STOCKHOLM UNIVERSITY

BACHELOR THESIS

Useful Applications in Statistical Learning with Reproducing Kernel Hilbert Spaces

Author:
Yingjie CAO

Supervisor:
Yishao ZHOU

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Sciences

in Mathematics

April 12, 2016

Declaration of Authorship

I, Yingjie CAO, declare that this thesis titled, "Useful Applications in Statistical Learning with Reproducing Kernel Hilbert Spaces" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

STOCKHOLM UNIVERSITY

Abstract

Department of Mathematics

Bachelor of Sciences

Useful Applications in Statistical Learning with Reproducing Kernel Hilbert Spaces

by Yingjie CAO

This paper presents a general reproducing kernel Hilbert Spaces (RKHS) framework with its various applications in statistical learning area. This theory has been around for quite some time and has been widely used in nonlinear regression and classification problems. Kernel methods, which map data from low-dimensional space into higher-dimensional space (RKHS), can be transferred in many classical statistical learning algorithms. This paper can be roughly divided into two parts. In the first part, the writer attempts to take the reader from a very basic understanding of fields through Hilbert spaces, into reproducing kernel Hilbert spaces. In the second part, the writer want to show reader the abundant applications of kernel methods in statistical learning algorithms, with algorithms and real-world examples.

Keywords: RKHS, kernel methods, statistical learning, SVM

Acknowledgements

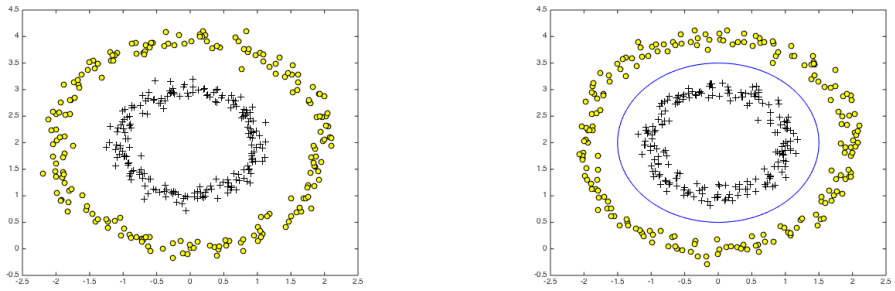
This project would not have been possible without the support of many people. Many thanks to my adviser, Yishao Zhou, who read my numerous revisions and helped make some sense of the confusion. Also thanks to my roommate, Jia Xu, who offered guidance and support. Thanks to School of Mathematical Sciences, Fudan University which offers me a great experience exchanging to Stockholm University. Thanks to the Department of Mathematics, Stockholm University which provides me comfortable and convenient research environment. And finally, thanks to my parents who endured this long process with me, always offering support and love.

Chapter 1

Introduction

1.1 Nonlinear Problem

Consider two concentric clouds of points (Figure3.1), how can we make a boundary between these data points with minimum wrongly-matched error? In other word, what is the best strategy to classify nonlinear datasets?



(A) Two Groups of Data (B) Boundary of Groups

FIGURE 1.1: Example of Nonlinear Data

As shown above, they are linearly non-separable. Intuitively, we know a good boundary is a circle, something like that (Figure1.1b).

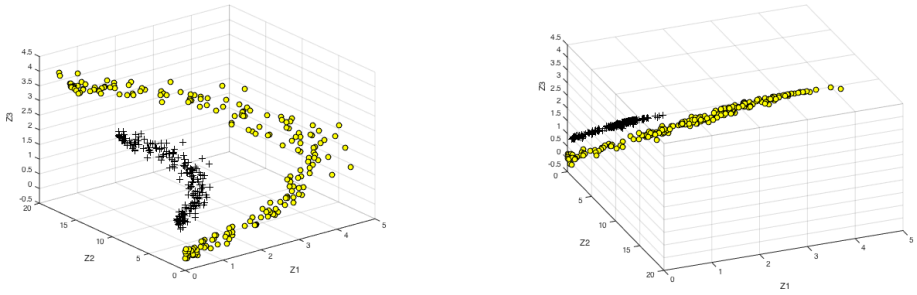
Though this classification example is simple, it is still tricky for computers to solve it. The first thing is, the boundary is no longer a line. How can we apply our previous knowledge in linear classification problems to this example? Well, we might think about adding features, from previous two features X_1, X_2 to five features: $X_1, X_2, X_1^2, X_2^2, X_1X_2$. Therefore, the boundary can be written in this form:

$$a_1X_1 + a_2X_1^2 + a_3X_2 + a_4X_2^2 + a_5X_1X_2 + a_6 = 0 \tag{1.1}$$

This idea can be approached in another way. Let $Z_1 = X_1, Z_2 = X_1^2, Z_3 = X_2, Z_4 = X_2^2, Z_5 = X_1X_2$, (1.1) can be written as:

$$\sum_{1}^{5} a_iZ_i + a_6 = 0 \tag{1.2}$$

Given a map $\varphi : \mathbb{R}^2 \to \mathbb{R}^5$, the two groups of data points can be easily divided by the five dimensional hyperplane. In our example, we can simplify the five-dimension space into three dimensions, with $Z_1 = X_1^2, Z_2 = X_2^2, Z_3 = X_2$, the two groups of data are shown as (Figure1.2a). Through rotation (Figure1.2b), the data points can be easily separated by a plane.



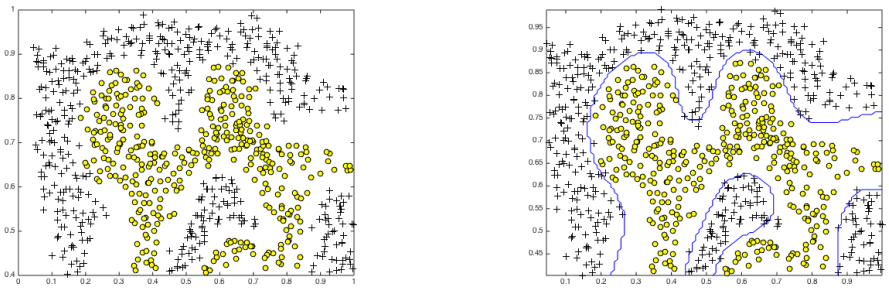
(A) AZ=-37.5° , EL=30° (B) AZ=22° , EL=-40°

FIGURE 1.2: Nonlinear Data in 3D Space

Amazing? It is. Through the map $\varphi : \mathbb{R}^2 \to \mathbb{R}^3$, linearly non-separable data groups can be separated by a hyperplane. Keep this point in mind and let's continue with another example which might be more complicated.

1.2 Complex System

Adding more features seems like a feasible idea, yet it fails encountering with complex system. Another example of two groups of data are given below (Figure1.3a), since the shape of boundary (Figure1.3b) is no longer simple geometry, it is hard to tell how many polynomial features we should add. Though we can give an answer in practice via method like Bias-Variance Tradeoff, it is still bothering and confusing.



(A) Two Groups of Data (B) Boundary of Groups

FIGURE 1.3: Example of Complex System

1.3 The Organization of the Paper

In Chapter 2, we introduce reproducing kernel Hilbert spaces(RKHS), which is a core concept in this paper. In Chapter 3, we develop kernel method which own its name from using kernel functions on RKHS, and then apply kernel method to several classical statistical learning algorithms. In Chapter 4, we make an experiment by using these modified algorithms into a real-world example, in the meanwhile showing readers how to use kernel function in practice.

