matrix, it is possible to obtain a matrix from another one which has the highest amount of similarity in the structure. It also allows the user to constrain the resulting matrix with other criteria.

3.2.8 Geometry-aware metric learning

One aspect of high importance is the use of intrinsic structure of dataset to be able to learn the optimal kernel of the choice. In order to measure the distance between two matrices the LogDet matrix divergence is used which is defined as

$$D_{\ell d}(K, \mathcal{K}) = \text{tr}(K\mathcal{K}^{-1}) - \log\det(K\mathcal{K}^{-1}) - n \tag{3.37}$$

LogDet divergence is a class of Bregman matrix divergence that can specifically be used for the positive definite matrices. The advantage of using such a divergence is that it stays invariant to rescaling of the feature space. In the choice of K the important aspect is to be able to capture the data structure, e.g. graph Laplacian or in a more general case **?** (as will be discussed later):

$$K = \left\{ K_0 - K_0(I + TK_0)^{-1}TK_0 \quad | \quad T = \sum_{i=1}^{r} \lambda_i v_i v_i^\top, \lambda_1 \geq \ldots \geq \lambda_r \geq 0 \right\} \tag{3.38}$$

The significance of these kernels are their ability to be easily extended to unseen examples by

$$k(x, x') = k_0(x, x') - k_0(x, .)(I + TK)^{-1}Tk_0(., x') \tag{3.39}$$

The convexity of this set is certain as long as $\{v_1, \ldots, v_r\}$ are orthogonal. To obtain the result, it should be firstly noted that in general $D_{\ell d}(K, \mathcal{K})$ is not jointly convex in K and $\mathcal{K}$. Therefore, the goal of this algorithm is formulated as finding T and consequently λ. In order to solve the problem a cyclic projection algorithm is used where at each step the solution is projected on the constraints. The optimization to solve the divergence of $D_{\ell d}(K, \mathcal{K})$ is similar to the one used in **?**.

3.2.9 Information-theoretic metric learning

From another aspect, the problem can be seen as a metric learning problem with an information theoretic approach **?**. A specific class of distance functions known as Mahalanobis distance is defined as

$$d_A(x, x') = (x - x')^\top A(x - x') \tag{3.40}$$

Therefore the problem in this case is to find the positive definite matrix A that parameterizes the Mahalanobis distance. In cases where data is Gaussian, this parameter can be selected to be the inverse of the covariance matrix. It is assumed that a base matrix A_0 is given and A is sought to be as similar as possible to A_0. In this case the KL-divergence between two distributions as defined as

$$KL(p(x; A_0) \| p(x; A)) = \int \mathcal{N}(x; A_0) \log \frac{\mathcal{N}(x; A_0)}{\mathcal{N}(x, A)} dx \tag{3.41}$$

where $\mathcal{N}(x; A) = \frac{1}{Z} \exp(-\frac{1}{2} d_A(x, \mu))$ is the multivariate Gaussian distribution with normalizing constant Z and mean μ, A^{-1} is also the covariance matrix. In case the mean of two

distributions are the same, the above equation is related to the LogDet divergence, *i.e.*

$$KL(p(x;A_0)\|p(x;A)) = \frac{1}{2}D_{\ell d}(A_0^{-1},A^{-1}) = \frac{1}{2}D_{\ell d}(A,A_0) \tag{3.42}$$

This problem can be optimized using the optimization method proposed by **?**. In addition to the case considered earlier for learning the kernel, the kernel in the form of $K = \mathcal{X}^\top A\mathcal{X}$ and $K_0 = \mathcal{X}^\top\mathcal{X}$ could also be considered. Replacing x in all computations with $\phi(x)$ will pave the way for simple kernel based solution. This method produces similar optimization that iteratively optimizes the kernel function.

3.3 Learning kernels from multiple sources

3.3.1 Beyond the point cloud: semi supervised

In an interesting work of **?**, the structure of the dataset is taken into consideration to construct a semi-supervised setting. In this algorithm, a small number of labeled examples are used as the points of kernel's construction to start and led by the unlabeled examples. In this method, unlike the approach we proposed, the cluster assumption plays a crucial rule. The kernel function is constructed based on the RKHS and the assumption that the evaluation functional is bounded at each point, *i.e.*

$$|f(x)| \leq C\|f\|_{\mathcal{H}} \tag{3.43}$$

Let the function S be the operator mapping each point from the inner product space to the space of quadratic inner product space and denote the obtained space as $\mathcal{H}^*$, *i.e.*

$$\langle f,g\rangle_{\mathcal{H}^*} = \langle f,g\rangle_{\mathcal{H}} + \langle Sf,Sg\rangle \tag{3.44}$$

For the purpose of this method, S is defined as the evaluation map $S(f) = S_f = (f(x_1), f(x_2), \ldots, f(x_n))$. Therefore the following (semi-)norm holds

$$\|Sf\|^2 = S_f^\top M S_f \tag{3.45}$$

where M left to be set later. In order to obtain the new kernel in $\mathcal{H}^*$, the linear combination of predefined kernels in $\mathcal{H}$ is considered as

$$\kappa^*(x, .) = k(x, .) + \sum_j \beta_j(x) k(x_j, .) \tag{3.46}$$

In this equation, β is data-dependant function of x . By defining the inner product over $k(x_i, .)$ at x the ultimate kernel in $\mathcal{H}^*$ is then obtained as

$$\kappa^*(x, z) = k(x, z)?k_x^\top (I + MK)^{-1} M K_z \tag{3.47}$$

The value of the matrix $M = L^p$, where p is the integer value and L is the Laplacian graph defining the geometry of the points.

This work, in addition to defining a family of kernel functions that favors the geometrical structure of the dataset, set the stage for future works as the geometry and the structure of the points in a set with regard to the RKHS is definable.

3.3.2 learning the kernel via regularization

Relate the papers to the DC-prograaming above **??**

3.3.3 Cross-validation

Cross-validation has traditionally been the solution to determine parameters in model selection for most of the well-known machine learning algorithms, not surprisingly they found application in kernels methods and specifically SVM **?**. There are variations of cross-validations, e.g. , k-fold cross-validation and leave-one-out. In which dataset is split into k subsets (folds) and the SVM decision rule is obtained from the $k-1$ folds leaving one fold out for test. This procedure is repeated k times until all subsets are tested. In the leave-one-out method, the number of k in k-fold is set to be the number of examples. Hence, n times of training and testing is required. However, the parameters determined in leave-one-out are unbiased of the expected error.

3.3.4 learning kernel hyper-parameters in a graph based semi supervised approach

Kapoor *et al.* **?** a Bayesian approach is proposed to semi-supervised learning and hyper-parameter selection of the kernel function. Along with other semi-supervised methods, the cluster assumption is made and from that it is assumed that the labels of the unlabeled data ξ_u–called hard labels–are related through hidden layer called soft labels h as

$$p(\xi_u|D) = \int_h p(\xi_u|h)p(h|D) \tag{3.48}$$

where D is the set of all labeled and unlabeled data. Using the Bayes rule the following holds

$$p(h|D) = p(h|\mathcal{X}, \xi) \propto p(y|\mathcal{X})p(\xi|h) \tag{3.49}$$

In order to calculate $p(h|\mathcal{X})$, a transformed graph Laplacian $r(L) = \sum_{i=1}^{n} r(\lambda_i) v_i^\top v_i$ is considered which is modeled as follows

$$p(h|\mathcal{X}) \propto \exp(-\frac{1}{2} h^\top r(L) h) \tag{3.50}$$

The likelihood $p(\xi|h)$ ca be written as $p(\xi|h) = \prod_{i=1}^{n} p(\xi_i|h_i)$ which is approximated as

$$p(\xi_i|h_i) = \epsilon(1 - \psi(h_i.\xi_i)) + (1 - \epsilon)\psi(\xi_i.h_i) \tag{3.51}$$

where ψ is the step function and ϵ is the labeling error rate. Consequently, denoted by Θ the parameters for the graph Laplacian and the kernel parameters and $p(h|D) = \mathcal{N}(\bar{h}, \Sigma_h)$, the posterior probability of the soft labels, the objective function is modeled as: $\arg\max_\Theta \log[p(\xi|\mathcal{X}, \Theta)]$. Subsequently, the EM algorithm is used to maximize this problem in following steps

