# Kernel Learning: Methods and Applications

A thesis submitted
for a degree of Doctor of Philosophy
by

**Zhuang Jinfeng**

School of Computer Engineering
Nanyang Technological University

August 2011

# Abstract

Kernel plays a central role in the successes of kernel-based techniques and applications. Conventional kernel-based techniques often adopt some single parametric kernel where the kernel parameters are usually fixed by some heuristic approach. The conventional single parametric kernels with fixed parameters limit their capacity of the subsequent kernel-based techniques when fitting complex data in a real-world application. Thus, the goal of this thesis is to address the open research challenge of learning effective kernels for real-world applications. The main contribution of this thesis is to investigate novel and effective kernel learning methods for improving the performance of kernel-based techniques towards real applications. The details of our contributions are summarized as follows.

First of all, to address the limitation of conventional parametric kernel methods, we first present a family of simple non-parametric kernel learning algorithms (SimpleNPKL), which are able to learn non-parametric kernels from side information by solving a special kind of Semi-Definite Programs (SDP) in a fairly efficient way. We apply SimpleNPKL to several applications, including clustering, classification, and data embedding, and conducted extensive experiments on publicly available data sets to validate the effectiveness and efficiency of our techniques.

Besides, this thesis also explores two new directions of kernel learning in learning effective kernels from data. The first technique is the deep multiple kernel learning, which is inspired by recent developments in deep learning architectures and multiple kernel learning. The second technique is the unsupervised multiple kernel learning, which learns kernels from the data in an unsupervised manner and can be used as a useful tool for data pre-processing. Our experiments on publicly available data sets show that the two new directions are promising.

Finally, we also investigate the applications of kernel learning techniques to some real-world problems. In particular, the first application is to apply the non-parametric kernel learning techniques for image re-ranking in a social image retrieval task. The second application is to explore the applications of kernel learning for social strength modelling in a social media application. Our extensive empirical studies on large real data further validate the effectiveness of kernel learning techniques in solving real-world applications.

# Acknowledgments

First, I would like to thank my supervisor Dr Steven C.H. Hoi, who is probably the best supervisor in Nanyang Technological University. Dr. Hoi brought me to the world of machine learning three years ago and worked closely with me for all my published research papers. During my PhD career, he not only inspires me on innovative research ideas with his profound knowledge and amazing wisdom, but also encourages me for being a decent and successful person.

Second, I would like to thank my non-official co-supervisor Dr. Ivor W.H. Tsang, who is one of the most descent researchers in the related areas. The discussion with Dr. Tsang would be the most memorable moments during my research career. Many of the ideas in our collaborated work were generated from our fruitful discussions.

Third, I would like to thank my old friends, especially Min Wu, Juyong Zhang, Zheng Li, Molly Zhou, Sasa Hu and Xiaoxiao Li. You guys shed light into my life. I always cherish the time of eating crabs and drinking beer with Juyong and Min. I thank Molly and Sasa. With you two women of different roles, I never feel lonely. I thank Xiaoxiao Li, as the blue side of life is indispensable and valuable. I sincerely wish you guys would accompany with me in the rest of my life.

I thank the relatively new friends, Linhong Zhu, Shuguo Han, Peilin Zhao, Hao Xia, Pengcheng Wu, Shumo Chu, and Bin Li. I really enjoy the time spent with everyone of you. You guys bring energy to our lab. I thank the friends known in Singapore, Sophia Wan, Louise Huang, Joyce Jia etc. The time with you always flies so quick. I believe we can meet frequently in the future. I thank my mentors in Microsoft Research Asia: Tao Mei, Xian-sheng Hua, Ying-qing Xu, and Shipeng Li. Without the discussions with you, I cannot work out the fruitful publications there. I also thank the great friends made in MSRA: Mingyan Gao, Jialei Wang, Binxing Jiao, Ju Sun, Zhaoyan Ming, and Qiong Wu.

At last, I would like to thank my parents. No matter when I am frustrated, you are always there with me. I pray that you both are healthy and happy forever. I love you.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Over the last decade kernel based learning methods are one of the most popular research topics in machine learning. According to a statistic[1], "kernel" is one of the most popular keywords in ICML conference since 1988.A number of kernel machines, e.g., support vector machines (SVMs) [CV95], kernel Fisher discriminant (KFD) [Mik99], and kernel principal component analysis (KPCA) [SSM98], have been proposed and become state-of-the-art methodologies. These approaches have shown practical relevance not only for classical classification and regression problems but also, in clustering, dimensionality reduction, etc. Kernel based learning algorithms have been successfully applied to a wide range of applications in various fields, such as pattern and object recognition, text categorization, time-series prediction, finance, gene expression profile analysis, DNA and protein analysis, etc. Some good surveys on kernel machine learning topics can be found in [Bur98, rMMR$^+$01, SS02, Mm06].

The key idea of kernel methods can be briefly explained as follows [HSS08]. Classically, theories and algorithms of machine learning and statistics have been very well developed for the linear case. However, real world data analysis problems often require nonlinear methods to detect the kind of dependencies that allow successful prediction of properties of interest. By using a positive semi-definite (PSD) kernel, one can sometimes have the best of both worlds. The kernel corresponds to a dot product in a (usually high-dimensional) feature space. In this space, our estimation methods are linear, but as long as we can formulate the computation in terms of kernel evaluations, we never explicitly have to compute in the high-dimensional feature space.

The kernel function plays a central role in kernel machines. Empirical evidence shows that the performance of a kernel method relies more on the kernel function rather than the kernel machine. For example, kernelized logistic regression, kernelized regularized least square, and support vector machine often result in similar performance. Most of the off-the-shelf algorithms have been kernelized if they could be. Therefore, how to design or learn a good kernel has become an important research topic in machine learning research.

[1]http://learning.eng.cam.ac.uk/zoubin/talks/ICML-Presentation.pdf

In this chapter, we introduce kernels informally as similarity measures that arise from a particular representation of patterns. The goal is to provide an overview of the basic concepts. Then we describe the problem of *kernel learning* and summarize our contributions. Finally we give organization and some notation table.

## 1.1 Preliminaries

Supervised classification is one of the fundamental problems of learning theory: Suppose we are given two classes of objects. We are then faced with a new object, and we have to assign it to one of the two classes. This problem can be formalized as follows: we are given empirical data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \{+1, -1\}$. Here, $\mathcal{X}$ is some nonempty set from which the patterns $\mathbf{x}_i$ (sometimes called cases, inputs, instances, observations, points or samples) are taken, usually referred to as the *domain*; the $y_i$ are called *labels, targets, outputs* or sometimes also *observations.* Note that we only consider two classes of patterns, +1 and -1 respectively. This simple situation is referred to as *binary classification.*

It should be emphasized that the patterns are just abstraction which could be about anything, and we have not made any assumptions on $\mathcal{X}$ other than it being a set. For instance, the task might be to categorize sheep into two classes, in which case the patterns $\mathbf{x}_i$ would simply be sheep. In order to study the problem of learning, however, we need an additional type of structure. In learning, we want to be able to generalize to unseen data points. In the case of pattern recognition, this means that given some new pattern $\mathbf{x} \in \mathcal{X}$, we want to predict the corresponding $y \in \{\pm 1\}$. By this we mean, loosely speaking, that we choose $y$ such that $(\mathbf{x}, y)$ is in some sense similar to the training examples. To this end, we need notions of similarity in $\mathcal{X}$ and in $\{\pm 1\}$. It is easy to characterize the similarity of the outputs $\{\pm 1\}$: in binary classification, there are only two situations can occur: two labels can either be identical or different. The choice of the similarity measure for the inputs, however, is a deep question that lies at the core of the field of machine learning.

Let us consider a similarity measure of the form

$$\begin{aligned} k: \ & \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ & (\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}') \end{aligned} \tag{1.1}$$

that is, a function that, given two patterns $\mathbf{x}$ and $\mathbf{x}'$, returns a real number characterizing their similarity. Unless stated otherwise, we will assume that $k$ is *symmetric*, that is, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. For reasons that will become clear later, the function $k$ is called a *kernel.*

General similarity measures of this form are rather difficult to study. Let us therefore start from a particularly simple case, and generalize it subsequently. A simple type of similarity measure that is of particular mathematical appeal is a *dot product*. For instance, given two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, the canonical dot product is defined as

$$\langle \mathbf{x}, \mathbf{x}' \rangle := \sum_{i=1}^{d} x_i x_i'.$$

here $x_i$ is the $i$th component of $\mathbf{x}$. The geometric interpretation of the canonical dot product is that it computes the cosine of the angle between the vectors $\mathbf{x}$ and $\mathbf{x}'$, provided they are normalized to length 1. A similarity function plays a central role in learning. For example, with a well defined kernel $k$, one can use nearest neighbor for classification, i.e., the label of a test point $\mathbf{x}$ is predicted as the label of its nearest neighbor in the training set. Furthermore, we can compute the average similarity between $\mathbf{x}$ and the positive/negative training set.

With the similarity function $k$, one can proceed on devising learning frameworks. Among the large volume of classification algorithms, *support vector machine* is one of the most influential techniques. Refer to [SS02] for a more detailed introduction.

## 1.2 Motivation: Why Kernel Learning?

A large volume of works on kernel design/learning has emerged and exhibited strengths in a variety of applications in the machine learning community. In the standard framework of kernel methods, the choice of an appropriate kernel is left to the user and a poor selection may lead to sub-optimal performance. Moveover, empirical evidence shows that the performance is often dominated by the kernel being used rather than the type of the kernel machine. Therefore, how to choose or design or learn a "good" kernel is a crucial problem. Instead of using a fixed kernel function, sample points can be used to select or construct a kernel function/matrix suitable for the learning task.

The focus of this thesis is how to learn a kernel from data, either a closed-from kernel function or a kernel matrix (the value of each entry is the kernel evaluation result). In the sequel, we do not distinguish kernel function from kernel matrix if it is clear in the context. Informally, the problem of *kernel learning* is to devise a closed-form kernel function, or construct/learn the kernel matrix on a sample $\mathbf{X}$ (typically in the transductive setting), given an accessible sample of data (both labeled and unlabeled), a prior kernel, or any form of useful prior information to help to determine the kernel.

The kernel is the prior knowledge we have about a problem and its solution. Accordingly, just as there is no "free lunch" in learning, there is also no free lunch in kernel choice. Kernel learning is very challenging and still under exploration. First, a kernel has to guarantee the positive definiteness, which is a necessary constraint to make the kernel valid. This is often difficult to justify. The resultant optimization problem is often very difficult to solve, for example, learning the kernel matrix is in nature a semidefinite program[BV04]. Second, the design of good kernel requires deep insights about the problem at hand, for instance, the manifold structure of the data. We know that the kernel essentially maps the data into a high-dimensional space. However, this property does not provide semantic hints on which kind of kernels are useful for classification. Third, it calls for theories that connect the kernel and the error risk on test data such that the learning of kernel is principled and essentially effective. Due to these difficulties, kernel learning is still being actively studied.

Researchers have proposed various methods on this topic. One approach is to construct kernels based on known ones guided by some general tricks. A successful work is Multiple Kernel Learning (MKL), which learns a convex combination of some basic kernels (e.g., [LCB$^+$04]). Another scheme is to specify a parametrization of the kernel based on some insights of the problem (e.g., [KL02]). Recently, nonparametric kernel learner has emerged to learn a kernel matrix directly from training data with few assumptions on the kernel structure (e.g, [TRW05][KSD06][HJL07][LYW07]). In chapter 2, we conduct a thorough survey on background knowledge and recent advances on representative kernel learning methods.

## 1.3 Problems and Research Scope

In general, our research focus in this thesis is about to design and develop *effective* and *efficient* kernel learning algorithms. More formally, the general problem can be described in the following.

*Given a set of training samples (possibly together with side information), how to learn an effective kernel for classification, clustering, or data embedding efficiently?*

Specifically, this thesis aims to tackle the above challenge by the following approaches:

- to study efficient non-parametric kernel learning (NPKL) algorithms. Unlike parametric MKL algorithms, the variables to be solved in NPKL is a positive semi-definite matrix, which is usually very hard to solve. A typical interior-point solver for NPKL has the time complexity of $O(N^{6.5})$, which makes NPKL prohibitive for large-scale real applications;

- To explore new heuristics for designing more effective multiple kernel learning (MKL) algorithms. Most existing MKL algorithms follow the same optimization framework as SVM. Despite being studied extensively, their empirical performance is not always satisfied. We aim to study more effective MKL algorithms beyond the SVM-based framework.

Besides the investigation of novel algorithms, another important research scope of this thesis is to examine the efficacy of kernel learning techniques when be applied to solve some challenging problems in real-world applications. Specifically, this thesis will explore kernel learning techniques to tackle challenges in two important real applications: (i) image re-ranking task in social image retrieval, and (ii) social strength modeling task in social media community mining. We have conducted extensive experiments on large-scale real datasets to validate the efficacy of the proposed solutions.

## 1.4 Summary of Contributions

The major contributions of this thesis include:

- We conduct a thorough survey on both the foundations of kernel machines and recent development of kernel learning, including concepts, algorithms and theories. We believe such a survey with consistent notations and organizations are valuable;
- We propose a family of non-parametric kernel learning (NPKL) algorithms, named SimpleNPKL, which avoids semi-definite programming in traditional NPKL methods, such that NPKL can be significantly boosted to handle data set of 10K samples. The framework is general enough to apply to classification, clustering, and data embedding tasks;
- We explore two novel directions of multiple kernel learning algorithms (MKL): *deep* MKL and *unsupervised* MKL, beyond the traditional SVM based and kernel target alignment based principles. Empirical results on public available data sets shows the proposed directions are promising;
- We apply kernel learning techniques to solve two important applications: image re-ranking and social strength modeling, and conduct extensive sets of empirical studies on large-scale real datasets.

In next section, we introduce the organizations of the thesis. Readers can jump to the part of their particular interest.

## 1.5 Thesis Organizations

In chapter 2 we introduce the basic concepts and theories on kernel machines and statistical machine learning. We first give formal definitions on positive definite kernel and related concepts (RKHS). Then we describe the most popular kernel method–support vector machine and related work on kernel learning. To provide deep insights on such traditional supervised learning algorithms, we also discuss the classical statistical learning theory. Since optimization techniques have already tightly intertwined with machine learning research. We introduce the general form of convex problems and *semi-definite program* (SDP). After that we describe the transductive learning problem with insights. Our SimpleNPKL framework is a natural recipe in this problem setting. It avoids heavy computation cost of SDP and yields satisfactory performance in transductive learning.

In chapter 3, we propose a family of Simple Non-Parametric Kernel Learning (SimpleNPKL) algorithms for efficient and scalable non-parametric kernel learning. With our new solution, the NPKL can be solved efficiently. Therefore, NPKL is made as a practical learning scheme for real application. We conduct extensive experiments, which show that SimpleNPKL is significantly more efficient than existing NPKL methods. We extend the proposed SimpleNPKL scheme to resolve other non-parametric kernel learning problems, including *colored maximum variance unfolding* [SSBG07], *minimum volume embedding* [SJ07], and *structure preserving embedding*[SJ09]. The encouraging results show that our technique is able to speed up the existing non-parametric kernel learning solutions significantly for several real-world applications.

In chapter 4 and 5, we explore two novel directions towards more effective kernel learning algorithms, beyond the traditional linear multiple kernel combinations in SVM framework. The former one is deep MKL, inspired by recent research in deep learning, while the second is unsupervised MKL, inspired by recent advances in local coordinate coding techniques. We evaluate these techniques on several publicly available datasets.

In chapter 6 and 7, we apply kernel learning (parametric or non-parametric) algorithms to two important applications: image re-ranking and social strength modeling on multimedia platforms. Empirical results verify both the efficiency and effectiveness of our techniques in real applications.

In the last chapter, we make the conclusion of this thesis and discuss future directions towards improving kernel learning.

To summarize, Figure 1.5 presents a high-level organization of this thesis. Our contribution can be categorized into the nodes in this figure. Specifically,

- 9: chapter 3 presents a fast algorithm for non-parametric kernel learnring, SimpleNPKL;
- 7: Chapter 4 presents a deep multiple-layer multiple kernel learning framework;
- 5: Chapter 7 presents an application of MKL for social strength modeling;
- 6: Chapter 5 presents an unsupervised MKL foramework;
- 10: Chapter 6 presents an image ranking framework with the idea of non-parametric kernel learning, using the idea of SimpleNPKL.

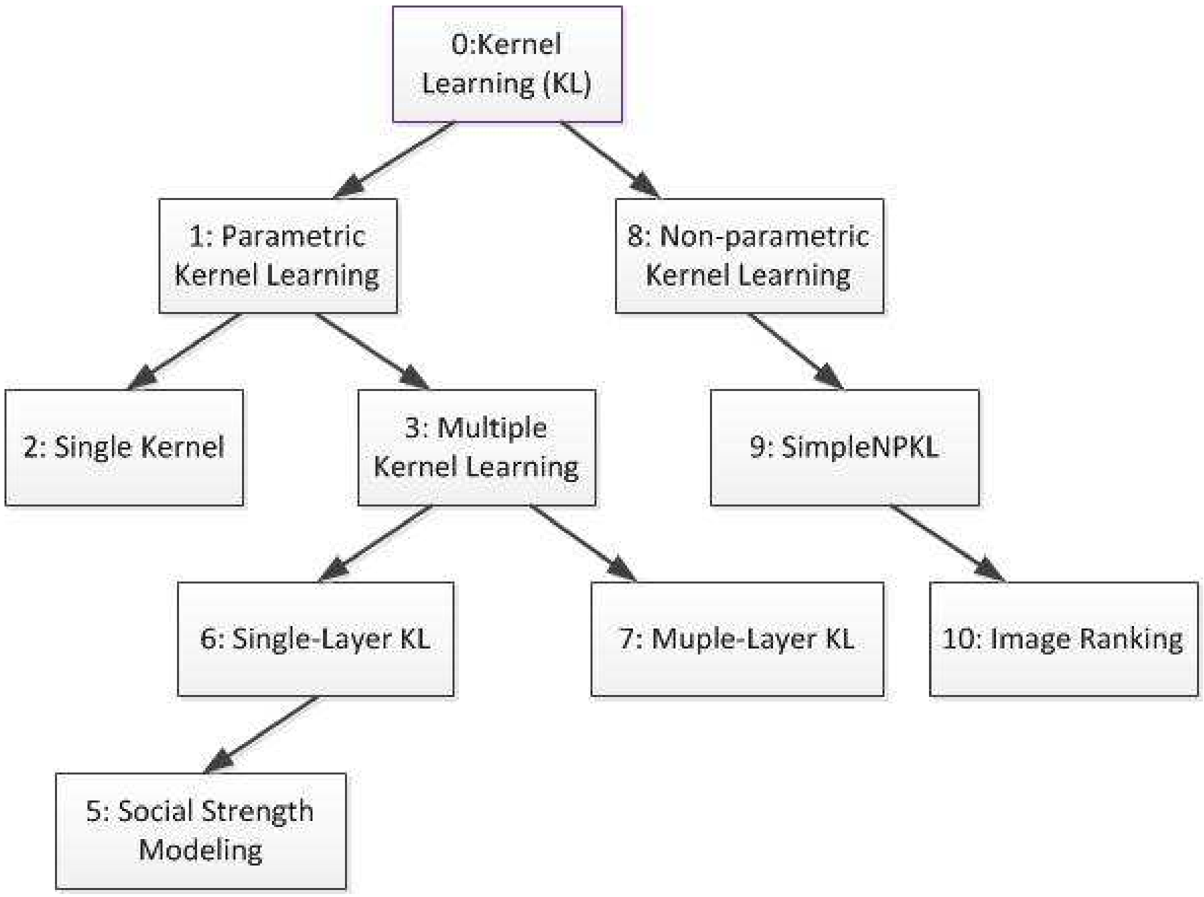

0:Kernel Learning (KL)
1: Parametric Kernel Learning
8: Non-parametric Kernel Learning
2: Single Kernel
3: Multiple Kernel Learning
9: SimpleNPKL
6: Single-Layer KL
7: Muple-Layer KL
10: Image Ranking
5: Social Strength Modeling

Figure 1.1: Organization on the topic kernel learning.

## 1.6 Notation Table

To make the rest presentation clear, we give the common notations in Table 7.1 and common abbreviations in Table 1.2 for reference.

In the whole thesis, we use bold upper case letters to denote matrix, bold lower case letters to denote vector, and curlicue letters to denote a vector space (could be for training samples, decision functions, etc.).

Table 1.1: Some common notations used in the thesis.

| Notation | Explanation |
|---|---|
| $\mathbb{R}$ | domain of real number |
| $\mathbb{N}$ | domain of negative integer |
| $\mathbb{S}^n_+$ | $n$-dimensional symmetric positive semi-definite matrix |
| $\mathcal{X}$ | domain from which the samples come from |
| $\mathbf{X}$ | a set of points |
| $\mathbf{x}$ | a single point, typically belonging to $\mathbb{R}^d$ |
| $x_i$ | the $i$th component of $\mathbf{x}$ |
| $y$ | the label of $\mathbf{x}$, taking value $\pm 1$ in binary classification case |
| $\mathbf{x}^\top$ | transpose of $\mathbf{x}$ |
| $\langle \mathbf{x}, \mathbf{x}' \rangle$ | $:= \mathbf{x}^\top \mathbf{x}'$, inner product between vectors (matrix) |
| $\mathbf{x}_i \circ \mathbf{x}_j$ | element-wise multiplication between vectors (matrix) |
| $k$ | kernel function |
| $\Phi$ | the implicit mapping function of some kernel $k$ |
| $\mathbf{K}$ | kernel matrix, the value of $(i, j)$ entry is the evaluation $k(\mathbf{x}_i, \mathbf{x}_j)$ |
| tr $\mathbf{K}$ | the trace of the matrix $\mathbf{K}$ |
| $\|\mathbf{K}\|_F$ | the Frobenius norm of the matrix $\mathbf{K}$ |
| $f$ | a decision function defined on $\mathcal{X}$ |
| $\mathcal{H}_K$ | Hilbert space induced by some kernel $\mathbf{K}$ |

Table 1.2: Some common abbreviations used in the thesis.

| Abbreviation | Explanation |
|---|---|
| SVM | support vector machine |
| KL | kernel learning |
| MKL | multiple kernel learning |
| $L_p$-MKL | $L_p$-norm regularized MKL |
| NPKL | non-parametric kernel learning |
| DMKL | deep multiple kernel learning |
| UMKL | unsupervised multiple kernel learning |
| SDP | semi-definite programming |
| PSD | positive semi-definite |
| PDS | positive definite and symmetric |
| SSM | social strength modeling |

# Chapter 2

# Literature Review

In this chapter we would establish the key components of kernel machines and the underlying statistical machine learning theory. We first give formal definitions on positive definite kernel and then introduce support vector machine (SVM), which is one of the most successful and influential discriminative models. The concept of kernel is tightly coupled with such kernel machines and plays a central role for their success. Then we explore classical statistical learning theories, which in turn would fit the case of kernel learning. We would like to emphasize the importance of unifying the risk bounds of learning algorithms as it may provide insight about to which extent kernel learning is possible or effective. Since optimization techniques tightly interplay with machine learning algorithms, we briefly introduce the convex optimization problems. We pay special attention to *semidefinite program* due to its importance in kernel learning. Then we discuss the transductive inference issues with some insights, which are related to our major contribution of simpleNPKL algorithms.

## 2.1 Positive Semidefinite Kernels

Now we give formal definitions on kernels and related concepts.

### 2.1.1 Positive Definite Kernels

Suppose we are given empirical data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a nonempty set from which the inputs $\mathbf{x}_i$ are taken, $y_i \in \mathcal{Y}$ are class labels. In the case of binary classification, our goal is to predict the corresponding $y \in \{\pm 1\}$ for some new input $\mathbf{x} \in \mathcal{X}$. To this end, we need some similarity measure defined on $\mathcal{X}$:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}') \tag{2.1}$$

satisfying, for $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$,

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle, \tag{2.2}$$

where $\Phi$ is a map from the *input space* $\mathcal{X}$ to some dot product space $\mathcal{H}$, sometimes called the *feature space*. $\langle , \rangle$ is a valid inner product in feature space. The similarity measure $k$ is usually referred as *kernel*, and $\Phi$ is called its *feature map*.

The advantage induced by kernel is that it allows us to construct algorithms in dot product spaces. Typically the mapped space is high dimensional. Therefore, the data has a higher chance to be separated by a linear classifier in the feature space comparing with the input space. Famous example of such algorithms include support vector machine (SVM)[Vap98], kernel logistic regression (KLR)[ZH01], kernel principle component analysis (KPCA)[SSM98], etc..

We have required that a kernel satisfies (2.2), that is, correspond to a dot product in some dot product space. In the present section we show that the class of kernels that can be written in the form (2.2) coincides with the class of positive definite kernels. This has far-reaching consequences. There are examples of positive definite kernels which can be evaluated efficiently even though they correspond to dot products in infinite dimensional dot product spaces. In such cases, substituting $k(\mathbf{x}, \mathbf{x}')$ for $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ is crucial. In the machine learning community, this substitution is called the *kernel trick*.

**Definition 2.1** *Given the kernel function $k$ and inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, the $n \times n$ matrix $\mathbf{K}$ defined by*

$$[\mathbf{K}]_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.3}$$

*is called the Gram matrix (or kernel matrix) of $k$ w.r.t. $\mathbf{x}_1, \ldots, \mathbf{x}_n$.*

**Definition 2.2** *A real $n \times n$ symmetric matrix $\mathbf{K}$ satisfying*

$$\sum_{i,j=1}^{n} c_i c_j K_{ij} \geq 0 \tag{2.4}$$

*for all $c_i \in \mathbb{R}$ is called* positive definite. *If equality in (2.4) only occurs for $c_1 = \ldots = c_n = 0$, then we shall call the matrix* strictly positive definite.

**Definition 2.3** *Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which for all all $n \in \mathbb{N}, \mathbf{x}_i \in \mathcal{X}, i \in [n]$ gives rise to a positive definite Gram matrix is called a* positive definite kernel. *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which for all $n \in \mathbb{N}$ and distinct $\mathbf{x}_i \in \mathcal{X}$ gives rise to a strictly positive definite Gram matrix is called a* strictly positive definite kernel.

We simply use kernels for positive definite kernels when no confusion is arising in the context.

### 2.1.2 Reproducing Kernel Hilbert Space

When we design a machine learning algorithm, the first problem is the existence and uniqueness of the solution. In this section, we introduce the concept of *reproducing kernel Hilbert space* (RKHS), which can be constructed from a kernel $k$. The solution of many algorithms, like SVM, can be proved unique in such a space according to *representer theorem*[SHS01].

We now define a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$, denoted as $\mathbb{R}^{\mathcal{X}}$, via

$$\begin{aligned}\Phi:\ & \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \\ & \mathbf{x} \mapsto k(\cdot, \mathbf{x}),\end{aligned}$$

here $\Phi(\mathbf{x}) = k(\cdot, \mathbf{x})$ denotes the function that assigns the value $k(\mathbf{x}', \mathbf{x})$ to $\mathbf{x}' \in \mathcal{X}$.

We next construct a dot product space containing the images of the inputs under $\Phi$. For this purpose, we first make it a vector space by forming linear combinations

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i), \tag{2.5}$$

here $n \in \mathbb{N}, \alpha_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathcal{X}$.

We turn the space of $f$ in form of (2.5) into a inner product space by defining the dot product between $f$ and another function $g(\cdot) = \sum_{j=1}^{n'} k(\cdot, \mathbf{x}'_j)$ as

$$\langle f, g \rangle := \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}'_j). \tag{2.6}$$

One can easily show that (2.6) is well defined, bilinear, symmetric, and positive definite. Furthermore, the operation of (2.6) is an inner product.

In particular, when $g(\cdot) = k(\cdot, \mathbf{x})$, we have

$$\langle k(\cdot, \mathbf{x}), f \rangle = f(\mathbf{x}), \text{ and } \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}'). \tag{2.7}$$

By virtue of these properties, $k$ is called a *reproducing kernel.*

One can get a Hilbert space $\mathcal{H}_K$ of $f$ in form of (2.5) by making this space complete in the norm corresponding to the dot product (2.6). We call $\mathcal{H}_K$ a reproducing kernel Hilbert space (RKHS). The Moore-Aronszajn theorem [Aro50] states that, for every positive definite kernel on $\mathcal{X} \times \mathcal{X}$, there exists a unique RKHS and vice versa.

### 2.1.3 Some Examples

Some kernels are widely applied. For example:

- *Linear Kernel*: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$. The kernel is the inner product in the input space.
- *Polynomial Kernel*: $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^d$.
- *Gaussian Kernel*: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma\}$.

The linear kernel is very effective in text categorization because text data is usually high-dimensional already. Gaussian kernel is widely used in image data, in constructing similarity matrix.

### 2.1.4 General Tricks for Constructing Kernels

We now gather a number of results useful for designing positive definite kernels. Many of these techniques concern manipulations that preserve the positive definiteness of Definition 2.2; which is to say, closure properties of the set of admissible kernels. Thus we can operate known kernels to obtain a new one. A more detailed summarization can be found in [SS02]. We summarize the key theorems therein to make this thesis complete.

It is easy to justify that linear combination and product of kernels preserve psd, as is stated in the following theorem:

**Theorem 2.1** *The set of kernels forms a convex cone, closed under pointwise convergence. In other words,*

- *if $k_1$ and $k_2$ are kernels, and $\alpha_1, \alpha_2 \geq 0$, then $\alpha_1 k_1 + \alpha_2 k_2$ is a kernel;*
- *if $k_1, k_2, \ldots$ are kernels, and $k(\mathbf{x}, \mathbf{x}') := \lim_{n\to\infty} k_n(\mathbf{x}, \mathbf{x}')$ exists for all $\mathbf{x}, \mathbf{x}'$, then $k$ is a kernel.*

**Theorem 2.2** *If $k_1$ and $k_2$ are kernels, then $k_1 k_2$, defined by $(k_1 k_2)(\mathbf{x}, \mathbf{x}') := k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$, is a kernel.*

Note that the corresponding result for positive definite matrices concerns the positivity of the matrix that is obtained by element-wise products of two positive definite matrices.

Some kernels are defined as functions of dot product, e.g., polynomial kernel. The following theorem provides necessary conditions for such kernels:

**Theorem 2.3** *A differentiable function of the dot product $k(\mathbf{x}, \mathbf{x}') = k(\langle \mathbf{x}, \mathbf{x}' \rangle)$ has to satisfy $k(t) \geq 0, k'(t) \geq 0$ and $k'(t) + tk''(t) \geq 0$ for any $t \geq 0$, in order to be a positive definite kernel.*

Now we consider a different form of product, the tensor product, which also works if the two kernels are defined on different domains.

**Theorem 2.4** *If $k_1$ and $k_2$ are kernels defined respectively on $\mathcal{X}_1 \times \mathcal{X}_1$ and $\mathcal{X}_2 \times \mathcal{X}_2$, then their tensor product,*

$$(k_1 \otimes k_2)(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}'_1, \mathbf{x}'_2) = k_1(\mathbf{x}_1, \mathbf{x}'_1) k_2(\mathbf{x}_2, \mathbf{x}'_2),$$

*is a kernel on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$. Here, $\mathbf{x}_1, \mathbf{x}'_1 \in \mathcal{X}_1$ and $\mathbf{x}_2, \mathbf{x}'_2 \in \mathcal{X}_2$.*

This result follows from the fact that the product of kernels is a kernel. There also exits a corresponding generalization from the sum of kernels to their *direct sum*.