Chapter 2

Reproducing Kernel Hilbert Spaces

2.1 Regularization Problems

A general class of regularization problems has the form (for more detail, see Appendix A)

$$\min_{f \in \mathcal{H}} \left[\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \right] \tag{2.1}$$

where $L(y, f(x))$ is a loss function, $J(f)$ is a penalty functional, and $\mathcal{H}$ is a space of functions on which $J(f)$ is defined.

2.2 Some Facts About RKHS

Definition 2.2.1. *An evaluation functional over the Hilbert space $\mathcal{H}$ is a linear functional $\mathcal{F}_x : \mathcal{H} \to \mathbb{R}$ that evaluates each function in the space at the point x, or*

$$\mathcal{F}_x[f] = f(x) \quad \textit{for all } f \in \mathcal{H} \tag{2.2}$$

Definition 2.2.2. *A Hilbert space $\mathcal{H}_K$ is a reproducing kernel Hilbert space (RKHS) if the evaluation functions are bounded, i.e. if for all x there exists some $M > 0$ such that*

$$|\mathcal{F}_x[f]| = |f(x)| \leq M \|f\|_{\mathcal{H}_K} \quad \textit{for all } f \in \mathcal{H}_K \tag{2.3}$$

While this condition might seem obscure or specific, it is actually quite general and is the weakest possible condition that ensures us both the existence of an inner product and the ability to evaluate each function in the space at every point in the domain. In practice, it is difficult to work with this definition directly. We would like to establish an equivalent notion that is more useful in practice. To do this, we need the "reproducing kernel" from which the reproducing kernel Hilbert space takes its name.

First, from the definition of the reproducing kernel Hilbert space, we can use the Riesz representation theorem to prove the following property.

Theorem 2.2.1. *[9] Let Ω be a set, for example a subset of $\mathbb{R}^d$ or $\mathbb{R}^d$ itself. If $\mathcal{H}_K$ is a RKHS, then for each $x \in \Omega$ there exists a function $K_x \in \mathcal{H}_K$ (called the representer of x) with the reproducing property*

$$\mathcal{F}_x[f] = \langle K_x, f \rangle_{\mathcal{H}_K} = f(x) \quad \textit{for all } f \in \mathcal{H}_K \tag{2.4}$$

This allows us to represent our linear evaluation functional by taking the inner product with en element of $\mathcal{H}_K$. Since K_x is a function in $\mathcal{H}_K$, by

the reproducing property, for each $y \in \Omega$ we can write

$$K_x(y) = \langle K_x, K_y \rangle_{\mathcal{H}_K} \tag{2.5}$$

The reproducing kernel of $\mathcal{H}_K$ is a function $K : \Omega \times \Omega \to \mathbb{R}$, defined by $K(x, y) := K_x(y)$. In general, we have the following definition of a reproducing kernel.

Definition 2.2.3. *A function $K : \Omega \times \Omega \to \mathbb{R}$ is a reproducing kernel if it is symmetric, i.e. $K(x, y) = K(y, x)$, and positive definite:*

$$\sum_{i,j=1}^{N} a_i a_j K(x_i, x_j) \geq 0 \tag{2.6}$$

for any $n \in \mathbb{N}$ and every set of real numbers $\{a_1, a_2, \ldots, a_N\}$ and $\{x_1, x_2, \ldots, x_N\}$, $x_i \in \Omega$.

Having this general notion of a reproducing kernel is important because it allows us to define an RKHS in terms of its reproducing kernel, rather than attempting to derive the kernel from the definition of the function space directly. The following theorem formally establishes the relationship between the RKHS and a reproducing kernel.

Theorem 2.2.2 (Moore–Aronszajn Theorem)**.** *To every positive definite function K on $\Omega \times \Omega$ there corresponds a unique RKHS $\mathcal{H}_K$ of real valued functions on Ω and vice versa.*

Corollary 2.2.2.1. *Given K, we can construct $\mathcal{H}_K$ in the form of all finite linear combinations $\sum_{i=1}^{N} a_i K_{x_i}$ with $x_i \in \Omega$ and limits of such function as the x_i become dense in Ω, in the norm induced by the inner product*

$$\langle K(\cdot, x_i), K(\cdot, x_j) \rangle_{\mathcal{H}_K} = K(x_i, x_j) \tag{2.7}$$

If $g(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$, then

$$\begin{aligned} \|g\|^2_{\mathcal{H}_K} &= \langle g, g \rangle_{\mathcal{H}_K} = \langle \alpha_i K(\cdot, x_i), \alpha_j K(\cdot, x_j) \rangle_{\mathcal{H}_K} \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} \langle K(\cdot, x_i), K(\cdot, x_j) \rangle_{\mathcal{H}_K} \alpha_i \alpha_j \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} K(x_i, x_j) \alpha_i \alpha_j \end{aligned} \tag{2.8}$$

Corollary 2.2.2.2. *If $K(x, y)$ has a representation of the form*

$$K(x, y) = \sum_i \gamma_i \phi_i(x) \phi_i(y) \tag{2.9}$$

with

$$\int_\Omega \phi_\xi(s) \phi_\eta(s) d\mu(s) = \begin{cases} 1, & \xi = \eta \\ 0, & \textit{otherwise} \end{cases} \tag{2.10}$$

where μ is some measure on Ω, then

$$\langle f, g\rangle_{\mathcal{H}_K} = \sum_i \frac{f_i g_i}{\gamma_i} \tag{2.11}$$

where $f_i = \int \phi_i(s) f(s) d\mu(s)$ and similarly for g_i. In particular,

$$\langle \phi_\xi, \phi_\eta\rangle_{\mathcal{H}_K} = \begin{cases} \frac{1}{\gamma_\xi}, & \xi = \eta \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

Theorem 2.2.3. *[11] Let $\{\phi_i\}_{i=1}^M$ be M functions on Ω^2 with the property that the $M \times N$ matrix T with ijth entry $\phi_i(x_j)$ is of rank M. Let $L(y_j, f)$ be a functional of f which depends on f only through $f(x_j) = f_j$. Then the solution to the problem: find $f \in span\{\phi_i\} + h$ with $h \in \mathcal{H}_K$ to minimize*

$$\sum_{i=1}^N L(y_j, f(x_j)) + \lambda \|h\|^2_{\mathcal{H}_K} \tag{2.13}$$

has a representation of the form

$$f(\cdot) = \sum_{i=1}^M c_i \phi_i(\cdot) + \sum_{j=1}^N d_i K(\cdot, x_j) \tag{2.14}$$

Corollary 2.2.3.1. *Suppose that $\tilde{g}(x) = g(x) + \rho(x)$, with $\rho(x) \in \mathcal{H}_K$, and orthogonal in $\mathcal{H}_K$ to each of $K(x, x_i), i = 1, \ldots, N$. Then*

$$\sum_{i=1}^N L(y_i, \tilde{g}(x_i)) + \lambda J(\tilde{g}) \geq \sum_{i=1}^N L(y_i, g(x_i)) + \lambda J(g) \tag{2.15}$$

with equation iff $\rho(x) = 0$.

2.3 Apply RKHS in Regularization Problems

An important subclass of problems of the form (2.1) are generated by a positive definite kernel $K(x, y)$, and the corresponding space of functions $\mathcal{H}_K$ as a reproducing kernel Hilbert space. The penalty functional J is defined in terms of the kernel as well.