1. **E-Step** given the current Θ_i, approximate $\mathcal{N}(\bar{h}, \Sigma_h)$
2. **M-Step** update $\Theta_{i+1} = \arg\max_\Theta \int_h \mathcal{N}(\bar{h}, \Sigma_h) \log \frac{p(h|\mathcal{X}, \Theta) p(\xi|h, \Theta)}{\mathcal{N}(\bar{h}, \Sigma_h)}$

3.3.5 Discriminant kernel learning

Given a dataset of training points it is possible to use the Discriminant analysis to obtain the optimization problem to obtain the optimal kernel. Thus maximizing the following objective function

$$F_1(w, K) = \frac{(w^\top(\bar{\phi}^+ - \bar{\phi}^-))^2}{w^\top(n_+/n\Sigma^+ + n_-/n\Sigma^- + \lambda I)w} \tag{3.52}$$

In this equation, $\bar{\phi}$ is the mean, Σ is the covariance matrix in the positive or negative class and $\lambda > 0$ is a regularization parameter. The optimal weight vector is determined as

$$w^* = \arg\max_w F_1(w, K) \tag{3.53}$$

for a fixed kernel matrix K and regularization parameter λ. This is equal to

$$w^* = (n_+/n\Sigma^+ + n_-/n\Sigma^- + \lambda I)^{-1}(\bar{\phi}^+ - \bar{\phi}^-) \tag{3.54}$$

Fixing $w = w^*$, the optimal value for Ks is obtain as

$$F_1^*(K) = (\bar{\phi}^+ - \bar{\phi}^-)^\top (n_+/n\Sigma^+ + n_-/n\Sigma^- + \lambda I)^{-1}(\bar{\phi}^+ - \bar{\phi}^-) \tag{3.55}$$

In order to maximize $F_1^*(K)$, the following convex optimization is proposed in **?**

$$\begin{aligned} \text{minimize} \quad & (1/\lambda)(t - \sum_{i=1}^{m} \theta_i a^\top G_i a) \\ \text{subject to} \quad & \begin{pmatrix} \lambda I \sum_{i=1}^m \theta_i J G_i J & \sum_{i=1}^m \theta_i J G_i a \\ \sum_{i=1}^m \theta_i J G_i a & t \end{pmatrix} \succeq 0 \\ & \theta \geq 0 \\ & \mathbf{1}\theta = 1 \end{aligned} \tag{3.56}$$

?

CHAPTER 4

IMPLEMENTATION

In this paper, we propose an approach to learn the kernel function called *Transferred Learning of the Kernel*. Our proposed approach aims to refine the mapping function such that the following criteria are satisfied:

1. Maximize data separability among different class members of mapped labeled examples in the feature space
2. Keep the distribution of labeled data (in main dataset) as similar as possible to the unlabeled ones (in auxiliary dataset)

The first criterion is in line with the fact that SVM is a linear algorithm which searches for a discriminative line that separates classes in labeled examples. The second criterion, on the other hand, signifies that the data distribution of unlabeled examples is the constraint we pose on the labeled data. Intuitively, as the number of labeled examples is limited, the distribution of data in the feature space is constructed such that it best resembles the unlabeled data distribution. In this paper, we assume either unlabeled data is drawn from the same distribution (similar to semi-supervised case) or at least the main and auxiliary datasets are related. However, it should be noted that although the unlabeled samples are collected from a relevant distribution, there may be samples not closely related to the main distribution. We use KNN to select points from unlabeled data that are closer to the labeled data. Using KNN, we deter the algorithm from being misguided via unlabeled data.

As mentioned earlier, the objective of our Transferred Learning of the Kernel approach is to satisfy the two aforementioned criteria. This is accomplished using Fisher Discriminant Analysis (FDA) criterion **?** and Maximum Mean Discrepancy (MMD) **??**. In a nutshell, the FDA tries to maximize linear separability of a dataset while MMD measures discrepancy between two distributions. In our work, both FDA and MMD are used together to operate on the mapped points in the feature space that yields in selecting the best mapping function.

4.1 Transferred Learning of the Kernel

The basic assumption in the proposed approach is that two sets of data from two different but rather related distributions are given: the main dataset which is denoted by $\mathcal{X} = \{(x_i, \xi_i) \mid (x, \xi) \in \mathbb{R}^d \times \mathbb{R}, i = 1, 2, \ldots, n\}$ and the auxiliary dataset $\mathcal{Y} = \{y_i \mid y \in \mathbb{R}^d, i = 1, 2, \ldots, m\}$. Auxiliary dataset is going to be used as additional source of information and is supposed to aid in learning the kernel. Two datasets $\mathcal{X}$ and $\mathcal{Y}$ are jointly used to learn the optimal kernel for SVM in which the optimal kernel is subsequently used to classify points related to $\mathcal{X}$. In compliance with classification task, we assume labeled examples in $\mathcal{X}$ are categorized into l classes of n_i data points. Clearly, in case of binary classification, $l = 2$. In order to deter $\mathcal{Y}$ from misguiding $\mathcal{X}$ in learning the kernel, we apply the KNN algorithm on the auxiliary dataset to find the most related points in $\mathcal{Y}$. Therefore, throughout this paper by $\mathcal{Y}$ we mean the dataset obtained after applying KNN.

Learning the kernel function consists of refining a base kernel k to obtain $\kappa^* : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which could better define the similarity between pair of points in $\mathcal{X}$. In order to formulate the optimal kernel, we adapt the kernel optimization algorithm proposed in **?** to suit our purpose. Authors in this paper based their algorithm on usage of conformal transformation and FDA. In conformal transformation of kernels **??**, the mapping function ϕ is modified by a positive function q such that the metric around the decision boundaries, *i.e.* support vectors, are enlarged.

In other words, optimal kernel κ^* is factorized by a function q and the base kernel k such that between class distances are increased. Using conformal mapping of kernel the optimal kernel may be written as:

$$\kappa^*(\mathbf{x}, \mathbf{x}') = q(\mathbf{x})q(\mathbf{x}')k(\mathbf{x}, \mathbf{x}') \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d \tag{4.1}$$

Base kernel function k may be selected from various kernels available that may have produced good results in the related problem (e.g. polynomial or linear kernel) and $\mathbf{x}$ is any of the input data points either from $\mathcal{X}$ or $\mathcal{Y}$. The positive function $q(\mathbf{x})$ is defined as:

$$q(\mathbf{x}) = \alpha_0 + \sum_{\ell=1}^{|\mathcal{S}|} \alpha_\ell h(\mathbf{x}, s_\ell) \quad \forall\, \mathbf{x} \in \mathbb{R}^d \tag{4.2}$$

In this equation, $h(\mathbf{x}, s_\ell) = e^{-||\mathbf{x}-s_\ell||^2/2\sigma}$ and s_ℓ denotes each support vector in the set of support vectors $\mathcal{S}$ obtained from training the SVM on $\mathcal{X}$ ($|\mathcal{S}|$ denotes the size of $\mathcal{S}$). The coefficient α_ℓ expresses the relationship between each of the support vectors s_ℓ and $x \in \mathcal{X}$. Therefore, the value of $q(\mathbf{x})$ is eventually dependant on values of α_ℓ as a linear combination of Gaussian kernels h defined over support vectors. The initial support vectors obtained from training the SVM are used as a basis for further refinement because they may be regarded as the initial guesses for the choice of decision boundaries. In the case where the number of labeled data is limited, resembling the distribution of data in $\mathcal{Y}$ could assist modifying the decision making region.