**Theorem 2.5** *If $k_1$ and $k_2$ are kernels defined respectively on $\mathcal{X}_1 \times \mathcal{X}_1$ and $\mathcal{X}_2 \times \mathcal{X}_2$, then their direct sum,*

$$(k_1 \oplus k_2)(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}'_1, \mathbf{x}'_2) = k_1(\mathbf{x}_1, \mathbf{x}'_1) + k_2(\mathbf{x}_2, \mathbf{x}'_2),$$

*is a kernel on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$. Here, $\mathbf{x}_1, \mathbf{x}'_1 \in \mathcal{X}_1$ and $\mathbf{x}_2, \mathbf{x}'_2 \in \mathcal{X}_2$.*

This construction can be useful if the different parts of the input have different meanings, and should be dealt with differently. Some classical kernels (e.g., [Hau99]) are based on the theorems in this section.

## 2.2 Support Vector Machine

In this section, we summarize the key ideas of the well-known SVM algorithm. It has shown strengths in a large variety of applications and become a standard method. Its *loss + regularization* framework has revolutionized the field of pattern recognition. Therefore, we first introduce SVM before discussing kernel learners. Classical results on statistical learning theory are also included.

### 2.2.1 Maximum Margin Classifier

Now we are ready to introduce SVM [CV95, Vap98]. Actually this is not an easy task due to the large amount of works based on this famous algorithm. Refer to [Bur98] and [Mm06] for very good surveys. In this section, we describe the key features of SVM. The content here overlaps with [Mm06].

We still restrict ourselves to binary classification, where the discriminant function is linear in the feature space induced by the mapping $\Phi$. Among the infinite number of existing separating hyperplanes, the support vector machine looks for the plane that lies furthermost from both classes, known as the maximal margin hyperplane. To be more specific, denote by $\mathbf{w}^\top \Phi(\mathbf{x}) + b = 0$ any separating hyperplane in the feature space. Under the assumption of separability, we can rescale $\mathbf{w}$ and $b$ so that $|\mathbf{w}^\top \Phi(\mathbf{x}) + b| = 1$ for those points in each class nearest to the hyperplane. Accordingly, it holds that for every $i \in [n]$,

$$\mathbf{w}^\top \Phi(\mathbf{x}_i) + b \begin{cases} \geq 1, & \text{if } y_i = +1 \\ \leq -1, & \text{if } y_i = -1 \end{cases} \tag{2.8}$$

After the rescaling, the distance from the nearest point in each class to the hyperplane is $1/\|\mathbf{w}\|$. Hence, the distance between the two boundary hyperplances is $2/\|\mathbf{w}\|$, which is called the *margin*. To maximize the margin, the following optimization problem has to be solved:

$$\begin{array}{ll} \min_{\mathbf{w},b} & \|\mathbf{w}\|^2 \\ \text{s.t.} & y_i(\mathbf{w}^\top \Phi(\mathbf{x}_i) + b) \geq 1, i \in [n] \end{array} \tag{2.9}$$

where the square in the norm of $\mathbf{w}$ has been introduced to make the problem quadratic. Notice that due to the convexity, the problem above has a unique global optimal solution. Let $\mathbf{w}^*$ and $b^*$ denote the solution. Then the decision surface in the feature space is determined by $f(\mathbf{x}) = (\mathbf{w}^*)^\top \Phi(\mathbf{x}) + b^* = 0$. Points $\Phi(\mathbf{x}_i)$ which satisfy the equalities $y_i((\mathbf{w}^*)^\top \Phi(\mathbf{x}_i) + b^*) = 1$ are called *support vectors* (SV), which could be automatically identified from the solution of the optimization problem. The SVs often represent a small fraction of the sample. The hyperplane $f(\mathbf{x}) = 0$ is completely determined by the subsample made up of the SVs. This fact implies that, for many applications, the evaluation of the decision function $f(\mathbf{x})$ is computationally efficient, allowing the use of SVMs on large data sets in real-time environments.

We can extend (2.9) to non-separately case by using slack variables for classification error:

$$\begin{array}{ll} \min_{\mathbf{w},b,\xi} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} & y_i(\mathbf{w}^\top \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, i = 1, \ldots, n \end{array} \tag{2.10}$$

This is the most widely used SVM formulation. Now we have to face the key issues of the formulation above: how to use $\Phi(\mathbf{x})$ to map the data into a probably high-dimensional space. Apparently, it is far from trivial to find such a nonlinear transformation. However, by introducing the Lagrangian multipliers $\alpha_i$ for each inequality constraint, and use the kernel evaluation $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \rangle \Phi(\mathbf{x}_j)$, we arrive at the dual problem of (2.10) (for the detailed derivation, refer to [Bur98]):

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i \tag{2.11}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_i\alpha_i = 0,$$

$$0 \le \alpha_i \le C, i \in [n]. \tag{2.12}$$

This problem is in form of a quadratic program on $\boldsymbol{\alpha}$ [BV04]. Therefore, it has unique global optimal solution. From the KKT condition for optimality [BV04], the primal variable $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \Phi(\mathbf{x}_i)$, and the decision function becomes $f(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) + b^*$. Some good solvers for (2.11) have been developed, for example, LibSVM[CL11], SVM$^{light}$[Joa08], Torch[RCM02], etc.

Note, that the potential high dimension (could be infinite dimensional) operations are replaced by a simple kernel evaluation. We even do not necessarily known the mapping function $\Phi$ explicitly. One can just focus on selecting a proper kernel function $k$ to get good performance. This provides flexibility of the SVM algorithm, which consequently promotes the whole kernel learning topic.

The dual formulation (2.11) provides another interpretation, i.e., regularization in the Hilbert space induced by the kernel $k$. Now we extend SVM to the general learning problem in RKHS.

### 2.2.2 Supervised Learning in RKHS

With the notions in previous sections, we now consider the standard supervised learning task of which the solution exists in an RKHS:

$$\min_{f} L(f,\mathbf{y}) + C\|f\|_K^2 \tag{2.13}$$

here the first term $L(f,\mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, f_i)$ is the empirical loss, $\|f\|_H$ is the norm of $f$ in $\mathcal{H}$ for regularization purpose. This *loss + regularization* framework enjoys great popularity due to both its theoretical generalization ability and good empirical performance. Among the large volume of works, SVM described in last section is one of the most successful work in recent machine learning research[CV95, Bur98, Mm06].

We obtain a non-linear SVM by replacing the dot product with kernel evaluation in (2.11), which exactly coincides with the function learning in RKHS, i.e., learning problem (2.13), in form of *quadratic programming*[BV04]. Standard optimization techniques, e.g., *interior point method*, handles $n$ of moderate size very well.

To summarize, SVM combines the maximum margin principle with convex optimization techniques, and the idea of a kernel mapping. From its dual formulation we conclude that it is a regularization problem in an RKHS. The generalization ability is bounded by the theories of VC-dimension or Rademacher complexity. The well-known *representer theorem* guarantees the existence and uniqueness of the solution [SHS01].

**Theorem 2.6** *The optimal solution $f^*$ of (2.13) admits the kernel expansion:*

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}).$$

The good theoretical foundation of the loss + regularization framework has been established in *statistical learning theory* [Vap98]. We briefly describe some classical kernel machines. The difference is incurred by different loss functions in the framework (2.13).

*Hinge loss.* In this case, we have

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i f(\mathbf{x}_i))_+ + \frac{C}{2}\|f\|^2_{\mathcal{H}_\mathcal{K}}.$$

This is exactly the standard SVM.

*Logistic loss.*, In this case, the learning task is:

$$\min_{f\in\mathcal{H}} -\frac{1}{n}\sum_{i=1}^{n}[y_i f(\mathbf{x}_i) - \ln(1 + \exp(f(\mathbf{x}_i)))] + \frac{C}{2}\|f\|^2_{\mathcal{H}_\mathcal{K}}.$$

This is kernelized logistic regression. Due to the logistic loss function, we lose sparsity for the solution. However, it offers a natural estimate of the probability as $p(y|\mathbf{x}) = 1/(1 + \exp(-\sum_{i=1}^{n}\alpha_i k(\mathbf{x}_i, \mathbf{x})))$, which could be more important than the classification rule in some applications.

*Square loss.* In this case, we obtain regularized least square problem:

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}(y - f(\mathbf{x}_i))^2 + \frac{C}{2}\|f\|^2_{\mathcal{H}_\mathcal{K}}.$$

## 2.3 Generalization Risk of Learning in RKHS

We commence by formulating the target of analysis explicitly in classical supervised setting. Given a set of data $\mathbf{X}^n = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$ drawn from the space $\mathcal{X} \times \mathcal{Y}$ according to some fixed distribution $\mathbb{P}$, the problem of *supervised classification* is to create a function $f : \mathcal{X} \to \mathcal{Y}$ which predicts the label $y_i \in \mathcal{Y}$ of given $\mathbf{x}_i \in \mathcal{X}$. The function $f$ is called a classifier. Error occurs on $\mathbf{x}_i$ if $f(\mathbf{x}_i) \neq y_i$. In this chapter, we focus on the binary case, that is, $\mathcal{Y} = \{-1, 1\}$.

Let $(\mathbf{x}, y)$ be an $\mathcal{X} \times \mathcal{Y}$-valued random pair. We measure the performance of $f$ by its *risk* (or more clearly, *probability of error*)

$$L(f) = \mathbb{P}\{f(\mathbf{x}) \neq y\}. \tag{2.14}$$

Let $f^* = \arg\min_f L(f)$ be the *Bayes classifier* and $L^*$ be the corresponding Bayes risk. In *structural error decomposition*, we write the excess error as [BBL03, OBL05]

$$L(f_n) - L^* = [L(f_n) - \inf_{f \in \mathcal{F}}(f)] + [\inf_{f \in \mathcal{F}}(f) - L^*], \tag{2.15}$$

here the subscript $n$ of $f$ indicates that $f$ depends on the training data $D_n$. The first term on the right side is called the *estimation error*, while the second is the *approximation error*. In order to minimize (2.15), we need to choose the function class $\mathcal{F}$ wisely [BBL02]:

- $\mathcal{F}$ should be sufficiently large such that the loss of the best function in $\mathcal{F}$ is close to $L^*$, and
- $\mathcal{F}$ should be sufficiently small such that finding the best candidate in $\mathcal{F}$ based on the data $D$ is still possible.

These two requirements are clearly in conflict. Consider two extreme cases: 1) $|\mathcal{F}| = 1$: we only have one classifier in $\mathcal{F}$, which means we are simply guessing $f^*$; 2) $\mathcal{F} = \{-1, 1\}^{\mathcal{X}}$: $\mathcal{F}$ consists of all the possible classifiers, which means the approximation error is zero. However, the estimation error could be very large. Given the data $D_n$, one wishes to selection a good $f$ from one $\mathcal{F}_i$ among a candidate sequence of model classes $\mathcal{F}_1, \mathcal{F}_2, \ldots$ so that $L(f_n)$ is minimized as much as possible. This is the problem of *model selection* [BBL02].

Estimating the approximation error is usually hard since it requires knowledge about the target $f^*$. Classically, in statistical learning theory it is preferable to avoid making specific assumptions about the target (such as its belonging to some model class $\mathcal{F}$), but the assumptions are rather on the value of $L^*$. It is also known that for any (consistent) algorithm, the rate of convergence to zero of the approximation error can be arbitrarily slow if one does not make assumptions about the regularity of the target, while the rate of convergence of the estimation error can be computed without any such assumption. We will thus focus on the estimation error [BBL03]. Moreover, instead of (2.15), we have the following decomposition of the risk in the sequel:

$$L(f) = L_n(f) + [L(f) - L_n(f)], \tag{2.16}$$

where $L_n$ is the *empirical error*

$$L_n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{f(\mathbf{x}_i) \neq y_i\}. \tag{2.17}$$

In *empirical minimization*, one estimates the risk by its empirical counterpart and some quantity which approximates or upper bounds $L(f) - L_n(f)$. Let $f_n^* = \arg\min_{f\in\mathcal{F}} L_n(f)$, it is easy to justify that [OBL05]

$$L(f_n^*) - \inf_{f\in\mathcal{F}} L(f) \leq 2\sup_{f\in\mathcal{F}} |L_n(f) - L(f)|,$$

and

$$L(f_n^*) \leq L_n(f_n^*) + \sup_{f\in\mathcal{F}} |L(f) - L_n(f)|. \tag{2.18}$$

If the term $\sup_{f\in\mathcal{F}} |L_n(f) - L(f)|$ is large, $f_n^*$ tends to fail in generalization ability, i.e., *overfitting* occurs. To compensate this effect, we need to penalize a complexity measure over $\mathcal{F}$ to bound the term $\sup_{f\in\mathcal{F}} |L_n(f) - L(f)|$, which is the key idea of *complexity regularization.*

**Remark 2.7:** □ To summarize, our first target of analysis is the risk bound stated in the form: With probability of at least $1-\delta$,

$$L(f_n) \leq L_n(f_n) + \Omega(n, \mathcal{F}; \delta),$$

where $\Omega$ is non-increase in sample size $n$ and non-decreasing in the richness of $\mathcal{F}$. Naturally we choose $\Omega$ relating to the deviation $\sup_{f\in\mathcal{F}} |L_n(f) - L(f)|$.

Now we are ready to present some classical results about the risk bound. The computation of $\Omega$ often involves complexity / capacity measure of the functions class $\mathcal{F}$. For example [BM02], the *Rademacher complexity*, which uses the ability of $\mathcal{F}$ to fit real random noise to measure its capacity:

**Definition 2.4** *For a sample $\mathbf{X}^n = \{\mathbf{x}_i\}_{i=1}^n$, the* **empirical Rademacher complexity** *of $\mathcal{F}$ is the random variable*

$$\hat{R}_n(\mathcal{F}) = \mathbb{E}_\sigma \left[ \sup_{f\in\mathcal{F}} \left| \frac{2}{n} \sum_{i=1}^n \sigma_i f(\mathbf{x}_i) \right| \right]$$

*where $\sigma$ is independent uniform $\{\pm 1\}$-valued random variables. The* **Rademacher complexity** *of $\mathcal{F}$ is $R_n(\mathcal{F}) = \mathbb{E}_{\mathbf{X}^n} \hat{R}_n(\mathcal{F})$.*

The Rademacher complexity and its empirical version is connected by the following Lemma:

**Lemma 2.1** *For classes of functions $\mathcal{F}$ mapping to $[0,1]$ and a sample $\mathbf{X}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we have with probability of at least $1-\delta$*

$$R_n(\mathcal{F}) \leq \hat{R}_n(\mathcal{F}) + 2\sqrt{\frac{\ln 1/\delta}{2n}}$$

With the notion of Rademacher complexity, the following theorem gives a generalization bound [KP00, BM02]:

**Theorem 2.8** *Let $\mathcal{F}$ be a $\{\pm 1\}$-valued functions defined on $\mathcal{X}$, with probability of at least $1-\delta$, every function $f \in \mathcal{F}$ satisfies*

$$L(f) \leq L_n(f) + \frac{R_n(\mathcal{F})}{2} + \sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{2.19}$$

*and*

$$L(f) \leq L_n(f) + \frac{\hat{R}_n(\mathcal{F})}{2} + 3\sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{2.20}$$

Classically, the capacity measure is given by the following notion [Vap98]:

**Definition 2.5** *Assume we are given a set of $n$ points which can be labeled in all possible $2^n$ ways, and for each labeling, a member of the set $\mathcal{F}$ can be found which correctly assigns those labels, we say that the set of points is* shattered *by that set of functions. The* **VC dimension** *of $\mathcal{F}$ is defined as the maximum number of training points that can be shattered by $\mathcal{F}$. A sample $\{\mathbf{x}_i^n\}$ is shattered by $\mathcal{F}$ if and only if for every possible labels, there exists at least one function $f \in \mathcal{F}$ that can classify $\{\mathbf{x}_i^n\}$ correctly.*

With this notion, a classical result is given by

**Theorem 2.9** *Let $\mathcal{F}$ be a $\{\pm 1\}$-valued functions defined on $\mathcal{X}$, with probability of at least $1-\delta$, every function $f \in \mathcal{F}$ satisfies*

$$L(f) \leq L_n(f) + \sqrt{\frac{2V(\mathcal{F})\log(n+1)}{n}} + \sqrt{\frac{\ln(1/\delta)}{2n}}, \tag{2.21}$$

*where $V(\mathcal{F})$ denotes the Vapnik-Chervonenkis dimension of $\mathcal{F}$.*

A more recently developed complexity measure over $\mathcal{F}$ is the *local Rademacher complexity* (LRC)[BBM02]. Different from global Rademacher complexity, LRC computes for a sub-class $\mathcal{F}_r \subset \mathcal{F}$ satisfying $\mathbb{E}_n(f) \leq r, f \in \mathcal{F}_r$, where $r$ is a threshold. Due to the limitation of space, refer to [BBM02] for further reading.

Note, the probability of error of $f \in \mathcal{F}$ is bounded by the empirical error and a complexity term defined over the solution space $\mathcal{F}$. However, from the proof of Theorem (2.8) and (2.9), it is clear that different capacity measures results in different risk bounds. Moreover, from the proof of Theorem 2.9, one can see that the bound (4.9) is tighter than (2.21). This fact inspires us that a proper capacity measure could be important for computing the risk bound.

The VC-Dimension is the maximal number of samples that can be classified in all $2^n$ possible ways, which implies that even noise can be classified in all possible ways, though it is the worst possible noise. For the popular RKHS $\mathcal{H}_B = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq B\}$, VC-dim($\mathcal{H}_B$) is $O(N)$ independently of B, here $N$ is the dimensionality of $\mathcal{H}$. Moveover, if $N$ is infinite, the VC-dimension is infinite for any positive $B$. This means the popular regularization by norm in $\mathcal{H}$ is meaningless, from the perspective of VC-dimension.

With risk bounds similar to Theorem (2.9) and (2.8), the risk analysis of $\mathcal{F}$ is reduced to a properly defined complexity measure. Researchers have proposed various complexity measure to derive tighter risk bounds, such as covering number, localized Rademacher complexity, pseudo-dimension, second order Rademacher complexity, etc.

The 0-1 loss in the forgoing analysis is difficult to handle. In real case, we often find a real-valued function $f \in \mathcal{F} \subset \mathbb{R}^{\mathcal{X}}$ rather than a binary classifier. For a prediction function $f : \mathcal{X} \to \mathbb{R}$, we can use $g_f(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$ as the classifier, i.e.,

$$g_f(\mathbf{x}) = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{2.22}$$

It is important to define a proper loss function for real-valued prediction functions [OBL05, YC09]:

**Definition 2.6** *A function $l : \mathbb{R} \to [0, \infty)$ is called a normalized classifying loss if it is convex, $l'(0) < 0$, $\inf_{t \in \mathbb{R}} l(t) = 0$, and $l(0) = 1$.*

We do not expose the theoretical interpretations of loss functions.

**Example 3.1:** Some commonly used loss functions:
Hinge loss: $l(t) = \max(1 - t, 0)$ for soft margin SVM;
Square loss: $l(t) = (1 - t)^2$ for regression. □

With the loss function $l$, the *empirical l-risk* and *$\phi$-risk* can be defined accordingly:

$$L_n^l(\mathbf{X}^n) = \frac{1}{n}\sum_{i=1}^{n} l(f_i, y_i), \quad L^l = \mathbb{E}_{\mathbf{X}^n} L_n^l(\mathbf{X}^n)$$

Thus we can use $L^l(f)$ as a surrogate of $L(g_f)$ to facilitate the optimization of learning. First we need to establish the relationship between *$\phi$-risk* and the real risk.

**Theorem 2.10** *Let $C_l$ denote the uniform upper bound of $l(f(\mathbf{x})y)$, $D_l$ be the Lipschitz constant of $l$. With probability at least $1 - \delta$,*

$$L(f) \leq L_n^l(f) + 2D_l R_n(\mathcal{F}) + C_l\sqrt{\frac{\log 1/\delta}{2n}} \tag{2.23}$$

For the loss functions we consider, one can show that $L^l_n \to L^l$ implies $L_n \to L$. Refer to [Zha04] and [PBM06] for further analysis of convex loss functions. With Theorem 2.10, we can just focus on the Rademacher complexity of real-valued $\mathcal{F} \subset \mathbb{R}^{\mathcal{X}}$ for Hinge loss. For other loss functions, asymptotically, we can analyze the properties of $L^l_n$ and $L^l$ instead of $L_n$ and $L$.

The learning in *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}_k$ associated with some kernel $k$ refers to the following learning problem:

$$\min_{f \in \mathcal{H}} L^l_n(f) + \lambda \|f\|^2_{\mathcal{H}_k}, \tag{2.24}$$

here $\|f\|_{\mathcal{H}_k}$ is the norm of $f$ in $\mathcal{H}_k$ . When taking $l$ as hinge loss, we obtain one of the well-known support vector machine formulations.

**Theorem 2.11** *[SHS01] The optimal solution $f^*$ of (2.24) admits the kernel expansion:*

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}).$$

Comparing with the error decomposition (2.16), the objective (2.24) minimizes the empirical loss and a regularization term.

The following theorem reveals how the regularization $\|f\|_{\mathcal{H}_K}$ relates to the Rademacher complexity measure:

**Lemma 2.2** *Consider the subset $\mathcal{H}_B \subseteq \mathcal{H}_K$:*

$$\mathcal{H}_B := \{\mathbf{x} \to \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) : \boldsymbol{\alpha}' K \boldsymbol{\alpha} \leq B^2\},$$

*the empirical Rademacher complexity of the class $\mathcal{H}_B$ satisfies*

$$\hat{R}_n(\mathcal{H}_B) \leq \frac{2B}{n} \sqrt{\sum_{i=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j)} = \frac{2B}{n} \sqrt{tr\, K},$$

where $[K]_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ is the Gram matrix corresponding to kernel function $k$, tr $K$ is the trace of the matrix $K$.

Thus the regularization $\|f\|_{\mathcal{H}_K}$ term affects the risk bound of $f \in \mathcal{H}_K$ through the Rademacher complexity measure. Due to this solid theoretical interpretation and good empirical performance, the *loss + regularization* framework enjoy grate popularity and achieves great success.

**Remark 2.12:** □ With the forgoing analysis, we conclude:

- In seek of good generalization ability, one needs to bound some effective complexity measure on $\mathcal{F} \subset \{\pm 1\}^{\mathcal{X}}$.
- We can use convex loss function $l$ as a surrogate of 0-1 loss to make the learning problem solvable. For proper defined $l$, it is guaranteed that $L_n^l \to L^l$ implies $L_n \to L$.
- For the *loss + regularization* framework in an RKHS:
  - VC-dimension cannot interpret the success because $\|f\|_{\mathcal{H}_K}$ cannot bound VC-dimension of $\mathcal{H}$, whilst
  - Rademacher complexity works because it increases with $\|f\|_{\mathcal{H}_K}$.

Thus, it is important to derive proper complexity measure in order to characterize the estimation error of the selected function class $\mathcal{F}$ well.

## 2.4 Transductive Learning

Traditional classifiers use only labeled data (feature / label pairs) to train. Labeled instances however are often difficult, expensive, or time consuming to obtain, as they require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect, but there has been few ways to use them. *Semi-supervised learning* addresses this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers. Because semi-supervised learning requires less human effort and gives higher accuracy, it is of great interest both in theory and in practice. A large volume of works on this topic emerges on the top conferences and journals on machine learning research. A good survey is given by [Zhu05].

We categorize a semi-supervised learning (SSL) algorithm according to whether it can handle unseen data readily, i.e., SSL can be either *inductive* or *transductive*. Inductive learners can naturally handle unseen data whilst transductive learners only works on the fixed set of labeled and unlabeled training data and cannot handle unseen data. Graph-based methods are often transductive. Transductive support vector machine (TSVM) is actually inductive [Vap98, OCK08, CSWB06] (the term "S$^3$VM" is also popularly used. We uniformly use the term TSVM in this thesis. Though motivated by the same assumption, it has several different versions depending on different optimization schemes).

Transductive learning is an important fundamental problem. The mechanism that provides an advantage to the transductive mode of inference over the inductive mode was clear from the very beginning of statistical learning theory. It can be seen in the proof of the very basic theorems on uniform convergence.

**Theorem 2.13** *Let $f$ be $\{\pm 1\}$-valued functions defined on a set $\mathcal{X}$. Let $P$ be a probability distribution on $\mathcal{X}\times\{\pm 1\}$, and suppose that $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n+1}, y_{n+1}), \dots, (\mathbf{x}_{2n}, y_{2n})\}$ and $(\mathbf{x}, y)$ are chosen independently according to $P$. Then we have*

$$P(\sup_f |P(f(\mathbf{x}) \neq y) - P_n(f)| \geq \varepsilon) \leq 2P(\sup_f |P_n(f) - P_{n'}(f)| \geq \frac{\varepsilon}{2}), \tag{2.25}$$

*where*

$$P_n(f) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\mathbf{x}_i)|$$

*and*

$$P_{n'}(f) = \frac{1}{n}\sum_{i=n+1}^{2n} |y_i - f(\mathbf{x}_i)|$$

*are the empirical risks constructed using two different samples.*

The bound for uniform convergence was obtained as an upper bound of the right hand side of (2.25). It implies that to obtain a bound for inductive inference we first obtain a bound for transductive inference and then obtain an upper bound for it. This means that transductive inference is a fundamental step in machine learning. Due to the access of the test data, it is often easier to learn in this setting comparing with inductive learning. Most of the nonparametric kernel methods fit this setting naturally.

We have shown the importance of kernels in the case of supervised learning. As we expected, it also widely used in semi-supervised learning. TSVM[CSWB06] is one of the most mature SSL algorithms. A typical version of TSVM is to enforce the decision boundary to go through the areas of low density. The semi-supervised effect can also be achieved with a two stage procedure: (1) use unlabeled data to design a kernel $\hat{k}(\cdot, \cdot)$; (2) replace $k(\cdot, \cdot)$ with $\hat{k}(\cdot, \cdot)$ in the supervised formula (2.13). Representer theorems and various SVM solvers are readily extended to SSL. Thus we obtain a new "TSVM" by designing a new kernel that involves unlabeled data.

In the transductive setting, since both the training and test data are given to the learner, one can just focus on learning a kernel matrix, instead of devising a parametric kernel function, which could be more difficult in general. This transductive kernel provides more flexibility comparing with closed-form kernel functions. Empirical evidence shows such methods are promising[HJL07].

Due to the positive semi-definiteness of the kernel matrix, the resultant optimization problem often admits a convex problem, in which the global optimum is solved by the state-of-the-art interior point algorithm in polynomial time[BV04]. However, such methods fail for even datasets of moderate size. Therefore, kernel learning calls for efficient optimization techniques based on the structure of the problem at hand.

## 2.5 Convex Optimization

The above analysis focuses on the effectiveness aspect of learning. The other dimension is the efficiency aspect, i.e., how to solve the optimization problem. Indeed, the fields of machine learning and mathematical programming are increasingly intertwined[BPH06]. Optimization problems lie at the heart of most machine learning approaches. Machine learning researchers have embraced the advances in mathematical programming allowing new types of models to be pursued. The special topic includes models using quadratic, linear, second-order cone, semidefinite, and semi-infinite programs, though the qualities of good optimization algorithms from the machine learning and optimization perspectives can be quite different. A discussion on this interplay is conducted by [BPH06]. Specifically, we briefly introduce the concept of convex optimization, which is in the following form[BV04]:

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & f_i(\mathbf{x}) \le 0, i = 1, \ldots, m \\ & \mathbf{a}_i^\top \mathbf{x} = b_i, i = 1, \ldots, p, \end{aligned}$$

where $f_0, \ldots, f_m$ are convex functions.

Convex optimization techniques are widely used in learning algorithms. The global optimum is unique and can be efficiently solved. If a problem has formulated into an convex optimization problem, it means it is solved with the off-the-shelf optimization techniques. However, some types problem, for instance, semi-definite programming (SDP), is still inefficient for real applications. Due to the importance of such problems, it is worthy of describing it specifically.

The standard primal-dual pair of semidefinite programs are given by:

$$\begin{aligned} \max \quad & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{X}) = \mathbf{b} \\ & \mathbf{X} \succeq \mathbf{0}. \end{aligned} \tag{2.26}$$

and

$$\begin{aligned} \min \quad & \langle \mathbf{b}, \mathbf{y} \rangle \\ \text{s.t.} \quad & \mathbf{Z} = \mathcal{A}^\top(\mathbf{y}) - \mathbf{C} \\ & \mathbf{Z} \succeq \mathbf{0}. \end{aligned} \tag{2.27}$$

Here the linear operator $\mathcal{A}$ is defined by

$$\mathcal{A}(\mathbf{X}) = \begin{bmatrix} \langle \mathbf{A}_1, \mathbf{X} \rangle \\ \vdots \\ \langle \mathbf{A}_m, \mathbf{X} \rangle \end{bmatrix} \quad \text{and} \quad \mathcal{A}^\top(\mathbf{y}) = \sum_{i=1}^m y_i \mathbf{A}_i. \tag{2.28}$$

The constraints on kernel matrix can be readily expressed using the inner product tr $\mathbf{AK}$. For example, the constraint $\mathbf{K}_{ij} \leq b$ can be written as tr $\mathbf{KA} \leq b$ where $\mathbf{A} = \mathbf{x}\mathbf{y}^\top$, with $x_j = 1, y_i = 1$ and all other entries of $\mathbf{x}$ and $\mathbf{y}$ are 0. For the distance constraint $\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} \leq b$, it can be written as tr $\mathbf{KA} \leq b$, where $\mathbf{A} = \mathbf{z}\mathbf{z}^\top, z_i = 1, z_j = -1$, and all other components of $\mathbf{z}$ are 0. Thus it is not surprising that many kernel learning methods are formulated into SDP (e.g., [LCB$^+$02, HJL07]).

Some good solvers for SDP are available (e.g., [Stu99]), which often implements the classical interior point methods. The complexity of such methods could be as high as $O(n^{6.5})$. Researchers have proposed many other techniques, for example, cutting plane methods[HR00, KK06]. It is a active research topic in numerical optimization research. Due to the necessary positive definiteness of the kernel matrices, SDP arises frequently in kernel learning framework. However, the heavy computational cost makes it prohibitive for large scale applications. In chapter 4, we propose a closed-form solution which avoids the heavy computation of SDP solvers.

## 2.6 Kernel Learning

Classical kernel machines, for instance, support vector machines (SVM), view a kernel as mapping data points into a possibly very high dimensional space implicitly, and describe a kernel function as being good for a given learning problem if data is better separated by a large margin in that implicit space. However, while seems elegant, this theory does not directly correspond to one's intuition of a good kernel as a good similarity function (refer to the argument in [BBV08]). Furthermore, it may be difficult for a domain expert to use the theory to help design an appropriate kernel for the learning task at hand since the implicit mapping may not be easy to calculate. Therefore, there exists a gap between the design of kernels as a similarity functions and the underlying RKHS theories. Moreover, the positive definiteness is often difficult to justify.