Let $x, y \in \mathbb{R}^p$. We consider the space of functions generated by the linear span of $K(\cdot, y), y \in \mathbb{R}^p$. Suppose that K has an eigen-expansion

$$K(x, y) = \sum_{i=1}^\infty \gamma_i \phi_i(x) \phi_i(y) \tag{2.16}$$

with $\gamma_i \geqslant 0, \sum_{i=1}^\infty \gamma_i^2 \leqslant \infty$. Elements of $\mathcal{H}_K$ have an expansion in terms of these eigen-functions,

$$f(x) = \sum_i^\infty c_i \phi_i(x) \tag{2.17}$$

According to (**??**), the norm induced by K can be shown as

$$\|f\|^2_{\mathcal{H}_K} = \langle f, f\rangle_{\mathcal{H}_K} = \sum_{i=1}^{\infty} \frac{f_i^2}{\gamma_i} = \sum_{i=1}^{\infty} \frac{c_i^2}{\gamma_i} < \infty \tag{2.18}$$

where

$$f_i = \int \phi_i(s) f(s) d\mu(s) = \int \phi_i(s) \big(\sum_j c_j \phi_j(s)\big) d\mu(s) = c_i \tag{2.19}$$

The penalty functional in (2.1) for the space $\mathcal{H}_K$ is defined to be the squared norm $J(f) = \|f\|^2_{\mathcal{H}_K}$. The quantity $J(f)$ can be interpreted as a generalized ridge penalty, where functions with large eigenvalues in the expression (2.18) get penalized less, and vice versa.

Rewriting (2.1) we have

$$\min_{f \in \mathcal{H}_K} \left[\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|^2_{\mathcal{H}_K} \right] \tag{2.20}$$

or equivalently,

$$\min_{\{c_j\}_1^\infty} \left[\sum_{i=1}^{N} L(y_i, \sum_{j=1}^{\infty} c_j \phi_j(x_i)) + \lambda \sum_{j=1}^{\infty} \frac{c_j^2}{\gamma_j} \right] \tag{2.21}$$

Accordance to Theorem 2.2.3 and Corollary 2.2.3.1, the solution for (2.20) can be shown as finite-dimensional, and has the form

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i). \tag{2.22}$$

with Corollary 2.2.2.1,

$$\|f\|^2_{\mathcal{H}_K} = \sum_{i=1}^{N} \sum_{j=1}^{N} K(x_i, x_j) \alpha_i \alpha_j. \tag{2.23}$$

Therefore, the regularization problem can be simplified into a finite-dimensional criterion

$$\min_{\boldsymbol{\alpha}} L(\boldsymbol{y}, \boldsymbol{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{\alpha} \tag{2.24}$$

where $\boldsymbol{\alpha}$ is a vector with ith entry $\alpha(i)$, $\boldsymbol{K}$ is a $N \times N$ matrix with ijth entry $K(x_i, x_j)$.

2.4 Custom Kernels

When we are dealing with regularization problem with kernel methods, it is always important to choose appropriate kernel functions, since the performance of kernel algorithm is highly dependent on the kernel. Though so called "best kernel function" always relys on the specific problem, it is better to have some prior knowledge of most commonly used kernels.

Below is a list of positive definite kernels available from the existing literature.

1. Linear Kernel
   The Linear kernel is the simplest kernel function. Kernel algorithms using a linear kernel are often equivalent to their non-kernel counterparts.
   $$k(x, y) = x^{\mathrm{T}} y + c$$

2. Polynomial Kernel
   The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized.
   $$k(x, y) = (\alpha x^{\mathrm{T}} y + c)^d$$

3. Radial Basis Function Kernel

   (a) Gaussian Kernel
   It is the most commonly used kernel function and also acts well in many cases.
   $$k(x, y) = \exp(-\frac{\|x - y\|^2}{2\sigma^2})$$
   The adjustable parameter σ plays a major role in the performance of the kernel, and should be carefully tuned to the problem at hand. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. In the other hand, if underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noise in training data.

   (b) Exponential Kernel
   The exponential kernel is closely related to the Gaussian kernel, with only the square of the norm left out.
   $$k(x, y) = \exp(-\frac{\|x - y\|}{2\sigma^2})$$

   (c) Laplacian Kernel
   The Laplace Kernel is completely equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter.
   $$k(x, y) = \exp(-\frac{\|x - y\|}{\sigma})$$

   (d) ANOVA Kernel
   The ANOVA kernel is said to perform well in multidimensional regression problems (Hofmann, 2008).
   $$k(x, y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

4. Rational Quadratic Kernel
   The Rational Quadratic kernel is less computationally intensive than

the Gaussian kernel and can be used as an alternative when using the Gaussian becomes too expensive.

$$k(x,y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2 + c}$$

5. Cauchy Kernel
   The Cauchy kernel comes from the Cauchy distribution (Basak, 2008). It is a long-tailed kernel and can be used to give long-range influence and sensitivity over the high dimension space.

$$k(x,y) = \frac{1}{1 + \frac{\|x-y\|^2}{\sigma^2}}$$

Below are some non-positive definite kernels which should be considered carefully when applying them as reproducing kernels, since they are conditionally positive definite. Besides, these kernels have specific role in some machine learning algorithms, i.e. neural network.

1. Hyperbolic Tangent (Sigmoid) Kernel
   The Sigmoid Kernel comes from the Neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons. Since SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network, this kernel was quite popular for support vector machines. Also, despite being only conditionally positive definite, it has been found to perform well in practice.

$$k(x,y) = \tanh(\alpha x^{\mathrm{T}} + c)$$

   There are two adjustable parameters in the sigmoid kernel, the slope α and the intercept constant c. A common value for α is $1/N$, where N is the data dimension.

2. Log Kernel
   The Log kernel seems to be particularly interesting for images, but is only conditionally positive definite.

$$k(x,y) = -\log(\|x-y\|^d + 1)$$

Chapter 3

Applications in Different Statistical Learning Algorithms

3.1 Kernel PCA

Standard linear principal components are obtained from the eigenvectors of the covariance matrix, and give directions in which the data have maximal variance. Kernel principal component analysis (kernel PCA) is the method to expand the features by non-linear transformations, and then apply PCA in this transformed feature space.

Lemma 3.1.1. *Given a data matrix X, the inner-product (gram) matrix $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} = \langle x_i, x_j \rangle$. The double-centered version of the gram matrix can be written as:*

$$\tilde{\boldsymbol{K}} = (\boldsymbol{I} - \boldsymbol{1_N})\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{1_N}) = \boldsymbol{U}\boldsymbol{D}^{\boldsymbol{2}}\boldsymbol{U}^{\mathrm{T}} \tag{3.1}$$

with 1_N a $N \times N$ matrix for which each element takes value $1/N$.