The definition of κ^* in the matrix representation could be rewritten as:

$$\mathcal{K}^*_{\mathcal{D}} = \mathcal{Q}_{\mathcal{D}} K_{\mathcal{D}} \mathcal{Q}_{\mathcal{D}} \tag{4.3}$$

where $\mathcal{K}^*_{\mathcal{D}}$ is the optimal kernel matrix and $K_{\mathcal{D}}$ is the initial kernel matrix created from dataset $\mathcal{D}$. Matrix $\mathcal{Q}_{\mathcal{D}}$ is also a diagonal matrix with entries $\{q(\mathbf{x}_1), q(\mathbf{x}_2), \ldots, q(\mathbf{x}_n)\}$. We also denote vector $q_{\mathcal{D}} = [q(\mathbf{x}_1), q(\mathbf{x}_2), \ldots, q(\mathbf{x}_n)]^\top$ for $\mathbf{x}_i \in \mathcal{D}$, and $\alpha = [\alpha_0, \alpha_1, \ldots, \alpha_{|\mathcal{S}|}]^\top$. From equations (4.1) and (4.3) we can easily infer:

$$\mathcal{K}^*_{\mathcal{D}} = [q(\mathbf{x}_i) q(\mathbf{x}_j) k(\mathbf{x}_i, \mathbf{x}_j)] = {q_{\mathcal{D}}}^\top K_{\mathcal{D}} q_{\mathcal{D}} \quad \forall \mathbf{x} \in \mathcal{D} \tag{4.4}$$

Consequently, $q_{\mathcal{D}}$ for $\mathbf{x}_i \in \mathcal{D}$ could be rewritten as:

$$q_{\mathcal{D}} = \begin{pmatrix} 1 & h(\mathbf{x}_1, s_1) & \ldots & h(\mathbf{x}_1, s_{|\mathcal{S}|}) \\ 1 & h(\mathbf{x}_2, s_1) & \ldots & h(\mathbf{x}_2, s_{|\mathcal{S}|}) \\ \vdots & \vdots & \ldots & \vdots \\ 1 & h(\mathbf{x}_n, s_1) & \ldots & h(\mathbf{x}_n, s_{|\mathcal{S}|}) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{|\mathcal{S}|} \end{pmatrix} = H_{\mathcal{D}}.\alpha \tag{4.5}$$

In order to be able to consider datasets $\mathcal{X}$ and $\mathcal{Y}$, two different aspects of the problem should be noted: on one hand minimal number of main training examples should be made, and on the other hand, the auxiliary dataset $\mathcal{Y}$ should be utilized in addition to $\mathcal{X}$. Consequently, we propose to jointly optimize two criteria, MMD and FDA between two distributions. FDA is a method that maximizes the distance of between class distributions while keeping data with similar labels as close as possible. However, FDA is useful in cases where labeled data from a distribution is available. In order to use the distributions $\mathcal{X}$ and $\mathcal{Y}$ jointly, one could estimate the density of the distributions and use the well-known methods like KL-divergence to guide two distributions to be close. However, accuracy of density estimation is highly dependent on the number of examples *i.e.* larger number of examples produces better density function. In

this paper, on the other hand, we propose to use MMD which is a test of statistics that measures similarity between two distributions. The advantage of using MMD over the former methods is that the density estimation is not required.

In order to achieve the objective of the algorithm in jointly optimizing MMD and FDA, we propose to use the following criteria:

$$J = \frac{tr(S_b)}{tr(S_w) + \lambda tr(\Delta)} \tag{4.6}$$

where S_b denotes *"between-class scatter matrix"* and S_w *"within-class scatter matrix"*, standard matrices in FDA. In this equation, λ used to control model complexity or the amount of influence expected from transferred knowledge and Δ, on the other hand, denotes the measure of proximity of labeled and unlabeled data distributions obtained from MMD measure.

Each term in equation (4.6) may formally be written as:

$$\begin{aligned}
tr(S_b) &= \sum_{i=1}^{l} (\bar{\phi}(x^i) - \bar{\phi}(x))(\bar{\phi}(x^i) - \bar{\phi}(x))^\top \\
tr(S_w) &= \sum_{i=1}^{l} \sum_{j=1}^{n_i} (\phi(x_j^i) - \bar{\phi}(x^i))(\phi(x_j^i) - \bar{\phi}(x^i))^\top \\
tr(\Delta) &= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \phi(x_i)\phi(x_j) + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \phi(y_i)\phi(y_j) \\
&\quad - \frac{2}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} \phi(x_i)\phi(y_j)
\end{aligned} \tag{4.7}$$

where $\phi(x_i)$ and $\phi(y_i)$ are the images of x and y in the feature space, $\bar{\phi}(x^i)$ is the average of

data in its respective class and $\bar{\phi}(x)$ is the centre of images of entire training samples. Formally:

$$\begin{aligned}\bar{\phi}(x^i) &= \frac{1}{n_i}\sum_{j=1}^{n_i}\phi(x_j)\\ \bar{\phi}(x) &= \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\\ \bar{\phi}(y) &= \frac{1}{m}\sum_{i=1}^{m}\phi(y_i)\end{aligned} \tag{4.8}$$

In equation (4.7), S_b and S_w represent standard formulation of FDA in the feature space and Δ defines MMD statistics test. For two probability distributions $p(\mathcal{X})$ and $p(\mathcal{Y})$ defined on sample sets $\mathcal{X}$ and $\mathcal{Y}$, MMD is defined as **??**:

$$\mathrm{MMD}[\mathcal{F}, p(\mathcal{X}), p(\mathcal{Y})] = \sup_{f\in\mathcal{F}}(\mathbf{E}_{x\sim\mathcal{X}}[f(x)] - \mathbf{E}_{y\sim\mathcal{Y}}[f(y)]) \tag{4.9}$$

The value of $\mathrm{MMD}[\mathcal{F}, p(\mathcal{X}), p(\mathcal{Y})]$ is zero if $p(\mathcal{X}) = p(\mathcal{Y})$. Selecting $\mathcal{F}$ to be a unit ball in the universal *Reproducing Kernel Hilbert Space (RKHS)* establishes the basis for selecting $f(x) = \langle k(x,.), f\rangle$ that simplifies the equation in (4.9) to be rewritten as equation (4.10) (details on obtaining equation (4.10) from (4.9) is given in **?**):

$$\mathrm{Dist}(\phi(\mathcal{X}), \phi(\mathcal{Y})) \quad = \quad \|\bar{\phi}(x) - \bar{\phi}(y)\| \tag{4.10}$$

where $\phi(\mathcal{X})$ and $\phi(\mathcal{Y})$ denotes the mapping of the whole training data from two distributions $\mathcal{X}$ and $\mathcal{Y}$ in the feature space. We seek to minimize $\mathrm{Dist}(\phi(\mathcal{X}), \phi(\mathcal{Y}))$ in equation (4.10) as we want to decrease discrepancy between two distributions $\mathcal{X}$ and $\mathcal{Y}$. From replacing (4.8)

in (4.10), it could be easily inferred that minimization of $\mathrm{Dist}(\phi(\mathcal{X}), \phi(\mathcal{Y}))$ in equation (4.10) is equal to minimizing the trace of the following matrix:

$$(\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{i=1}^{m}\phi(y_i))^\top(\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{i=1}^{m}\phi(y_i))$$

Considering the definition of kernel function $\kappa^* = \langle\phi(\mathbf{x}_i)\phi(\mathbf{x}_j)\rangle$ and definition of conformal transformation given in equation (4.4), Δ could be rewritten as:

$$\begin{aligned}\Delta &= \sum_{i=1}^{3}\Delta_i = \frac{1}{n^2}\mathcal{K}^*_{\mathcal{X}} + \frac{1}{m^2}\mathcal{K}^*_{\mathcal{Y}} - \frac{2}{mn}\mathcal{K}^*_{\mathcal{X}\mathcal{Y}} \\ &= \frac{1}{n^2}q_{\mathcal{X}}^\top K_{\mathcal{X}} q_{\mathcal{X}} + \frac{1}{m^2}q_{\mathcal{Y}}^\top K_{\mathcal{Y}} q_{\mathcal{Y}} - \frac{2}{mn}q_{\mathcal{X}}^\top K_{\mathcal{X}\mathcal{Y}} q_{\mathcal{Y}}\end{aligned}$$

where $K_{\mathcal{X}\mathcal{Y}}$ is an $(n+m)\times(n+m)$ matrix constructed from base kernel on $\mathcal{X}$ and $\mathcal{Y}$. Furthermore, we denote each $\{\mathcal{K}^*_{ij} \mid i,j = 1,2,\ldots,l\}$ as a matrix with dimensions $n_i \times n_j$. Each $\mathcal{K}^*_{ij}$ represents a sub-matrix in $\mathcal{K}^*$ that is computed from the data of its respective class. Consequently, traces of matrices S_b and S_w is rewritten as **?**:

$$\begin{aligned}tr(S_b) &= \begin{pmatrix} \frac{1}{n1^2}\mathcal{K}^*_{11} & 0 & \ldots & 0 \\ 0 & \frac{1}{n2^2}\mathcal{K}^*_{22} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & \frac{1}{nl^2}\mathcal{K}^*_{ll} \end{pmatrix} \\ &\quad -\frac{1}{n^2}\begin{pmatrix} \mathcal{K}^*_{11} & \mathcal{K}^*_{12} & \ldots & \mathcal{K}^*_{1l} \\ \mathcal{K}^*_{21} & \mathcal{K}^*_{22} & \ldots & \mathcal{K}^*_{2l} \\ \vdots & \vdots & \ldots & \vdots \\ \mathcal{K}^*_{l1} & \mathcal{K}^*_{l2} & \ldots & \mathcal{K}^*_{ll} \end{pmatrix} = q_{\mathcal{X}}^\top \mathcal{B} q_{\mathcal{X}}\end{aligned} \tag{4.11}$$