Researchers have proposed various techniques to learn or construct a kernel. Due to the diversity of this topic, it is not easy to categorize all the works in a neat way. One possible method is to distinguish kernels by the ability whether they could handle unseen data. In the first case the goal is to learn a kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ defined on the whole space $\mathcal{X} \times \mathcal{X}$. In the other case, given both the $n$ (feature / label ) pairs of labeled and $u$ unlabeled test data $\mathbf{X} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n), \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+u}\}$, or just a known kernel matrix $\mathbf{K}$, or a similarity graph $\mathbf{G}$ defined on $\mathbf{X}$, or a set of side information $\mathbf{T}$ over $\mathbf{X}$, the goal is to learn a kernel $k : \mathbf{X} \times \mathbf{X} \mapsto \mathbb{R}$ which is defined only on the fixed set $\mathbf{X} \times \mathbf{X}$. In the former case, people need to devise new parametric-form kernel functions (for instance, [GFKS02, LSST$^+$02, Gär03, CKM07, GD07, SARK08]), which often involves some tricks (e.g.,convolutional kernel[Hau99, SK08]), or new methods of aggregating known kernels (for instance, multiple kernel learning method, [BLJ04, LCB$^+$04, SRSS06, RBCG08, XJKL08]). In the second case, one group of recent studies focuses on semi-supervised learning settings where the kernels are learned from mixtures of labeled and unlabeled data (example techniques include diffusion kernels [KL02], cluster kernels [CWS02], and gaussian random field technique [ZGL03]). These techniques often assume certain *parametric* forms of the target kernel functions and thus limit their capacity of fitting diverse patterns. Instead of assuming parametric forms for target kernels, an emerging group of kernel learning studies are devoted to *nonparametric* kernel learning [CSTEK01, LCB$^+$04, ZKGL04, KSD06, HJL07, HJ08]. One can see that in either case, the learning can be supervised or unsupervised.

## 2.6.1 Multiple Kernel Learning

The target of kernel learning is to make the learned kernel reveal the inherent similarity between instances drawn from some space $\mathcal{X}$. Thus "flexibility" is an important guideline of kernel learning. *Multiple kernel learning (MKL)*[LCB$^+$04], targeting at a linear combination of given kernels, is a representative work along this line. It often produces better result than single kernel. It also provides a more elegant method for hyper-parameter tuning by constructing base kernels with varying hyper-parameters.

### 2.6.1.1 Convex Multiple Kernel Learning

Depending on the selection of kernel class $\mathcal{K}$, we obtain different kernel learning algorithms. The most widely used case is the convex combination of a series of base kernels $\mathcal{K}_{base} = \{k_t : 1 \leq t \leq m\}$:

$$\mathcal{K} = \{k = \sum_{t=1}^{m} \theta_t k_t : d \geq 0, \boldsymbol{\theta}'\mathbf{1} = 1, \mathrm{tr}\,\mathbf{K}_t = \tau\} \tag{2.29}$$

here $\boldsymbol{\theta}$ is the weighing vector of candidate kernels, $\mathbf{K}_t$ is the kernel matrix of the $t$-th kernel function evaluated on the training data. A bunch of works have been done on how to solve $\mathbf{d}$ efficiently, for example, [SRSS06], [RBCG08] and [XJKL08].

Apparently, the richness of $\mathcal{K}$ provides more flexibility to fit the complex data. Meanwhile, it induces more risk of estimating the optimal classifier. Here our focus is theoretical guarantees, that is, we consider the risk bound of the solution $f_k^l$ of (2.46). Note, both $k$ and $f$ are optimization variables in learning, i.e., the risk is not only affected by the capacity of $\mathcal{H}_k$ for a fixed $k$, but also the richness of the candidate set of kernels $\mathcal{K}$. In the sequel of this section, we summarize some existing work on this topic. The theoretical foundations for supervised classification case are established in [MP05, AMP05].

Substituting the target kernel $k = \sum_{i=1}^{m} \theta_t k_t$ into the dual formulation of SVM, MKL problem is formulated into a saddle-point problem:

$$\min_{\boldsymbol{\theta}} \max_{\alpha} \quad \alpha^\top \mathbf{1} - \frac{1}{2}(\boldsymbol{\alpha} \circ \mathbf{y})^\top (\sum_{i=1}^{m} \theta_i \mathbf{K}_i)(\boldsymbol{\alpha} \circ \mathbf{y}) \tag{2.30}$$

$$\text{s.t.} \quad \boldsymbol{\alpha}^\top \mathbf{y} = 0, 0 \le \alpha_i \le C, i = 1, \dots, n \tag{2.31}$$

$$\boldsymbol{\theta}^\top \mathbf{1} = 1, 0 \le \theta_t \le 1, t = 1, \dots, m \tag{2.32}$$

This is min-max saddle point problem. It is convex on $\boldsymbol{\theta}$ and concave in $\alpha$. Therefore, the saddle-point is the unique global optimal.

The learned kernel is able to handle unseen data naturally due to the parametric form of the basic kernels. Most of the existing MKL research focus on the efficiency aspect, i.e., how to solve (2.29) efficiently. Many algorithms have been proposed to solve the presenting problem, for instance, SMO-similar method [BLJ04], SDP [LCB+04], QCQP [LBC+04] (multiclass case [ZO07]), iterative method with DC programming [AHMP06], SILP [SRS05, SRSS06], Subgradient Decent [RBCG07, RBCG08], level method [XJKL08]. Among these methods, the extended level algorithm proposed by Xu et. al. [XJKL08] has shown the state-of-the-art performance. It is quicker than SILP and SD method. The level method is from the family of bundle methods, which have recently been employed to efficiently solve regularized risk minimization problems.

The level method of MKL (MKL$^{\text{Level}}$) is an iterative approach designed for optimizing a non-smooth objective function. Let $J(\boldsymbol{\theta}, \boldsymbol{\alpha})$ denote the objective function of MKL. According to van Neuman Lemma, for the optimal solution $(\boldsymbol{\theta}^*, \boldsymbol{\alpha}^*)$, we have

$$J(\boldsymbol{\theta}, \boldsymbol{\alpha}^*) = \max_{\boldsymbol{\alpha}} J(\boldsymbol{\theta}, \boldsymbol{\alpha}) \ge J(\boldsymbol{\theta}^*, \boldsymbol{\alpha}^*) \ge J(\boldsymbol{\theta}^*, \boldsymbol{\alpha}) = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\alpha}).$$

The key idea of MKL$^{\text{Level}}$ is to iteratively update both the lower and the upper bound of $J(\boldsymbol{\theta}, \boldsymbol{\alpha})$[XJKL08] in order to find the saddle point $(\boldsymbol{\theta}^*, \boldsymbol{\alpha}^*)$.

MKL is also applied in the semi-supervised setting by Dai and Yeung [DY07]. The key problem in semi-supervised learning is how to make use of the unlabeled data to speed up the learning. The author makes use of the clustering assumption [DY07]. The unknown label $\hat{y}$ is also an optimization variable. Therefore, the convexity is violated. They used an annealing method to overcome this difficulty. MKL methods can also been extended to other problems, e.g., KFDA [FDBR04, KMB06], Regression [SRS05]. Besides the toy set, application of MKL on bioinformatics is reported in [LBC+04, OZ08].

In seek of flexibility, researchers have proposed a method by which each instance $\mathbf{x}$ has its own kernel weight $\theta(\mathbf{x})$. See [LJN06, GA08] for examples. The work on constructing Laplacian matrix can also be deemed as a MKL problem [AHP05]. It also has been shown that MKL can be applied in maximum margin clustering and exhibits very good performance [YFL09].

#### 2.6.1.2 $L_p$-norm Multiple Kernel Learning

The $L_1$ norm over the kernel weight $\boldsymbol{\theta}$ encourages the sparsity over the selected kernels. However, such sparse combination does not always produces good performance. It can even be worse than trivial uniform combination[Cor09], i.e., each kernel has the same weight in the target kernel. A natural try to solve this problem is to generalize $L_1$ norm to $L_p$ norm for $p > 1$:

$$\|\boldsymbol{\theta}\|_p = \sqrt[p]{\sum_{t=1}^{m} \theta_t^p}.$$

The development of $L_p$-norm MKL including algorithms and theories mainly attributes to [KBS+09, KRB10, MKZ11]. With the $L_p$-norm constraint over $\boldsymbol{\theta}$, we obtain the $L_p$-norm MKL formulation:

$$\begin{array}{ll} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} & \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top \sum_{t=1}^{m} \theta_t \mathbf{Q}_t \boldsymbol{\alpha} \\ \text{s.t.} & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \mathbf{y}^\top \boldsymbol{\alpha} = 0, \boldsymbol{\theta} \geq 0, \|\boldsymbol{\theta}\|_p^p \leq 1, \end{array}$$

where $\mathbf{Q}_t = \text{diag}(\mathbf{y})\mathbf{K}_t\text{diag}(\mathbf{y})$.

The above problem can be solved by adopting the idea of *semi-infinite programming*[SRSS06]. We first re-write the problem in the form

$$\min_{\eta} \quad \eta \text{ s.t. } \eta \geq \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^\top \sum_{t=1}^{m} \theta_t \mathbf{Q}_t \boldsymbol{\alpha} \tag{2.33}$$

With a fixed $\eta$, we solve $\boldsymbol{\alpha}$ to find the most violated constraint by SVM solver. Then minimize the objective w.r.t. $\eta$ and $\boldsymbol{\theta}$. However, it is not easy as the $L_p$-norm induces non-linearity and unlikely can be solved by off-the-shelf toolkits which often only contains solvers for linear or quadratic problem. Here we approximate the $L_p$-norm by two-order Taylor expansion[KBS+09]:

$$\|\boldsymbol{\theta}\|_p^p \approx 1 + \frac{p(p-3)}{2} - \sum_{t=1}^{m} p(p-2)(\tilde{\theta}_t)^{p-1}\theta_t + \frac{p(p-1)}{2} \sum_{t=1}^{m} \tilde{(\theta)}_t^{p-2}\theta_t^2,$$

where the exponential $\boldsymbol{\theta}^p$ is defined element-wise, i.e., $\boldsymbol{\theta} := [\theta_1^p, \ldots, \theta_m^p]^\top$. After initializing $\|\boldsymbol{\theta}(0)\|_p^p = 1$ with uniform weight, the successive $\boldsymbol{\theta}(k+1)$ at the $k+1$ step is computed using $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}(k+1)$.

The experiments according to [KBS+09] show $L - p$-norm regularization over $\boldsymbol{\theta}$ provides more flexibility than traditional $L_1$-norm regularization and often yields better performance.

#### 2.6.1.3 Polynomial Multiple Kernel Learning

Following the line of exploring kernel class $\mathcal{K}$, Cortes et. al. studied *polynomial multiple kernel learning* (PMKL) beyond the traditional linear kernel combination in MKL. Given the base kernels $\mathcal{K}_{base}$, the kernel class of PMKL is defined as

$$\mathcal{K}_{poly} = \{\mathbf{K} = \sum_{0 \sum_{t=1}^{m} p_t \leq d, p_t \geq 0, t \in [0,m]} \theta_{\mathbf{p}} \mathbf{K}_1^{p_1} \dots \mathbf{K}_m^{p_m} \ : \ \theta \geq 0\}.$$

According to [CCR09], the kernel of this form is *positive definite and symmetric (PDS)*, which implies the exponential operation is entry-wise. However, the number of base number of kernel is in order of $\mathcal{O}(m^d)$, which can be too large for estimating the weight $\boldsymbol{\theta}$. For practical purpose, Cortes et. al. choose the case $d = 2$, where $\mathbf{K}$ comes to a simpler expression:

$$\mathbf{K} = \sum_{a,b=1}^{m} \theta_a \theta_b \mathbf{K}_a \circ \mathbf{K}_b.$$

Instead of minimizing the SVM-based objective function, the authors choose to minimize square loss to learn the kernel weight

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{y}^\top (\mathbf{K}_{\boldsymbol{\theta}} + \lambda \mathbf{I})^{-1} \mathbf{y},$$

which in turn can be solved by *projection-based gradient descent*, i.e, the $\boldsymbol{\theta}$ at each step is projected to satisfy the constraints.

The empirical performance of the above polynomial MKL is comparable with previous results.

#### 2.6.1.4 More Generalized Multiple Kernel Learning

The $L_p$-norm MKL and PMKL inspire that we can consider the problem in a more general manner. Suppose the target kernel $\mathbf{K}_{\boldsymbol{\theta}}$ is parameterized by a parameter vector $\boldsymbol{\theta}$ and a set of base kernels $\mathcal{K}_{base}$, the SVM-based kernel learning framework can be expressed as[VB09]

$$\min_{\boldsymbol{\theta}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{Y} \mathbf{K}_{\boldsymbol{\theta}} \mathbf{Y} \boldsymbol{\alpha} + \Omega(\boldsymbol{\theta}). \tag{2.34}$$

Note here the combination can be in any form, including non-convex function over $\boldsymbol{\theta}$. While the candidate kernel class $\mathcal{K} = \{\mathbf{K}_{\boldsymbol{\theta}}\}$ introduces more flexibility than single kernel, it also incurs the higher risk of overfitting. The last term $\Omega(\boldsymbol{\theta})$ is for regularization purpose.

Eqn (2.34) is in form of Tikhonov regularization and MKL with hard constraint on $\boldsymbol{\theta}$ is in form of Ivanov regularization. Thus, the aforementioned various MKL algorithms are special cases of (2.34) as the Ivanov and Tikhonov regularization can be shown equivalent to each other[MKZ11]. Varma and Bahu use entry-wise product between Gaussian kernels with varying band-width parameter. In some UCI datasets, it achieves better performance than MKL.

### 2.6.2 Nonparametric Kernel Learning

In classical methods, we have a closed-form kernel function $k$. The similarity between any pair of instances $(\mathbf{x}, \mathbf{x}')$ can be evaluated by $k(\mathbf{x}, \mathbf{x}')$. This parametric form limits their capacity of fitting diverse patterns. It is guaranteed that any positive definite matrix corresponds to a valid kernel. Therefore, in the transductive setting, one can just focus on learning a p.s.d. matrix.

Such nonparametric kernels usually make use of information like: (1) the marginal distribution of the data, or prior parametric kernel. For example, Gaussian kernel is widely used for many kinds of data. One can rectify the kernel matrix evaluated from Gaussian to fit the data, for example, the possible manifolds of data; (2) side information. Empirical observations about the data can be provided by domain experts. Such information may not be directly useful for building a classifier, but can provide hints to constrain the kernel.

An important technique is the kernel target alignment criterion proposed in [CSTEK01], which guides the kernel learning task to optimize the kernel by maximizing the alignment between the training data examples and the class labels of the training examples:

$$\max_{\mathbf{K}\succeq\mathbf{0}} \quad \frac{\langle \mathbf{K}_{N_l}, \mathbf{T}\rangle}{\sqrt{\langle \mathbf{K}_{N_l}, \mathbf{K}_{N_l}\rangle \langle \mathbf{T}, \mathbf{T}\rangle}}, \tag{2.35}$$

where $\mathbf{T} = \mathbf{y}\mathbf{y}'$ is the outer product of labels, $\mathbf{K}_{N_l}$ is the sub-matrix of which the entry values are kernel evaluation on $N_l$ labeled data examples. Note that $\mathbf{T}$ could be obtained by empirical experiments and more general than class labels. The objective (2.35) only involves the labeled data.

A popular assumption is to treat $\mathbf{K}$ to be spanned by the eigen-vectors of some known kernel defined over both the labeled and unlabeled data [CWS02, ZKGL04, HLC06, ZA05, JZ08]: $\mathcal{K} = \{\sum_i \lambda_i \mathbf{v}\mathbf{v}' : \lambda_i \geq 0\}$. Thus the optimization variables are reduced from the entire kernel matrix $\mathbf{K}$ to the kernel spectrum $\boldsymbol{\lambda}$. More generally, we construct the target kernel as $\mathbf{K} = \sum_{i=1}^{n} \psi(\lambda_i)\mathbf{v}_i\mathbf{v}_i^\top$. [CWS02][HLC06] proposed some examples. For example, we could set $\psi(\lambda) = 1$, or a polynomial function $\psi(\lambda) = \lambda^t$. Zhu et. al. enforces $\psi$ to be a decreasing function[ZKGL04]. The resultant optimization problems is solved by QCQP. Though empirical improvements are obtained, the theoretical analysis is developed by [ZA05, JZ08].

Another scheme is to minimize the distance between the target $\mathbf{K}$ and the prior kernel $\mathbf{K}_0$ and enforce some side constraints or heuristics. A recent approach, called *indefinite kernel learning* [Ld07, CY08, YYG09], extends the MKL formulation to learn non-parametric kernels from an indefinite kernel $\mathbf{K}_0$, which does not assume the convex hull assumption. The indefinite kernel learning rewrites the objective function of MKL as follows:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \max_{\boldsymbol{\alpha}} \quad \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\langle(\boldsymbol{\alpha}\circ\mathbf{y})(\boldsymbol{\alpha}\circ\mathbf{y})', \mathbf{K}\rangle + \gamma\|\mathbf{K} - \mathbf{K}_0\|_F^2, \tag{2.36}$$

where $\mathbf{K}_0$ is an initial kernel, which could be indefinite. Here the Euclidean distance is adopted to regularize the target kernel.

Kulis et. al. used Bregman divergences to measure the distance between kernels [KSD06]:

$$\begin{aligned} \min\nolimits_{\mathbf{K}\succeq\mathbf{0}} \quad & D_\phi(\mathbf{K},\mathbf{K}_0) && (2.37)\\ \text{s.t.} \quad & \text{tr }(\mathbf{K}\mathbf{A}_i) \le b_i, 1 \le i \le c, && (2.38)\\ & \text{rank}(\mathbf{K}) \le r, && (2.39)\end{aligned}$$

where each $\mathbf{A}$ matrix enforces a linear constraint over $\mathbf{K}$. They restrict the rank of $\mathbf{K}$ explicitly by (2.39). Thus the learned kernel may speed up training by a low-rank representation of the $\mathbf{K}$.

Hoi et. al. constructed the graph Laplacian $\mathbf{L}$ first and minimized tr $\mathbf{LK}$ [HJL07]. They also made use of pair-wise constraints to fit the data better:

$$\min\nolimits_{\mathbf{K}\succeq\mathbf{0}} \quad \text{tr } \mathbf{LK} + C\sum_{ij}\max(0, 1 - A_{ij}K_{ij}) \tag{2.40}$$

here $\mathbf{A}$ is a constraint matrix taking values in $\{\pm 1\}$. If two points $(\mathbf{x}_i, \mathbf{x}_j)$ have the same label, $\mathbf{A}_{ij} = 1$; if they cannot have the same label, $\mathbf{A}_{ij} = -1$. Such side information is more general than class label. In real application, the label is often difficult to obtain, while the pairwise constraints can be observed by some experiments. Hoi evaluate this method by clustering. Active learning in this framework was proposed later [HJ08].

Tsuda et. al. proposed method by maximizing the von Neumann entropy[TN04]:

$$\begin{aligned} \min\nolimits_{\mathbf{K}\succ\mathbf{0}} \quad & \text{tr }(\mathbf{K}\log\mathbf{K}) && (2.41)\\ & \text{tr }\mathbf{KA} \le b_i, i = 1, \ldots, c && (2.42)\end{aligned}$$

Roughly speaking, maximizing entropy amounts to placing the samples in the feature space as evenly as possible. In biological networks, this heuristic was shown to be useful.

Very recently, Li et. al. formulated the kernel learning into a more general framework:

$$\min\nolimits_{\mathbf{K}\succeq\mathbf{0}} \quad \frac{\gamma}{c}\sum_{i=1}^{c} L(\text{tr }\mathbf{KA}_i) + D_\phi(\mathbf{K},\mathbf{K}_0) \tag{2.43}$$

here $L()$ is any convex loss function. Instead of using general SDP solver, they used an inexact Newton method to speed up efficiency.

Methods belonging to this category are still under explorations. First, the learning of the kernel and the classifier are isolated. Thus the resultant kernel may not be optimized for classification. We need theoretical analysis to connect $\mathbf{K}$ and the classification performance. Moreover, it is imperative to develop quicker solvers to make NPK practical for real applications.

The optimization over the above learning objective function (2.35) or (2.37) will simply return the trivial solution $\mathbf{K}_0$ without additional constraints, which would make NPKL meaningless. In practice, some prior knowledge about the target kernel will be added to constrain the solution space $\mathcal{K}$. Most of the existing constraints over the entries of $\mathbf{K}$ could be expressed by $\mathrm{tr}\,\mathbf{KT} \leq b$. For example, as discussed in [KT03], the square of distance between two data examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space can be expressed by $\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|_2^2 = K_{ii} + K_{jj} - 2K_{ij} = \mathrm{tr}\,\mathbf{KT}_{ij}$, where $\mathbf{T}_{ij}$ is a matrix of $N \times N$ only taking non-zeros at $T_{ii} = T_{jj} = 1, T_{ij} = T_{ji} = -1$. Moreover, one can introduce slack variables for soft constraints.

Besides, some regularization terms over kernel $\mathbf{K}$ are often included during the optimization phase. For example, fixing the trace $\mathrm{tr}\,\mathbf{K} = 1$ is rather common in SDP solvers.

At last, we summarize the typical schemes of existing NPKL methods:

- To encourage the similarity (e.g., kernel target alignment) or penalize the distance (e.g., Bregman divergence) to some prior similarity information;

- To enforce some constraints to the kernel $\mathbf{K}$ with prior heuristics, such as distance constraint $K_{ii} + K_{jj} - 2K_{ij} = d_{ij}^2$, or side information, etc; and

- To include regularization terms over $\mathbf{K}$ to control capacity, such as $\mathrm{tr}\,\mathbf{K} = 1$.

By the above steps, NPKL provides a flexible scheme to incorporate more prior information into the target kernel. Due to the non-parametric nature, the solution space $\mathcal{K}$ is capable of fitting diverse empirical data such that the learned kernel $\mathbf{K}$ can be more effective and powerful to achieve better empirical performance than traditional parametric kernel functions.

Despite the powerful capacity achieved by NPKL, one key challenge with NPKL is the difficulty of the resulting optimization problem, in which

- the whole gram matrix $\mathbf{K}$ is treated as the optimization variable, that is, $O(N^2)$ variables;

- the kernel matrix $\mathbf{K}$ must be positive semi-definite.

As a result, NPKL is often turned into a Semi-Definite Programming (SDP) problem. For instance, a NPKL problem to learn a kernel matrix $\mathbf{K}$ with $m$ linear constraints is written as follows:

$$\max_{\mathbf{K} \succeq \mathbf{0}} \mathrm{tr}\,\mathbf{CK} \quad : \quad \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K} = b_{ij}, \tag{2.44}$$

where $\mathbf{C}$ and $\mathbf{T}_{ij}$'s are $N \times N$ symmetric matrices and $b_{ij}$'s are scalars, and its dual problem can be rewritten as:

$$\min_{\mathbf{y}} \mathbf{b}'\mathbf{y} \quad : \quad \mathbf{C} - \sum_{(i,j)} \mathbf{T}_{ij} y_{ij} \succeq \mathbf{0}, \tag{2.45}$$

where $\mathbf{y}$ is the vector of dual variables $y_{ij}$'s for the linear constraints in (2.44) and $\mathbf{b}$ is the vector of $b_{ij}$'s.

Typically, this SDP problem of NPKL is usually solved by applying a general purpose SDP solver. Among various SDP solvers, the *interior-point algorithm* is one of the state-of-the-art solutions [BV04]. From [LVBL98], the time complexity per iteration of the SDP problem (2.45) is $O(m^2N^2)$. Using the primal-dual method for solving this SDP, the accuracy of a given solution can be improved by an absolute constant factor in $O(\sqrt{N})$ iterations [NN94]. When $m$ approaches to $O(N^2)$, the overall computation complexity is often as high as $O(N^{6.5})$, which makes NPKL inapplicable to real applications.

### 2.6.3 Generalization Bounds of Multiple Kernel Learning

In last section we introduce the learning in RKHS. For this kind of learning scheme, the kernel function plays a central role for the performance. However, it is nontrivial to select a good kernel. An inappropriate kernel could result in suboptimal or bad performance. Recently, the idea of learning / selecting a kernel from given data has attracted much attention in machine learning research.

In the problem of *kernel learning*, we learn the classifier $f \in \mathcal{H}_K$ and the associate kernel $K$ simultaneously. Thus the learning is cast as a two-layer minimization problem:

$$\min_{k\in\mathcal{K}} \min_{f\in\mathcal{H}_K} \quad L_n^l(f) + \lambda\|f\|_{\mathcal{H}_k} \tag{2.46}$$

Typically, the optimization domain $\mathcal{K}$ is defined over a set of basic kernels $\mathcal{K}_{base} = \{k_1, \ldots, k_m\}$. Throughout this section, we use $\mathcal{F}_{\mathcal{K}}$ to denote the function class with bounded norm in Hilbert space depending the kernel class $\mathcal{K}$:

$$\mathcal{F}_{\mathcal{K}} := \{\sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot) : \|f\|_{\mathcal{H}_{\mathcal{K}}} \leq B, k \in \mathcal{K}\} \tag{2.47}$$

#### 2.6.3.1 Transductive Bounds

The kernel learning is initiated by [LCB+02]. By assuming the size of test set equals to the size of training set in the transductive setting, the risk on test data is bounded via the Rademacher complexity defined on the training data and test data [LCB+02, LCB+04, BH02]. Later [SBD06] proved the above bounds are vacuous. Recently [EYP07] proposed a notion of *transductive Rademacher complexity*, by which the restriction over the number of test points can be removed and a tighter bound is derived.

The focus of this chapter is the generalization ability of kernel learning schemes. We do not expose the details on transductive setting.

#### 2.6.3.2 Risk Bounds with Rademacher Complexity

Recall Remark (2.3), a natural choice is to compute the risk bound using Rademacher complexity involving kernels. With the results shown Theorem 2.8 and 2.10, we follow Definition 2.4 to compute the Rademacher complexity of $\mathcal{F}$ defined by (2.47).

**Theorem 2.14** *[BH02, Theorem 2]Let $\mathbf{K}_1, \ldots, \mathbf{K}_m$ be some fixed kernel matrices and $\mathcal{K}$ as defined by (2.29) then*

$$\hat{R}(\mathcal{F}_{\mathcal{K}}) \leq 2B \max_{i=1,\ldots,m} \sqrt{\|\mathbf{K}_i\|/n} \tag{2.48}$$

*here $\|\mathbf{K}\| = \sup_{\|\mathbf{v}\|_2 \leq 1} \mathbf{v}'\mathbf{K}\mathbf{v} = \max \lambda(\mathbf{K})$ is the induced norm.*

We can obtain the risk bound with Rademacher complexity by plugging (2.48) into Theorem 2.8. Unfortunately, the following theorem shows that simply applying Rademacher complexity is meaningless [SBD06].

**Theorem 2.15** *For the multiple kernel combination defined in (2.29), the learned classifier by solving (2.24) with Hinge loss results in a risk bound always greater than 1 when applying Rademacher complexity in Theorem 2.10.*

**Remark 2.16:** $\square$ When we take the kernel class $\mathcal{K}$ into optimization, Rademacher complexity can fail to give meaningful risk bound with a candidate kernel class $\mathcal{K}$, which implies we need to define better complexity measure that takes $\mathcal{K}$ into account.

Very recently, Cortes proved a tighter bound with Rademacher complexity[CMR10a], which shows the complexity increases with log factor over the number of kernels $m$.

**Theorem 2.17** *Let $m > 1$ and assume that $k_m(\mathbf{x}, \mathbf{x}) \leq \kappa^2$ for all $x \in \mathcal{X}$. For any sample of size $n$, the Rademacher complexity of the hypothesis class $\mathcal{F}_{\mathcal{K}}$ can be bounded as follows (where $c := 23/22$ and $\lceil \cdot \rceil$ rounds to the next largest integer:*

$$R(\mathcal{F}_{\mathcal{K}}) \leq \sqrt{\frac{ce\lceil \log m \rceil kappa^2}{n}}$$

#### 2.6.3.3 Risk Bounds with Covering Number and Pseudo-Dimension

**Definition 2.7** *A subset $\tilde{A} \subset A$ is an $\epsilon$-net of $A$ under the metric $d$ if for any $a \in A$ there exist $\tilde{a} \in \tilde{A}$ with $d(a, \tilde{a}) \leq \epsilon$. The* **covering number** *$\mathcal{N}(A, \epsilon, d)$ is the size of the smallest $\epsilon$-net of $A$. The* **uniform $l_\infty$ covering number** *$\mathcal{N}_n(\mathcal{F}, \epsilon, d_\infty^{\mathbf{X}^n})$ of a predictor class $\mathcal{F}$ is given by considering all possible samples $\mathbf{X}^n$ of size $n$:*

$$\mathcal{N}_n(\mathcal{F}, \epsilon, , d_\infty^{\mathbf{X}^n}) = \sup_{\mathbf{X}^n} \mathcal{N}(\mathcal{F}, \epsilon, , d_\infty^{\mathbf{X}^n}),$$

*where the metric is*

$$d_\infty^{\mathbf{X}^n}(f_1, f_2) = \max_{\mathbf{x}_i \in \mathbf{X}^n} \left| f_1(\mathbf{x}_i) - f_2(\mathbf{x}_i) \right|.$$

**Definition 2.8** *Let $\mathcal{K}$ be a set of kernel functions mapping from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$. We say that $S_{n^2} = \{(\mathbf{x}_i, \mathbf{x}'_i) \in \mathcal{X} \times \mathcal{X}\}$ is pseudo-shattered by $\mathcal{K}$ if there are real numbers $\{r_i \in \mathbb{R}\}$ such that for any $b \in \{-1, 1\}^{n^2}$ there is a function $\mathbf{K} \in \mathcal{K}$ with property $sgn(k(\mathbf{x}_i, \mathbf{x}'_i) - r_i) = b_i$ for any $i \leq n^2$. Then, we define a* **pseudo-dimension** *$d_{\mathcal{K}}$ of $\mathcal{K}$ to be the maximum cardinality of $S_{n^2}$ that is pseudo-shattered by $\mathcal{K}$.*

With the notion *pseudo-dimension*, we have

**Theorem 2.18** *[SBD06] With probability at least $1 - \delta$,*

$$L(f) \leq L_n(f) + \sqrt{\tilde{O}(d_{\mathcal{K}} + 1/\gamma^2)/n},$$

*here $\gamma$ is the margin which relates to the loss function $\phi(t) = \max(\gamma - t, 0)$.*

The complete proof can be found in [SBD06]. Instead of the technical details, we sketch the attack plans: 1) use an $\epsilon$-net of *kernels* to construct an $\epsilon$-net of *classifiers* with respect to the kernels, where the key observation is that kernels that are close as real-valued functions also yield similar classes of classifiers; 2) apply standard results bounding the estimation error in terms of covering numbers; 3) bound the covering number in terms of pseudo-dimension; 4) instantiate the pseudo-dimension for the specific kernel class (2.29).