Proof. Let $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} = \langle x_i, x_j \rangle$,

$$\begin{aligned} \Delta_{ij}^2 &= \|x_i - x_j\|^2 = \|(x_i - \bar{x}) - (x_j - \bar{x})\|^2 \\ &= \|x_i - \bar{x}\|^2 + \|x_j - \bar{x}\|^2 - 2\langle x_i - \bar{x}, x_j - \bar{x} \rangle \end{aligned} \tag{3.2}$$

Then we have double-centered matrix $\boldsymbol{K}$

$$\begin{aligned} \tilde{\boldsymbol{K}} &= \langle x_i - \bar{x}, x_j - \bar{x} \rangle \\ &= \frac{1}{2}\left[\|x_i - \bar{x}\|^2 + \|x_j - \bar{x}\|^2 - \Delta_{ij}^2 \right] \\ &= \langle x_i, x_j \rangle - \frac{2}{N}\langle x_i, x_j \rangle + \frac{1}{N^2}\langle x_i, x_j \rangle \\ &= \boldsymbol{K} - \boldsymbol{1_N}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1_N} + \boldsymbol{1_N}\boldsymbol{K}\boldsymbol{1_N} \\ &= (\boldsymbol{I} - \boldsymbol{1_N})\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{1_N}) \end{aligned} \tag{3.3}$$

The centered version of $\boldsymbol{X}$ can be decomposed by its singular value decomposition (SVD) in the form $\tilde{\boldsymbol{X}} = (\boldsymbol{I} - \boldsymbol{1_N})\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}}$, then $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{D}$ is the matrix of principal components variables. And it follows that $\tilde{\boldsymbol{K}} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{D}^2\boldsymbol{U}^{\mathrm{T}}$, we can compute $\boldsymbol{Z}$ from the eigen decomposition of $\boldsymbol{K}$. □

Kernel PCA simply mimics this procedure, interpreting the kernel matrix $\boldsymbol{K} = \{K(x_i, x_j)\}$ as an inner-product matrix of the implicit features

$\langle \phi(x_i), \phi(x_j) \rangle$ and finding its eigenvectors. The elements of the mth component $\boldsymbol{z}_m$ (mth column of $\boldsymbol{Z}$) can be written as $z_{im} = \sum_{j=1}^{N} \alpha_{jm} K(x_i, x_j)$, where $\alpha_{jm} = u_{jm}/d_m$. Now we will show the process of derivation.

Assume that our data mapped into feature space, $\phi(x_1), \ldots, \phi(x_N)$, is centered, i.e. $\sum_{k=1}^{N} \phi(x_k) = 0$ (if not, using the formula of centralization by lemma 3.1.1). To do PCA for the covariance matrix

$$C = \sum_{i=1}^{N} \phi(x_i)\phi(x_i)^{\mathrm{T}} \tag{3.4}$$

we have to find eigenvalues $\lambda \geq 0$ and eigenvectors $\boldsymbol{V} \in \mathcal{F} \setminus \{0\}$ satisfying $\lambda \boldsymbol{V} = C\boldsymbol{V}$. Note that all solutions $\boldsymbol{V}$ lie in the span of $\phi(x_1), \ldots, \phi(x_N)$, we may consider the equivalent system

$$\lambda(\phi(x_k) \cdot \boldsymbol{V}) = (\phi(x_k) \cdot C\boldsymbol{V}) \quad \text{for all } k = 1, \ldots, N. \tag{3.5}$$

and there exist coefficients $\alpha_1, \ldots, \alpha_N$ such that

$$\boldsymbol{V} = \sum_{i=1}^{N} \alpha_i \phi(x_i) \tag{3.6}$$

considering the $N \times N$ matrix $\boldsymbol{K}$ with ijth entry

$$\boldsymbol{K}_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = (\phi(x_i) \cdot \phi(x_j)) \tag{3.7}$$

we arrive at

$$N\lambda \boldsymbol{K}\boldsymbol{\alpha} = \boldsymbol{K}^2\boldsymbol{\alpha} \tag{3.8}$$

where $\boldsymbol{\alpha}$ denotes the column vector with entires $\alpha_1, \ldots, \alpha_N$. To find solutions of (3.8), we solve the eigenvalue problem

$$N\lambda\boldsymbol{\alpha} = \boldsymbol{K}\boldsymbol{\alpha} \tag{3.9}$$

for nonzero eigenvalues. Clearly, all solutions of (3.8) do satisfy (3.8).

We normalize the solutions $\boldsymbol{\alpha}^k$ belonging to nonzero eigenvalues by requiring that the corresponding vectors in F be normalized, i.e. $(\boldsymbol{V}^k \cdot \boldsymbol{V}^k) = 1$. By virtue of (3.6), (3.7) and (3.9), this translates into

$$1 = \sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k (\phi(x_i) \cdot \phi(x_j)) = (\boldsymbol{\alpha}^k \cdot \boldsymbol{K}\boldsymbol{\alpha}^k) = \lambda_k(\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k) \tag{3.10}$$

For principle component extraction, we compute projections of the image of a test point $\phi(x)$ onto the eigenvectors $\boldsymbol{V}^k$ in $\mathcal{F}$ accordance to

$$(\boldsymbol{V}^k \cdot \phi(x)) = \sum_{i=1}^{N} \alpha_i^k (\phi(x_i) \cdot \phi(x)) \tag{3.11}$$

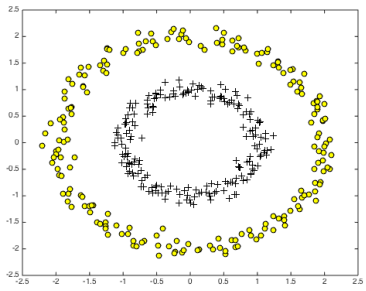
Consider two concentric clouds of points (has been shown in Introduction). We wish to use kernel PCA to identify these groups. Here we use two kinds of kernel, they are both positive-definite.

- Polynomial Kernel:

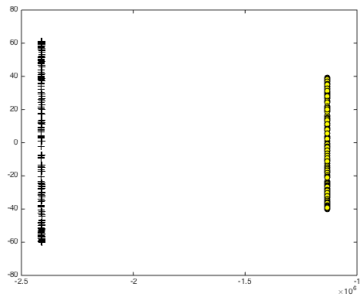
$$k(x, y) = (\alpha x^{\mathrm{T}} y + c)^d$$

- Gaussian Kernel:

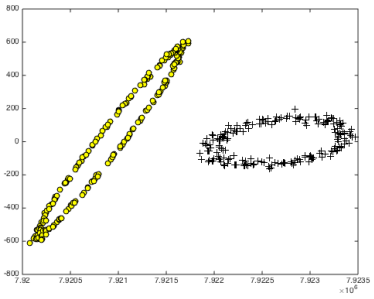
$$k(x, y) = \exp(-\frac{\|x - y\|^2}{2\sigma^2})$$



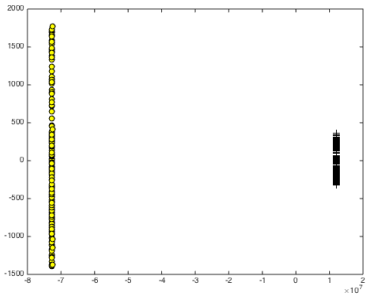
(A) Input points before kernel PCA



(B) Output points after kernel PCA with Gaussian Kernel $k(x, y) = \exp(-\|x - y\|^2)$



(C) Output points after kernel PCA with Linear Kernel $k(x, y) = x^{\mathrm{T}} y + 1$



(D) Output points after kernel PCA with Polynomial Kernel $k(x, y) = (x^{\mathrm{T}} y + 1)^2$

FIGURE 3.1: Kernel PCA

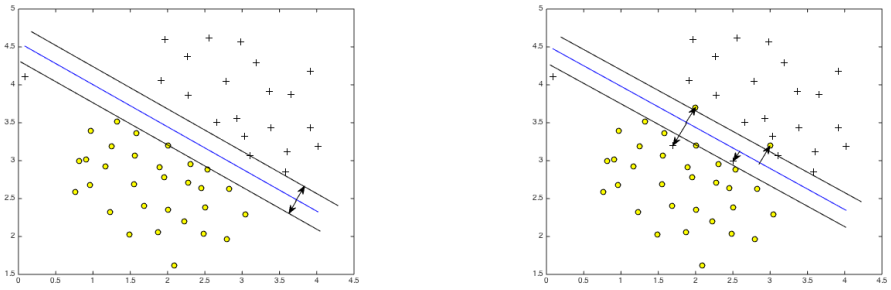
3.2 Kernel SVM

3.2.1 Support Vector Classifier

The left panel (Figure3.2a) shows the separable case. The linear decision boundary is $x^{\mathrm{T}}\beta + \beta_0 = 0$, which bound the maximal margin of width $2M = 2/\|\beta\|$, where the margin been determined by the points with its margin width M for one side. The right panel (Figure3.2b) shows the non-separable (overlap) case. Some points are on the wrong side of their margin by ξ_i; points on the correct side have $\xi_i = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq constant$.