and

$$
\begin{aligned}
tr(S_w) \quad = \quad & \begin{pmatrix} \kappa^*_{11} & 0 & \dots & 0 \\ 0 & \kappa^*_{22} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \kappa^*_{ll} \end{pmatrix} \\
& - \begin{pmatrix} \frac{1}{n1^2}\mathcal{K}^*_{11} & 0 & \dots & 0 \\ 0 & \frac{1}{n2^2}\mathcal{K}^*_{22} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \frac{1}{nl^2}\mathcal{K}^*_{ll} \end{pmatrix} = q_{\mathcal{X}}^{\top}\mathcal{W}q_{\mathcal{X}}
\end{aligned}
\tag{4.12}
$$

where $\mathcal{W}$ and $\mathcal{B}$ are the matrices computed from the base kernel k. Consequently from $\mathcal{W}$ and $\mathcal{B}$, maximization of J in equation (4.6) could be performed. With slightly misuse of notation, the final value for α could be obtained from $\frac{\partial J(\alpha)}{\partial \alpha} = 0$ as:

$$
\frac{(\frac{\partial S_b}{\partial \alpha}(S_w + \lambda\Delta) - \frac{\partial (S_w + \lambda\Delta)}{\partial \alpha} S_{\mathcal{B}})}{(S_w + \lambda\Delta)^2} = 0
$$

where

$$
\begin{aligned}
\frac{\partial S_b}{\partial \alpha} &= \frac{\partial (q_{\mathcal{X}}^{\top}\mathcal{B}q_{\mathcal{X}})}{\partial (q_{\mathcal{X}})}\frac{\partial (q_{\mathcal{X}})}{\partial \alpha} = 2H_{\mathcal{X}}^{\top}\mathcal{B}H_{\mathcal{X}}\alpha = S'_b\alpha \\
\frac{\partial S_w}{\partial \alpha} &= \frac{\partial (q_{\mathcal{X}}^{\top}\mathcal{W}q_{\mathcal{X}})}{\partial (q_{\mathcal{X}})}\frac{\partial (q_{\mathcal{X}})}{\partial \alpha} = 2H_{\mathcal{X}}^{\top}\mathcal{W}H_{\mathcal{X}}\alpha = S'_w\alpha \\
\frac{\partial \Delta_1}{\partial \alpha} &= \frac{\partial (q_{\mathcal{X}}^{\top}K_{\mathcal{X}}q_{\mathcal{X}})}{\partial (q_{\mathcal{X}})}\frac{\partial (q_{\mathcal{X}})}{\partial \alpha} = \frac{2}{n^2}H_{\mathcal{X}}^{\top}K_{\mathcal{X}}H_{\mathcal{X}}\alpha \\
\frac{\partial \Delta_2}{\partial \alpha} &= \frac{\partial (q_{\mathcal{Y}}^{\top}K_{y}q_{\mathcal{Y}})}{\partial (q_{\mathcal{Y}})}\frac{\partial (q_{\mathcal{Y}})}{\partial \alpha} = \frac{2}{m^2}H_{\mathcal{Y}}^{\top}K_{\mathcal{Y}}H_{\mathcal{Y}}\alpha \\
\frac{\partial \Delta_3}{\partial \alpha} &= \frac{\partial (q_{\mathcal{X}}^{\top}K_{\mathcal{X}\mathcal{Y}}q_{\mathcal{Y}})}{\partial (q_{\mathcal{X}})}\frac{\partial (q_{\mathcal{X}})}{\partial \alpha} + \frac{\partial (q_{\mathcal{X}}^{\top}K_{\mathcal{X}\mathcal{Y}}q_{\mathcal{Y}})}{\partial (q_{\mathcal{Y}})}\frac{\partial (q_{\mathcal{Y}})}{\partial \alpha} \\
&= \frac{-4}{nm}H_{\mathcal{X}\mathcal{Y}}^{\top}K_{\mathcal{X}\mathcal{Y}}H_{\mathcal{X}\mathcal{Y}}\alpha
\end{aligned}
\tag{4.13}
$$

Denoting $\Delta'\alpha = \sum_{i=1}^{3} \frac{\partial \Delta_i}{\partial \alpha}$, the optimal solution to the problem is given by:

$$(S'_b - J(S'_w + \lambda\Delta'))\alpha = 0 \quad \Rightarrow \quad S'_b(S'_w + \lambda\Delta')^{-1}\alpha = J\alpha \tag{4.14}$$

In order to find α which corresponds to the maximum value of J, one could find the largest eigenvalue of the matrix $S'_b(S'_w + \lambda\Delta')^{-1}$ and the corresponding eigenvector is α. On the other hand, in cases where $\lambda\Delta$ is close to zero, this matrix may be singular, therefore a gradient ascend algorithm could be used to obtain α. On each iteration of gradient ascend algorithm, value of α may be updated as:

$$\alpha^{n+1} = \alpha^n + \eta(t)\left(\frac{S'_b}{(S_w + \lambda\Delta)} - J(\alpha)\frac{(S'_w + \lambda\Delta')}{(S_w + \lambda\Delta)}\right)\alpha^n \tag{4.15}$$

where

$$\eta(t) = \eta_0\left(1 - \frac{t}{N}\right)$$

In which N is the maximum number of iterations for optimizing α and t is the current iteration. It is obvious in update criterion (4.15), only values of $J(\alpha)$ and $(S_w + \lambda\Delta)$ need to be computed in each iteration.

4.2 Experimental Evaluation

The empirical evaluation of the proposed approach is performed on two different sets of datasets. The first dataset corresponds to a real world data generated from Landsat Multi-Spectral Scanner image. Landsat image **?** is a remote sensing database consisting of color feature of the images. It has 36 columns consisting of 4 spectral bands of 9 pixels in neighbors ranging from 0 to 255. This dataset is categorized into 7 classes. For the purpose of this evaluation, we split

the dataset into 3 categories with a pair of classes each. As we have assumed to have binary classification, a pair of these classes is taken as the training samples and another pair, as the auxiliary unlabeled data.

In the second experiment, we use a synthetic dataset. We create a random 2-dimensional dataset with normal distribution $\mathcal{N}(c, 10.0)$ for each dimension where c is set to be $0, 5, 10, 15$. In order to test the classification, we label $\mathcal{N}(0, 10.0)$ and $\mathcal{N}(10, 10.0)$ as ± 1 to be the main dataset $\mathcal{X}$ and the rest is left unlabeled as the auxiliary dataset $\mathcal{Y}$.

In order to test the performance of the proposed approach in optimizing kernel functions, we modified the LibSVM **?**, an implementation of SVM algorithm in Java. The modified SVM is capable of calling the custom kernel functions particularly transferred kernel obtained from the proposed approach. In these experiments, we use C-SVM with default parameter values, *i.e.* $C = 1$. The proposed approach is tested in two settings: firstly, a FDA-based optimized kernel similar to **?** which does not use the unlabeled data. Secondly, the proposed algorithm that uses transferred data from unlabeled examples. The default kernel for both optimized and transferred kernel is selected to be a polynomial kernel with degree 2. The parameter σ in $h(\mathbf{x}, x)$ is set to be 5 and neighborhood size in KNN algorithm empirically decided to be 15. Java implementation of KNN available in JavaML[1] with Euclidean distance is used to select neighbors in $\mathcal{Y}$.

In order to compare the performance of the proposed kernel, 3 other popular kernels Linear kernel, Polynomial kernel and the RBF kernel are also used. The results of classification obtained from running the kernels with SVM is shown in Table 4.1 and Figure **??**.