In the first tip, for any two kernel functions $k$ and $\tilde{k}$, for any predictor $f \in \mathcal{F}_k$, there exists a predictor $\tilde{f} \in \mathcal{F}_{\tilde{k}}$ satisfy

$$d_{\infty}^{\mathbf{X}^n}(f, \tilde{f}) \leq \sqrt{n d_{\infty}^{\mathbf{X}^n}(k, \tilde{k})} := \sqrt{n \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}^n} |k(\mathbf{x}_i, \mathbf{x}_j) - \tilde{k}(\mathbf{x}_i, \mathbf{x}_j)|}$$

The above inequality implies the distance between two functions contained in some RKHS can actually be bounded by the distance between their underlying kernel functions. Consequently, the covering number $\mathcal{N}_{\mathcal{F}_{\mathcal{K}}}$ between function classes can be bounded by the covering number between kernel classes $\mathcal{N}_{\mathcal{K}}$:

$$\mathcal{N}_n(\mathcal{F}_{\mathcal{K}}, \epsilon) \leq 2\mathcal{N}_n\left(\mathcal{K}, \frac{\epsilon^2}{4n}\right) \cdot \left(\frac{16n\kappa}{\epsilon^2}\right)^{\frac{64\kappa}{\epsilon^2}\log\left(\frac{\epsilon e n}{8\sqrt{\kappa}}\right)},$$

where $\kappa = \max_{\mathbf{x}_i \in \mathbf{X}^n} k(\mathbf{x}_i, \mathbf{x}_i)$.

For the second step, we have the following result bounding estimation error with uniform $l_\infty$ covering number [AB99]: for a function class $\mathcal{F}$ and fixed $\gamma > 0$ in hinge loss, with probability at least $1 - \delta$

$$L(f) \leq L_n(f) + \sqrt{8\frac{1 + \log \mathcal{N}_{2n}(\mathcal{F}, \gamma/2) - \log \delta}{n}}$$

Therefore, all we need to do is to bound the covering number for the function class under consideration.

In the third step, for any kernel family $\mathcal{K}$ bounded by $\kappa$ with pseudo-dimension $d_{\mathcal{K}}$ we have the result

$$\mathcal{N}_n(\mathcal{K}, \epsilon) \leq \left(\frac{en^2\kappa}{\epsilon d_{\mathcal{K}}}\right)^{d_{\mathcal{K}}}$$

Finally, for the kernel class defined in (2.29), we bound its pseudo-dimension by Theorem 2.19.

For the kernel family $\mathcal{K}$ defined in (2.29), we have

**Theorem 2.19** *For the kernel class $\mathcal{K}$ defined by (2.29), the pseudo-dimension is bounded by*

$$d_{\mathcal{K}} \leq m.$$

**Remark 2.20:** □ We conduct two key observations:

- Pseudo-dimension relates the properties of candidate kernel class $\mathcal{K}$ to the risk bound. This verifies the conjecture that we should characterize the richness of $\mathcal{K}$ to compute the risk bound. Including pseudo-dimension of $\mathcal{K}$ makes a better capacity measure than Rademacher complexity on $\mathcal{F}_{\mathcal{K}}$.
- Pseudo-dimension can be computed for convex kernel combination. However, it could be difficult to derive in general.

#### 2.6.3.4 Risk Bounds with Rademacher Chaos Complexity

Recall that we introduce random label (the Rademacher variables) to measure to which extent a classifier $f$ fits noise data, which leads to the definition of Rademacher complexity. Similarly, one can introduce random kernel evaluation results to test the capacity of kernel functions. This motivates the *Rademacher chaos complexity.*

**Definition 2.9** *[YC09] Given a sample $\mathbf{X}^n$, the homogeneous Rademacher chaos process of order two, with respect to the Rademacher variable $\sigma$, is a random variable system defined by*

$$\{\hat{U}_k(\mathbf{X}^n) = \frac{1}{n}\sum_{i<j} \sigma_i\sigma_j k(\mathbf{x}_i, \mathbf{x}_j) : k \in \mathcal{K}\},$$

*and we refer to the expectation of its superma*

$$\hat{U}_n(\mathcal{K}) = \mathbb{E}_\sigma[\sup_{k\in\mathcal{K}} |\hat{U}_k(\sigma)|]$$

*as the empirical* **Rademacher chaos complexity** *over $\mathcal{K}$.*

With $\hat{\mathcal{U}_n(\mathcal{K})}$, we can bound the $l$-risk:

**Theorem 2.21** *[YC09] With probability at least $1-\delta$,*

$$L^l \leq L^l_n + 4C_l B\left(\frac{2\hat{U}_n(\mathcal{K})}{n}\right)^{1/2} + 4\kappa C_l B\left(\frac{2}{n}\right)^{1/2} + 3D_l\left(\frac{\ln 1/\delta}{2n}\right)^{1/2} + \frac{2}{\sqrt{n}},$$

*where $D_l = \sup\{|l(t)| : |t| \leq \kappa B\}$, $C_l = \sup\{|l(t) - l(t')|/|t-t'| : \forall |t|, |t'| \leq \kappa B\}$, $B = \sqrt{1/\lambda}$.*

Rademacher chaos complexity can be bounded by covering number. To show this, we first define the empirical metric w.r.t. the sample $\mathbf{X}^n$ for $\mathcal{K}$:

$$d_2^{\mathbf{X}^n}(K_1, K_2) = \left(\frac{1}{n^2}\sum_{i<j}^{n}(K_{1ij} - K_{2ij})^2\right)^{1/2} \tag{2.49}$$

Be ware of the difference with the metric in Definition 2.7. Then we have

**Theorem 2.22** *There exist an absolute constant $C$ such that, for any $\mathbf{X}^n$, there holds*

$$\hat{\mathcal{U}}_n(\mathcal{K}) \leq C\int_0^\infty \log \mathcal{N}(\mathcal{K}\cup\{\mathbf{0}\}, \epsilon, d_2^{\mathbf{X}^n})d\epsilon. \tag{2.50}$$

Therefore, Rademacher chaos complexity can be further bounded in terms of pseudo-dimension by the following theorem:

**Theorem 2.23** *There exist a universal constant $C$ such that for any sample $\mathbf{X}^n$, there holds*

$$\hat{\mathcal{U}}_n(\mathcal{K}) \leq C(1+\kappa)^2 d_{\mathcal{K}} \ln(2en^2) \tag{2.51}$$

This provides us two choices for bounding the risk: 1) Compute $d_{\mathcal{K}}$. Substituting $d_{\mathcal{K}}$ into either Theorem 2.18 or Theorem 2.21 with (2.51) yields generalization risk; 2) when $d_{\mathcal{K}}$ is difficult to compute, one can estimate $\hat{\mathcal{U}}$ directly.

**Theorem 2.24** *For the kernel classes given by (2.29) with a sample $\mathbf{X}^n$, its Rademacher chaos complexity is bounded by*

$$\hat{\mathcal{U}}_n(\mathcal{K}) \leq C\kappa^2 \ln(m+1).$$

**Remark 2.25:** □ The Rademacher chaos complexity leads to bound related to the kernel class $\mathcal{K}$.

- Rademacher complexity yields a bound over the trace or induced norm of $K$, which is shown to be vacuous by [SBD06]. Rademacher chaos complexity avoids this problem by measuring the capacity of $K$ through its ability of fitting random similarity between pairs of points.
- The main result of Theorem 2.21 is meaningful only if we can estimate $\hat{\mathcal{U}}_n$ directly. If the only way to use it is (2.51), we need NOT define $\hat{\mathcal{U}}_n$.

# Chapter 3

# Simple Non-Parametric Kernel Learning Algorithms

As introduced in chapter 2, to address the problem of kernel learning, a bunch of research on learning effective kernels from data automatically has been actively explored recently. An example technique is *Multiple Kernel Learning* (MKL) [LCB$^+$04, BLJ04], which aims at learning a convex combination of several predefined parametric kernels in order to identify a good target kernel for the applications. MKL has been actively studied in many applications, including bio-informatics [SRS05, SRSS06], computer vision [DTXM09, SWY$^+$09, VGVZ09], and natural language processing [MT11a], etc. Despite some encouraging results reported, these techniques often assume the target kernel function is of some *parametric* forms, which limits their capacity of fitting diverse patterns in real complex applications.

Instead of assuming some parametric forms for the target kernel, an emerging group of kernel learning studies are devoted to *Non-Parametric Kernel Learning* (NPKL) methods, which aim to learn a Positive Semi-Definite (PSD) kernel matrix directly from the data. Example techniques include [CSTEK01], [LCB$^+$04], [ZKGL04], [ZA05], [KSD06], [HJL07], [BKD09] and [FLW09, MT11b]. NPKL provides a flexible learning scheme of incorporating prior/side information into the known similarity measures such that the learned kernel can exhibit better ability to characterize the data similarity. However, due to the PSD constraint, the resulting optimization task of NPKL is often in the form of Semi-Definite Programing (SDP). Many existing studies have simply solved such SDP problems by some general purpose SDP solvers, which often have the time complexity of $O(N^{6.5})$, making the NPKL solution infeasible to real world large-scale applications.

In this chapter, we aim at addressing the efficiency and scalability issues related to the NPKL techniques proposed by [HJL07] and [ZTH09], which have shown the state-of-the-art empirical performance in several applications [ZH10]. In particular, the main contributions of this chapter include:

(i) We propose a family of Simple Non-Parametric Kernel Learning (SimpleNPKL) algorithms for efficient and scalable non-parametric kernel learning.

(ii) We present the first SimpleNPKL algorithm with linear loss function to learn non-parametric kernels from pairwise constraints. The algorithm enjoys a *closed-form* solution that can be computed efficiently by sparse eigen-decomposition techniques, for example, the *Lanczos* algorithm.

(iii) To achieve more robust performance, we propose the second SimpleNPKL algorithm that has other loss functions (including square hinge loss, hinge loss and square loss), which can be re-formulated as a *mini-max optimization* problem. This optimization can be solved by an efficient iterative projection algorithm that mainly involves the computation of sparse eigen decomposition.

(iv) To further speed up the SimpleNPKL algorithm of other loss functions, we investigate some active constraint selection techniques to reduce the computation cost at each iteration step.

(v) We conducted extensive experiments, which show that SimpleNPKL is significantly more efficient than existing NPKL methods. With the same linear loss function, SimpleNPKL is on average 40 times faster than the original NPKL using a standard SDP solver. This makes the NPK learning techniques practical to large-scale applications.

(vi) We extend the proposed SimpleNPKL scheme to resolve other non-parametric kernel learning problems, including *colored maximum variance unfolding* [SSBG07], *minimum volume embedding* [SJ07], and *structure preserving embedding*[SJ09]. The encouraging results show that our technique is able to speed up the existing non-parametric kernel learning solutions significantly for several real-world applications.

The rest of this chapter is organized as follows. Section 3.1 introduces a framework of Non-parametric Kernel Learning (NPKL) from pairwise constraints proposed by [HJL07]. Section 3.2 describes our proposed SimpleNPKL algorithms, which aim to resolve the NPKL task efficiently. Section 3.3 discusses some implementation issues for developing a fast solver in practice. Section 3.4 extends our technique to speed up other kernel learning methods. Section 3.5 gives our empirical results and Section 3.6 concludes this work.

## 3.1 Non-Parametric Kernel Learning from Pairwise Constraints

In this Section, we introduce the framework of Non-Parametric Kernel Learning (NPKL) [HJL07, ZTH09], which aims to learn non-parametric kernels from side information, which is presented in a collection of must-link and cannot-link pairs.

### 3.1.1 Side / Label Information

Let $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ denote the entire data collection, where each data point $\mathbf{x}_i \in \mathcal{X}$. Consider a set of $N_l$ labeled data examples, $\mathcal{L} = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_{N_l}, y_{N_l})\}$, one can use $y_i y_j$ as the similarity measurement for any two patterns $\mathbf{x}_i$ and $\mathbf{x}_j$. Sometimes, it is possible that the class label information is not readily available, while it is easier to obtain a collection of similar (positive) pairwise constraints $\mathcal{S}$ (known as "must-links", that is, the data pairs share the same class) and a collection of dissimilar (negative) pairwise constraints $\mathcal{D}$ (known as "cannot-links", that is, the data pairs have different classes). These pairwise constraints are often referred to as *side information*.

In general, kernel learning with labeled data can be viewed as a special case of kernel learning with side information [KT03, KSD06, HJL07], that is, one can construct the sets of pairwise constraints $\mathcal{S}$ and $\mathcal{D}$ from $\mathcal{L}$. In real applications, it is often easier to detect pairwise constraint while the class label is difficult to obtain. For example, in bioinformatics, the interaction between two proteins can be identified by empirical experiments. These interactions are expressed naturally by pairwise constraints. However, it could be very difficult to judge the protein function, which corresponds to class labels. In the sequel, we focus on learning kernels from pairwise constraints.

Given $\mathcal{S}$ and $\mathcal{D}$, we construct a similarity matrix $\mathbf{T} \in \mathbb{R}^{N\times N}$ to represent the pairwise constraints, that is,

$$T_{ij} = \begin{cases} +1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ -1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

A straightforward and intuitive principle for kernel learning is that the kernel entry $K_{ij}$ should be aligned with the side information $T_{ij}$ as much as possible [CSTEK01], that is, the alignment $T_{ij}K_{ij}$ of each kernel entry is maximized.

### 3.1.2 Locality Preserving Regularization

In addition to side/label information, preserving the intrinsic geometric structure of the data have also been explored to improve the performance of kernel learning. Typically, most existing kernel learning approaches [KSD06, HJL07, HJ08] adopt the low dimensional data manifold [SNB05] for preserving the locality of the data in kernel learning. The following reviews an approach for exploring low dimensional data manifold in kernel learning [HJL07].

Let us denote by $f(\mathbf{x}, \mathbf{x}')$ a similarity function that measures the similarity between any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathbf{S} \in \mathbb{R}^{N\times N}$ is a similarity matrix where each element $S_{ij} = f(\mathbf{x}_i, \mathbf{x}_j) \geq 0$. Note that $f(\cdot,\cdot)$ does not have to be a kernel function that satisfies the Mercer's condition. For a given $N$ data examples, a kernel matrix $\mathbf{K}$ can be expressed as $\mathbf{K} = \mathbf{V}'\mathbf{V}\succeq \mathbf{0}$, where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$ is the matrix of the embedding of the $N$ data examples. The regularizer of the kernel matrix $\mathbf{K}$, which captures the local dependency between the embedding of $\mathbf{v}_i$ and $\mathbf{v}_j$ (i.e., the low dimensional embedding of similar data examples should be similar w.r.t. the similarity $S_{ij}$), can be defined as:

$$\begin{aligned}\Omega(\mathbf{V}, \mathbf{S}) &= \frac{1}{2}\sum_{i,j=1}^{N} S_{ij} \left\| \frac{\mathbf{v}_i}{\sqrt{D_i}} - \frac{\mathbf{v}_j}{\sqrt{D_j}} \right\|_2^2 \\ &= \text{tr}\,(\mathbf{V}\mathbf{L}\mathbf{V}') = \text{tr}\,(\mathbf{L}\mathbf{K}), \end{aligned} \tag{3.2}$$

where $\mathbf{L}$ is the graph Laplacian matrix defined as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}, \tag{3.3}$$

where $\mathbf{D} = \text{diag}(D_1, D_2, \ldots, D_N)$ is a diagonal matrix with the diagonal elements defined as $D_i = \sum_{j=1}^{N} S_{ij}$.

### 3.1.3 Formulation of Non-Parametric Kernel Learning

Taking into consideration of both the side information in (3.1) and the regularizer in (3.2), the NPKL problem is then formulated into the *loss + regularization* framework [HJL07] as follows:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \;\; \text{tr}\,\mathbf{L}\mathbf{K} + C\textstyle\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \ell\big(T_{ij}K_{ij}\big), \tag{3.4}$$

which generally belongs to a Semi-Definite Programming (SDP) problem [BV04]. Here, $C > 0$ is a tradeoff parameter to control the empirical loss[1] $\ell(\cdot)$ of the alignment $T_{ij}K_{ij}$ of the target kernel and the dependency among data examples with respect to the intrinsic data structure.

[1]The common choice of the loss function $\ell(\cdot)$ can be hinge loss, square hinge loss or linear loss.

## 3.2 SimpleNPKL: Simple Non-Parametric Kernel Learning

In this Section, we present a family of efficient algorithms for solving the NPKL problem in (3.4). We refer to the proposed efficient algorithms as "SimpleNPKL" for short.

### 3.2.1 Regularization on K

As aforementioned, the solution space of NPKL has been relaxed to boost its flexibility and capacity of fitting diverse patterns. However, arbitrarily relaxing the solution space $\mathcal{K}$ could result in over-fitting. To alleviate this problem, we introduce a regularization term:

$$\text{tr}\ (\mathbf{K}^p), \tag{3.5}$$

where $p \geq 1$ is a parameter to regulate the capacity of $\mathbf{K}$. Similar regularization terms on $\mathbf{K}$ have also been adopted in previous kernel learning studies. For example, [CSTEK01] used $\|\mathbf{K}\|_F = \sqrt{\langle \mathbf{K}, \mathbf{K} \rangle} = \sqrt{\text{tr}\ (\mathbf{K}^2)}$ in the objective to penalize the complexity of $\mathbf{K}$; while [LCB$^+$04] proposed to adopt a hard constraint $\text{tr}\ (\mathbf{K}) \leq B$, where $B > 0$ a constant, to control the capacity of $\mathbf{K}$.

We refer the modified NPK learning problem with the regularization term (3.5) either in the objective or in the constraint to as Simple Non-Parametric Kernel Learning (SimpleNPKL), which can be solved efficiently without engaging any standard SDP solvers. Next we present two SimpleNPKL algorithms that adopt several different types of loss functions.

### 3.2.2 SimpleNPKL with Linear Loss

First of all, we consider a linear loss function $\ell(f) = -f$, and rewrite the formulation of (3.4) as the SimpleNPKL formulation:

$$\min_{\mathbf{K}} \text{tr} \left( \left( \mathbf{L} - C \sum_{(i,j) \in (\mathcal{S} \cup \mathcal{D})} \mathbf{T}_{ij} \right) \mathbf{K} \right) \ :\ \mathbf{K} \succeq \mathbf{0},\ \text{tr}\ \mathbf{K}^p \leq B, \tag{3.6}$$

where $\mathbf{T}_{ij}$ is the matrix of setting the $(i, j)$-th entry to $T_{ij}$ and other entries to 0. To solve this problem, we first present a proposition below.

**Proposition 3.26:** *Given* $\mathbf{A}$ *is any symmetric matrix such that* $\mathbf{A} = \mathbf{P}\, diag(\boldsymbol{\sigma})\mathbf{P}'$, *where* $\mathbf{P}$ *contains columns of orthonormal eigenvectors of* $\mathbf{A}$ *and* $\boldsymbol{\sigma}$ *is a vector of the* *corresponding eigenvalues, and $B$ is any positive constant, the optimal solution $\mathbf{K}^*$ to the following SDP problem for $p > 1$:*

$$\max_{\mathbf{K}} tr\, \mathbf{AK} \quad : \quad \mathbf{K} \succeq \mathbf{0}, \quad tr\, \mathbf{K}^p \leq B, \tag{3.7}$$

*can be expressed as the following closed-form solution:*

$$\mathbf{K}^* = \left( \frac{B}{tr\, \mathbf{A}_+^{\frac{p}{p-1}}} \right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}} \tag{3.8}$$

*where $\mathbf{A}_+ = \mathbf{P}\, diag(\boldsymbol{\sigma}_+)\mathbf{P}'$, and $\boldsymbol{\sigma}_+$ is a vector with entries equal to $\max(0, [\boldsymbol{\sigma}]_i)$.*

*For $p = 1$, the optimal solution $\mathbf{K}^*$ can be expressed as the following closed-form solution:*

$$\mathbf{K}^* = B\mathbf{A}_1$$

*where $\mathbf{A}_1 = \mathbf{P}\, diag(\boldsymbol{\sigma}_1)\mathbf{P}'$, and $\boldsymbol{\sigma}_1$ is a vector with entries equal to $\frac{1}{\sum_{i:[\boldsymbol{\sigma}]_i = \max_i [\boldsymbol{\sigma}]_i} 1}$ for all $i$ that $[\boldsymbol{\sigma}]_i = \max_i [\boldsymbol{\sigma}]_i$; otherwise, the entries are zeros.* □

**Proof:** By introducing a dual variable $\gamma \geq 0$ for the constraint tr $\mathbf{K}^p \leq B$, and $\mathbf{Z} \in \mathcal{S}_+^n$ ($\mathcal{S}_+^n$ is self-dual) for the constraint $\mathbf{K} \succeq \mathbf{0}$, we have the Lagrangian of (6.7):

$$\mathcal{L}(\mathbf{K}; \gamma, \mathbf{Z}) = \text{tr } \mathbf{AK} + \gamma(B - \text{tr } \mathbf{K}^p) + \text{tr } \mathbf{KZ}.$$

By the Karush-Kuhn-Tucker (KKT) conditions, we have:

$$\mathbf{A} - \gamma p \mathbf{K}^{p-1} + \mathbf{Z} = \mathbf{0} \quad \text{and} \quad \text{tr } \mathbf{KZ} = 0.$$

First, we show that tr $(\mathbf{KZ}) = 0$ is equivalent to $\mathbf{KZ} = \mathbf{ZK} = \mathbf{0}$. Since $\mathbf{K} \succeq \mathbf{0}, \mathbf{Z} \succeq \mathbf{0}$, we have tr $(\mathbf{KZ})$ = tr $(\mathbf{K}^{1/2}\mathbf{K}^{1/2}\mathbf{Z}^{1/2}\mathbf{Z}^{1/2}) = \|\mathbf{K}^{1/2}\mathbf{Z}^{1/2}\|_F^2$. Thus, tr $(\mathbf{KZ}) = 0$ follows that $\mathbf{K}^{1/2}\mathbf{Z}^{1/2} = \mathbf{0}$. Pre-multiplying by $\mathbf{K}^{1/2}$ and post-multiplying by $\mathbf{Z}^{1/2}$ yields $\mathbf{KZ} = \mathbf{0}$, which in turn implies $\mathbf{KZ} = \mathbf{0} = (\mathbf{KZ})' = \mathbf{ZK}$. Hence, $\mathbf{K}$ and $\mathbf{Z}$ can be simultaneously diagonalized by the same set of orthonormal eigenvectors [AHO97]. From the first KKT condition we have $\mathbf{A} = \gamma p \mathbf{K}^{p-1} - \mathbf{Z}$. Consequently, $\mathbf{A}$ can also be diagonalized with the same eigenvectors as $\mathbf{K}$ and $\mathbf{Z}$.

Assume $\mathbf{A} = \mathbf{P}\text{diag}(\boldsymbol{\sigma})\mathbf{P}'$, where $\mathbf{P}$ contains columns of orthonormal eigenvectors of $\mathbf{A}$, and $\boldsymbol{\sigma}$ is the vector of the corresponding eigenvalues. Then, $\mathbf{K} = \mathbf{P}\text{diag}(\boldsymbol{\lambda})\mathbf{P}'$ and $\mathbf{Z} = \mathbf{P}\text{diag}(\boldsymbol{\mu})\mathbf{P}'$, where $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\mu} \geq \mathbf{0}$ denote the vector of the eigenvalues of $\mathbf{K}$ and $\mathbf{Z}$ respectively. Therefore, we have

$$\begin{aligned} \text{tr } \mathbf{K}^p &= \|\boldsymbol{\lambda}\|_p^p \leq B, && (3.9) \\ \text{tr } \mathbf{AK} &= \boldsymbol{\lambda}'\boldsymbol{\sigma}, && (3.10) \\ \boldsymbol{\sigma} &= \gamma p \boldsymbol{\lambda}^{p-1} - \boldsymbol{\mu}, && (3.11) \\ \boldsymbol{\lambda}'\boldsymbol{\mu} &= 0. && (3.12) \end{aligned}$$

Together with $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\mu} \geq \mathbf{0}$, and from (3.12), $[\boldsymbol{\lambda}]_i$ and $[\boldsymbol{\mu}]_i$ cannot be both non-zeros. Hence, from (3.11), we know $\boldsymbol{\sigma}_+ = \gamma p \boldsymbol{\lambda}^{p-1}$ contains all positive components of $\boldsymbol{\sigma}$. Moreover, from (3.10) and $\boldsymbol{\lambda} \geq \mathbf{0}$, together with the constraint (3.9), the SDP problem (6.7) is reduced to

$$\max_{\boldsymbol{\lambda}} \boldsymbol{\lambda}'\boldsymbol{\sigma}_+ \quad : \quad \|\boldsymbol{\lambda}\|_p^p \leq B.$$

By Hölder inequality, we have $\boldsymbol{\lambda}'\boldsymbol{\sigma}_+ \leq \|\boldsymbol{\lambda}\|_p \|\boldsymbol{\sigma}_+\|_q$, where it holds for $1/p + 1/q = 1$. The equality is achieved if and only if $|\boldsymbol{\lambda}|^p$ and $|\boldsymbol{\sigma}_+|^q$ are linearly dependent. Thus we can scale $\mathbf{K}$ satisfying (3.9) to arrive at the closed-form solution of $\mathbf{K}$ in (3.8) for $p > 1$.

For $p = 1$, from Equations (3.9) and (3.10), the optimization task is simplified as $\max \boldsymbol{\lambda}'\boldsymbol{\sigma} \, : \, \boldsymbol{\lambda} \geq 0, \|\boldsymbol{\lambda}\|_1 \leq B$. Due to the linearity, the maximum objective value is obtained by choosing $[\boldsymbol{\lambda}]_i = B / \sum_{i:[\boldsymbol{\sigma}]_i = \max_i [\boldsymbol{\sigma}]_i} 1$ for all $i$ that $[\boldsymbol{\sigma}]_i = \max_i [\boldsymbol{\sigma}]_i$; otherwise, $[\boldsymbol{\lambda}]_i = 0$.

Based on Proposition 3.26, we can easily solve the SimpleNPKL problem. In particular, by setting $\mathbf{A} = C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij} - \mathbf{L}$, we can directly compute the optimal $\mathbf{K}^*$ to SimpleNPKL of (3.6) using sparse eigen-decomposition as in (3.8). Thus the computation cost of SimpleNPKL with linear loss is dominated by eigen-decomposition. It is clear that this can significantly reduce the time cost for the NPKL tasks. Alternatively, we add tr $(\mathbf{K}^p)$ directly into the objective, and arrive at the following formulation:

$$\min_{\mathbf{K}} \operatorname{tr} \left( \left( \mathbf{L} - C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij} \right) \mathbf{K} \right) + \frac{G}{p} \operatorname{tr} \mathbf{K}^p \ : \ \mathbf{K} \succeq \mathbf{0},$$

where $G > 0$ is a tradeoff parameter. To solve this problem, we first present a proposition below.

**Proposition 3.27:** *Given* $\mathbf{A}$ *is any symmetric matrix such that* $\mathbf{A} = \mathbf{P}\,diag(\boldsymbol{\sigma})\mathbf{P}'$, *where* $\mathbf{P}$ *contains columns of orthonormal eigenvectors of* $\mathbf{A}$ *and* $\boldsymbol{\sigma}$ *is a vector of the corresponding eigenvalues, and* $B$ *is any positive constant, the optimal solution* $\mathbf{K}^*$ *to the following SDP problem for* $p > 1$*:*

$$\max_{\mathbf{K}} tr\, \mathbf{A}\mathbf{K} - \frac{G}{p} tr\, \mathbf{K}^p \quad : \quad \mathbf{K} \succeq \mathbf{0}, \tag{3.13}$$

*can be expressed as the following closed-form solution:*

$$\mathbf{K}^* = \left( \frac{1}{G} \mathbf{A}_+ \right)^{\frac{1}{p-1}} \tag{3.14}$$

*where* $\mathbf{A}_+ = \mathbf{P}\,diag(\boldsymbol{\sigma}_+)\mathbf{P}'$, *and* $\boldsymbol{\sigma}_+$ *is a vector with entries equal to* $\max(0, [\boldsymbol{\sigma}]_i)$. □

Following the techniques in the proof of Proposition 3.26, we obtain (3.14) immediately. If we set $G = \left(\frac{1}{B}\text{tr}\ \mathbf{A}_+^{\frac{p}{p-1}}\right)^{\frac{p-1}{p}}$, these two formulations result in exactly the same solution. Moreover, if we set $B = \text{tr}\ \mathbf{A}_+^{\frac{p}{p-1}}$, it means we just use the projection $\mathbf{A}_+$ as $\mathbf{K}$. No re-scaling of $\mathbf{A}_+$ is performed. In the sequel, we consider the regularization tr $\mathbf{K}^p$ with $p = 2$ for its simplicity and smoothness.

### 3.2.3 SimpleNPKL with Square Hinge Loss

Although the formulation with linear loss in (3.6) gives rise to a closed-form solution for the NPKL, one limitation of the NPKL formulation with linear loss is that it may be sensitive to noisy data due to the employment of the linear loss function. To address this issue, in this section, we present another NPKL formulation that uses (square) hinge loss $\ell(f) = (\max(0, 1 - f))^d/d$, which sometimes can be more robust, where $d = 1$ (hinge loss) or 2 (square hinge loss). We first focus on the NPKL formulation with square hinge loss, which can be written into the following constrained optimization:

$$\min_{\mathbf{K},\epsilon_{ij}} \ \text{tr}\ \mathbf{LK} + \frac{C}{2} \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \epsilon_{ij}^2 \tag{3.15}$$

$$\text{s.t.} \quad \forall(i,j)\in(\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \geq 1-\epsilon_{ij}, \tag{3.16}$$

$$\mathbf{K}\succeq \mathbf{0}, \text{tr}\ \mathbf{K}^p \leq B.$$

Note that we ignore the constraints $\epsilon_{ij} \geq 0$ since they can be satisfied automatically. However, (3.15) is not in the form of (6.7), and thus there is no longer a closed-form solution for $\mathbf{K}$.

#### 3.2.3.1 Dual Formulation: The Saddle-Point Minimax Problem

By Lagrangian theory, we introduce dual variables $\alpha_{ij}$'s ($\alpha_{ij} \geq 0$) for the constraints in (3.16), and derive a partial Lagrangian of (3.15):

$$\text{tr}\ \mathbf{LK} + \frac{C}{2}\sum_{(i,j)} \epsilon_{ij}^2 - \sum_{(i,j)} \alpha_{ij}(T_{ij}K_{ij} - 1 + \epsilon_{ij}). \tag{3.17}$$

For simplicity, we use $\sum_{(i,j)}$ to replace $\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})}$ in the sequel. By setting the derivatives of (3.17) w.r.t. the primal variables $\epsilon_{ij}$'s to zeros, we have

$$\forall(i,j) \in (\mathcal{S}\cup\mathcal{D}),\ C\epsilon_{ij} = \alpha_{ij} \geq 0$$

and substituting them back into (3.17), we arrive at the following saddle-point minimax problem $J(\mathbf{K}, \boldsymbol{\alpha})$:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \min_{\mathbf{K}} \ & \mathrm{tr}\left(\left(\mathbf{L} - \sum_{(i,j)} \alpha_{ij} \mathbf{T}_{ij}\right)\mathbf{K}\right) - \frac{1}{2C}\sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} \\ \text{s.t.} \quad & \mathbf{K} \succeq \mathbf{0},\ \mathrm{tr}\, \mathbf{K}^p \leq B,\ \forall (i,j) \in \mathcal{S} \cup \mathcal{D},\ \alpha_{ij} \geq 0, \end{aligned} \tag{3.18}$$

where $\boldsymbol{\alpha} = [a_{ij}]$ denotes a matrix of dual variables $\alpha_{ij}$'s for $(i,j) \in \mathcal{S} \cup \mathcal{D}$, and other entries are zeros. This problem is similar to the optimization problem of DIFFRAC [BH08], in which $K$ and $\boldsymbol{\alpha}$ can be solved by an iterative manner.