Our training data consists of N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Define a hyperplane by

$$\{x : f(x) = x^{\mathrm{T}}\beta + \beta_0 = 0\} \tag{3.12}$$



(A) (B)

FIGURE 3.2: Support Vector Classifier

where β is a unit vector: $\|\beta\| = 1$. A classification rule induce by $f(x)$ is

$$G(x) = sign[x^{\mathrm{T}}\beta + \beta_0] \tag{3.13}$$

For the separable case (Figure3.2a), we can find a function $f(x) = x^{\mathrm{T}}\beta + \beta_0$ with $y_i f(x_i) > 0 \; \forall i$. The optimization problem can be shown as

$$\begin{aligned} &\max_{\beta,\beta_0,\|\beta\|=1} M \\ &\text{subject to } y_i(x_i^{\mathrm{T}}\beta + \beta_0) \geq M, i = 1, \ldots, N \end{aligned} \tag{3.14}$$

or equivalently,

$$\begin{aligned} &\min_{\beta,\beta_0} \|\beta\| \\ &\text{subject to } y_i(x_i^{\mathrm{T}}\beta + \beta_0) \geq 1, i = 1, \ldots, N \end{aligned} \tag{3.15}$$

Suppose now the classes overlap in feature space (Figure3.2b), we can modify (3.15) with a little change

$$\begin{aligned} &\min_{\beta,\beta_0} \|\beta\| \\ &\text{subject to } \begin{cases} y_i(x_i^{\mathrm{T}}\beta + \beta_0) \geq 1 - \xi_i, \forall i \\ \xi_i \geq 0, \sum \xi_i \leq constant \end{cases} \end{aligned} \tag{3.16}$$

The problem (3.16) is quadratic with linear inequality constraints, hence it is a convex optimization problem. We describe a quadratic programming solution using Lagrange multipliers. Computationally it is convenient to re-express (3.16) in the equivalent form

$$\begin{aligned} &\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{subject to } \xi_i \geq 0, y_i(x_i^{\mathrm{T}}\beta + \beta_0) \geq 1 - \xi_i, \forall i \end{aligned} \tag{3.17}$$

where the "cost" parameter C replaces the constant in (3.16); the separable case corresponds to $C = \infty$.

The Lagrange function is

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^{\mathrm{T}}\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i \quad (3.18)$$

which we minimize w.r.t β, β_0 and ξ_i. Setting the respective derivatives to zero, we get

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i, \quad (3.19)$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i, \quad (3.20)$$

$$\alpha_i = C - \mu_i, \forall i \quad (3.21)$$

as well as the positivity constraints α_i, μ_i, $\xi_i \geq 0\ \forall i$. By substituting (3.19)-(3.20) into (3.18), we obtain the Lagrangian dual object function

$$\begin{aligned} L_D &= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}}x_j \\ &\text{subject to } 0 \leq \alpha_i \leq C, \text{ and } \sum_{i=1}^{N}\alpha_i y_i = 0 \end{aligned} \quad (3.22)$$

From (3.19) we see that the solution for β has the form

$$\hat{\beta} = \sum_{i=1}^{N}\hat{\alpha}_i y_i x_i \quad (3.23)$$

Given the solutions $\hat{\beta}_0$ and $\hat{\beta}$, the decision function can be written as

$$\hat{G(x)} = sign[\hat{f}(x)] = sign[x^{\mathrm{T}}\hat{\beta} + \hat{\beta}_0] \quad (3.24)$$

3.2.2 Support Vector Machines and Kernels

The support vector classifier described so far finds linear boundaries in the input feature space. Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space. Once the basis functions $h_m(x)$, $m = 1, \ldots, M$ are selected, the procedure is the same as before. We fit the SV classifier using input features $h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$, $i = 1, \ldots, N$, and produce the nonlinear function $\hat{f}(x) = h(x)^{\mathrm{T}}\hat{\beta} + \hat{\beta}_0$. The classifier is $\hat{G}(x) = sign(\hat{f}(x))$ as before.

The support vector machine (SVM) classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases. It might seem that the computations would become prohibitive. It would also seem that with sufficient basis functions, the data would be separable, and overfitting would occur.

We can represent the optimization problem (3.18) and its solution in a special way that only involves the input features via inner products. We do

this directly for the transformed feature vectors $h(x_i)$. We then see that for particular choices of h, these inner products can be computed very cheaply.

With prior knowledge of the kernel function

$$K(x, y) = \langle h(x), h(y) \rangle \tag{3.25}$$

The Lagrange dual function (3.22) has the form

$$\begin{aligned} L_D &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle h(x_i), h(x_j) \rangle \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \end{aligned} \tag{3.26}$$

The solution function $f(x)$ can be written as

$$\begin{aligned} f(x) &= h(x)^{\mathrm{T}} \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + \beta_0 \end{aligned} \tag{3.27}$$

As before, given α_i, β_0 can be determined by solving $y_i f(x_i) = 1$ in (3.26) for all x_i for which $0 < \alpha_i < C$.

With $f(x) = h(x)^{\mathrm{T}} \beta + \beta_0$, we have

$$\begin{cases} \xi_i \geq 0 \\ y_i f(x_i) \geq 1 - \xi_i \end{cases} \quad \forall i \tag{3.28}$$

Therefore, the optimization problem (3.17) can be rewritten as the form

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ \tag{3.29}$$

where the subscript "+" indicates positive part. With $\lambda = 1/C$, we have

$$\min_{\beta, \beta_0} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2} \|\beta\|^2 \tag{3.30}$$

This has the form with a loss function and a shrinkage, which is a familiar paradigm in function estimation. Next, we will use the knowledge of reproducing kernel Hilbert spaces to solve this regularization problem.