Table 4.1 presents the results of classification using SVM with various kernels. For best evaluation, the proposed algorithm is tested with different number of training examples, each

[1] http://java-ml.sourceforge.net

Table 4.1: This table shows the error percentage of each of the datasets and the average error percentage. Each row is dedicated to the results collected for each of the kernels described

Kernel Type	**Synthetic Dataset (test set size: 1500)**					
No. of Training Examples	**16-35**	**36-55**	**56-75**	**76-95**	**96-115**	**Average**
Transferred Kernel	8.57%	8.54%	8.44%	8.58%	8.34%	**8.49%**
Optimized Kernel	8.89%	9.09%	8.73%	8.93%	8.28%	**8.78%**
Linear Kernel	8.94%	9.08%	8.73%	8.93%	8.28%	**8.79%**
Polynomial Kernel	22.09%	13.77%	9.63%	9.56%	9.06%	**12.82%**
RBF Kernel	24.82%	19.88%	15.70%	15.46%	14.25%	**18.02%**
	SatImage (test set size: 448)					
Transferred Kernel	9.59%	9.68%	8.57%	7.58%	7.81%	**8.65%**
Optimized Kernel	9.6%	9.67%	9.6%	8.03%	8.03%	**9.0%**
Linear Kernel	10%	10.17%	10.04%	8.13%	8.48%	**9.36%**
Polynomial Kernel	9.59%	9.68%	9.6%	8.03%	8.03%	**9.0%**
RBF Kernel	44.24%	41.29%	38.43%	35.67%	32.86%	**38.5%**

synthesized from a pair of selected classes. As shown in Table 4.1, the columns represent the number of training examples used to test data. Each entry in the table is computed from the average value of prediction errors obtained from 10 times running the algorithm with the training size defined in the boundary. As it is observed, in some cases predictions of FDA-based and transferred kernel are the same. In these cases, the distribution of unlabeled data enforces the same support vectors as the FDA-based kernel, thus result of classification remains similar.

Performance of the proposed approach in comparison with other kernels are plotted in Figure **??**. The performance of the algorithm is evaluated in terms of the accuracy of classification results using SVM obtained from different kernels. As it is shown, the correct predictions of the proposed algorithm are higher specifically when the number of training examples is limited. It is also worth noting that as the number of training examples increased, the influence we are expecting from the transferred training points are decreased. This means that as the number of training examples increase, the accuracy of SVM using the optimized kernel obtained from our approach becomes comparable to the results obtained from the FDA-based optimized kernel.

4.3 Conclusion

Learning the kernels is important in model selection of the kernel-based algorithms as the type of kernel used highly influences the classification results. Finding a way to automate this step will pave the way towards achieving better accuracy and performance in kernel methods. In this paper, we proposed the Transferred Learning of the kernel approach to optimize a base kernel. This approach is a novel application of transfer learning using unlabeled examples in the context of kernel optimization. Transfer learning in this approach is specifically used to cope with the scarcity of data in the main dataset. On the other hand, developing such methods to learn the kernel, paves the way to the use of learning algorithms with a very small training examples.

The empirical evaluation of the algorithm showed it could effectively work with real world and artificial datasets. The proposed algorithm clearly boosts the accuracy compared to the base kernel. It is even proved to be better than standard FDA in terms of achieving lower error rates.

Acknowledgments

This research has been made possible through the Science Fund Grant “Delineation and 3D Visualization of Tumor and Risk Structures” (DVTRS), No: 1001/PKOMP/817001 by the Ministry of Science, Technology and Innovation of Malaysia.

CHAPTER 5

METHOD 2

5.1 Methodology

In general, to learn the kernel we seek to find the optimal feature space that the inner product of points in that space best defines the similarity between pairs of points in the input space. In order to find the optimal feature space, one has to define the mapping function which is dependent on the nature of the problem and the characteristics of underlying data (that may not be known a priori). This mapping function is implicitly defined in a kernel. In addition, a given kernel function defines relationships between two points within a unique feature space. Therefore, instead of finding the corresponding mapping function, one may find the optimal positive definite kernel to define the feature space of choice. Formally, given an unlabeled dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_n \in \mathbb{R}^d\}$, we search for $\mathcal{H}^*$ as the optimal feature space:

$$\phi^* : \mathcal{X} \rightarrow \mathcal{H}^*, \quad \kappa^*(x_i, x_j) = \langle \phi^*(x_i), \phi^*(x_j) \rangle \tag{5.1}$$

where κ^* is the optimal kernel function (or matrix $\mathcal{K}^*$ accordingly). In order to obtain κ^*, we propose an unsupervised approach which uses the intrinsic characteristics of dataset to learn the optimal kernel. Intrinsic characteristics of the dataset are captured through the determination of influence of each point in input space and reflecting it in the feature space. In this case, if the influence of data points in the structure of dataset is calculated, this measure is then used to determine the relationship between data points. The influence of points in the input space and the feature space are thereafter probabilistically determined *i.e.* the intrinsic structure of the dataset in the feature space is dependent on the structure of the input space determined by the

measure we will introduce. Consequently, the linear combination of data points in the feature space is inferred from the input space which intuitively amounts to inferring the local relation from a global perspective.

Therefore, in the proposed algorithm, we initially calculate the influence of each point on the dataset, thereafter it will be used to guide the learning of κ^*. We model the dataset as a graph and subsequently use *Random Walks* **??** to calculate the influence of each point in the dataset as detailed in Subsection 5.1.1. To perform the random walks, we must first define the distance between points in the input and feature space. In our case, the Euclidean distance is used that its definition in the feature space is described in Subsection 5.1.2. Subsequent to obtaining the measure of influence, the optimal kernel is obtained through linear combination of base kernel functions as the justification is provided in Subsection 5.1.3. As it will be shown, the linear combination of kernel functions could be regarded as concatenation of various feature spaces that ultimately produces $\mathcal{H}^*$. Finally, the optimal combination of kernels is determined through an optimization problem which will be discussed in Subsection 5.1.4.

5.1.1 Determining Structure of a Dataset through Random Walks on the Graph

In order to determine the influence of each point in the dataset, we define a probability space $(\mathcal{D}, \mathcal{C}, \mu)$. In this space, μ is the probability measure defined on the dataset $\mathcal{D}$ and $\mathcal{C}$ is the sets of events *i.e.* the connectivity between points. The probability measure μ is the notion we interchangeably use to indicate the influence of each point in the dataset $\mathcal{D}$.

In order to determine the measure μ for each point in the dataset $\mathcal{D}$, we represent our dataset as a weighted graph. In this representation, each data point corresponds to a vertex $\{v_1, v_2, \ldots, v_n \,|\, v_i \in \mathcal{D}\}$ and each edge between v_i and v_j in the graph is weighted with a function $w(v_i, v_j)$ that satisfies **?**:

1. Matrix $W = (w(v_i, v_j))_{ij}$ is symmetric *i.e.* $W = W^\top$.

2. Function w satisfies $w(v_i, v_j) \geq 0$.

We also introduce $d(v_i)$ to denote the degree of a node in the graph as:

$$d(v_i) = \sum_{j=1}^{n} w(v_i, v_j). \tag{5.2}$$

Having defined w and d, probability of transition between node i and node j at timestamp τ is denoted by p_τ. That is, we define how probable it is to select node j while node i is already selected. Thus, p_1 is calculated as:

$$p_1(v_i, v_j) = \frac{w(v_i, v_j)}{d(v_i)}. \tag{5.3}$$

It is clear that the following holds:

$$\sum_{j=1}^{n} p_\tau(v_i, v_j) = 1. \tag{5.4}$$

Denoting P to be the transition matrix constructed from p_1 for all vertices, one may calculate the transition matrix in successive timestamps (P is a Markov matrix). By treating this model as *Markov Chain* of the consequent probabilities of walking from a vertex to another in Eq. 5.3, the matrix of proximities P^τ at τ could be obtained. As τ increases, the local geometry information is being propagated and accumulated to obtain the global characterization of a graph.

Furthermore, a similar formulation has been used in the diffusion maps **???**. It is shown that the probability p_τ induced from each timestamp t may be used to define a *diffusion distance*

between points:

$$D_\tau^2(v_i, v_j) = \left\| p_\tau(v_i, .) - p_\tau(v_j, .) \right\|^2_{1/\mu(.)} \tag{5.5}$$

Diffusion distance between two points reflects the connectivity in graphs where larger distances imply higher probability of transition between those points.

Since the increasing value of τ reveals the global characteristics of the dataset, we are specifically interested in the probability values where $\tau \to \infty$. To calculate this probability, we define the vector u such that $u = uP$. The vector u is in fact the left eigenvector of the transition matrix P where the eigenvalue is 1. As the graph is connected, from the definition of u, it is clear that:

$$\lim_{\tau \to \infty} P^\tau = \mathbf{1}u \tag{5.6}$$

where $\mathbf{1}$ is the column vector of length n with all the entries are equal to 1. The following also holds for each entry of u:

$$\lim_{\tau \to \infty} p_\tau(v_j, v_i) = u_i. \tag{5.7}$$

In this equation, u_i is called the *Stationary Distribution* of v_i mainly in the parlance of random walks. The value of u_i is given by:

$$u_i = \frac{d(v_i)}{\sum_{j=1}^{n} d(v_j)}. \tag{5.8}$$

Hence, u_i is dependent on the degree of v_i. If w is a shift invariant kernel function referring to similarity of points, u_i is equal to the *Parzen Window* that estimates the density of the respective point **?**.