#### 3.2.3.2 Iterative Algorithm

In this subsection, we present an iterative algorithm which follows the similar update strategy in [BX05]: 1) For a fixed $\boldsymbol{\alpha}_{t-1}$, we can let $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^{t-1} \mathbf{T}_{ij} - \mathbf{L}$. Based on Proposition 3.26, we can compute the closed form solution $\mathbf{K}_t$ to (3.18) using (3.8); 2) For a fixed $\mathbf{K}_t$, we can update $\boldsymbol{\alpha}_t$ using $\boldsymbol{\alpha}_t = (\boldsymbol{\alpha}_{t-1} + \eta_t \nabla J_t)_+$; 3) Step 1) and 2) are iterated until convergence. Here $J$ denotes the objective function (3.18), $\nabla J_t$ abbreviates the derivative of $J$ at $\boldsymbol{\alpha}_t$, and $\eta_t > 0$ is a step size parameter. The following Lemma guarantees the differentiable properties of the optimal value function [BS96, YYG09]:

**Lemma 3.3** *Let $\mathcal{X}$ be a metric space and $\mathcal{U}$ be a normed space. Suppose that for all $x \in \mathcal{X}$ the function $f(x, \cdot)$ is differentiable and that $f(x,u)$ and $\nabla_u f(x,u)$ are continuous on $\mathcal{X} \times \mathcal{U}$, and $Q$ be a compact subset of $\mathcal{X}$. Then the optimal value function $f(u) := \inf_{x \in Q} f(x,u)$ is differentiable. When the minimizer $x(u)$ of $f(\cdot, u)$ is unique, the gradient is given by $\nabla f(u) = \nabla_u f(u, x(u))$.*

From Proposition 3.26, we see that the minimizer $\mathbf{K}(\boldsymbol{\alpha})$ is unique for some fixed $\boldsymbol{\alpha}$. Together with the above lemma, we compute the gradient at the point $\boldsymbol{\alpha}$ by:

$$\nabla J_{ij} = 1 - \mathrm{tr}\, \mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}, \tag{3.19}$$

where $\mathbf{K} = \left(\frac{B}{\mathrm{tr}\, \mathbf{A}_+^{\frac{p}{p-1}}}\right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}}$, $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^t \mathbf{T}_{ij} - \mathbf{L}$.

Similarly, for the another formulation:

$$\begin{aligned} \min_{\mathbf{K}, \epsilon_{ij}} \ & \mathrm{tr}\, \mathbf{LK} + \frac{C}{2} \sum_{(i,j) \in (\mathcal{S} \cup \mathcal{D})} \epsilon_{ij}^2 + \frac{G}{p} \mathrm{tr}\, \mathbf{K}^p \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S} \cup \mathcal{D}),\ T_{ij} K_{ij} \geq 1 - \epsilon_{ij}, \end{aligned} \tag{3.20}$$

we can derive the corresponding saddle-point minimax problem of (3.20):

$$\begin{array}{ll} \max_{\boldsymbol{\alpha}} \min_{\mathbf{K}} \text{ tr } \left( \left( \mathbf{L} - \sum_{(i,j)} \alpha_{ij} \mathbf{T}_{ij} \right) \mathbf{K} \right) - \frac{1}{2C} \sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} + \frac{G}{p} \text{tr } \mathbf{K}^p \\ \text{s.t.} \qquad \mathbf{K} \succeq \mathbf{0}, \ \forall (i,j) \in \mathcal{S} \cup \mathcal{D}, \ \alpha_{ij} \geq 0. \end{array}$$

Again, from the Proposition 3.27, we observe that the minimizer $\mathbf{K}(\boldsymbol{\alpha})$ is unique for some fixed $\boldsymbol{\alpha}$. Together with Lemma 3.3, we compute the gradient at the point $\boldsymbol{\alpha}^t$ in the same way as in (3.19) by setting $\mathbf{K} = \left(\frac{1}{G}\mathbf{A}_+\right)^{\frac{1}{p-1}}$, $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^t \mathbf{T}_{ij} - \mathbf{L}$. The alternative optimization algorithm is summarized in Algorithm 5.

**Protocol 1** SimpleNPKL with (square) hinge loss.

```
Input: Pairwise constraint matrix $\mathbf{T}$, parameters $C$ and $B$ (or G), $k$
Output: $\boldsymbol{\alpha}$ and $\mathbf{K}$.
1: Construct graph Laplacian $\mathbf{L}$ using $k$ nearest neighbors;
2: Initialize $\boldsymbol{\alpha}^0$;
3: repeat
4:     Set $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^{t-1} \mathbf{T}_{ij} - \mathbf{L}$;
5:     Compute the closed-form solution $\mathbf{K}_t = \left(B/\text{tr } \mathbf{A}_+^{p/(p-1)}\right)^{1/p} \mathbf{A}_+^{1/(p-1)}$
       //For the formulation (3.13), use $\mathbf{K}_t = \left(\mathbf{A}_+/G\right)^{1/(p-1)}$ instead;
6:     Compute the gradient $\nabla J_{ij} = 1 - \text{tr } \mathbf{T}_{ij}\mathbf{K}_t - \frac{1}{C}\alpha_{ij}$;
7:     Determine a step size $\eta_t$, update $\alpha_{ij}^t$ using $\alpha_{ij}^t = \left(\alpha_{ij}^{t-1} + \eta_t \nabla J_{ij}\right)_+$;
8: until convergence
```

#### 3.2.3.3 Estimating the Rank of $K$

According to Proposition 3.26 or Proposition 3.27, we are required to locate the positive spectrums of $\mathbf{A}$, which can be achieved by full eigen-decomposition of $\mathbf{A}$. However, this can be computationally prohibitive for large scale data sets. Moreover, the computation on the negative eigen-vectors of $\mathbf{A}$ should be avoided. The following proposition [Pat95] bounds the rank of matrix $\mathbf{K}$ in a general SDP setting.

**Proposition 3.28:** *The rank $r$ of $\mathbf{K}$ in the SDP problem: $\max_{\mathbf{K}\succeq\mathbf{0}} tr\,(\mathbf{A}_0\mathbf{K})$ with $m$ linear constraints on $K$, follows the bound $\begin{pmatrix} r+1 \\ 2 \end{pmatrix} \leq m$.* □

Moreover, from the empirical study in [AHO97], the rank $r$ is usually much smaller than this bound. This implies that the full decomposition of matrix $\mathbf{A}_0$ is not required. Our formulation (3.15) has an additional constraint: tr $\mathbf{K}^2 \leq B$ for $p = 2$. This condition equivalently constraints tr $(\mathbf{K})$, which is a common assumption in SDP problems [KK06].

To show this, we have $B \geq \text{tr}\,\mathbf{K}\mathbf{K} = \frac{1}{N}\sum_i \lambda_i^2 N \geq \frac{1}{N}(\sum_i \lambda_i \cdot 1)^2 = \frac{1}{N}(\text{tr}\,\mathbf{K})^2$, where the second inequality is resulted from the Cauchy inequality. Hence, we have $\text{tr}\,\mathbf{K} \leq \sqrt{BN}$. Therefore, we can make use of the $r$ estimated from Proposition 3.28 as a suggestion to estimate the rank of $\mathbf{K}$.

#### 3.2.3.4 Determining the Convergence Properties

When the $\eta_t$ is small enough or a universal choice of $\eta_t = O(1/t)$ is used, the whole optimization problem is guaranteed to converge [BX05]. Practically, the value of $\eta$ plays an important role for the convergence speed. Therefore, it is worth studying the influence of $\eta$ on the convergence rate, which requires to lower bound the increment of $J_{\boldsymbol{\alpha}_t}$ at each step. We first establish the Lipschitz property of $\nabla J(\boldsymbol{\alpha})$.

**Lemma 3.4** *Assume we use the formulation of Proposition 3.27 at each iteration of Algorithm 5, then the gradient of the objective function given by (*3.19*) is Lipschitz continuous with Lipschitz constant $L = \frac{m}{G} + \frac{1}{C}$, where $m = |\mathcal{S} \cup \mathcal{D}|$ is the number of nonzeros in $\mathbf{T}$. That is,*

$$\|\nabla J(\boldsymbol{\alpha}_1) - \nabla J(\boldsymbol{\alpha}_2)\|_F \leq \big(\frac{m}{G} + \frac{1}{C}\big)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_F.$$

**Proof:** For an $\boldsymbol{\alpha}_t$, we use $\mathbf{K}_t$ denote the corresponding minimizer of $J$ computed by (3.8). For a spectral function $\lambda$ defined on $\mathbb{S}_+$, which is Lipschitz continuous with Lipschitz constant $\kappa$, we have

$$\|\lambda(\mathbf{K}_1) - \lambda(\mathbf{K}_2)\|_F \leq \kappa\|\mathbf{K}_1 - \mathbf{K}_2\|_F.$$

For our case, the p.s.d. projection is defined by $\lambda(\mathbf{K}) = \sum_i \max(0, \lambda_i)^2$. The Lipschitz constant $\kappa$ of this function is 1. Therefore, for any $\mathbf{K}_1$ and $\mathbf{K}_2$ given by (3.8), we have

$$\begin{aligned}
\|\mathbf{K}_1 - \mathbf{K}_2\|_F &= \|\mathbf{A}_{1+} - \mathbf{A}_{2+}\|_F \\
&\leq \Big\|\frac{1}{G}\Big(\sum_{(i,j)} \alpha_{ij}^{(1)}\mathbf{T}_{ij} - \mathbf{L}\Big) - \frac{1}{G}\Big(\sum_{(i,j)} \alpha_{ij}^{(2)}\mathbf{T}_{ij} - \mathbf{L}\Big)\Big\|_F \\
&= \frac{1}{G}\Big\|\sum_{(i,j)} (\alpha_{ij}^{(1)} - \alpha_{ij}^{(2)})\mathbf{T}_{ij}\Big\|_F \\
&\leq \frac{1}{G}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_F\|\mathbf{T}\|_F = \frac{\sqrt{m}}{G}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_F.
\end{aligned}$$

Consequently, we have,

$$\begin{aligned}\|\nabla J(\boldsymbol{\alpha}_1) - \nabla J(\boldsymbol{\alpha}_2)\|_F &= \sqrt{\sum_{(i,j)} \Big( \big(1 - \mathrm{tr}\, \mathbf{T}_{ij}\mathbf{K}_1 - \frac{1}{C}\alpha_{ij}^{(1)}\big) - \big(1 - \mathrm{tr}\, \mathbf{T}_{ij}\mathbf{K}_2 - \frac{1}{C}\alpha_{ij}^{(2)}\big)\Big)^2} \\ &= \sqrt{\sum_{(i,j)} \Big( \mathrm{tr}\, \mathbf{T}_{ij}\big(\mathbf{K}_2 - \mathbf{K}_1\big) + \frac{1}{C}\big(\alpha_{ij}^{(2)} - \alpha_{ij}^{(1)}\big)\Big)^2} \\ &\leq \|\mathbf{T}\|_F \|\mathbf{K}_1 - \mathbf{K}_2\|_F + \frac{1}{C}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_F \\ &\leq \big(\frac{m}{G} + \frac{1}{C}\big)\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_F.\end{aligned}$$

With the Lipschitz property of $\nabla J$, we can further show each iteration of Algorithm 5 makes progress towards the optimal solution. Interestingly, we are aware that the proof is very similar to the analysis of indefinite kernel learning, which is proposed very recently by [YYG09]. This result is developed based on non-smooth optimization algorithm of [Nes05]. To make the chapter complete, we expose the detailed proof in the following proposition.

**Proposition 3.29:** *Assume we use the formulation of Proposition 3.27, and* $\eta \geq \frac{m}{G} + \frac{1}{C}$ *at each iteration of Algorithm 5. The iteration sequence* $\{\boldsymbol{\alpha}_t\}$ *generated in Algorithm 5 satisfy:*

$$J(\boldsymbol{\alpha}_{t+1}) \geq J(\boldsymbol{\alpha}_t) + \frac{\eta}{2}\|\boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t\|_F^2,$$

*and*

$$\max_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) - J(\boldsymbol{\alpha}_t) \leq \frac{\eta}{2t}\|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}^*\|_F^2,$$

*where* $\boldsymbol{\alpha}^*$ *is the optimal solution of* $\max_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha})$. □

**Proof:** Let $L = \frac{m}{G} + \frac{1}{C}$ abbreviate the Lipschitz constant of $\nabla J(\boldsymbol{\alpha})$, then we have

$$\begin{aligned}J(\boldsymbol{\alpha}) - J(\boldsymbol{\alpha}_t) - \langle \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha} - \boldsymbol{\alpha}_t\rangle &= \int_{\boldsymbol{\alpha}_t}^{\boldsymbol{\alpha}} \nabla J(\boldsymbol{\alpha}) d\boldsymbol{\alpha} - \langle \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha} - \boldsymbol{\alpha}_t\rangle \\ &= \int_0^1 \langle \nabla J(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\alpha}_t) - \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha} - \boldsymbol{\alpha}_t\rangle d\theta \\ &\geq -\int_0^1 \|\nabla J(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\alpha}_t) - \nabla J(\boldsymbol{\alpha}_t)\| \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_t\|_F d\theta \\ &\geq -L\int_0^1 \theta \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_t\|_F^2 d\theta \\ &\geq -\frac{\eta}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_t\|_F^2.\end{aligned}$$

Applying this inequality with $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{t+1}$, we have

$$-J(\boldsymbol{\alpha}_t) - \langle \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t \rangle \geq -J(\boldsymbol{\alpha}_{t+1}) - \frac{\eta}{2}\|\boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t\|_F^2. \tag{3.21}$$

From step 5 in Algorithm 5, it is easy to verify that

$$\begin{aligned}
\boldsymbol{\alpha}_{t+1} &= \arg\min_{\boldsymbol{\alpha}} \|(\boldsymbol{\alpha} - \boldsymbol{\alpha}_t) - \nabla J(\boldsymbol{\alpha}_t)/\eta\|_F^2 \\
&= \arg\min_{\boldsymbol{\alpha}} -2\langle \boldsymbol{\alpha} - \boldsymbol{\alpha}_t, \nabla J(\boldsymbol{\alpha}_t)/\eta \rangle + \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_t\|_F^2 \\
&= \arg\min_{\boldsymbol{\alpha}} -\nabla J(\boldsymbol{\alpha}_t) - \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}_t, \nabla J(\boldsymbol{\alpha}_t) \rangle + \frac{\eta}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_t\|_F^2. \qquad (3.22)
\end{aligned}$$

Let $f(\boldsymbol{\alpha})$ denote the right side of (3.22). From the first-order optimality condition over $\boldsymbol{\alpha}_{t+1}$, for any $\boldsymbol{\alpha}$ we have $\langle \nabla f(\boldsymbol{\alpha}_{t+1}), \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t+1} \rangle \geq 0$, that is,

$$-\langle \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t+1} \rangle \geq \eta \langle \boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t, \boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha} \rangle. \tag{3.23}$$

Adding (3.21) and (3.23) together yields that $-J(\boldsymbol{\alpha}_t) - \langle \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha} - \boldsymbol{\alpha}_t \rangle \geq -J(\boldsymbol{\alpha}_{t+1}) + \eta\langle \boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t+1}, \boldsymbol{\alpha} - \boldsymbol{\alpha}_t \rangle + \frac{\eta}{2}\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t+1}\|_F^2$. Note that $-J$ is convex, $-J(\boldsymbol{\alpha}) \geq -J(\boldsymbol{\alpha}_t) - \langle \nabla J(\boldsymbol{\alpha}_t), \boldsymbol{\alpha} - \boldsymbol{\alpha}_t \rangle$. Thus we have

$$J(\boldsymbol{\alpha}_{t+1}) \geq J(\boldsymbol{\alpha}) + \eta\langle \boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t+1}, \boldsymbol{\alpha} - \boldsymbol{\alpha}_t \rangle + \frac{\eta}{2}\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t+1}\|_F^2.$$

Applying $\boldsymbol{\alpha} = \boldsymbol{\alpha}_t$, we have that

$$J(\boldsymbol{\alpha}_{t+1}) \geq J(\boldsymbol{\alpha}_t) + \frac{\eta}{2}\|\boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t\|_F^2.$$

Applying $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$, we have that

$$J(\boldsymbol{\alpha}^*) - J(\boldsymbol{\alpha}_{i+1}) \leq -\eta\langle \boldsymbol{\alpha}_i - \boldsymbol{\alpha}_{i+1}, \boldsymbol{\alpha}^* - \boldsymbol{\alpha}_i \rangle - \frac{\eta}{2}\|\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_{i+1}\|_F^2 = \frac{\eta}{2}\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}_i\|_F^2 - \frac{\eta}{2}\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}_{i+1}\|_F^2. \tag{3.24}$$

Taking summation over $i$ from 0 to $t-1$, we have

$$\sum_{i=0}^{t-1}(J(\boldsymbol{\alpha}^*) - J(\boldsymbol{\alpha}_{i+1})) \leq \frac{\eta}{2}\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}_0\|_F^2.$$

From (3.24), we see that the sequence $\{J(\boldsymbol{\alpha}_t)\}$ increase monotonically. Thus we obtain

$$t(J(\boldsymbol{\alpha}^*) - J(\boldsymbol{\alpha}_t)) \leq \frac{\eta}{2}\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}_0\|_F^2,$$

which completes the proof.

### 3.2.4 SimpleNPKL with Square Loss

In this subsection, we consider square alignment loss for the SimpleNPKL framework:

$$\begin{aligned} \min_{\mathbf{K},\epsilon_{ij}} \quad & \operatorname{tr} \mathbf{L}\mathbf{K} + \frac{C}{2} \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \epsilon_{ij}^2 \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} = 1 - \epsilon_{ij}, \\ & \mathbf{K} \succeq \mathbf{0}, \operatorname{tr} \mathbf{K}^p \le B. \end{aligned}$$

Here we need not to enforce $\epsilon \ge 0$. With the standard techniques of Section 3.2.3, we derive the following min-max problem:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{K}} \operatorname{tr} \left( \mathbf{L} - \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \right) \mathbf{K} + \sum_{ij} \alpha_{ij} - \frac{1}{2C} \sum_{ij} \alpha_{ij}^2 \quad : \quad \mathbf{K} \succeq \mathbf{0}, \operatorname{tr} \mathbf{K}^p \le B.$$

Therefore, we can compute the gradient of $J$ w.r.t. $\boldsymbol{\alpha}$:

$$\nabla J_{ij} = 1 - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}.$$

The whole analysis of Section 3.2.3 still holds. The difference just lies in the way of computing gradient $\nabla J$. We will show an application of square loss in Section 3.4.

### 3.2.5 SimpleNPKL with Hinge Loss

In this subsection, we consider hinge loss for the SimpleNPKL framework:

$$\begin{aligned} \min_{K,\epsilon_{ij}} \quad & \operatorname{tr} \mathbf{L}\mathbf{K} + C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \epsilon_{ij} \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \ge 1 - \epsilon_{ij}, \epsilon_{ij} \ge 0 \\ & \mathbf{K} \succeq \mathbf{0}, \operatorname{tr} \mathbf{K}^p \le B. \end{aligned}$$

Following the standard techniques of Lagrangian dual, we arrive at the min-max problem:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{K}} \operatorname{tr} \left( \mathbf{L} - \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \right) \mathbf{K} + \sum_{ij} \alpha_{ij} \quad : \quad \mathbf{K} \succeq \mathbf{0},\ \operatorname{tr} \mathbf{K}^p \le B,\ 0 \le \alpha_{ij} \le C.$$

Therefore, we can compute the gradient of $J$ w.r.t. $\boldsymbol{\alpha}$:

$$\nabla J_{ij} = 1 - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K}$$

The whole analysis of Section 3.2.3 still holds. The difference just lies in the way of computing gradient $\nabla J$. Note that the gradient updating $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \eta \nabla J$ may jump out of the range $[0, C]$. We need to project $\boldsymbol{\alpha}$ into this region at each iteration. We will also show an example of Hinge loss in Section 3.4.

## 3.3 Implementation Issues

In this Section, we discuss some implementation issues that are important to the success of the proposed SimpleNPKL algorithms.

### 3.3.1 Building a Sparse Graph Laplacian

Recall that the graph Laplacian $\mathbf{L}$ in (3.3) is often sparse, in particular, which is usually computed by finding $k$-nearest neighbors for the purpose of constructing the similarity matrix $\mathbf{S}$. Specifically, an entry $S(i,j) = 1$ if and only if data examples $i$ and $j$ are among each other's $k$-nearest neighbors; otherwise, it is set to 0. So, there are at most $k$ nonzero entries on each row of $\mathbf{L}$.

A naive implementation of finding $k$-nearest neighbors often takes $O(N^2 \log N)$ time. To enforce the data examples $i$ and $j$ are among each other's $k$-nearest neighbors, one can use B-matching algorithm [JS06] to find the $k$-nearest neighbors. However, when the data set is very large, the construction of $\mathbf{L}$ becomes non-trivial and very expensive. To address this challenge, we suggest to first construct the *cover tree* structure [BKL06], which takes $O(N \log N)$ time. The similar idea to construct a tree structure for distance metric learning was discussed in [WS08]. With the aid of this data structure, the batch query of finding $k$-nearest neighbors on the whole data set can be done within $O(N)$ time. Hence, the graph Laplacian $\mathbf{L}$ can be constructed efficiently for large-scale problems.

### 3.3.2 Fast Eigendecomposition by Lanczos Algorithm

Among various existing SDP approaches [BV04], the interior-point method is often deemed as the most efficient one. However, as discussed in previous subsection, the graph Laplacian $\mathbf{L}$ is often sparse. In addition, the number of pairwise constraints is usually small due to expensive cost of human labels. Therefore, $\mathbf{L} - \sum_{(i,j)} \alpha_{ij}\mathbf{T}_{ij}$ is also sparse. Such sparse structure is not yet exploited in such general algorithms. According to Proposition 3.26, the time cost of each iteration in Algorithm 5 is dominated by eigen-decomposition. Moreover, from Proposition 3.28, the rank $r$ of the kernel matrix $\mathbf{K}$ is upper bounded by the number of active constraints. Therefore, we can estimate the rank for sparse eigen-decomposition, which can be solved efficiently using the so-called *Implicitly Restarted Lanczos Algorithm* (IRLA) [LSY98]. Its computational cost is dominated by matrix-vector multiplication. Specifically, the time cost of IRLA is linear with the number of non-zeros in $\mathbf{A}$. Assume $k$ nearest neighbors are used to construct the graph Laplacian $\mathbf{L}$, then the number of non-zeros in $\mathbf{A}$ is at most $Nk + m$, where $m$ is the number of nonzeros in $\mathbf{T}$, and $\mathbf{A}$ is very sparse. Moreover, the time cost of computing gradient is $O(m)$. Therefore, the time complexity per iteration of SimpleNPKL is $O(Nk + m)$.

### 3.3.3 Active Constraint Selection

As shown in Algorithm 5, the computational cost of the update procedure is highly depends on the number of pairwise constraints. However, some less informative constraints often do not contribute much to the learning of the kernel matrix $\mathbf{K}$, and fitting some noisy pairwise constraints may also lead to the poor generalization. Moreover, as discussed in Section 3.2.3.3, the rank of $\mathbf{K}$ is lower when there are fewer active constraints in (3.16). Therefore, selecting pairwise constraints for SimpleNPKL may improve both the efficiency and the generalization of the NPK learning.

To speed up the eigen-decomposition process, instead of engaging all pairwise constraints, we propose to sample a subset of $T_{ij}$'s for SimpleNPKL. Instead of acquiring class label information for kernel learning; here, we consider another simple active constraint selection scheme. Recall that a general principle in active learning is to request the label of the data points that are most uncertain for their predictions. Following this idea, we adopt the margin criterion to measure the uncertainty of the prediction value on a data point. In particular, given a data point $\mathbf{x}_i$, assume that we have the prediction function in the form:

$$f(\mathbf{x}_i) = \sum_j y_j k(\mathbf{x}_i, \mathbf{x}_j).$$

We can use $|y_i f(\mathbf{x}_i)|$ to measure the uncertainty of prediction, where $y_i \in \{-1, +1\}$ is the class label of data point $\mathbf{x}_i$. As a result, for a data point $\mathbf{x}_i$, we choose the constraints involving point $i$:

$$\begin{aligned} i^* &= \arg\min_i \left| \frac{1}{l_i} \sum_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \right| \\ &= \arg\min_i \left| \frac{1}{l_i} \sum_{j, T_{ij} \neq 0} T_{ij} k(\mathbf{x}_i, \mathbf{x}_j) \right|, \end{aligned}$$

where we deem $T_{ij}$ as an entry of $yy'$, and $l_i = |\{j : (i,j) \in \mathcal{S} \cup \mathcal{D}\}, T_{ij} \neq 0\}|$ is used as a normalization of the margin value. Based on the above formula, we choose a subset of $k$ data points $\mathcal{S}_k$ that are most uncertain according to the margin measure. Then, we choose all the $T_{ij}$'s that involve any point $i \in \mathcal{S}_k$ as pairwise constraints to form a new set of constraints. Finally, we run SimpleNPKL based on this new set of constraints.

### 3.3.4 Low Rank Approximation of K

Since the rank $r$ of $\mathbf{K}$ often satisfies $r < n$, we may express $\mathbf{K}$ as $\mathbf{K} = \mathbf{VEV}'$, where the columns of $\mathbf{V}_{n\times r}$ are eigenvectors of $\mathbf{K}$. If we fix the base $\mathbf{V}$, the number of variables is reduced from $n^2$ to $r^2$. With this approximation scheme, the $\mathbf{A}$ matrix in Algorithm 5 becomes $\mathbf{A} = \mathbf{V}'(\mathbf{L} - \sum \alpha_{ij}\mathbf{T}_{ij})\mathbf{V}$. Note $\mathbf{V}'\mathbf{LV}$ can be pre-computed and $\mathbf{V}' \sum \alpha_{ij}\mathbf{T}_{ij}\mathbf{V}$ can be computed efficiently by virtue of the sparseness. Therefore, SimpleNPKL can be significantly faster with this approximation.

## 3.4 Applications of SimpleNPKL

In this Section, we extend the proposed SimpleNPKL technique to other similar machine learning problems where the goal of the optimization is to find an optimal matrix such that its inner product with another matrix is maximized or minimized. In particular, we consider the data embedding problems, where the goal is to find a new data representation that preserves some similarity/distance constraints between pairs of data points. These problems typically can be implemented by constraining the alignment of the target kernel matrix to some prior affinity or distance structures. As a result, the kernel matrix $\mathbf{K} = \mathbf{V}'\mathbf{V}$ implies a data embedding with a natural interpretation, in which the column vector of $\mathbf{V}$ corresponds to the new data representation. We discuss several important data embedding methods below.

### 3.4.1 Colored Maximum Variance Unfolding

Colored MVU [SSBG07] is an improvement of Maximum Variance Unfolding (MVU) [WSS04], which produces a low-dimensional representation of the data by maximizing the trace of a matrix $\mathbf{K}$ subject to some positive definiteness, centering and distance-preserving constraints, that is:

$$\min_{\mathbf{K}} \quad -\mathrm{tr}\,\mathbf{K} \quad : \quad \mathbf{K}\succeq \mathbf{0}, \sum_{ij} \mathbf{K}_{ij} = 0,\ \mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = D_{ij},\ \forall (i,j) \in \mathcal{N}.$$

where $\mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = K_{ii} + K_{jj} - 2K_{ij}$ is the square distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

CMVU interprets MVU from a statistical perspective. It maximizes the dependence between the domain of input pattern $\mathbf{x}$ and the domain of label $y$, which is measured by the *Hilbert- Schmidt Independence Criterion* [GBSS05, SSBG07]. Here we introduce slack variables $\boldsymbol{\xi}$ to measure the violations of distance constraints and penalize the corresponding square loss. Consequently the optimization task of colored MVU is reformulated as:

$$\min_{\mathbf{K},\boldsymbol{\xi}} \quad -\mathrm{tr}\,\mathbf{HKHY} + \frac{C}{2}\sum \xi_{ij}^2, \quad : \quad \mathbf{K}\succeq \mathbf{0},\ \mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = D_{ij} - \xi_{ij},\ \forall (i,j) \in \mathcal{N}$$

where $H_{ij} = \delta_{ij} - N^{-1}$ such that $\mathbf{HKH}$ centers $\mathbf{K}$, $\mathbf{Y} = \mathbf{y}\mathbf{y}'$ is the kernel matrix over labels. Apparently this belongs to an SDP problem.

Following the SimpleNPKL algorithms, we derive the minimax optimization problem by introducing dual variables for the inequality constraints:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{K}} \ \mathrm{tr}\left(-\mathbf{HYH} - \sum_{ij} \alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K} + \sum_{ij} \alpha_{ij} D_{ij} - \frac{1}{2C}\sum_{ij} \alpha_{ij}^2 : \mathbf{K}\succeq \mathbf{0}, \mathrm{tr}\,\mathbf{KK} \leq B. \tag{3.25}$$

By substituting the following results

$$\mathbf{A} = \mathbf{HYH} + \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \text{ and } \nabla J_{ij}^t = D_{ij} - \text{tr } \mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}^t$$

back into Algorithm 5, the problem of (3.25) can be solved immediately.

### 3.4.2 Minimum Volume Embedding

Minimum Volume Embedding (MVE) is another improvement of MVU [SJ07]. One limitation of MVU is that it simply maximizes the trace of $\mathbf{K}$, which may result in the solution that engages considerably more dimensions than necessity. To address this problem, [SJ07] proposed to grow the top few eigenvalues of $\mathbf{K}$ while shrinking the remaining ones. In particular, let $\mathbf{K} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i'$, $\lambda_1 \geq, \ldots, \geq \lambda_n$, and $\mathbf{K}_0 = \sum_{i=1}^d \mathbf{v}_i\mathbf{v}_i' - \sum_{i=d+1}^n \mathbf{v}_i\mathbf{v}_i'$. When the intrinsic dimensionality $d$ is available, MVE formulates the data embedding problem as follows:

$$\min_{\mathbf{K}} \quad -\text{tr } \mathbf{KK}_0 \quad : \quad \text{the same set of constraints of MVU.} \tag{3.26}$$

After obtaining the solution $\mathbf{K}^t$ at each step, MVE proceeds by substituting $\mathbf{K}_0 = \mathbf{K}^t$ back to the optimization of (3.26) and repeatedly solving the optimization. Hence, MVE improves MVU by decreasing the energy of the small eigen components of $\mathbf{K}$. To find the solution, every $\mathbf{K}^t$ is computed by applying a general SDP solver in [SJ07].

To speed up the solution, following the similar derivation in the above CMVU, we can solve (3.26) by eigen-decomposition in an iterative manner. Specifically, we make the following modifications:

$$A = \mathbf{K}_0 + \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \text{ and } \nabla J_{ij}^t = D_{ij} - \text{tr } \mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}^t$$

By substitute the above results back into Algorithm 5, we can solve the MVE problem efficiently.