3.2.3 Reproducing Kernels in SVM

Suppose the basis h arises from the eigen-expansion of a positive definite kernel K,

$$K(x, y) = \sum_{j=1}^{\infty} \gamma_j \phi_j(x) \phi_j(y) \tag{3.31}$$

and $h_j(x) = \sqrt{\gamma_j}\phi_j(x)$. Then with $\theta_j = \sqrt{\gamma_j}\beta_j$, we can write (3.30) as

$$\min_{\beta_0,\{\theta_j\}_1^\infty} \sum_{i=1}^{N}\left[1 - y_i(\beta_0 + \sum_{j=1}^{\infty}\theta_j\phi_j(x_j))\right]_+ + \frac{\lambda}{2}\sum_{j=1}^{\infty}\frac{\theta_j^2}{\gamma_j} \tag{3.32}$$

The theory of reproducing kernel Hilbert spaces described there guarantees a finite-dimensional solution of the form

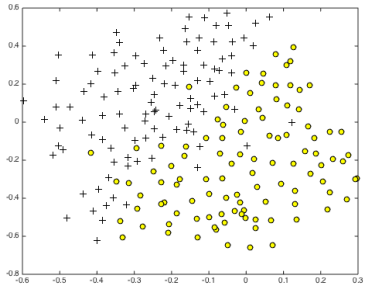
$$f(x) = \beta_0 + \sum_{i=1}^{N}\alpha_i K(x, x_i) \tag{3.33}$$

Also the equivalent version of the optimization criterion of (3.32) is

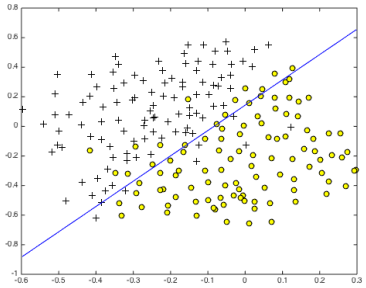
$$\min_{\beta_0,\theta} \sum_{i=1}^{N}(1 - y_i f(x_i))_+ + \frac{\lambda}{2}\alpha^{\mathrm{T}}\boldsymbol{K}\alpha \tag{3.34}$$

where $\boldsymbol{K}$ is the $N \times N$ matrix of kernel evaluations for all pairs of training features.

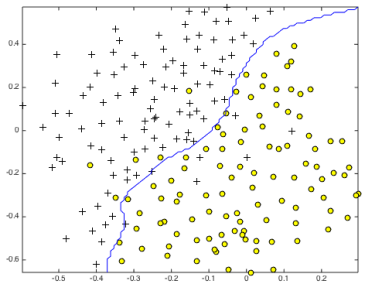
Here is an example that classes overlap in feature space, we will see how the decision boundary changes with different kind of kernels.



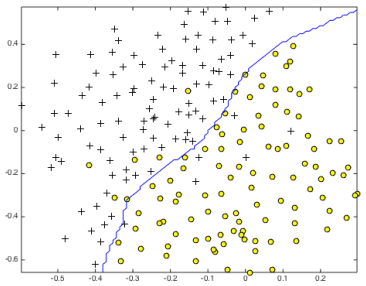
(A) Input points: non-separable case



(B) Training SVM with Linear Kernel $k(x, y) = x^{\mathrm{T}}y + 1$



(C) Training SVM with Gaussian Kernel $k(x, y) = \exp(-\frac{\|x-y\|^2}{2*0.1^2})$



(D) Training SVM with Log Kernel $k(x, y) = -\log(\|x - y\|^{0.5} + 1)$

FIGURE 3.3: Training SVM with Different Kernels

Chapter 4

Real-World Example: handwritten digits recognition[1]

The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image, so it is easy to apply our supervised statistical learning algorithms with minimal efforts on preprocessing and formatting. We want to compare the effects conducted by different statistical learning algorithms, in the meanwhile using with and without kernel. Here (Figure4.1) are the first 100 digits in training dataset.

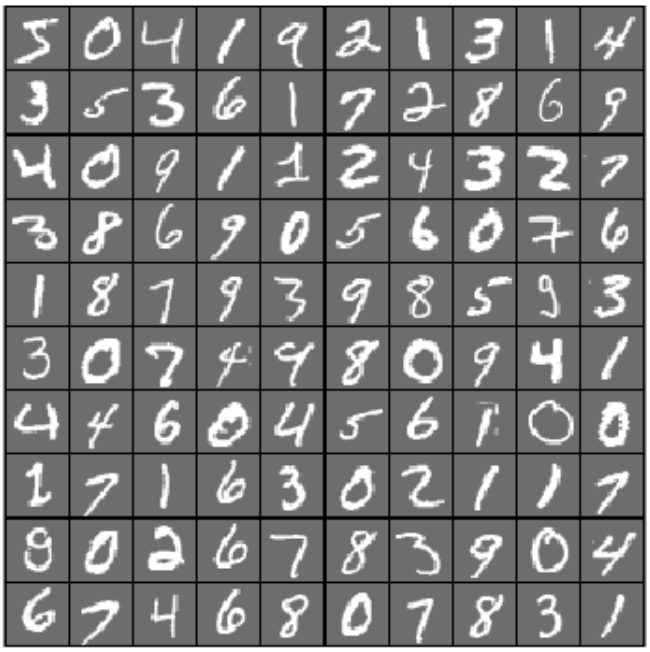


FIGURE 4.1: MNIST database: handwritten digits

The following algorithms only use the training set to train the model (tune the parameters, cross-validation, etc.), and then make a prediction by applying the trained model on testing set features. Since we already have the labels of testing set, we can therefore get the accuracy by comparing our predictions and known labels. Note that testing set is in black box when

[1]All the code and data used in this chapter can be find here: https://github.com/Louise222/Bachelor-Thesis.git

training the model, we can assume that our trained model can acquire a similar prediction accuracy with other testing sets.

4.1 SVM: No kernel v.s. Kernel

LIBSVM is a powerful software for support vector classification, it uses one v.s. one strategy to deal with multi-class classification problem. We use this package to train two models, one without kernel and one with gaussian kernel. By comparing the predicted label and real label in the testing dataset, we can get accuracy of these two models.

Before training model, we need to preprocess the data. Our training dataset is large, it is time consuming either training model or searching for a better parameters. Thus, we subset the training datasets into two parts, and only use 1000 digits among 60,000 to search for good parameters. Later in training model process, we still use the whole training dataset.

```
% step1: turn into LIBSVM format
% <label> <index1>:<value1> <index2>:<value2> ...
train_sparse = sparse(train_images_scale);
libsvmwrite('train',train_labels,train_sparse);
% step2: .../tool/subset.py for a subset of data
>subset.py train 1000 output1 output2
% step3: .../tool/grid.py for searching parameters
>grid.py -log2c -2,10,1 -log2g -4,-14,-1 output1
```

This figure (4.2) is plotted by gnuplot followed by ">grid.py" command. It finds the best parameters for SVM with RBF kernel.

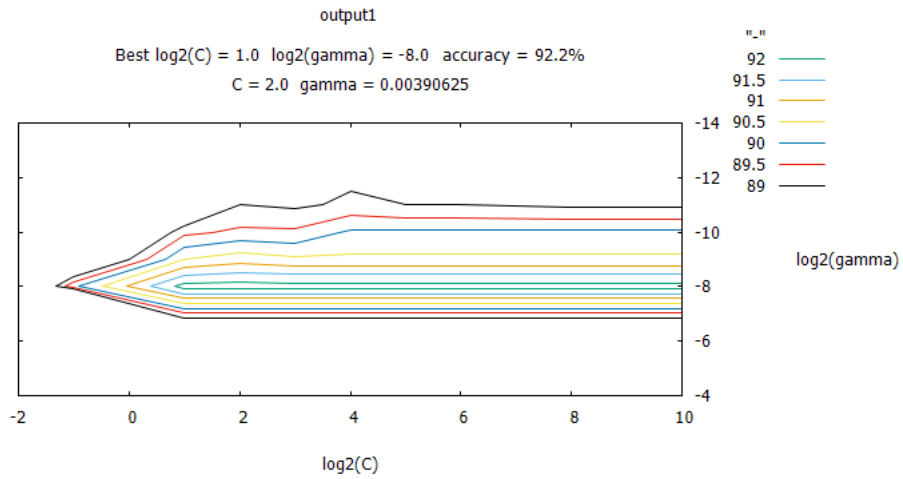


FIGURE 4.2: Searching for the Best Parameters

In our example, we get $C = 2$, $g = 0.0039$, that is to say, it is a good idea to set $C = 2$ for shrinkage parameter and $\gamma = 0.0039$ for Gaussian kernel parameter.

```
% Train and Predict, linear kernel
model1 = svmtrain(train_labels, train_images_scale, ...
```