Stationary distribution of each point has the highest potential to reveal the intrinsic structure of a dataset. Hence, we use the stationary distribution of each point to denote the influence of that point in the dataset. In case w is a similarity function, a larger u_i yields greater influence

on the structure of the graph . In the proposed approach, we use Euclidean distance between two points as function w. When Euclidean distance is used for w, p_τ shows the probability of node i *not* transiting to node j. That is, omitting a point with smaller value of u_i more strongly influences the connectivity and consequently the structure of the graph than points with larger value of u_i. A point with smaller u_i is in close proximity to other points in the graph, therefore, it could be inferred that it is positioned in a dense area. In the definition of μ, we use u_i subject to the use of distance function in calculation of d. Therefore, μ is defined as:

$$\mu(v_i) = 1 - u_i = 1 - \frac{d(v_i)}{\sum_{j=1}^{n} d(v_j)}. \tag{5.9}$$

5.1.2 Euclidean distances and kernels

Kernel functions could be used to define distances between points. As described, $\phi(x)$ is a nonlinear mapping to the feature space. Therefore, the Euclidean distance between two points could be defined between mapped points in the feature space, *i.e.*

$$\Delta_{ij} = \left\| \phi(x_i) - \phi(x_j) \right\| \tag{5.10}$$

which easily could be shown to be equal to:

$$\Delta_{ij}^2 = K_{ii} + K_{jj} - 2K_{ij} \tag{5.11}$$

where Δ_{ij} is the $n \times n$ matrix of distances between two points i and j in the feature space. This equation implies kernel functions may be used to infer the distance between points.

It is also worth stressing that inferring distances from kernel functions may generally be seen as introducing a nonlinear metric. However, the problem of defining a kernel function is

more generic than finding an appropriate distance metric since different kernel functions may produce similar metrics **?**. Therefore, the proposed approach in this paper may be applied in the learning algorithms that heavily depend on distance metrics between data points for their computation.

5.1.3 Combination of Kernels

Linear combinations of kernel functions have recently been used as the basis for the introduction of new kernels. Each kernel in the combination has a specific mapping and a geometric representation in the Hilbert space yielding various characteristics. For example, the linear kernel is considered to assess similarity between points even if they are located distant from each other, while the Gaussian kernel is well-suited for assessing points in proximity to each other. In a combination, each kernel could represent a set of features in which summation corresponds to the fusion of them. Therefore, by combining several mapping functions a feature space that best serves one's goals could be found.

Formally, the combination of kernels is a set of linear summation of b base kernel matrices:

$$G(\mathcal{K}) = \{\mathcal{K} \,|\, \mathcal{K} = \sum_{\ell=1}^{b} \alpha_\ell K_\ell \quad \forall \alpha \in \mathbb{R}\} \tag{5.12}$$

Each kernel function κ in the set $G(\mathcal{K})$ could be interpreted as concatenation of various feature spaces defined by base kernels:

$$\begin{aligned} \kappa(x_i, x_j) &= \overbrace{\langle \alpha_1 \phi_1(x_i), \phi_1(x_j) \rangle}^{\alpha_1 k_1} + \overbrace{\langle \alpha_2 \phi_2(x_i), \phi_2(x_j) \rangle}^{\alpha_2 k_2} + \dots \\ &= \left\langle \begin{pmatrix} \alpha_1 \phi_1(x_i) \\ \alpha_2 \phi_2(x_i) \\ \vdots \end{pmatrix}, \begin{pmatrix} \phi_1(x_j) \\ \phi_2(x_j) \\ \vdots \end{pmatrix} \right\rangle \end{aligned} \tag{5.13}$$

In the set $G(\mathcal{K})$, α_ℓ weighs each of the base kernel matrices K_ℓ with respect to others. In general, two assumptions could be made for the base kernels. In the first case, each kernel is dedicated to a set of distinct feature sets obtained from heterogeneous data sources. In this case, each base kernel is assumed to be suitable for a specific data source. In the second case, base kernels are presumed to be suitable for the problem at hand, *i.e.* , these kernels could be among those that have given good results for similar problems. In this case, all base kernels utilize a single data source. Our approach is in the line of the later case. It should be noted that we do not impose any constraints on the type of base kernels, thus, they could consist same kernels with varying parameters. Specifically, in this paper, we focus on two subsets of $G(\mathcal{K})$:

1. Affine Combination of kernels

$$\mathcal{K} = \{\mathcal{K} \in G(\mathcal{K}),\ \sum_{\ell=1}^{b} \alpha_\ell = 1\} \tag{5.14}$$

   In general, the subspace obtained from affine combination of positive definite matrices in $G(\mathcal{K})$ are symmetric, therefore it is required to constrain the resulting optimal kernel to satisfy positive definiteness condition of a kernel matrix.

2. Convex Combination of kernels

$$\mathcal{K} = \{\mathcal{K} \in G(\mathcal{K}),\ \sum_{\ell=1}^{b} \alpha_\ell = 1, \alpha_\ell \geq 0\} \tag{5.15}$$

   In this case, the set $G(\mathcal{K})$ is always positive definite, therefore the search space to find the optimal kernel is smaller.

In the kernel combinations we consider in this paper, each of the base kernel matrices K_ℓ

is normalized according to the normalization formula **?**:

$$(K'_\ell)_{ij} = \frac{(K_\ell)_{ij}}{\sqrt{(K_\ell)_{ii}(K_\ell)_{jj}}}. \tag{5.16}$$

Normalization of base kernels deters one kernel to dominate the final result in the optimal solution.

5.1.4 Learning the Kernel through influence determination

The objective of the proposed approach is finding the optimal combination of base kernels. In other words, the objective is to find the vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]^\top$ which are the coefficients of the linear combination of base kernels.

As described earlier, the optimal mapping function ϕ^* obtained from combination of base kernels gives rise to the optimal kernel function κ^* and vice versa. We denote two sets $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $\Phi = \{\phi^*(x_1), \phi^*(x_2), \ldots, \phi^*(x_n)\}$ as the sets of points in the input and feature space respectively. The probability of each point $\phi^*(x_i)$ being placed in the correct position in the feature space could be written as a mixture of m distributions, *i.e.* :

$$p(\phi^*(x_i)) = \sum_{c=1}^{m} p(c)p(\phi^*(x_i) \mid c) \tag{5.17}$$

in which $p(\phi^*(x_i) \mid c)$ is an independent distribution. In this case, the probability density is marginalized over parameter c. Probability of c denoted by $p(c)$ is called the mixing coefficient and can be interpreted as the prior probability of points in Φ given c. In general, Eq. (5.17) always holds for a given m. In this paper, we take $m = n$ denoting the assumption that each point is dependent on parameter c. Using Bayes theorem, for a given $\phi^*(x_i)$ we have:

$$p(c \mid \phi^*(x_i)) = \frac{p(c)p(\phi^*(x_i) \mid c)}{\sum_{j=1}^{n} p(j)p(\phi^*(x_i) \mid j)}. \tag{5.18}$$

We specify $p(\phi^*(x_i) \mid c)$ with respect to the probability space $(\Phi, \mathcal{C}, \mu)$ as:

$$p(\phi^*(x_i) \mid c) = \begin{cases} 1 - \frac{d(\phi^*(x_i))}{\sum_{j=1}^{n} d(\phi^*(x_j))} & \text{if} \quad i = c \\ 0 & \text{if} \quad i \neq c. \end{cases} \tag{5.19}$$

This equation could be interpreted as an extreme case of Gaussian distribution that could be further simplified as:

$$p(\phi^*(x_i) \mid c) = \delta_{ci}\mu(\phi^*(x_i)) \tag{5.20}$$

where δ_{ci} is the Kronecker delta. This value determines the probability of parameter c for a given point in the feature space. Moreover, from Eq. (5.18) and (5.20) we have:

$$p(c \mid \phi^*(x_i)) \propto p(c)\delta_{ci}\mu(\phi^*(x_i)) \tag{5.21}$$

In order to determine the utmost probability for each point, we maximize the conditional probability $p(c \mid \phi^*(x_i))$ as:

$$\max \sum_{i=1}^{n}\sum_{c=1}^{n} p(c)\delta_{ci}\mu(\phi^*(x_i)) \tag{5.22}$$

It should be noted that in contrary to the maximum likelihood paradigm that the learning algorithm is formulated as a single probability density function and the objective is to maximize the joint probabilities of each feature in that distribution, in Eq. (5.22) we formulate

the objective as several independent probability distributions that we intend to maximize the union of them. In other words, the probability that each point is placed in its correct position is identified as an event which we seek to maximize its occurrence.