### 3.4.3 Structure Preserving Embedding

Structure Preserving Embedding (SPE) [SJ09] is a machine learning technique that embeds graphs in low-dimensional Euclidean space such that the embedding preserves the global topological properties of the input graph. Suppose we have a connectivity matrix $\mathbf{W}$, where $W_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected and $W_{ij} = 0$ otherwise. SPE learns a kernel matrix $\mathbf{K}$ such that the similarity tr $\mathbf{KW}$ is maximized while the global topological properties of the input graph are preserved. More formally, the SPE problem is formulated into the following SDP optimization:

$$\min_{\mathbf{K}} \quad -\text{tr } \mathbf{KW} + C\xi \quad : \quad D_{ij} > (1 - W_{ij}) \max_m (W_{im} D_{im}) - \xi, \;\; \xi \geq 0$$

where $D_{ij} = K_{ii} + K_{jj} - 2K_{ij} = \mathrm{tr}\ \mathbf{KT}_{ij}$ is the squared distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Let $[n] = \{1, \ldots, n\}$ and $\mathcal{N}_i$ denote the set of indices of points which are among the nearest neighbors of $\mathbf{x}_i$. Then for each point $\mathbf{x}_i$, SPE essentially generates $(n-|\mathcal{N}_i|)\times|\mathcal{N}_i|$ constraints:

$$\mathrm{tr}\ \mathbf{KT}_{ij} > \mathrm{tr}\ \mathbf{KT}_{ik} - \xi, \ \ \forall i \in [n], j \in [n] - \mathcal{N}_i, k \in \mathcal{N}_i.$$

In order to speed up the SPE algorithm, we apply the SimpleNPKL technique to turn the SPE optimization into the following minimax optimization problem:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{K}} \ \ \mathrm{tr}\left(\sum_i \sum_{k\in\mathcal{N}_i} \sum_{j\notin\mathcal{N}_i} \alpha_{ijk}(\mathbf{T}_{ik}-\mathbf{T}_{ij})-\mathbf{W}\right)\mathbf{K} : \mathbf{K}\succeq \mathbf{0}, \mathrm{tr}\ \mathbf{KK}\leq B, \sum \alpha_{ijk} \in [0, C].$$

Similarly, we can derive the following results:

$$\mathbf{A} = \mathbf{W} - \sum_{ijk} \alpha_{ijk}(\mathbf{T}_{ik} - \mathbf{T}_{ij}) \ \ \text{and} \ \ \nabla J^t_{ijk} = \mathrm{tr}\ \mathbf{K}(\mathbf{T}_{ik} - \mathbf{T}_{ij}).$$

Substituting them back into Algorithm 5 leads to an efficient solution for the SPE problem.

## 3.5 Experiments

In this Section, we conduct extensive experiments to examine the efficacy and efficiency of the proposed SimpleNPKL algorithms.

### 3.5.1 Experimental Setup

We examine both efficacy and efficiency of the proposed SimpleNPKL using side information to learn a kernel matrix for kernel $k$-means clustering. As shown in [HJL07], the learned kernel matrix of the Non-Parametric Kernel Learning (NPKL) outperforms other kernel learning methods in the task of clustering using side information. For simplicity, we only compare our proposed SimpleNPKL algorithms with the NPKL method in [HJL07] for kernel $k$-means clustering. The results of $k$-means clustering and constrained $k$-means clustering using Euclidean metric are also reported as the performance of the baseline methods. The abbreviations of different approaches are described as follows:

- ***k*-means**: $k$-means clustering using Euclidean metric;
- **c*k*-means**: The constrained $k$-means clustering algorithm using Euclidean metric and side information;
- **SimpleNPKL+LL**: The proposed SimpleNPKL with linear loss defined in (3.6);
- **SimpleNPKL+SHL**: The proposed SimpleNPKL with squared hinge loss defined in (3.15);
- **NPKL+LL**: NPKL in (3.4) using linear loss;
- **NPKL+HL**: NPKL in (3.4) using hinge loss.

To construct the graph Laplacian matrix $\mathbf{L}$ in NPKL, we adopt the cover tree data structure.[2] The sparse eigen-decomposition used in SimpleNPKL is implemented by the popular *Arpack* toolkit.[3] We also adopt the standard SDP solver, SDPT3,[4] as the baseline solution for NPKL. The pair-wise constraint is assigned for randomly generated pairs of points according to their ground truth labels. The number of constraints is controlled by the resulted amount of connected components as defined in previous studies [XNJR02, HJL07]. Note that typically the larger the number of constraints, the smaller the number of connected components.

Several parameters are involved in both NPKL and SimpleNPKL. Their notation and settings are given as follows:

[2]The cover tree data structure is described at `http://hunch.net/~jl/projects/cover_tree/cover_tree.html`.

[3]The *Arpack* toolkit can be found at `http://www.caam.rice.edu/software/ARPACK/`.

[4]SDPT3 can be found at `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

- $k$ : The number of nearest neighbors for constructing the graph Laplacian matrix $\mathbf{L}$, we set it to 5 for small data sets in Table 3.1, and 50 for Adult database in Table 3.6;
- $r$ : The ratio of the number of connected components compared with the data set size $N$. In our experiments, we set $r \approx 70\%N$ which follows the setting of [HJL07];
- $B$ : The parameter that controls the capacity of the learned kernel in (3.5). We fix $B = N$ for the adult data sets and fix $B = 1$ for the data sets in Table 3.1 and;
- $C$ : The regularization parameter for the loss term in NPKL and SimpleNPKL. We fix $C = 1$ for the adult data sets and several constant values in the range (0, 1] for the data sets in Table 3.1.

In our experiments, all clustering results were obtained by averaging the results from 20 different random repetitions. All experiments were conducted on a 32bit Windows PC with 3.4GHz CPU and 3GB RAM.

### 3.5.2 Comparisons on Benchmark Data Sets

To evaluate the clustering performance, we adopt the clustering accuracy used in [HJL07]:

$$\text{Cluster Accuracy} = \sum_{i>j} \frac{\mathbf{1}\{c_i = c_j\} = \mathbf{1}\{\hat{c_i} = \hat{c_j}\}}{0.5n(n-1)}.$$

This metric measures the percentage of data pairs that are correctly clustered together. We compare the proposed SimpleNPKL algorithms with NPKL on the nine data sets from UCI machine learning repositories,[5] as summarized in Table 3.1. The same data sets were also adopted in the NPKL study of [HJL07].

The clustering accuracy and CPU time cost (the clustering time was excluded) of different NPKL methods are reported in Table 3.2 and 3.3. As can be observed from Table 3.2, all NPKL methods outperform the baseline $k$-means clustering and the constrained $k$-means clustering methods, which use Euclidean metric for $k$-means clustering. The proposed SimpleNPKL with square hinge loss produces very competitive clustering performance to the results of NPKL with hinge loss (as reported in [HJL07]). SimpleNPKL with square hinge loss and NPKL with hinge loss often perform better than the NPKL methods using linear loss.

For the CPU time cost, the time costs of SimpleNPKL and NPKL using linear loss are usually lower than those of their counterparts with (square) hinge loss. Regarding the efficiency evaluation in Table 3.3, our SimpleNPKL with linear loss or squared hinge loss is about 5 to 10 times faster than NPKL using the SDPT3 solver. For some cases of linear loss, SimpleNPKL can be even 100 times faster.

| Data Set | #Classes | #Instances | #Features |
|---|---|---|---|
| Chessboard | 2 | 100 | 2 |
| Glass | 6 | 214 | 9 |
| Heart | 2 | 270 | 13 |
| Iris | 3 | 150 | 4 |
| Protein | 6 | 116 | 20 |
| Sonar | 2 | 208 | 60 |
| Soybean | 4 | 47 | 35 |
| Spiral | 2 | 100 | 3 |
| Wine | 3 | 178 | 12 |

Table 3.1: The statistics of the data sets used in our experiments.

| Data Set | $k$-means | c$k$-means | NPKL | | SimpleNPKL | |
|---|---|---|---|---|---|---|
| | | | LL | HL | LL | SHL |
| Chessboard | 49.8 ±0.2 | 50.1±0.3 | **61.1± 6.9** | 56.3± 6.1 | 60.2± 0.0 | 58.8± 0.8 |
| Glass | 69.7 ±1.9 | 69.2±1.7 | 74.4± 3.7 | **79.1± 4.9** | 73.0± 2.5 | 73.5± 2.9 |
| Heart | 51.5 ±0.1 | 52.3±3.7 | 86.0± 0.3 | 86.2± 0.0 | 86.8± 0.0 | **89.4± 0.1** |
| Iris | 84.5 ±6.5 | 89.4±8.5 | 96.0± 6.1 | **97.4± 0.0** | **97.4± 0.0** | **97.4± 0.0** |
| Protein | 76.2 ±2.0 | 80.7±3.1 | 78.2± 3.2 | **86.4± 3.8** | 81.8± 1.8 | 75.9± 2.0 |
| Sonar | 50.2 ±0.1 | 50.8±0.2 | 76.8± 0.3 | 64.5± 6.8 | 70.2± 10 | **78.0± 0.0** |
| Soybean | 82.1 ±6.1 | 83.8±8.3 | 90.2± 7.7 | **100.0± 0.0** | 95.3± 5.1 | 95.4± 4.9 |
| Spiral | 50.1 ±0.6 | 50.6±1.3 | 86.5± 0.0 | **94.1± 0.0** | 92.2± 0.0 | **94.1± 0.0** |
| Wine | 71.2 ±1.2 | 76.1±2.8 | 78.1± 1.7 | **85.5± 5.3** | 83.7± 4.8 | 85.0± 2.6 |

Table 3.2: Clustering accuracy of SimpleNPKL, compared with the results of NPKL in (3.4) using a standard SDP solver, and $k$-means.

Recall that our key Proposition 3.26 provides a closed-form solution to the learned kernel matrix $\mathbf{K}$ for $p \geq 1$, in which the capacity parameter $B$ can be omitted for SimpleNPKL+linear loss. To show the influence of the capacity parameter $B$ for SimpleNPKL + square hinge loss, we present some results in Table 3.4 with a fixed $p = 2$. To clearly show the influence on convergence, we present the number of iterations instead of elapsed CPU time. We observe that SimpleNPKL + square hinge loss is not sensitive to $B$ on the both *Iris* and *Protein* data sets. It even produces the identical accuracy on the *Iris* data set for $B \in \{2.5, 3, 3.5, 4\}$. However, it affects the number of steps it takes to converge. Similar phenomena can be observed on other data sets.

We also study the clustering performance of varying $p$ in Table 3.5. We fixed $B = 1$ in this experiment. From Table 3.5, we can observe that SimpleNPKL+square hinge loss produces the best clustering accuracy for the *Iris* data set when $p = 4$, but the

[5]The data sets are available at `http://archive.ics.uci.edu/ml/`.

| Data Set | NPKL | | SimpleNPKL | | Speedup |
|---|---|---|---|---|---|
| | LL | HL | LL | SHL | |
| Chessboard | 1.38±0.07 | 5.23±0.06 | **0.05±0.00** | 0.13±0.00 | 27.6 |
| Glass | 1.85±0.04 | 32.36±0.37 | **0.11±0.00** | 2.95±0.00 | 16.8 |
| Heart | 2.64±0.10 | 63.84±0.68 | **0.17±0.01** | 13.15±0.08 | 15.5 |
| Iris | 1.36±0.03 | 1.65±0.04 | **0.04±0.00** | 3.45±0.01 | 34.0 |
| Protein | 1.80±0.06 | 8.16±0.11 | **0.05±0.00** | 1.32±0.00 | 36.0 |
| Sonar | 1.77±0.08 | 30.38±0.24 | **0.11±0.00** | 3.91±0.03 | 16.1 |
| Soybean | 1.51±0.05 | 3.25±0.04 | **0.01±0.00** | 0.16±0.00 | 151.0 |
| Spiral | 1.78±0.10 | 6.23±0.08 | **0.05±0.00** | 1.95±0.00 | 36.6 |
| Wine | 2.54±0.04 | 30.91±1.30 | **0.09±0.00** | 1.53±0.01 | 28.2 |

Table 3.3: CPU time of SimpleNPKL, compared with the results of NPKL in (3.4) using a standard SDP solver. (The best results are in bold and the last "Speedup" column is listed only for the linear loss case.)

| DataSet | $B$ | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 |
|---|---|---|---|---|---|---|---|---|
| Iris | Accur.(%) | 94.8±0.0 | 94.8±0.0 | 95.2±0.4 | **95.7±0.0** | 95.7±0.0 | 95.7±0.0 | 95.7±0.0 |
| | #Iterations | 11 | 13 | 14 | **10** | 10 | 31 | 26 |
| Protein | Accur.(%) | **74.5±0.8** | 73.6±1.6 | 74.4±0.8 | 74.3±0.9 | 74.1±1.0 | 73.7±1.1 | 73.7±1.0 |
| | #Iterations | 51 | 32 | 51 | **11** | 14 | 27 | 19 |

Table 3.4: Results of varying capacity parameter $B$ with fixed $p = 2$ and $C = 1$ on *Iris* and *protein* data sets.

improvement is not significant comparing with $p = 2$. For the *Protein* data set, our algorithm achieves the best results when $p = 2$. In general, when $p < 2$, the clustering performance drops significantly.

### 3.5.3 Scalability Study on Adult Data Set

In this Section, we evaluate our SimpleNPKL algorithms on another larger data set to examine the efficiency and scalability. We adopt the *Adult* database, which is available

| Data Set | $p$ | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 |
|---|---|---|---|---|---|---|---|---|
| Iris | Accur.(%) | 61.6±3.5 | 58.6±4.0 | 94.8±0.0 | 94.8±0.0 | 95.1±0.4 | 94.8±0.0 | **95.6±0.2** |
| | #Iterations | 51 | **6** | 11 | 9 | 19 | 10 | 9 |
| Protein | Accur.(%) | 72.3±1.3 | 72.8±2.2 | **74.5±0.8** | 73.6±1.5 | 73.6±1.6 | 73.5±1.6 | 73.5±1.6 |
| | #Iterations | 32 | 35 | 51 | 40 | **11** | **11** | 21 |

Table 3.5: Results of varying $p$ in the $p$-norm regularization over $\mathbf{K}$ with fixed $B = 1$ and $C = 1$ on *Iris* and *protein* data sets.

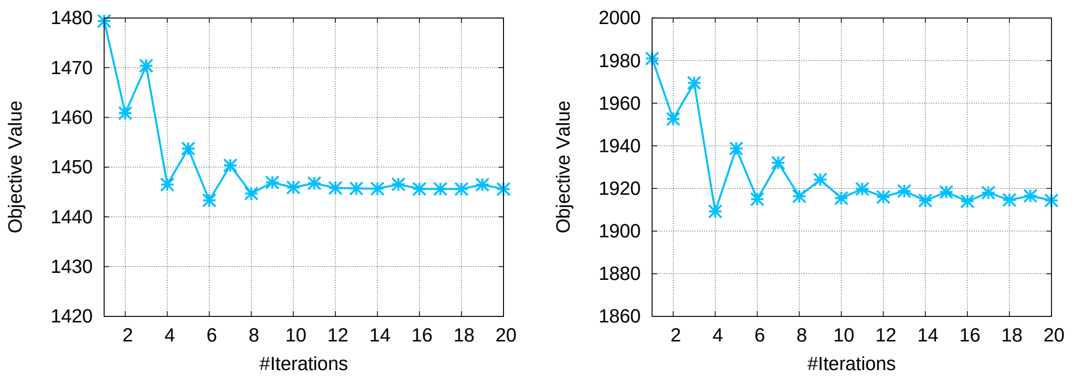
| Data Set† | A1a | A2a | A3a | A4a | A5a |
|---|---|---|---|---|---|
| #Instances | 1,605 | 2,265 | 3,185 | 4,781 | 6,414 |

†: #Classes=2, #Features=123

Table 3.6: The statistics of the *Adult* database.

| Data Set | #Constraints | Accuracy(%) | | | | CPU Time(s) | |
|---|---|---|---|---|---|---|---|
| | | $k$-means | $ck$-means | SimpleNPKL | | SimpleNPKL | |
| | | | | LL | SHL | LL | SHL |
| A1a | 4,104 | 56.4±3.5 | 59.0±2.3 | **61.4±1.7** | 60.7±2.7 | **8.5** | 322.9 |
| A2a | 5,443 | 57.3±3.6 | 60.2±0.1 | 61.1±1.3 | **61.4±1.2** | **15.3** | 637.2 |
| A3a | 7,773 | 57.8±3.5 | 59.2±3.0 | 61.1±1.7 | **61.5±2.0** | **28.8** | 1,160.8 |
| A4a | 12,465 | 58.8±1.6 | 59.3±3.9 | **61.6±1.3** | 61.4±1.5 | **66.3** | 2,341.3 |
| A5a | 16,161 | 57.7±3.1 | 59.8±2.2 | 60.8±3.1 | **61.9±1.7** | **79.6** | 3,692.1 |

Table 3.7: Evaluation results on *Adult* data set. (The best results are in bold.)



(a) A1a (b) A2a

Figure 3.1: Convergence of SimpleNPKL using square hinge loss on *A1a* and *A2a*. The parameters are $C = 1$, $B = N$.

at the website of LibSVM.[6] The database has a series of partitions: A1a, A2a, $\cdots$, A5a (see Table 3.6). Since the training time complexity of NPKL using standard SDP solvers is $O(N^{6.5})$, which cannot be applied on this database for comparison. We only report the results of both $k$-means and constrained $k$-means clustering as the baseline comparison.

Table 3.7 shows the clustering performance and CPU time cost (the clustering time was excluded) of SimpleNPKL on the *Adult* database. From the results, we can draw several observations. First of all, we can see that by learning better kernels from pairwise constraints, both SimpleNPKL algorithms produce better clustering performance than that of $k$-means clustering and constrained $k$-means clustering methods using Euclidean

[6]LibSVM can be found at `http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/`.

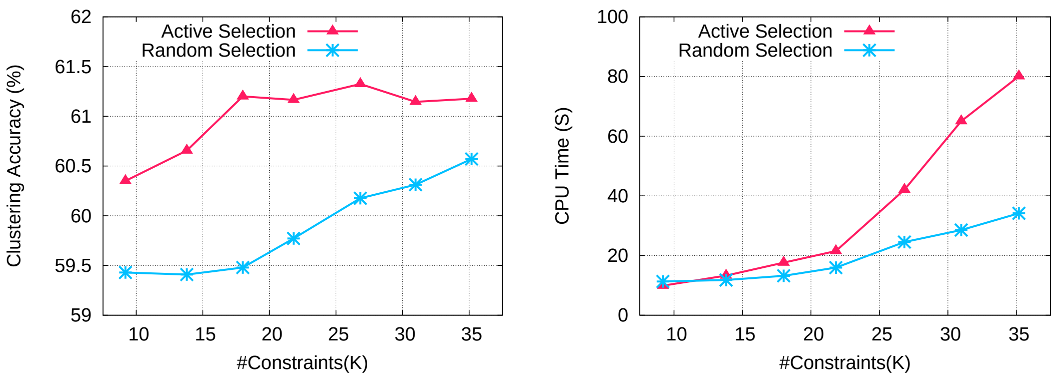
metric. Further, comparing the two algorithms themselves, in terms of clustering accuracy, they perform quite comparably, in which SimpleNPKL+SHL outperforms slightly. However, in terms of CPU time cost, SimpleNPKL+LL with linear loss is considerably lower than SimpleNPKL+SHL using square hinge loss.

We also plot the objective value $J(K,\boldsymbol{\alpha})$ of SimpleNPKL on two data sets *A1a* and *A2a* in Figure 3.1. We observe that SimpleNPKL with square hinge loss often converges quickly within 10 iterations. Similar results can be observed from the other data sets.

### 3.5.4 Comparisons on Constraint Selection

In this Section, we study the active constraint selection scheme for SimpleNPKL. Figure 3.2 shows the clustering performance of active constraint selection by the approach described in Section 3.3.3.

Several observations can be drawn from the results: 1) Comparing with the original approach using all constraints, the computation time is reduced by choosing a small amount of pairwise constraints. This is because the Lanczos algorithm can perform the sparse eigen-decomposition faster on a sparse matrix with fewer nonzero entries; 2) Though the active constraint selection costs more time than random selection, the former usually achieves better clustering (accuracy) performance than the latter with the same amount of constraints; 3) Using the proposed active constraint selection method to choose about half of the pairwise constraints for SimpleNPKL can often produce comparable or even better clustering performance than that using all constraints.



(a) Clustering Accuracy (b) CPU Time

Figure 3.2: Comparisons of clustering accuracy and CPU time by active constraint selection and random selection (constraint selection time is included) on A1a with parameters: $B = N, C = 1, k = 20, r = 0.6$. Using all $3.9K$ constraints directly, the accuracy is $60.8 \pm 2.9$ and the CPU time is 81.6 seconds.

### 3.5.5 Evaluations on Data Embedding Applications

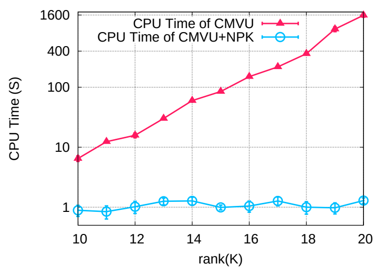
In this Section, we evaluate the performance of the proposed SimpleNPKL algorithms with applications to speed up three data embedding techniques, that is, CMVU, MVE, and SPE, respectively. Our goal is to show that SimpleNPKL is capable of producing similar empirical results to the baseline counterpart with significant efficiency gain. All the data sets are publicly available in the UCI machine learning repository. In all the experiments, we simply fix $C = 1$ for all the three methods, and set $B = m \times N$, $m \in \{0.1, 1, 2, 10\}$.

#### 3.5.5.1 Colored Maximum Variance Unfolding

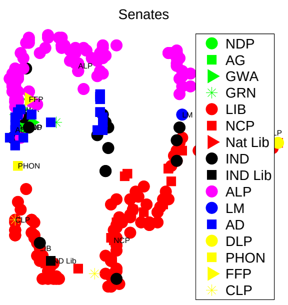
The first experiment is to examine the efficiency by applying the proposed SimpleNPKL technique to solve the CMVU problem. In particular, we examine the CMVU task for learning low-dimensional embedding on three data sets which were used in [SSBG07]. Two approaches are compared:

- **CMVU**: An approximate efficient method employed by [SSBG07]. Suppose $\mathbf{K} = \mathbf{VAV}'$, where $\mathbf{V}$ (of size $n \times d$, $d < n$) is fixed to be the bottom $d$ eigenvectors of the graph Laplacian of the neighborhood graph via $\mathcal{N}$. Thus the number of variables is reduced from $n^2$ to $d^2$.
- **CMVU+NPKL**: Our SimpleNPKL method introduced in Section 3.4.1. Unlike the above CMVU algorithm by approximation, our method is able to obtain the global optimal solution using the SimpleNPKL scheme without approximation.

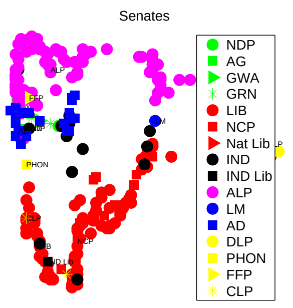
Figure 3.3, 3.4 and 3.5 show the experimental results of visualizing the embedding results in a 2D space and the CPU time cost of CMVU. The time costs of CMVU+NPKL were also indicated in the captions of those figures. As we can observe from the visualization results, the proposed CMVU+NPKL is able to produce comparable embedding results as those by the original CMVU in most cases. Further, by examining the time cost, we found that the time cost of CMVU increases with dimensionality $d$ exponentially due to the intensive computational cost of solving the SDP problem. In contrast, the proposed CMVU+NPKL is able to find the global optima efficiently, which is much faster than CMVU when $d$ is large. Although CMVU could be faster than CMVU+NPKL for very small $d$ values, it is important to note that the optimal $d$ value is often unknown for many applications. The proposed CMVU+NPKL approach can efficiently and directly resolve the CMVU problem without soliciting the approximation step.



3.3.a: Time cost of CMVU with the rank.

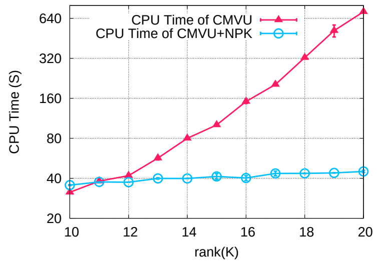


3.3.b: Embedding of CMVU.

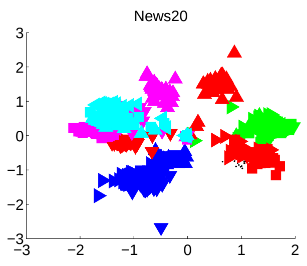


3.3.c: Embedding of CMVU+NPKL.

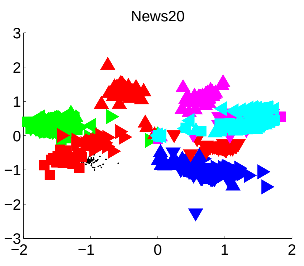
Figure 3.3: Comparisons of CMVU and CMVU+NPKL on *senate* data set. **Time cost of CMVU+NPK is** $1.50 \pm 0.06$ **seconds.**



3.4.a: Time cost of CMVU with the rank.



3.4.b: Embedding of CMVU.



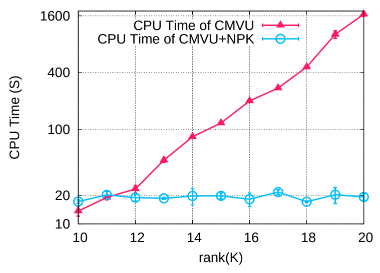
3.4.c: Embedding of CMVU+NPKL.

Figure 3.4: Comparisons of CMVU and CMVU+NPK on *news20* data set. **Time cost of CMVU+NPKL is** $120.4 \pm 1.7$ **seconds.**

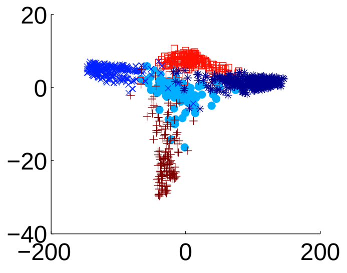
#### 3.5.5.2 Minimum Volume Embedding and Structure Preserving Embedding

This experiment is to examine the embedding performance of the SimpleNPKL technique with applications to MVE [SJ07] and SPE [SJ09] tasks. In particular, five approaches are compared:

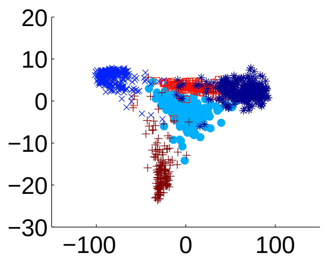
- **KPCA**: The classical Kernel Principle Component Analysis algorithm;
- **MVE**: The algorithm summarized in Table 1 in [SJ07]. Pay attention to the SDP solver in Step 5 and 6, which is the key for the success of MVE.
- **MVE+NPKL**: The embedding algorithm based on our SimpleNPKL algorithm. Refer to Section 3.4.2 for detailed discussion.
- **SPE**: The algorithm summarized in Table 1 of [SJ09].



3.5.a: Time cost of CMVU with the rank.



3.5.b: Embedding of CMVU.



3.5.c: Embedding of CMVU+NPKL.

Figure 3.5: Comparisons of CMVU and CMVU+NPKL on *usps* data set. **Time cost of CMVU+NPKL is $28.95 \pm 1.8$ seconds.**

- **SPE+NPKL**: The embedding algorithm based on the proposed SimpleNPKL algorithm. Refer to Section 3.4.3;

To examine the embedding results quantitatively, we follow the previous studies [SJ07, SJ09] to evaluate the classification performance on the embedding data by performing k-nearest neighbor classification. Similar to the settings in [SJ09], we randomly choose 100 points from the largest two classes for each data set, and then divide the data examples into training/validation/test sets at the ratios of 60:20:20. The validation set is used to find the best parameter of $k$ for $k$-NN classification.

Table 3.8 shows the comparison of the classification results by five different approaches. From the results, we can see that the two proposed algorithms, MVE+NPKL and SPE+NPKL, are generally able to achieve the competitive classification results that are comparable to the other two original algorithms using a standard SDP solver. Among all compared algorithms, MVE+NPKL tends to achieve slightly better performance than the other approaches. All these results show that the proposed algorithms are effective to produce comparable embedding performance.

Next we compare the computational cost of the proposed algorithms against their original methods, respectively. Table 3.9 shows the summary of average CPU time cost of the compared algorithms. From the results, it is clear to see that the two proposed algorithms, MVE+NPKL and SPE+NPKL, are significantly more efficient than the other two original algorithms, respectively. By comparing MVE and MVE+NPKL, we found that MVE+NPKL achieves about 10 to 30 times speedups over the original MVE algorithm; the speedup values are even more significant for the SPE problem, where the proposed SPE+NPKL algorithm attains about 50 to 90 times speedup over the original SPE algorithm. These promising results again validate the proposed SimpleNPKL is effective for improving the efficiency and scalability of the three data embedding tasks.

| Data Set | KPCA | MVE | MVE+NPKL | SPE | SPE+NPKL |
|---|---|---|---|---|---|
| Wine | 90.5 ±5.6 | **91.9 ±6.6** | 90.9±5.8 | 75.2 ±0.09 | 87.1 ±7.9 |
| Ionosphere | 79.8±7.3 | **86.3 ±7.3** | 84.2±8.5 | 80.4 ±10.4 | 83.6 ±7.8 |
| Heart | **65.6 ±8.4** | 62.4 ±9.8 | 62.9 ±9.8 | 54.9 ±10.2 | 62.2 ±11.1 |
| Sonar | 58.2 ±12.4 | 59.2 ±10.2 | **59.8 ±12.2** | 57.4 ±11.1 | 59.4 ±11.4 |
| Glass | 70.7 ±9.8 | 73.5 ±7.8 | **74.5 ±10.4** | 61.7 ±9.7 | 69.4 ±8.7 |
| Spiral | 98.7 ±2.4 | 69.1 ±9.8 | **98.8 ±2.4** | 76.7 ±0.07 | 82.9 ±8.4 |
| Australian | 63.2 ±9.8 | 61.3 ±8.2 | **63.8 ±9.3** | 60.1 ± 0.10 | 59.5 ±10.1 |
| Breast cancer | 91.9 ±5.4 | 92.9 ±4.6 | 92.4 ±5.8 | 93.4 ±0.07 | **94.4 ±5.5** |

Table 3.8: $k$-NN classification accuracy on the 2D embedded results. (The best results are bolded.)

| Data Set | MVE | MVE+NPKL | SpeedUp | SPE | SPE+NPKL | SpeedUp |
|---|---|---|---|---|---|---|
| Wine | 2.92 ±0.06 | **0.34 ±0.04** | 8.5 | 47.72 ±0.29 | 0.51 ±0.01 | 93.6 |
| Ionosphere | 16.98 ±0.16 | 1.14 ±0.01 | 14.9 | 30.07 ±1.25 | **0.60 ±0.01** | 50.1 |
| Heart | 9.64 ±0.00 | **0.38 ±0.02** | 25.3 | 48.18 ±0.31 | 0.51 ±0.11 | 94.5 |
| Sonar | 7.50 ±0.13 | **0.46 ±0.01** | 16.3 | 30.40 ±1.16 | 0.61 ±0.02 | 49.8 |
| Glass | 11.08 ±0.26 | **0.39 ±0.01** | 28.2 | 29.10 ±0.12 | 0.53 ±0.01 | 54.9 |
| Spiral | 18.28 ±0.28 | **0.46 ±0.00** | 39.7 | 47.91 ±0.91 | 0.48 ±0.01 | 99.8 |
| Australian | 4.61 ±0.03 | **0.30 ±0.02** | 15.4 | 28.94 ±0.11 | 0.53 ±0.01 | 54.6 |
| Breast cancer | 16.59 ±0.10 | **0.49 ±0.02** | 33.9 | 48.72 ±0.26 | 0.56 ±0.01 | 87.0 |

Table 3.9: The evaluation of CPU time cost of different algorithms and the speedup of the SimpleNPKL method over the standard SDP solver. (The best results are bolded.)