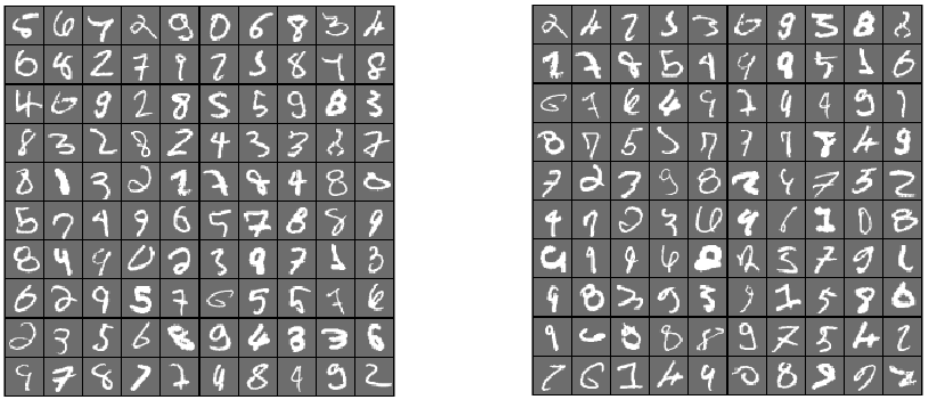
```
                        '-s 0 -t 0 -c 2');
[predicted_label1, accuracy1, decision_values1] = ...
            svmpredict(test_labels, test_images_scale, model1);
% Shown wrongly-matched digits
index1 = find(test_labels~=predicted_label1);
display_network(test_images(index1(1:100),:)');
```

Without kernel, we get accuracy of 93.16%.

```
% Train and Predict, gaussian kernel
model2 = svmtrain(train_labels, train_images_scale,...
                    '-s 0 -t 2 -g 0.0039 -c 2');
[predicted_label2, accuracy2, decision_values2] =...
     svmpredict(test_labels, test_images_scale, model2);
% Shown wrongly-matched digits
index2 = find(test_labels~=predicted_label2);
display_network(test_images(index2(1:100),:)');
```

With kernel, we get accuracy of 98.35%.

Here are some wrongly-matched digits predicted by SVM. The digits in the left panel (Figure4.3a) are wrongly predicted by linear SVM, and those in the right panel (Figure4.3b) are wrongly predicted by kernel SVM.



(A) No Kernel (B) Gaussian Kernel

FIGURE 4.3: Wrongly-matched Digits by SVM

In conclusion, we have 5.84% of predicting error with linear SVM, and only have 1.65% of predicting error with kernel SVM.

4.2 Comparison with Neural Network

Neural Network method uses sigmoid kernel, which is non-positive definite. We cannot apply our kernel method in neural network, but, since support vector classifier with a sigmoid kernel acts exactly with a two-layer neural network model, it is interesting to make a comparison between kernel SVM and neural network. In this section, we go further with more hidden layers, fine-tuning parameters towards neural network model, and see how high accuracy can get.

Here (Figure4.4) are learned features that resemble pen strokes. In other words, the model has learned to represent handwritten characters in terms of what pen strokes appear in an image.

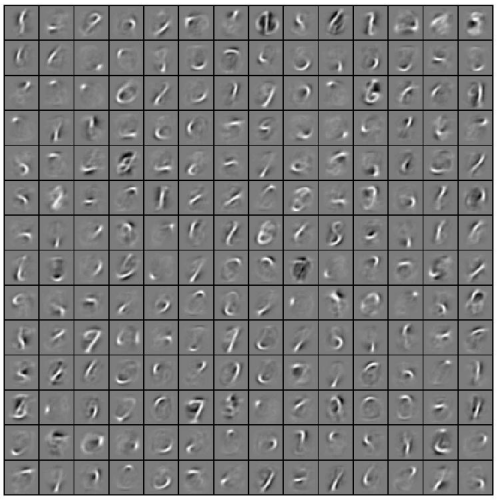


FIGURE 4.4: Learned Features of Handwritten Digits

The final model (Figure4.5) we use has 2 hidden layers, and a final softmax classifier layer capable of classifying the MNIST digits as desired.

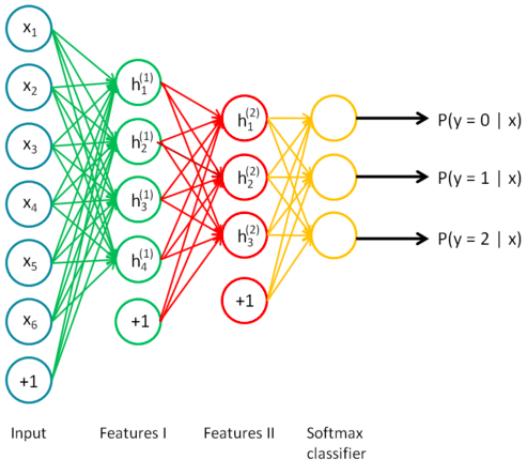


FIGURE 4.5: Deep Learning Model

With complexity goes up, the perform of model depends highly on parameters. Without fine-tuning, the model gives an accuracy of 87.77%, but with fine-tuning, it gives an accuracy of 98.38%, which is similar to SVM with Gaussian kernel.

Chapter 5

Conclusion

This paper focuses on spaces of functions generated by kernels, which are called reproducing kernel Hilbert spaces (RKHS). Upon these special spaces, we develop a kernel-based approach for classification problem in the machine learning area. The remarkable feature of this method is that while the criterion of classification problem is defined over an infinite-dimensional space, the solution can be finite-dimensional. With this notable feature, we can apply this method to large-scale nonlinear classification problems, which are complicated and time-consuming by ordinary classification algorithms.

The theoretical basis of RKHS lies on the research of Hilbert spaces in boundary value problems and reproducing property in the theory of integral equations. These spaces have wide applications, including complex analysis, harmonic analysis, and quantum mechanics. RKHS are particularly important in the field of statistical learning theory because of the celebrated Riesz representer theorem which states that every function in an RKHS can be written as a linear combination of the kernel function evaluated at the training points. Based on this property, we extend the theory into several widely-used statistical learning algorithms such as principle component analysis (PCA) and support vector machine (SVM), by the name of kernel methods. Kernel methods use an approach called "kernel trick" which enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates.

In our real-word example, we experiment on a large database *The MNIST Database of Handwritten Digits* (60,000 training, 10,000 testing observations), derived from standard NIST. These handwritten digits vary with their sizes and most of them have some degree of rotation, all these special characteristics make feature selection difficult to apply. Our experiments have shown that the kernel methods perform extremely well with less than 2 percent error, considering some handwritten digits are too hasty and careless to be labeled by humans, the result is really inspiring.