In order to be able to maximize (5.22), we set $p(c)$ to be the stationary distribution of x_c in the input space, *i.e.* $p(c) = \mu(x_c)$. Consequently, a point that has a larger influence in the input space will be given higher chance to be assigned to its correct position in the feature space or conversely it needs to be moved lesser.

Therefore, from Eq. (5.22) and Eq. (5.9) the objective of the proposed approach is formulated in an optimization that searches for an optimal α:

$$\max_{\alpha} \ (\mathbf{1} - u)^{\top}(\mathbf{1} - u^{*}) \tag{5.23}$$

where u^* is the vector constructed from stationary distributions of each point $\phi^*(x_i)$ in the feature space according to Eq. (5.8). With respect to Eq. (5.8), we can write each entry u_i^* as:

$$u_i^* \propto d(\phi^*(x_i)) \tag{5.24}$$

where d from Eq. (5.2) and Eq. (5.11) in the feature space is:

$$\begin{aligned} d(\phi^*(x_i)) &= \sum_{j=1}^{n} \Delta_{ij} = \sum_{j=1}^{n} \mathcal{K}_{ii}^* + \mathcal{K}_{jj}^* - 2\mathcal{K}_{ij}^* \\ &= \sum_{\ell=1}^{b} \alpha_\ell \sum_{j=1}^{n} (K'_\ell)_{ii} + (K'_\ell)_{jj} - 2(K'_\ell)_{ij} \\ &= \sum_{\ell=1}^{b} \alpha_\ell \sum_{j=1}^{n} (B_\ell)_{ij} \end{aligned} \tag{5.25}$$

where B_ℓ is the matrix of distances of ℓth normalized base kernel.

Consequently, Eq. (5.23) could be simplified as:

$$\begin{aligned} \max_{\alpha} \ (\mathbf{1}-u)^{\top}(\mathbf{1}-u^{*}) \ &\Rightarrow \ \max_{\alpha} \ (u-\mathbf{1})^{\top}u^{*} \\ &\Rightarrow \ \max_{\alpha} \ (u-\mathbf{1})^{\top}\Delta \end{aligned} \tag{5.26}$$

where it can be further simplified as:

$$\max_{\alpha} \ -\sum_{\ell=1}^{b}\alpha_{\ell}\sum_{i=1}^{n}\sum_{j=1}^{n}(1-u_{j})(B_{\ell})_{ij} = -\alpha^{\top}s \tag{5.27}$$

In this equation, s can be interpreted as a column vector with size b of weighted sum of distances obtained from each of the base kernels. Consequently, the value that has to be optimized is in fact a linear combination over α. In order to solve this optimization problem, we consider two cases based on the types of kernel combinations we discussed earlier:

1. Affine combination of base kernels: The main issue in this case is the constraints that must be considered on the kernel matrix to be positive definite. Therefore, the objective of the proposed approach in affine combination of kernels is formulated as an instance of Semidefinite Programming. Semidefinite programming searches for the optimal solution in the intersection of semidefinite cone and the affine constraints. That is, the goal is to optimize a linear function of positive semidefinite matrices subject to linear constraints. In this way the optimization is convex, therefore it does not suffer from local optima.

$$\begin{aligned} \min_{\alpha} \quad & \alpha^{\top}s \\ \text{subject to} \quad & \left(\sum_{\ell=1}^{b}\alpha_{\ell}K'_{\ell}\right) \succeq 0 \\ & \mathbf{1}^{\top}\alpha = 1 \end{aligned} \tag{5.28}$$

2. Convex Combination of base kernels: In this case, the search space is limited to the positive definite matrices, thus, the optimization could be solved simpler than SDP. Hence, the optimal kernel could be obtained from the following Linear Programming problem:

$$\begin{aligned} \min_{\alpha} \quad & \alpha^\top s \\ \text{subject to} \quad & \mathbf{1}^\top \alpha = 1, \alpha \geq 0 \end{aligned} \tag{5.29}$$

The solution to the above optimization is in fact lies in the intersection of the conic combination of α (in the objective function) and the hyperplane obtained from the affine combination of α.

It should be noted that one could describe the proposed approach as determining a weighted consensus of the base kernels. The consensus in the feature space is achieved using the guidance of the correspondence of the Euclidean distance in the feature space and the input space considering the influence of points in the input space.

5.2 Discussion

The intuition of the approach described in the previous section could be differently described. We compute the stationary distribution of each point u_i in the input space and reflect this value in the feature space such that those with smaller u_i remain in dense areas. This means increasing the density of points in the feature space with higher influence in structure of the dataset. That is, maximization of $\alpha^\top s$ increases the distances between each pair of data points according to their contribution to the structure of the dataset in the input space. It should also be noted that as indicated in Eq. (5.4), $\sum_{j=1}^{n} u_j^* = 1$, thus, maximizing Eq. (5.23) naturally decrease u_i^* in the dense areas. This could also be justified by the diffusion distance in Eq.

(5.5) where the point with smaller differences in u_i in the graph, are more strongly connected. Keeping points with small u_i in proximity creates a dense area through which the other points are connected.

As an example consider two distinct points x_1 and x_2 in the dataset, each with stationary distributions u_1 and u_2, respectively. Assuming $u_1 > u_2$ means x_2 is contributing more to the dataset than x_1. Maximizing their distances in the feature space would increase the distances between $\phi^*(x_1)$ and other points more than $\phi^*(x_2)$ *i.e.* $\mathcal{K}^*_{1j} > \mathcal{K}^*_{2j}$. Furthermore, if u_1 and u_2 are small with respect to other points, these points will be less stretched apart while the points with larger u_i will be more pushed away. Thus, points with less influence on the mapped dataset would scatter in less dense areas that have less effect in decision making processes.

Having the influential points concentrated in dense areas, it is expected that the proposed approach outperforms the existing kernels specifically in maximum margin algorithms and dimensionality reduction. Variations of maximum margin algorithms, either for classification or clustering finds a separating hyperplane in the area with lower density. Our approach seeks to find the influential points in the dataset and keep them in the dense areas while allowing the rest of points scatter around according to their probability of transition. Therefore, similarity between points is highly dependent on those with smaller stationary distribution. Similarly, this approach is justified for kernel-based dimensionality reduction algorithms like kernel-PCA. Kernel-PCA seeks to extract non-linear principle components in the dataset. Hence, concentrating on the points with higher influence on the dataset will increase the possibility of finding an intrinsic pattern in the data. Our empirical evaluation on SVM and kernel-PCA proved the effectiveness of the derived kernel from the proposed method.

5.3 Experiments

In this section, we present the empirical results obtained from applying the proposed approach to various datasets. In general, we go through the following stages to obtain the optimal kernel: firstly, kernel matrix K_c' for each of the normalized base kernels and the vectors u and correspondingly s from the dataset are constructed. In the next stage, the optimization problem in Eq. (5.28) or (5.29) is solved. The optimization is performed using `CVX`. `CVX` is a package for specifying and solving convex programs implemented in Matlab **??**. Finally, the coefficients of the optimal kernel obtained from the optimization process, is tested in the target kernel-based algorithms.

In order to show the effectiveness of the proposed approach, three experiments are performed. In the first experiment in Subsection 5.3.1, we illustrate the effectiveness of the optimal kernel obtained from the proposed approach in highly similar images. This experiment is performed to examine how the optimal kernel obtained from the proposed approach is able to discriminate input features which may be highly similar. It is expected that a good kernel returns close values for similar images while the distinction between them are well reflected in a specific dataset. In the second experiment, in Subsection 5.3.2, we illustrate the results of applying optimal kernel in an unsupervised algorithm *i.e.* Kernel-PCA for dimensionality reduction. A good kernel will capture the most valuable components of the data better in order to evaluate the results of Kernel-PCA. We assess the data obtained from Kernel-PCA in KNN to be able to compare results. In the last experiment in Subsection 5.3.3, the results of applying the proposed approach in a supervised algorithm *i.e.* SVM is illustrated. The best kernel is decided to be the one with higher accuracy in SVM. As it will be shown, the results of the optimal kernel are comparatively better than each of the base kernels in various cases.