To further illustrate the scalability of SPE+NPKL, we propose to solve a real-world embedding task on a large data set. In particular, we crawled a Flickr[7] data set, which consists of 3,660 Flickr user profiles and a collection of 3.7 million photos uploaded by these users. Each photo was annotated with a set of textual tags by users. Accordingly the photos for a particular Flickr user are described by tiling these tags. In total, our data set has 359,832 tags and 93,692 unique tags. Each Flickr user has a contact list, which is a collection of Flickr users who may share similar tastes / interests in their photo sharing. In our data set, every user has 19.1 contacts on average. We thus set $|\mathcal{N}|$ to 20 in both MVE and SPE. Moreover, there are $97,212$ interest groups, and each Flickr user could belong to one or more interest groups. We compute *tf-idf* weight for the tags to represent a Flickr user (here the document frequency for a tag is actually the number of users annotated with that tag, that is, one or more photos of this user annotated with the tag). The k-nearest neighbor graph for MVE is constructed using cosine similarity between Flickr users. For SPE, we further constrain that the intra-group distance is smaller than the inter-group distance. In general, people who are friends or similar to each other tend to join the same interest group. Our goal is to apply the proposed MVE+NPKL and SPE+NPKL algorithms on these Flickr users in order to draw the 2D embedding results of the Flickr users exclusively belonging to two different interest groups: *B&W*[8] and *Catchy Colors*[9] as shown in Figure 7.

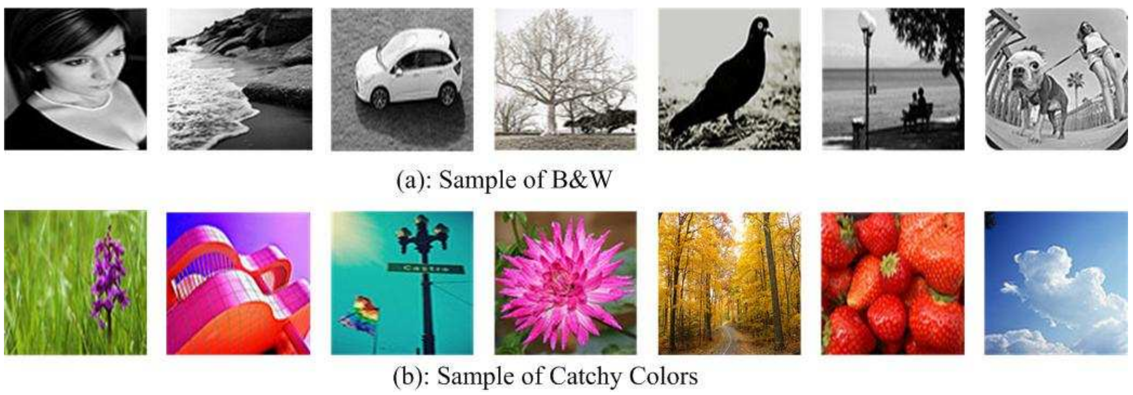
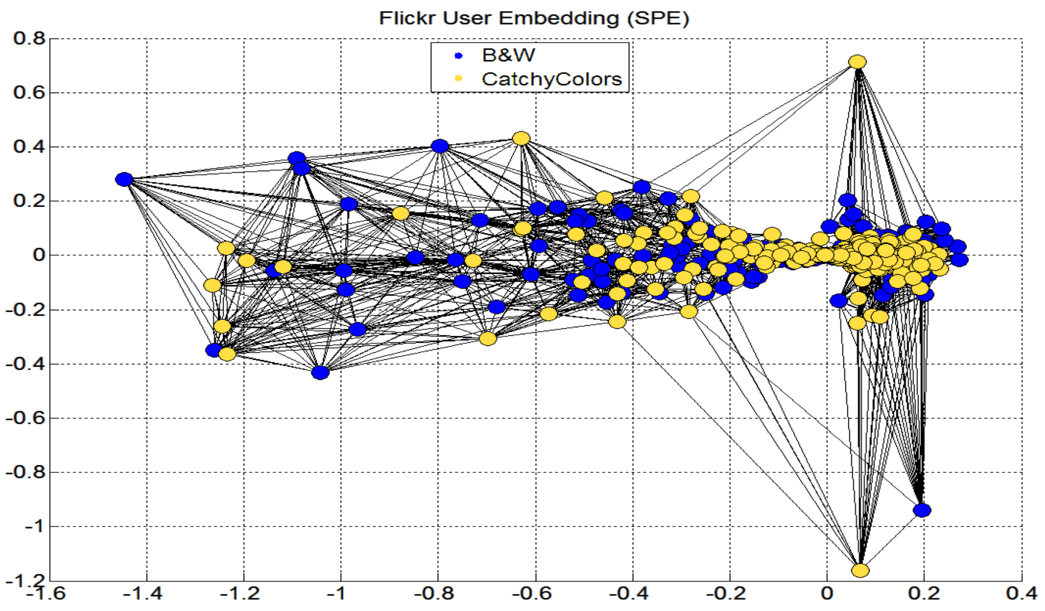


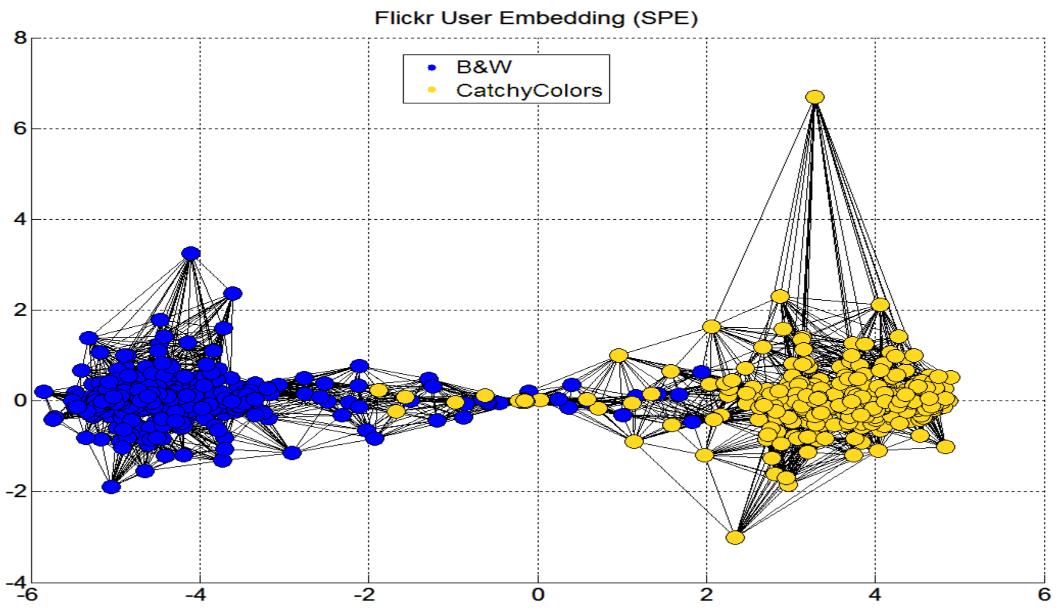
Figure 3.6: Sample photos from two Flickr interest groups: *B&W* and *Catchy Colors*.

Specifically, the theme of the group B&W is related to a collection of photos with black and white color only. The corresponding top 5 annotated tags for B&W are {*bw, black and white, black, white, portrait*}. In contrast, the top 5 tags for CatchyColors include {*red, blue, green, flower, yellow*}. Therefore, photos in the latter group are more

[7]Flickr can be found at `http://www.flickr.com/`.
[8]*B&W* can be found at `http://www.flickr.com/groups/blackwhite/`.
[9]*Catchy Colors* can be found at `http://www.flickr.com/groups/catchy/`.



3.7.a: MVE+NPKL



3.7.b: SPE+NPKL

Figure 3.7: The 2D embedding result of Flickr users exclusively belonging to the interest group *B&W* (blue points) and *Catchy Colors* (red points). The CPU time cost of MVE+NPKL and SPE+NPKL are 27.2 minutes and 196.4 minutes, respectively.

colorful than the ones in B&W. An illustration of photos belonging to these two groups are depicted in Figure 6. However, the semantics of photos of these two groups are highly overlapping. Accordingly, the embedding results of MVE are highly overlapped as shown in Figure 7 (a), though it drives the spectral information into the top eigenvectors of the learned kernel matrix. On the other hand, by constraining intra-group distance less that inter-group distance, SPE can preserve the topology structure of these two groups as shown in Figure 7 (b). The 2D embedding shows the cluster structure rather clearly.

Note that the original algorithms using general SDP solvers cannot directly apply on the above large real data set. The proposed SimpleNPKL framework makes it feasible to analyze the emerging social networking problems using kernel learning methods. We hope our preliminary attempt in this chapter could shed a light on a series of important applications in the future, including: 1) **Visualization:** as illustrated in Figure 7, we are able to obtain an intuitive understanding about the distribution of the entities in a social networking community. From Figure 7 (b), one can also observe the abnormal entities (e.g., the red dot on the right upper corner) and prototypes (the ones located at the centers of clusters). This may also benefit spam user detection and important user identification applications; 2) **Friend suggestion:** Given a Flickr user $U_i$, we can rank the other users $U_j$ according to their similarity to $U_i$ computed by the learned non-parametric kernel $K_{ij}$. With such information, a user can quickly find the other users of similar interests/tastes in photo sharing so as to facilitate the social communication between the users; 3) **Interest group recommendation:** It is interesting and beneficial to develop an intelligent scheme for recommending a Flickr user some interest groups. By applying the proposed kernel learning techniques to find similarity between Flickr users, it is possible for us to develop some recommendation scheme that suggests a Flickr user some interest groups that received the highest numbers of votes from its neighbors.

## 3.6 Summary

In this chapter, we investigated a family of SimpleNPKL algorithms for improving the efficiency and scalability of the Non-Parametric Kernel Learning (NPKL) from large sets of pairwise constraints. We demonstrated that the proposed SimpleNPKL algorithm with linear loss for the pairwise constraints enjoys a closed-form solution, which can be simply computed by efficient sparse eigen-decomposition, such as the Lanczos algorithm. Moreover, our SimpleNPKL algorithm using other loss functions (including square hinge loss, hinge loss, and square loss) can be transformed into a saddle-point minimax optimization problem, which can be solved by an efficient iterative optimization procedure that only involves sparse eigen-decomposition computation. In contrast to the previous standard SDP solution, empirical results show that our approach achieved the same/comparable accuracy, but is significantly more efficient and scalable for large-scale data sets. We also explore some active constraint selection scheme to reduce the pairwise constraints in SimpleNPKL, which can further improve both computational efficiency and the clustering performance. Finally, we also demonstrate that the proposed family of SimpleNPKL algorithms can be applicable to other similar machine learning problems, in which we studied three example applications on data embedding problems. In the future, we will extend our technique for solving other SDP related machine learning problems.

# Chapter 4

# Deep Multiple Kernel Learning

In chapter 2, we introduced related works on multiple kernel learning (MKL) algorithms and theories. Now we investigate two novel directions towards improving MKL: Deep MKL (DMKL, [ZHT11]) and Unsupervised MKL (UMKL, submitted to ACML'11). The former one is inspired by the recent development of deep learning[HOT06][Ben09][CS09]. We embed the summation of base kernels into a single (series of) base kernels for next layer of MKL. We introduce the framework of DMKL in this chapter and UMKL in chapter 5.

Multiple Kernel Learning (MKL) aims to learn kernel machines for solving a real machine learning problem (e.g. classification) by exploring the combinations of multiple kernels. The traditional MKL approach is in general "shallow" in the sense that the target kernel is simply a linear (or convex) combination of some base kernels. In this chapter, we investigate a framework of Multi-Layer Multiple Kernel Learning (MLMKL) that aims to learn "deep" kernel machines by exploring the combinations of multiple kernels in a multi-layer structure, which goes beyond the conventional MKL approach. Through a multiple layer mapping, the proposed MLMKL framework offers higher flexibility than the regular MKL for finding the optimal kernel for applications. As the first attempt to this new MKL framework, we present a Two-Layer Multiple Kernel Learning (2LMKL) method together with two efficient algorithms for classification tasks. We analyze their generalization performances and have conducted an extensive set of experiments over 16 benchmark datasets, in which encouraging results showed that our method performed better than the conventional MKL methods.

Despite being studied actively, unfortunately existing MKL methods do not always produce considerably better empirical performance when comparing with a single kernel whose parameters were tuned by cross validation [GN08]. There are several possible reasons to account for such failure. One conjecture is that the target kernel domain $\mathcal{K}$ of MKL using a linear combination may not be rich enough to contain the optimal kernel. Therefore, some emerging study has attempted to improve the regular MKL by explore more general kernel domain $\mathcal{K}$ using some nonlinear combination [VB09]. Following the similar motivation, we speculate that the insignificant performance gain is probably due to the *shallow* learning nature of regular MKL that simply adopts a flat (linear/nonlinear) combination of multiple kernels.

To this end, this chapter presents a novel framework of Multi-Layer Multiple Kernel Learning (MLMKL), which applies the idea of deep learning to improve the MKL task. Deep architecture has been being actively studied in machine learning community and has shown promising performance for some applications [HOT06, HS06, LEC+07, Ben09]. Our study was partially inspired by the recent work [CS09] that first explored kernel methods with the idea of deep learning. However, unlike the previous work, our study in this chapter mainly aims to address the challenge of improving the existing MKL techniques with deep learning. Specifically, we introduce a multilayer architecture for MLMKL, in which all base kernels in antecedent-layers are combined to form some inputs to other kernels in subsequent layers. We also provide an efficient alternating optimization algorithm to learn the decision function and the weight of base kernels simultaneously. To further minimize the requirement of domain knowledge to design the choices and the numbers of base kernels in each antecedent-layers, we also present an infinite base kernel learning algorithm for our proposed MLMKL framework.

## 4.1 Deep Learning and Multilayer Kernels

Recently, a lot of machine learning studies have addressed one limitation of conventional learning techniques (such as SVM) regarding their shallow learning architectures. It has been shown that the deep architecture, such as multilayer neural nets, is often more preferable over the shallow ones. Very recently, Cho and Saul [CS09, CS10] first introduced the idea of deep learning to kernel methods, which can be applied either in deep architectures or in shallow structures, like SVM. An $l$-layer kernel is the inner product after multiple feature mapping of inputs:

$$k^{(l)}(\mathbf{x}_i, \mathbf{x}_j) = \big\langle \underbrace{\Phi(\Phi(\ldots(\Phi(\mathbf{x}_i))))}_{l \text{ times}}, \underbrace{\Phi(\Phi(\ldots(\Phi(\mathbf{x}_i))))}_{l \text{ times}} \big\rangle,$$

here $\Phi$ is the underlying feature mapping function of $k$, $\langle\cdot,\cdot\rangle$ computes the inner product.

Specifically, we consider an example of two-layer RBF kernel. An RBF kernel is typically defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2},$$

where $\gamma > 0$ is the kernel parameter. By applying the idea of two-layer kernel with the RBF kernel, the composition yields

$$\begin{aligned}&\big\langle \Phi(\Phi(\mathbf{x}_i)), \Phi(\Phi(\mathbf{x}_j)) \big\rangle \\ &= e^{-\gamma\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2} = e^{-2\gamma(1 - k(\mathbf{x}_i, \mathbf{x}_j))} = \kappa e^{2\gamma k(\mathbf{x}_i, \mathbf{x}_j)},\end{aligned} \quad (4.1)$$

where $\kappa$ is a constant that can be omitted. The similar idea can be applied for other types of kernels. In [CS09, CS10], the authors provided a multiple layer composition approach with respect to a special family of arc-cosine kernel functions.

**Remark 4.30:** □

The work studied in [CS09] has some limitations. First, the proposed multi-layer kernel was applied to only a single type of kernel, typically some special kernel function, such as the arc-cosine kernel [CS09]. In a real application, a more desirable solution is to allow a combination of a variety of different kernels when designing the deep kernel. Second, the multi-layer kernel proposed in [CS09] is often "static", i.e., some fixed kernel (where degree $n$ and level $l$ parameters were chosen manually). No solution has been provided to optimize the kernel by learning the optimal parameters automatically. Our work was partially motivated by this work to address the above limitations.

## 4.2 Two-Layer Multiple Kernel Learning

In this Section, we first introduce a general framework of Multi-Layer Multiple Kernel Learning (MLMKL), and then present an MLMKL paradigm, i.e., Two-Layer Multiple Kernel Learning.

### 4.2.1 Framework

Following the optimization framework of MKL, the basic idea of MLMKL is to relax the optimization domain $\mathcal{K}$ in traditional MKL optimization by adopting a family of deep kernels. Specifically, we first define a domain of $l$-level multi-layer kernels as follows:

$$\mathcal{K}^{(l)} = \left\{ k^{(l)}(\cdot,\cdot) = g^{(l)}\big([k_1^{(l-1)}(\cdot,\cdot),\ldots,k_m^{(l-1)}(\cdot,\cdot)]\big)\right\},$$

where $g^{(l)}$ is some function to combine multiple $(l-1)$-level kernels, which must ensure the resulting combination is a valid kernel. With this domain, in a way similar to regular MKL, we can formulate the optimization problem of $l$-level MLMKL into:

$$\min_{k\in\mathcal{K}^{(l)}} \min_{f\in\mathcal{H}_k} \quad \lambda\|f\|_{\mathcal{H}_k} + \sum_{i=1}^{n} \ell(y_i f(\mathbf{x}_i)).$$

To explain it intuitively, Figure 1 illustrates the architecture of an example three-layer MKL paradigm.

Despite sharing the similar optimization form, MLMKL is much more challenging than the conventional shallow MKL. This is because there are many unknown structures and variables, including the initialization of base kernels, the unknown combination functions $g^{(l)}$ at each level, and the final prediction model $f$. Apparently, it is not possible to fully optimize every aspect. In the following, we attempt to attack this challenge by considering a simplified paradigm, i.e., Two-Layer Multiple Kernel Learning (2LMKL).

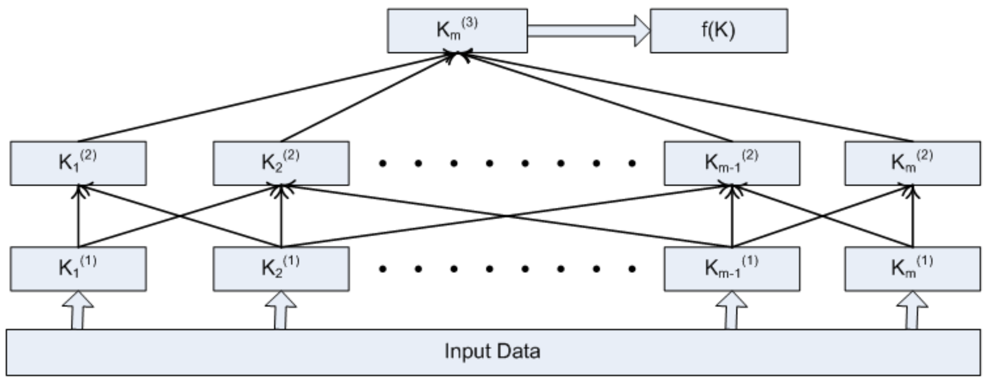


Figure 4.1: The architecture of the proposed deep multiple kernel learning framework. Here shows an example of three-layer MKL, and some connections are not displayed to simplify the figure.

### 4.2.2 Two-Layer Multiple Kernel Learning

To simplify the notations, we restrict our discussion on a Two-Layer Multiple Kernel Learning task in this Section. Our algorithm can also be extended to general multiple-layer MKL. Further, we employ an RBF kernel for the combination function $g^{(2)}$ and define the two-layer multiple kernel domain as follows:

$$\begin{aligned}\mathcal{K}^{(2)} = \Big\{ k^{(2)}(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\mu}) = \exp\left( \sum_{t=1}^{m} \mu_t k_t^{(1)}(\mathbf{x}_i, \mathbf{x}_j) \right) : \\ \boldsymbol{\mu} \in \mathbb{R}_+^m \Big\}, \end{aligned} \tag{4.2}$$

where $\mu_t$ denotes the weight of $t$-th antecedent-layer kernel. As aforementioned, the kernel in form of (4.2) is an inner product after two layer of feature mapping. Therefore, it is a PSD kernel. Thus we can formulate the two-layer MKL with the kernel $\mathcal{K}^{(2)}$:

$$\min_{k \in \mathcal{K}^{(2)}} \min_{f \in \mathcal{H}_k} \frac{1}{2} \|f\|_{\mathcal{H}_k}^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f(\mathbf{x}_i)) + \sum_{t=1}^{m} \mu_t \ .$$

Note the last term is introduced as a regularization to prevent the coefficients being too large. We can further turn the above formulation into the following equivalent min-max optimization:

$$\begin{aligned} &\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k^{(2)}(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\mu}) + \sum_{t=1}^{m} \mu_t \\ &\text{s.t. } 0 \le \alpha_i \le C, \sum_{i=1}^{n} \alpha_i y_i = 0, \mu_t \ge 0, t = 1, \ldots, m, \end{aligned} \tag{4.3}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top$ is the vector of dual variables and $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_m]^\top$. Once solving the above optimization to find the solutions for $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, it is straightforward to obtain the final decision function of the Two-Layer MKL machine:

$$f(\mathbf{x}; \alpha, \boldsymbol{\mu}) = \sum_{i=1}^{n} \alpha_i y_i k^{(2)}(\mathbf{x}_i, \mathbf{x}; \boldsymbol{\mu}) + b, \tag{4.4}$$

where the bias term $b$ can be easily determined from KKT conditions. Due to the nonlinearity of $\exp(\cdot)$ function, we expect that the decision function in (4.4) can represent richer prediction tasks than that in the RKHS induced from a one-layer kernel.

The next challenge is how to resolve the above optimization. We consider an alternative optimization scheme. That is, (1) fix $\boldsymbol{\alpha}$ and solve $\boldsymbol{\mu}$; and (2) fix $\boldsymbol{\mu}$ and solve $\boldsymbol{\alpha}$. Specifically, let us denote by $J(\boldsymbol{\alpha}, \boldsymbol{\mu})$ the following function:

$$J(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k^{(2)}(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\mu}) - \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{m} \mu_i.$$

Since $k^{(2)}$ is positive semi-definite, the objective is convex over $\boldsymbol{\alpha}$. Thus it can be solved by standard QP solvers for a fixed $\boldsymbol{\mu}$. However, for any pair $(i, j)$ of $y_i y_j = -1$, $\alpha_i \alpha_j y_i y_j k^{(2)}(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\mu}) \le 0$. Thus, $J$ is non-convex over $\boldsymbol{\mu}$. We simply compute the $d$-th component of the gradient w.r.t. $\boldsymbol{\mu}$:

$$\begin{aligned} \left[\nabla J_{\boldsymbol{\mu}^{t-1}}\right]_d &:= \left[\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^{t-1})\right]_d \\ &= \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k^{(2)}(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\mu}^{t-1}) k_d^{(1)}(\mathbf{x}_i, \mathbf{x}_j) - 1. \end{aligned} \tag{4.5}$$

Then we update $\boldsymbol{\mu}$ by gradient ascent:

$$\boldsymbol{\mu}^t = \max(\boldsymbol{\mu}^{t-1} + \eta \nabla J_{\boldsymbol{\mu}^{t-1}}, \mathbf{0}).$$

The step size $\eta$ can be set by Armijo's rule such that the convergence is guaranteed. Let $\Theta = \{\theta_1, \ldots, \theta_m\}$ denote the set of hyper-parameters corresponding to base kernels $k_t^{(1)}$'s inside $k^{(2)}$ in (4.2). Algorithm 2 shows the detailed optimization steps of the proposed two-layer MKL algorithm with the given $\Theta$.

## 4.3 Improve 2-layer MKL with Infinite Base Kernel Learning

Notice that, similar to traditional MKL algorithms, our proposed MLMKL algorithm also must assume a set of predefined base kernels inside $k^{(2)}$ as in $\mathcal{K}^{(2)}$ provided beforehand. If

**Protocol 2** Two-Layer MKL (2LMKL): $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ = TwoLayerMKLwithTheta$(\Theta_0; \mathbf{X}_1^n)$

**Input:** Training sample $\mathbf{X}_1^n$, initial set of base kernel parameters $\Theta_0 = \{\theta_1, \ldots, \theta_m\}$;
**Output:** weight vector $\boldsymbol{\mu}$ of base kernels, dual variables $\boldsymbol{\alpha}$ of SVM.

```
1: Randomly initialize $\boldsymbol{\mu}^0$, compute initial base kernels with $\Theta$;
2: repeat
3:     Compute the current kernel matrix with $\boldsymbol{\mu}^{t-1}$;
4:     $\boldsymbol{\alpha}^t = \arg\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}, \boldsymbol{\mu}^{t-1})$ by SVM solver;
5:     Compute $\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^{t-1})$ by (4.5) as descent direction;
6:     Determine the step size $\eta^t$ by Armijo's rule, update $\boldsymbol{\mu}^t = \max(\boldsymbol{\mu}^{t-1} + \eta^t \nabla J_{\boldsymbol{\mu}^{t-1}}, \mathbf{0})$;
7: until convergence
```

the number of base kernels is too small, $k^{(2)}$ may not be flexible enough to fit complicated patterns in a real problem. On the other hand, both the time cost and space cost of MKL/MLMKL increase with the cardinality of $\mathcal{K}_{conv}/\mathcal{K}^{(2)}$. This may be computationally inefficient when using too many base kernels. Moreover, though the proposed deep MKL architecture provides a flexibility for the design of multi-layer kernels, determining an appropriate set of base kernels usually require some domain knowledge, which may be difficult for some non-expert users.

To partially address some of the above challenges, here we propose to generate the base kernels inside $k^{(2)}$ iteratively. This can be done by selecting a base kernel that optimizes the objective function in (5.3), which is similar to the idea of infinite kernel learning [AMP05, GN08] except that our base kernels are in the antecedent layer. Assume the inner base kernel is continuously parameterized by $\theta$, for example, the bandwidth parameter of Gaussian kernel, or the degree of polynomial kernel. To expand the base kernel set $\mathcal{K}$, we choose a $\theta$ such that the resultant single kernel maximizes $J$ with the current solution $\boldsymbol{\alpha}$:

$$\max_{\theta \in \mathbb{R}_+} J(\boldsymbol{\alpha}, \theta) = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \exp(k(\mathbf{x}_i, \mathbf{x}_j; \theta)). \tag{4.6}$$

Again, this problem is non-convex over $\theta$. Similar to solving $\boldsymbol{\mu}$, we compute the gradient of (4.6) w.r.t. $\theta$ and do gradient ascent. For example, if the inner base kernel is a Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j; \theta) = \exp(-\theta \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, the gradient can be then computed as follows:

$$\nabla J_\theta = \frac{1}{2} \sum_{i,j \in \mathbb{N}_n} \alpha_i \alpha_j y_i y_j e^{(k(\mathbf{x}_i \mathbf{x}_j \theta))} k(\mathbf{x}_i, \mathbf{x}_j; \theta) \|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{4.7}$$

After that, we employ a line search approach to determining the step size for gradient ascent. The proposed improved two-layer MKL algorithm iterates between the following two steps: (1) iteratively solve the dual variables $\boldsymbol{\alpha}$ and the kernel weight $\boldsymbol{\mu}$ as similar to the previous algorithm; (2) add a new $\theta$ to $\Theta$ by the base kernel generation method. Finally, Algorithm 3 summarizes the details of the improved two-layer multiple kernel learning algorithm, which is denoted as 2LMKL$^{\text{Inf}}$ for short.

**Protocol 3** Infinite Two-Layer MKL (2LMKL$^{\text{Inf}}$): $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \Theta) = \text{TwoLayerMKL}(\Theta_0; \mathbf{X}_1^n)$

**Input:** Initial set of base kernel parameters $\Theta_0$, training sample $\mathbf{X}_1^n$;

**Output:** Final set of base kernel parameters $\Theta$, weight vector $\boldsymbol{\mu}$ of base kernels, dual variables $\boldsymbol{\alpha}$.

```
 1: Initialize $\Theta = \Theta_0$;
 2: while true do
 3:    $(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \text{TwoLayerMKLwithTheta}(\Theta; \mathbf{X}_1^n)$;
 4:    $\theta = \text{NewKernel}(\boldsymbol{\alpha}; \mathbf{X}_1^n)$;
 5:    if $J(\boldsymbol{\alpha}, \theta) \leq J(\boldsymbol{\alpha}, \boldsymbol{\mu})$ then
 6:       break;
 7:    end if
 8:    $\Theta = \Theta \cup \theta$;
 9: end while
10:
11: Function $\theta = \text{NewKernel}(\boldsymbol{\alpha}; \mathbf{X}_1^n)$;
12: Randomly initialize $\theta^0$;
13: while $J(\boldsymbol{\alpha}, \theta^{t-1})$ is improving do
14:    Compute the gradient $\nabla J_\theta$ by the similar approach in Eqn. (4.7);
15:    Determine a step size $\eta^t$, update $\theta^t = \theta^{t-1} + \eta^t \nabla J_\theta$;
16: end while
```

Table 4.1: The statistics of the 16 binary-class data sets used in our experiments.

| Data Set | Breast | Ionosphere | Diabetes | Waveform | Sonar | Adult | Liver | German |
|---|---|---|---|---|---|---|---|---|
| # instances | 683 | 351 | 768 | 400 | 208 | 1,605 | 345 | 1,000 |
| # dimensions | 10 | 33 | 8 | 21 | 60 | 123 | 6 | 24 |
| Data Set | Splice | Australian | Thyroid | Ringnorm | Heart | Banana | Titanic | FlareSolar |
| # instances | 1,000 | 690 | 140 | 400 | 270 | 400 | 150 | 666 |
| # dimensions | 60 | 14 | 5 | 20 | 13 | 2 | 3 | 9 |

## 4.4 Analysis of Generalization Performance

We are aware of the trend of seeking new kernel combination methods beyond the traditional MKL. However, the kernel is the prior knowledge of the data. The construction of kernel cannot be fully "automated". When we add more flexibility to kernel learning, we are also potentially increasing the difficulty of finding the optimal kernel. It calls for theoretical analysis on these generalized kernel combination methods. We base our analysis of two-layer MKL mainly on the notion of *pseudo-dimension* of the kernel optimization domain $\mathcal{K}^{(2)}$ [SBD06].