There are several problems which need to be addressed in further research on RKHS and their related algorithms. One of the challenges is that the kernels used in algorithms must be positive-definite. This property is necessary since the Riesz representation theorem proves that given a positive definite kernel K, there is a unique associated RKHS with K as a reproducing kernel. Yet, positive-definite kernels are limited and some widely-used kernels, such as sigmoid kernel, lack this property. If we can build up

a method linking non-positive definite kernels with RKHS (with some sort of limitation), we can apply this "kernel trick" into more algorithms and thus make it more powerful in real-world examples. Another difficulty lies in applying and evaluating algorithms. It's difficult to make a distinction between several algorithms with almost the same testing accuracy shown at the end, thus make it unclear to find the most appropriate model, especially in unsupervised learning. More visualization and middle procedure should be displayed so that engineers can have more measurement towards these kernel-based methods.

Appendix A

Loss Function with Shrinkage

In this Appendix, we describe the formula for regularization problems.

A.1 Statistical Decision Theory

We seek a function $f(X)$ for predicting Y given values of the input X. This requires a loss function $L(Y, f(X))$ for penalizing errors in prediction, and by far the most common and convenient is squared error loss: $L(Y, f(X)) = (Y - f(X))^2$.

- Least squares assumes $f(X)$ is well approximated by a globally linear function,
$$f(X) = X^{\mathrm{T}}\hat{\beta}$$
- k-nearest neighbors assumes $f(x)$ is well approximated by a locally constant function,
$$f(x) = \frac{1}{k}\sum_{x_i \in N_k(x)} y_i$$

A.2 Shrinkage Methods

Regularization is designed to address the problem of overfitting.

High bias or underfitting is when the form of our hypothesis maps poorly to the trend of the data. It is usually caused by a function that is too simple or uses too few features.

At the other extreme, overfitting or high variance is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

To address the issue of overfitting by regularization while still remaining all the features, we can keep all the features by reducing the parameters θ_j, this is exactly what shrinkage means. It works well when we have a lot of slightly useful features.

Using shrinkage methods, the cost function mentioned above can be shown as follows,

$$L(Y, f(X)) = \min_f \left\{ \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda J(f) \right\}$$

where $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage, and

thus cause higher bias and lower variance. The user-selected functional $J(f)$ will be large for functions f that vary too rapidly over small regions of input space.

A.3 Some Classical Regularization Problems

- Ridge Regression,

$$L(Y, f(X)) = \min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- The Lasso,

$$L(Y, f(X)) = \min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- Regularized Logistic Regression,

$$L(Y, f(X)) = \min_{\theta} \left\{ [\sum_{i=1}^{N} \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i))] + \lambda \sum_{j=1}^{n} \theta_j^2 \right\}$$

Appendix B

Gaussian Kernel

The Gaussian kernel $k(x, y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$ is very popular and makes a good default kernel especially in absence of expert knowledge about data and domain. What makes the Gaussian kernel universal? How does the Gaussian kernel separate seemingly any sort of nonlinear data exceptionally well?

(B.1) is an example of feature space produced by Gaussian kernel. The feature space has infinite dimensions (maximum to the size of training sample), through adjusting the parameter σ, we can have a tradeoff between bias and variance. Small σ corresponds to low bias and high variance (overfitting), vise visa.

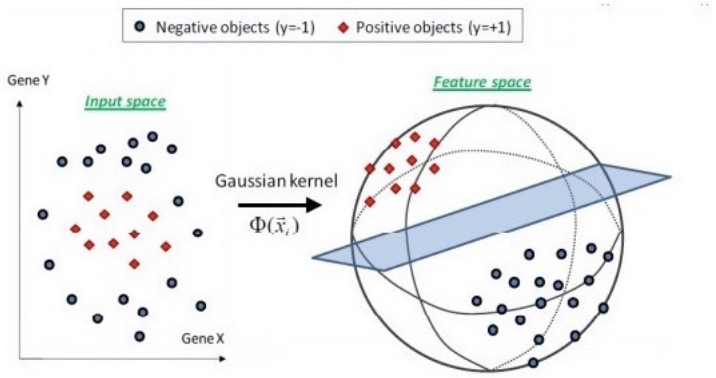


FIGURE B.1: Feature Space Produced by Gaussian Kernel

B.1 Locality

Given a bounded continuous function f, there exists a function $g \in \mathcal{H}$, such that f and g are close (in the sense of $\|\cdot\|_\infty$) up to arbitrary precision needed. Basically, this means Gaussian kernel gives functions which can approximate "nice" (bounded, continuous) functions arbitrarily well. Gaussian and Laplace kernels are universal. A polynomial kernel, for example, is not.

B.2 Separability

The Gaussian kernel for dimensions higher then one, say N, can be described as a regular product of N one-dimensional kernels. For example,

the Gaussian kernel be:

$$f(x;\sigma_1^2) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-x^2}$$

, then we have:

$$f(x,y;\sigma_1^2+\sigma_2^2) = f(x;\sigma_1^2) f(y;\sigma_2^2).$$

B.3 Anisotropy

Anisotropy of a Gaussian kernel means that the scales, or standard deviations, are different for the different dimensions. When they are the same in all directions, the kernel is called isotropic, which means that the behavior of the function is in any direction the same. For 2D this means the Gaussian function is circular, for 3D it looks like a fuzzy sphere.

B.4 The Fourier transform of the Gaussian kernel

The basis functions of the Fourier transform $\mathcal{F}$ are the sinusoidal functions $\exp(i\omega x)$. The definitions for the Fourier transform and its inverse are:

the Fourier transform:

$$F(\omega) = \mathcal{F}\{f(x)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{i\omega x} dx$$

the inverse Fourier transform:

$$\mathcal{F}^{-1}\{F(\omega)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) e^{-i\omega x} d\omega$$

So the Fourier transform of the Gaussian function is again a Gaussian function, but now of the frequency ω. Also, the fact is smaller kernel in the spatial domain gives a wider kernel in the Fourier domain, and vice versa.

If we consider a Laplace kernel $k(x,y) = \exp(-\frac{\|x-y\|}{\sigma})$ which is also radial basis function kernel, its Fourier transform is a Cauchy distribution which drops much slower than the exponential function in the Fourier transform of a Gaussian kernel. This means a function f will have more high-frequency components. As a result, the function given by a Laplace kernel is "rougher" than that given by a Gaussian kernel.

Reference

[1] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[2] Kenji Fukumizu, Francis R. Bach, and Michael I. Jordan. "Dimensionality Reduction for Supervised Learning with Reproducing Kernel Hilbert Spaces". In: *Journal of Machine Learning Research* 5 (2004), pp. 73–79.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2nd. Springer-Verlag, 2009, pp. 167–173, 417–455, 547–550. URL: http://statweb.stanford.edu/~tibs/ElemStatLearn/.

[4] Peter D. Lax. *Functional Analysis*. Wiley-Interscience, 2002, pp. 36–71.

[5] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of Handwritten Digits*. URL: http://yann.lecun.com/exdb/mnist/.

[6] Andrew Ng. *Machine Learning*. URL: http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning.

[7] Andrew Ng et al. *UFLDL Tutorial*. 2011. URL: http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial.

[8] Cheng Soon Ong et al. "Learning with Non-positive Kernels". In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. ACM, 2004, pp. 81–.

[9] Lorenzo Rosasco. *Reproducing Kernel Hilbert Spaces - MIT 9.520 Lecture Notes*. 2010.

[10] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Kernel Principal Component Analysis". In: *Lecture Notes in Computer Science* 1327 (June 2006), pp. 583–588.

[11] Grace Wahba. "Advances in Kernel Methods". In: ed. by Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola. MIT Press, 1999. Chap. Support Vector Machines, Reproducing Kernel Hilbert Spaces, and Randomized GACV, pp. 69–88.