5.3.1 Optimal kernel from Highly Similar Images

In the first experiment, we show the impact of running the proposed approach in highly similar images. The teapot dataset from Weinberger and Saul **?** is used to run this experiment. This dataset originally consists of RGB colored images of size 76×101. Images are taken sequentially from a 360° rotating teapot. We select 12 consecutive images from this dataset as illustrated in Fig. **??**. We use the following kernels: Linear, Polynomial and Gaussian kernel. These kernels are used to construct the kernel matrices as well as the distance matrix which are illustrated as colored block images in Fig. **??**. We wish to stress Gaussian kernel's bandwidth is selected manually to be two different values with a huge gap. This once more shows how effective the parameters of kernels are and how difficult it may be to find an appropriate value for that. For the polynomial kernel we set the degree parameter to 2. Having decided on the 3 base kernels, the optimal kernel is constructed using the proposed approach and the resulting matrix is illustrated in Fig. **??**. In Fig. **??**, points with similar colors have closer value in the kernel matrix. As it may clearly be inferred from Fig. **??**, although various images are used, the diversity has not been captures well in base kernels while the kernel constructed from the proposed method shows the differences and reflects similar colors for similar images. In general, it is expected that a good kernel function will return very close values for similar points while the difference between data points is also showed. Two eigenvectors corresponding to two largest eigenvalues are plotted in Fig. **??**. As it is expected, more influential points, that the similarity of other points depend on, are grouped closely and other points with less influence are scattered further.

5.3.2 Dimensionality reduction for Face Recognition

The formulation of an unsupervised method to learn the kernel paves the way for the use of the proposed approach in any of the learning algorithms that no labeled data is required. In order to

Table 5.1: In this table, results (%) of face dataset with specified kernel functions are shown.

Ex.	Dim.	L	P	G_1	G_2	Affine Comb.					Convex		
						LP	LG_1	LPG_1	LPG_2	LPG_1G_2	LP	LG_1	LPG
50	20	77.33	86	50	57.33	90.67	90	92	**93.33**	**93.33**	86	77.33	8
	30	90	92	50	58	92	77.33	92	92	**93.33**	92	90	9
	40	92	**92.67**	50	67.33	92	92	92	92	92	**92.67**	92	91.6
100	20	86	96	50	68	97.33	86	96.67	**97.33**	**97.33**	96	86	9
	30	95.33	96.67	50	68.67	96.67	95.33	**97.33**	**97.33**	**97.33**	96.67	95.33	96.6
	40	96.67	96.67	50	66	**97.33**	96.67	**97.33**	**97.33**	**97.33**	96.67	96.67	96.6
150	20	44	46.67	25	27.67	**47**	44	45.33	44.6	44.67	46.67	44	46.6
	30	47.33	47.67	25	27.67	**49.33**	47.33	**49.33**	49	49	46.67	47.33	47.6
	40	49	49.33	25	27	49.33	49	49.67	48	48.67	49.33	49	49.3

test our approach we used Kernel-PCA. Kernel-PCA is a non-linear dimensionality reduction technique that has recently been proposed which tries to find a non-linear projection of data that maximizes the variance.

Since dimensions of face images are huge but many of the neighboring pixels have similar colors, dimensionality reduction in general and Kernel-PCA in particular has been used in face recognition problems **?**. In our experiment, we use Carnegie Mellon PIE face dataset **?** to test our method. A subset of images in the dataset is selected and used to obtain the optimal kernel which is later used in Kernel-PCA. The feature vector of the images is constructed from raw pixel intensity values. Kernel-PCA is used to project the data to lower dimensions. Subsequently, the projected examples in the dataset is used to train the KNN classifier with neighborhood size 5. The result of classification of images is shown in Table 5.1. In this table, L stands for linear kernel and P for polynomial kernel. G_1 and G_2 are also indicate the Gaussian kernel with bandwidth 10000 and 150000 respectively. The combination of kernels specified using the concatenation of initial letters. Number of images used to derive the optimal kernel is shown in the first column. In the second column, various target dimensionalities are given. In other columns, percentage of correct classification obtained from applying KNN on the dataset of reduced dimensions is shown. The results show the proposed approach is achieving better results from KNN especially with smaller training examples. It could also be observed that the number of base kernels and their initial values are influencing results of the kernel learning.

Table 5.2: In this table results of running SVM with various kernels are illustrated. In last column, the results of proposed algorithm with different base kernels are reported

Ex.	L	P	G_1	G_2	Affine Comb.					Convex Com			
					LP	LG_1	LPG_1	LPG_2	LPG_1G_2	LP	LG_1	LPG_1	L
50	80.33	76.66667	80.33	80	**87.67**	82.33	82.33	82	79.33	80.33	46.6	40	
100	87.67	83	72	88	87.33	87.67	87.67	88	88.67	88.67	46.6	40	
150	87.67	84.33	72	72	**89.67**	89.33	**89.67**	**89.67**	89	87.67	46.6	46.67	
200	87	84.67	72	87	**89.67**	89	**89.67**	**89.67**	88.67	87.33	87	46.67	

5.3.3 Text Classification

In the last experiment, we evaluate our approach in a text classification task. We choose 20-newsgroup dataset collected from 20 different newsgroups represented in 26214 size feature vectors. For the purpose of this classification we use C-SVC setting in Java implementation of LibSVM **?**. To have an equal setting for all our experiments, we leave parameter C to its default value 1.0. To build training set for binary classification, we select different number of examples from the same class and set their class value to $+1$ as the positive examples. Negative training examples are also randomly selected from the other classes. Similarly we create a test set from examples that are not being selected in training set. The results of running SVM with various kernels are shown in Table 5.2. As it is generally expected for the text classification, the performance of linear kernel is comparatively better among the base kernels. As it is shown in the table, if the base kernels are selected appropriately the result of the kernel optimization proposed in this paper is noticeably better. It may also be seen that in the cases where two base kernels are provided, better pair of them produce better results although the classification result obtained from the combination of good and bad base kernels is also tends toward better one.

5.4 Conclusion

In this paper, we proposed an approach to learn the kernel function that seeks to find a mapping to the feature space with emphasis on influence of intrinsic characteristics of the dataset in the feature space. The global intrinsic of dataset are reflected in terms of stationary probability

distribution of each point. It could clearly be seen that the global characteristics, as used here, may be beneficial to the performance of the kernel especially in cases where labeled data is not available. In the proposed approach, the intrinsic characteristics captured in the input space used to find a combination of kernel functions to concatenate feature spaces using the available data. As it is discussed, the linear combination of kernels could be interpreted as weighting the mapping functions to form the feature space of choice for the dataset at hand. The empirical results in various real world problems showed effectiveness of the proposed method in finding an optimal kernel without any labeled data. The optimal kernel obtained from the proposed method was used in two different kernel-based algorithms: SVM and Kernel-PCA as two popular supervised and unsupervised algorithms respectively to demonstrate the performance of the approach.

Acknowledgment

This research has been made possible through the Science Fund Grant “Delineation and 3D Visualization of Tumor and Risk Structures“ (DVTRS), No: 1001/PKOMP/817001 by the Ministry of Science, Technology and Innovation of Malaysia.

CHAPTER 6

CONCLUSION

T-that's all folks. Have fun with LaTeX!

REFERENCES

IPS (2007). *A Guide to the Preparation, Submission and Examination of Theses*, Institute of Graduate Studies, Universiti Sains Malaysia, Penang, Malaysia.
URL: *http://www.ips.usm.my*

Lim, L. T. (2007). LaTeX: Beautiful typesetting, [Online]. [Accessed January 5, 2010]. Available from World Wide Web: http://liantze.googlepages.com/latextypesetting.

Mittelbach, F., Goossens, M., Braams, J., Carlisle, D. and Rowley, C. (2004). *The LaTeX Companion*, Addison-Wesley Series on Tools and Techniques for Computer Typesetting, 2nd edn, Addison-Wesley, Boston, MA, USA.

Oetiker, T., Partl, H., Hyna, I. and Schlegl, E. (2006). *The Not So Short Introduction to LaTeX* 2_ε, 4.2 edn.

Roberts, A. (2005). Getting to grips with LaTeX, [Online]. http://www.andy-roberts.net/misc/latex/index.html.
URL: *http://www.andy-roberts.net/misc/latex/index.html*

APPENDICES

APPENDIX A

DATA USED

Put some test data here.

APPENDIX B

UML DIAGRAMS

Yet another dummy placeholder for appendix material.