**Theorem 4.31** *[SBD06] Let $L(f) = \mathbb{P}(f(\mathbf{x}_i)y_i \leq 0)$ be the generalization risk of some prediction function $f$ learned by solving (2.30), and $L_n(f) = \frac{1}{n}\sum_{i=1}^n \mathbf{1}(f(\mathbf{x}_i)y_i < \gamma)$ be the empirical error. For a kernel family $\mathcal{K}$ with pseudo-dimension $d_{\mathcal{K}}$, the generalization risk of $f$ is bounded as:*

$$L(f) \leq L_n(f) + \sqrt{\tilde{O}(d_{\mathcal{K}} + 1/\gamma^2)/n},$$

*where $\gamma$ is the margin in loss function $l(t) = \max(0, \gamma - t)$, the $\tilde{O}$ notation hides logarithmic factors in its argument, the sample size and the allowed failure probability.*

Here the pseudo-dimension $d_{\mathcal{K}}$ measures the richness / complexity of a kernel domain $\mathcal{K}$:

**Definition 4.1** *Let $\mathcal{K}$ be a set of kernel functions mapping from $\mathbf{X} \times \mathbf{X}$ to $\mathbb{R}$. We say that a set of paired examples $\mathbf{S}_n = \{(\mathbf{x}_i, \mathbf{x}'_i) \in \mathbf{X} \times \mathbf{X}, i = 1, \dots, n\}$ is pseudo-shattered by $\mathcal{K}$ if there are real numbers $\mathbf{r} \in \mathbb{R}^n\}$ such that for any $\mathbf{b} \in \{-1, 1\}^n$ there is a function $k \in \mathcal{K}$ with property $sgn(k(\mathbf{x}_i, \mathbf{x}'_i) - r_i) = b_i$ for any $(\mathbf{x}_i, \mathbf{x}'_i) \in \mathbf{S}_n$. Then, we define a* **pseudo-dimension** *$d_{\mathcal{K}}$ of $\mathcal{K}$ to be the largest $n$ such that there exist a $\mathbf{S}_n$ that can be pseudo-shattered by $\mathcal{K}$.*

**Theorem 4.32** *Let $\mathcal{K}^{(1)} = \{k_1^{(1)}, \dots, k_m^{(1)}\}$ be the inner base kernel family for the 2-layer deep kernel $\mathcal{K}^{(2)}$ defined in (4.2), where $m$ is the number of base kernels. Assuming the evaluation of $k^{(1)}$ always positive, then the pseudo-dimension $d_{\mathcal{K}^{(2)}}$ is bounded by*

$$d_{\mathcal{K}^{(2)}} \leq m.$$

**Proof:** First, we re-write $\mathcal{K}^{(2)}$ as follows:

$$\mathcal{K}^{(2)} = \left\{ \prod \exp\left(\mu_t k_t^{(1)}\right) : k_t^{(1)} \in \mathcal{K}^{(1)} \right\} .$$

Thus each $k^{(2)} \in \mathcal{K}^{(2)}$ is the product of base kernels in form of $\exp(\mu k^{(1)})$. Consider the logarithmic operation $\ln \circ \mathcal{K}$, where for each kernel $k \in \mathcal{K}$ and any pair $(\mathbf{x}_i, \mathbf{x}_j)$, we have $(\ln \circ k)(\mathbf{x}_i, \mathbf{x}_j) = \ln k(\mathbf{x}_i, \mathbf{x}_j)$. Therefore, $\ln \circ \mathcal{K}^{(2)} = \left\{ \sum \mu_t k_t^{(1)} : k_t^{(1)} \in \mathcal{K}^{(1)} \right\}$. This is a linear space of dimension at most $m$ (with basis $\mathcal{K}^{(1)}$ when all $k^{(1)}$ are linearly independent). According to Theorem 11.4 of [AB99], the Pseudo-dimension [SBD06] of

$\ln \circ \mathcal{K}^{(2)}$ is bounded by $d_{\ln \circ \mathcal{K}^{(2)}} \leq m$. We can recover $\mathcal{K}^{(2)} = \exp \circ \ln \circ \mathcal{K}^{(2)}$. Since the exponential function is monotonic, by applying Theorem 11.3 of [AB99], we arrive at

$$d_{\mathcal{K}^{(2)}} = d_{\exp \circ \ln \circ \mathcal{K}^{(2)}} \leq d_{\ln \circ \mathcal{K}^{(2)}} \leq m. \tag{4.8}$$

*Remark*: Despite the simplicity of its proof, Theorem 2 implies that the outer exponential computation of our new kernel does not increase the complexity of the kernel domain in terms of pseudo-dimension. Comparing with MKL, our MLMKL method, with more flexible or richer optimization domain, would thus have a better chance in finding the best prediction function without increasing the generalization risk explicitly.

Recently, Ying and Campbell [YC09] employed the *Rademacher chaos complexity* $\widehat{\mathbf{u}}_n(\mathcal{K})$ to measure the richness of $\mathcal{K}$ through its ability of fitting noisy similarity value:

$$\widehat{\mathbf{u}}_n(\mathcal{K}; \mathbf{X}_1^n) = \mathbb{E}_{\boldsymbol{\varepsilon}} \sup_{k \in \mathcal{K}} \left| \frac{1}{n} \sum_{i<j} \varepsilon_i \varepsilon_j k(\mathbf{x}_i, \mathbf{x}_j) \right| : k \in \mathcal{K}, \mathbf{x}_i \in \mathbf{X}_1^n,$$

where $\varepsilon_i$ is a Rademacher random variable taking value $\pm 1$ with uniform probability. We have

**Corollary 4.1** *The Rademacher chaos complexity of $\mathcal{K}^{(2)}$ is bounded by*

$$\widehat{\mathbf{u}}_n(\mathcal{K}^{(2)}) \leq (192e + 1)\kappa^2 m, \tag{4.9}$$

*here $\kappa = \max_{\mathbf{x}} k^{(2)}(\mathbf{x}, \mathbf{x})$, $n$ is the size of training sample, $e$ is natural constant.*

The above corollary can be followed directly by combining Theorem 2 and the Rademacher chaos complexity result in [YC10, Theorem 3]. Finally, the generalization bound based on $\widehat{\mathbf{u}}_n(\mathcal{K}^{(2)})$ can be obtained immediately by combining the above Corollary with Lemma 9 of [YC09].

**Proof:** With the metric entropy integral techniques, it is proved by Ying et. al. that the Rademacher chaos complexity can be bounded by pseudo-dimension

$$\widehat{\mathbf{u}}_n(\mathcal{K}_{conv}) \leq \theta d_{\mathcal{K}} (1 + \kappa)^2 \ln(2en^2), \tag{4.10}$$

here $\theta$ is some constant. Thus, applying Theorem 4.32 yields the results immediately.

Table 4.2: The evaluation of classification performance by comparing with a number of different algorithms. Each element in the table shows the mean and standard deviation of classification accuracy (%). The relative ranking of different MKL algorithms on each data is shown in (). The last row shows the average rank score over all data sets achieved by each algorithm.

| Data Set | SVM | MKL$^{\text{level}}$ | LpMKL | GMKL | IKL | MKM | 2LMKL | 2LMKL$^{\text{Inf}}$ |
|---|---|---|---|---|---|---|---|---|
| Breast | 96.8±1.0 | 96.5±0.8 (5) | 96.2±0.7 (7) | **97.0±1.0** (2) | 96.5±0.7 (5) | **97.1±1.0** (1) | **97.0±1.0** (2) | **96.9±0.7** (4) |
| Diabetes | 76.7±1.8 | 75.8±2.5 (4) | 72.6±2.5 (6) | 66.4±2.5 (7) | 76.0±3.0 (3) | 75.8±2.5 (4) | **76.6±1.6** (1) | **76.6±1.9** (1) |
| Australian | 84.6±1.4 | 85.0±1.5 (5) | 84.5±1.6 (6) | 80.0±2.3 (7) | 85.4±1.2 (3) | 85.3±0.9 (4) | **85.5±1.6** (2) | **85.7±1.6** (1) |
| Splice | 85.0±1.4 | 88.4±2.4 (3) | 87.1±1.6 (4) | 92.4±1.4 (2) | 80.0±1.5 (7) | 84.6±1.5 (6) | **92.9±1.1** (1) | 84.7±1.2 (5) |
| FlareSolar | 67.5±2.0 | 67.6±2.0 (2) | 64.8±1.8 (5) | 65.3±1.8 (3) | 64.8±1.8 (5) | 64.4±1.7 (7) | **68.1±1.8** (1) | 65.3±1.8 (3) |
| Titanic | 78.0±3.2 | 77.1±2.9 (2) | 77.0±3.0 (5) | 76.7±3.1 (7) | 76.8±2.8 (6) | 77.1±3.0 (2) | 77.1±3.0 (2) | **77.8±2.6** (1) |
| Iono | 92.8±2.0 | 91.7±1.9 (7) | 92.6±1.4 (4) | 92.7±1.8 (3) | 93.7±1.0 (2) | 91.7±2.7 (6) | 92.3±1.5 (5) | **94.4±0.9** (1) |
| Banana | 89.7±1.5 | **90.2±2.0** (1) | 87.5±2.6 (4) | 83.4±2.7 (6) | **90.2±1.8** (1) | 80.5±5.3 (7) | 86.8±2.1 (5) | **90.2±1.6** (1) |
| Ringnorm | 98.5±0.7 | 98.1±0.8 (3) | 96.7±1.0 (7) | 97.5±1.0 (6) | **98.5±0.7** (1) | 97.7±1.0 (5) | 97.9±0.8 (4) | **98.5±0.8** (1) |
| Waveform | 89.0±1.8 | 88.2±1.6 (6) | 88.9±2.0 (4) | 88.2±1.8 (6) | 89.7±2.3 (3) | **90.0±1.6** (2) | 88.7±1.9 (5) | **90.4±1.6** (1) |
| Heart | 82.1±3.0 | 83.0±2.9 (4) | 76.7±3.8 (7) | 77.0±3.6 (6) | 83.3±2.1 (2) | 82.4±2.5 (5) | 83.1±2.5 (3) | **83.6±2.1** (1) |
| Sonar | 83.8±3.4 | 78.3±3.5 (7) | **84.8±3.2** (1) | 78.8±4.6 (6) | 81.0±5.0 (4) | 83.1±3.8 (3) | 79.0±4.3 (5) | **84.6±2.4** (2) |
| Thyroid | 93.9±2.9 | 92.9±2.9 (6) | 93.1±2.2 (5) | **94.6±2.1** (3) | **94.8±2.0** (1) | 92.6±3.0 (7) | 93.4±3.1 (4) | **94.8±2.2** (1) |
| Liver | 70.5±4.1 | 62.3±4.5 (6) | 69.4±2.9 (2) | 63.6±2.6 (4) | 60.0±2.9 (7) | **70.1±3.6** (1) | 66.0±3.4 (3) | 62.7±3.1 (5) |
| Adult | 82.0±0.7 | 81.5±1.0 (4) | **82.1±0.6** (1) | 75.5±1.1 (6) | 75.1±1.1 (7) | 81.7±0.9 (3) | 79.1±2.4 (5) | 81.8±1.0 (2) |
| German | 75.2±1.9 | 71.4±2.8 (5) | 74.3±1.4 (3) | 70.4±1.6 (6) | 70.0±1.5 (7) | **75.7±2.3** (1) | 74.8±1.8 (2) | 74.2±2.0 (4) |
| Rank | N/A | 4.38 | 4.44 | 5.00 | 4.00 | 4.00 | 3.13 | 2.13 |

## 4.5 Experiments

### 4.5.1 Experimental Testbed and Setup

We evaluate the performance of the proposed Two-Layer MKL algorithms for binary classification tasks over a testbed of 16 publicly available data sets as shown in Table 1[1] [2].

Following the settings of previous MKL studies [XJKL08], for each data set, we create the set of base kernels $\mathcal{K}$ as follows: (1) Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \dots, 2^{6}\}$) on all features and on each single feature; (2) polynomial kernels of degree 1 to 3 on all features and on each single feature. Each base kernel matrix is normalized to unit trace. For each data set, we randomly sample 50% of all instances as training data, and use the rest as test data. The training instances are normalized to be of zero mean and unit variance, and the test instances are also normalized using the same mean and variance of the training data. To get stable results, for each data set, we repeat each algorithm 20 times and compute the average results of the 20 runs.

For comparison, we have tried our best to compare as many state-of-the-art MKL methods as possible, which were proposed under different contexts for various applica-

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

[2] http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark

tions. The goal of our experiment is mainly to examine if deep MKL is effective for improving the performance of the shallow MKL techniques. Specifically, we have compared the following algorithms:

**SVM**: The Support Vector Machine algorithm with a single Gaussian kernel. The bandwidth parameter is selected via 5-fold cross validation on the training data;

**MKL**$^{\text{Level}}$: The convex multiple kernel learning algorithm, that is, the target kernel class is $\mathcal{K}_{conv}$ defined in (5.1). We use the extended level method [XJKL08] to learn the kernel;

**LpMKL**: The MKL algorithm with $L_p$ norm regularization over the kernel weight [KBS$^+$09]. We adopt their cutting plane algorithm with second order Taylor approximation of $L_p$;

**GMKL**: The Generalized MKL algorithm in [VB09]. The target kernel class is the Hadamard product of single Gaussian kernel defined on each dimension;

**IKL**: The Infinite Kernel Learning algorithm proposed by [GN08]. We use LevelMKL as the embedded algorithm to solve the kernel weight $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$;

**MKM**: The Multilayer Kernel Machine with deep learning [CS09], which essentially trained SVM with multilayer arc-cosine kernel functions;

**2LMKL**: The proposed Two-Layer MKL algorithm described in Algorithm 2;

**2LMKL**$^{\text{Inf}}$: The proposed Infinite Two-Layer MKL algorithm described in Algorithm 3.

For parameter settings, the regularization parameter $C$ in MKL or our 2LMKL algorithms is determined by 5-fold cross validation on the training data over the range of $\{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$. For a fair comparison, the same set of base kernels was adopted by MKL$^{Level}$, LpMKL, and 2LMKL. For LpMKL, we examine $p = 2, 3, 4$ and report the best result. For fair comparison, in MKM we chose the layer $l = 2$, and found the best degree parameter ({0,1,2}) by cross validation. For 2LMKL$^{\text{Inf}}$, the initial base kernel is a Gaussian kernel with 10 parameters and polynomial kernel with 3 degrees calculated on all the features. During the iterative process, we add one Gaussian kernel at each iteration.

### 4.5.2 Performance Analysis

Table 2 shows the detailed results of average classification accuracy and standard deviation values. To compare the overall performance, we count the ranks of the algorithms according to their performance on each data set. The average rank is included in the last row. From the results, we can draw several observations as follows.

First of all, by comparing the results between SVM and the four existing MKL methods, we found that the existing MKL algorithms do not always outperform SVM with an RBF kernel. For example, for the MKL$^{\text{Level}}$ algorithm, it outperformed SVM only over five data sets (*Australian, Splice, FlareSolar, Banana, and Heart*), and was surpassed significantly by SVM over several data sets (*German, Sonar, and Liver*, etc). Although it seems a little bit surprising, the similar observation was reported in some previous empirical study [GN08], which also found that regular MKL does not always outperform SVM with an RBF kernel who kernel parameter was tuned by cross validation. This observation validates our motivation for overcoming the *shallow* limitation of the regular MKL methods.

Second, by comparing the four existing MKL methods, we observed that IKL overall achieved the best performance among them, and GMKL tended to perform slightly worse than the other methods. Specifically, among all the MKL methods, IKL won 3 best cases, LpMKL won 2 best cases, while either MKL$^{\text{Level}}$ or GMKL won only 1 out of 16 cases. We conjecture the reason that IKL performed better is probably because IKL has the possibility of using a largely increased kernel set, which may be flexible for the classification task. Similarly, the reason that GMKL performed worse may be due to the relatively smaller kernel set, in which the base kernel set consists of only $d$ kernels each of them was defined on a single dimension.

Third, by examining the results achieved the two proposed algorithms, 2LMKL and 2LMKL$^{\text{Inf}}$, we found that both of them achieved rather impressive performance. Both of them considerably outperformed the other methods, including the existing MKL methods and the MKM method, over quite a number of data sets. Specifically, among all the MKL methods, 2LMKL won 5 best cases while 2LMKL$^{\text{Inf}}$ won 11 best cases out of 16 cases. The encouraging performance showed our 2LMKL method is more effective than the regular MKL methods through the exploration of deep kernel learning capability, and is also more effective than the previous MKM method with deep learning.

Finally, comparing the two proposed two-layer MKL algorithms themselves, we observed that 2LMKL$^{\text{Inf}}$ performed better than 2LMKL. This validates the efficacy of the proposed improvement by exploiting the idea of indefinite kernel learning.

### 4.5.3 Summary

This chapter presented a general framework of multi-layer multiple kernel learning (MLMKL) to overcome the shallow learning nature of regular MKL. Under the framework, we propose a Two-Layer Multiple Kernel Learning (2LMKL) method, and developed two effective algorithms to solve it. We analyzed the generalization risk of the proposed two-layer MKL algorithms and conducted an extensive set of experiments. Our empirical results showed that the proposed 2LMKL algorithms usually perform better than the existing shallow MKL methods, demonstrating the efficacy of the 2LMKL approach.

Despite the promising results, MLMKL remains a rather new area for future research. In our future work, we plan to extend the current two-layer MKL scheme to higher-layer MKL solutions to further enhance the efficacy. Akin to the training scheme of deep learning[HOT06], one can learn $\boldsymbol{\mu}^{(l)}$ in a bottom-up and layer-wise manner.In other words, we can embed the learned kernel $k_s^{(l)}$ (summation of base kernels at the current layer) into the base kernel functions to generate the candidate kernels $k_1^{(l+1)}, \ldots, k_n^{(l+1)}$ for the next layer. Then conventional MKL algorithms are adopted to solve the weight $\boldsymbol{\mu}^{(l+1)}$. This process can be repeated to construct deep kernels. We will also analyze the overfitting issue for MLMKL and investigate more theoretical insights about the power of multi-layer multiple kernel learning.

# Chapter 5

# Unsupervised Multiple Kernel Learning

Traditional multiple kernel learning (MKL) algorithms are essentially supervised in the sense that the kernel learning task requires class labels of the training data. However, class labels may not always be available prior to the kernel learning task in some real world scenarios, e.g., some early preprocessing step of a classification task or an unsupervisd learning task such as dimension reduction or clustering. In this chapter, we address a new problem of unsupervised multiple kernel learning (UMKL), which does not require class labels of training data as used in a conventional multiple kernel learning task. As a kernel essentially defines pairwise similarity between any two examples, our unsupervised kernel learning method mainly follows two intuitive principles: (1) a good kernel should allow every example to be well reconstructed from its localized bases weighted by the kernel values; (2) a good kernel should induce kernel values that are coincided with the local geometry of the data. We propose an efficient iterative algorithm to solve the optimization task of the unsupervised multiple kernel learning. Empirical results on a number of UCI data sets verifies the efficacy of our UMKL algorithm.

## 5.1 Introduction of UMKL

Despite being studied extensively, existing MKL methods are essentially supervised learning which requires class labels of training data available for the learning task. However, the class labels of training data may not always be available for some learning tasks. Examples include unsupervised learning tasks such as dimension reduction, clustering, or some early preprocessing step of a supervised classification task for choosing a kernel before the labeled data arrive. By the above motivations, in this chapter, we address a new problem of *Unsupervised Multiple Kernel Learning* (UMKL), which aims to determine a linear combination of multiple kernels by learning only from unlabeled data.

The UMKL task is more challenging than traditional MKL tasks since no class labels are available to guide the learning task. We propose to attack the unsupervised multiple kernel learning problem by exploiting two intuitions: (1) a good kernel should allow every example to be well reconstructed from its localized bases weighted by the kernel values; (2) a good kernel should induce kernel values that are coincided with the local geometry of the data. We formulate the problem as an optimization task by combining the above two intuitions and propose an iterative algorithm to efficiently solve the optimization task.

The rest of this chapter is organized as follows. Section 5.2 presents the problem formulation and the proposed UMKL algorithm. Section 5.3 includes our empirical evaluation of the proposed UMKL algorithm.Section 5.4 concludes this work.

## 5.2 Framework of UMKL

In this section, we formulate the problem of unsupervised multiple kernel learning and then present an iterative algorithm to solve the optimization task.

### 5.2.1 Problem Formulation

Consider a collection of $n$ training samples $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature vector and $y_i$ is the unknown class label of $\mathbf{x}_i$, and a set of $m$ predefined kernel functions $\{k_t(\cdot,\cdot), t = 1, \ldots, m\}$. The goal of an Unsupervised Multiple Kernel Learning (UMKL) task is to find an optimal linear combination of the $m$ kernel functions, i.e., $k^*(\cdot,\cdot) \in \mathcal{K}_{conv}$, where $\mathcal{K}_{conv}$ is defined as follows:

$$\mathcal{K}_{conv} = \left\{ k(\cdot,\cdot) = \sum_{t=1}^{m} \mu_t k_t(\cdot,\cdot) : \sum_{t=1}^{m} \mu_t = 1, \mu_t \geq 0 \right\} \tag{5.1}$$

Unlike supervised MKL, the key challenge of UMKL is how to seek the optimal kernel $k^*(\cdot,\cdot)$ purely from the unlabeled training data. In other words, we need some principles/intuitions to guide the kernel learning task, which should be independent of class labels.

To attack the challenge, we propose to formulate the unsupervised multiple kernel learning task by exploiting the following two intuitive principles:

- A good kernel should enable each training instance to be well reconstructed from the localized bases weighted by the kernel values. In other words, for each $\mathbf{x}_i$ we expect the optimal kernel can minimize the approximation error $\|x_i - \sum_j k_{ij}\mathbf{x}_j\|^2$;

- A good kernel should induce kernel values that are coincided with the local geometry of the training data. This is equivalent to finding the optimal kernel that minimizes the distortion over all training data $\sum_{ij} k_{ij}\|\mathbf{x}_i - \mathbf{x}_j|^2$.

Besides, we also exploit the locality preserving principle, which has been shown effective for many unsupervised dimension reduction and semi-supervised learning tasks.In particular, to infer such a local structure, we introduce a set of local bases for each $\mathbf{x}_i \in \mathcal{X}$, denoted $\mathcal{B}_i \subset \mathcal{X}$, which is used to reconstruct sample $\mathbf{x}_i$ and compute the distortion as well. Combining the above principles, we have the final optimization for UMKL:

$$\min_{k,\mathcal{B}} \frac{1}{2}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \sum_{\mathbf{x}_j\in\mathcal{B}_i} k_{ij}\mathbf{x}_j\right\|^2 + \gamma_1 \sum_{i=1}^{n}\sum_{\mathbf{x}_j\in\mathcal{B}_i} k_{ij}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 + \gamma_2 \sum_i |\mathcal{B}_i| \tag{5.2}$$

where both the target kernel $k \in \mathcal{K}_{conv}$ and the local basis set $\mathcal{B}_i$ are unknown variables to be optimized, $\gamma_1$ controls the trade-off between the coding error and the locality distortion, $\gamma_2$ controls the size of local basis set.

To simplify the formulation, for each $\mathbf{x}_i$, we introduce a vector $\mathbf{d} \in \{0,1\}^n$ to indicate its neighbors, i.e., $\mathcal{B}_i = \{\mathbf{x}_j : d_j \neq 0\}$. As a result, the optimization problem can be re-written in a matrix-based form:

$$\min_{\boldsymbol{\mu},\mathbf{D}} \frac{1}{2}\left\|\mathbf{X}\big(\mathbf{I} - \mathbf{K}\circ\mathbf{D}\big)\right\|_F^2 + \gamma_1 \mathrm{tr}\ \mathbf{K}\circ\mathbf{D}\circ\mathbf{M}\big(\mathbf{1}\mathbf{1}^\top\big) + \gamma_2\|\mathbf{D}\|_{1,1} \tag{5.3}$$

$$\text{s.t.} \qquad [\mathbf{K}]_{ij} = \sum_{t=1}^{m}\mu_t k^t(\mathbf{x}_i,\mathbf{x}_j), 1 \le i,j \le n, \boldsymbol{\mu}^\top\mathbf{1} = 1, \boldsymbol{\mu} \ge 0, \mathbf{D} \in \{0,1\}^{n\times n},$$

where the $i$-th column of $\mathbf{X}$ is the point $\mathbf{x}_i$, matrix $[\mathbf{M}]_{ij} := \mathbf{x}_i^\top\mathbf{x}_i + \mathbf{x}_j^\top\mathbf{x}_j - 2\mathbf{x}_i^\top\mathbf{x}_j$ is the Euclidean distance matrix defined on $\mathbf{X}$, $B \in \mathbb{N}_+$ is the parameter controls the size of $\mathcal{B}_i$ for each $\mathbf{x}_i$. For the above notations, $\circ$ denotes an element-wise multiplication of two matrices, $\|\cdot\|_F^2$ denotes the Frobenius-norm of a matrix, and *tr* denotes the trace of a matrix.

The presented formulation essentially treats the kernel evaluation $\boldsymbol{\kappa}_i$ as a local coding coordinate of $\mathbf{x}_i$. With the additional constraint $\boldsymbol{\kappa}^\top\mathbf{1} = 1$, such a local coding system allows a linear approximation of an target function, guaranteed by[YZG09]:

**Theorem 5.33** *Let $(\boldsymbol{\kappa},\mathcal{B})$ be an arbitrary coordinate coding on $\mathbb{R}^d$. Let $f$ be an $(\alpha,\beta,p)$-Lipschitz smooth function, i.e., $|f(\mathbf{x}) - f(\mathbf{x}')| \le \alpha\|\mathbf{x}-\mathbf{x}'\|$ and $|f(\mathbf{x}') - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{x}'-\mathbf{x})| \le \beta\|\mathbf{x}-\mathbf{x}'\|^{1+p}$, $\alpha,\beta > 0, p \in (0,1]$. We have for all $\mathbf{x} \in \mathbb{R}^d$:*

$$\left|f(\mathbf{x}) - \sum_{\mathbf{x}_j\in\mathcal{B}} \kappa(\mathbf{x}_i,\mathbf{x}_j)f(\mathbf{x}_j)\right| \le \alpha\|\mathbf{x} - \sum_j \kappa_j\mathbf{x}_j\| + \beta\sum_{\mathbf{x}_j\in\mathcal{B}} |\kappa(\mathbf{x}_i,\mathbf{x}_j)|\|\mathbf{x}_i - \mathbf{x}_i\|^{1+p}.$$

For practical purpose, we remove the constraint $\boldsymbol{\kappa}^\top\mathbf{1} = 1$. Instead, we deem the coding as kernel evaluation result, which is sampled from a predefined set $\mathcal{K}_{conv}$. Moreover, for each data point $\mathbf{x}_i$, we learn a set of local bases, rather than a global basis set. This allows further locality characterization of the data. Such local bases are also useful for discovering the data topology, which is essential for data embedding tasks.

### 5.2.2 Iterative Optimization Algorithm

It is not difficult to see that the optimization task (5.3) is essentially a mixed integer programming, which is not easy to solve directly. In our approach, we propose an iterative optimization algorithm by alternatively solving $\boldsymbol{\mu}$ and $\mathbf{D}$, which is in spirit to [TWT10].

#### 5.2.2.1 Solving $\boldsymbol{\mu}$ by Fixing D

We first solve $\boldsymbol{\mu}$ by assuming the values of $\mathbf{D}$ have been identified. The objective reduces to

$$J(\boldsymbol{\mu}) = \boldsymbol{\mu}^\top \Big( \sum_{t=1}^{m} \sum_{i=1}^{n} \boldsymbol{\kappa}_{t,i} \boldsymbol{\kappa}_{t,i}^\top \circ \mathbf{d}_i \mathbf{d}_i^\top \circ \mathbf{P} \Big)^\top \boldsymbol{\mu} + \mathbf{u}^\top \boldsymbol{\mu}, \tag{5.4}$$

where $[\mathbf{u}]_t := \sum_{i=1}^{n} (\mathbf{v} \circ \mathbf{d}_i - 2\mathbf{p}_i \circ \mathbf{d}_i)^\top \boldsymbol{\kappa}_{t,i}$, $\mathbf{P} = \mathbf{X}^\top \mathbf{X}$ is a linear kernel, $\boldsymbol{\kappa}_{t,i} := [k^t(\mathbf{x}_i, \mathbf{x}_1), \ldots, k^t(\mathbf{x}_i, \mathbf{x}_n)]^\top$ is the $i$-th column of the $t$-th kernel matrix, $\mathbf{p}$ and $\mathbf{v}$ are the columns of $\mathbf{P}$ and $\mathbf{M}$ corresponding to $\mathbf{x}_i$, separately.

As a result, the original optimization problem (5.3) w.r.t. $\boldsymbol{\mu}$ is re-formulated into a quadratic program [BV04] which can be solved efficiently with an off-the-shelf optimization toolkit.

#### 5.2.2.2 Solving D by Fixing the Kernel

The $i$-th column $\mathbf{d}_i$ of $\mathbf{D}$ is a binary vector indicating the set of localized bases of sample $\mathbf{x}_i$. Note that the columns of $\mathbf{D}$ are independent of each others, i.e., the solution $\mathbf{d}_i$ of a specific sample $\mathbf{x}_i$ would not affect the bases of other samples. Therefore, we can solve each column $\mathbf{d}_i$ of $\mathbf{D}$ separately.

At the beginning, we need to find the local basis set for each training sample. To this end, we adopt sparse coding to initialize $\mathbf{D}$, which is formulated into:

$$J(\mathbf{d}) = \frac{1}{2} \Big\| \mathbf{x}_i - \sum_{j \neq i} d_j \mathbf{x}_j \Big\|_2^2 + \gamma_2 \|\mathbf{d}\|_1. \tag{5.5}$$

Note here we relax the binary indicating vector $\mathbf{d}$ to continuously valued vector. This is a typical sparse coding problem, which can be solved efficiently by feature-sign algorithm [LBRN06].

Assume the we have learned a kernel with the feature mapping function $\Phi$, i.e., $k_{ij} = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$. The sparse coding (5.5) can be kernelized as

$$\begin{aligned} J(\mathbf{d}) &= \frac{1}{2} \|\Phi(\mathbf{x}_i) - \sum_{j \neq i} d_j \Phi(\mathbf{x}_j)\|_2^2 + \gamma_2 \|\mathbf{d}\|_1 \\ &= \frac{1}{2} \big( k_{ii} + \mathbf{d}_i^\top \mathbf{K} \mathbf{d}_i - 2\mathbf{d}_i^\top \boldsymbol{\kappa} \big) + \gamma_2 \|\mathbf{d}_i\|_1, \end{aligned}$$

where $\boldsymbol{\kappa} = [k(\mathbf{x}_i, \mathbf{x}_j)]_j$ abbreviates the $i$-th column of the kernel matrix $\mathbf{K}$. Besides the relaxation of $\mathbf{D}$, another scheme is to propose some greedy algorithm to tackle the original mixed integer programming problem. To this end, we formulate (5.3) w.r.t. $\mathbf{d}$ by

$$J(\mathbf{d}) = \mathbf{d}^\top \big(\boldsymbol{\kappa}\boldsymbol{\kappa}^\top \circ \mathbf{P}\big)\mathbf{d} + \big(\boldsymbol{\kappa} \circ \mathbf{v} - 2\boldsymbol{\kappa} \circ \mathbf{p}\big)^\top \mathbf{d}, \tag{5.6}$$

where the common subscript of $\mathbf{d}, \boldsymbol{\kappa}, \mathbf{p}, \mathbf{v}$ are omitted.

Let $\mathbf{U}$ abbreviate $\boldsymbol{\kappa}\boldsymbol{\kappa}^\top \circ \mathbf{P}$, and $\mathbf{v}$ abbreviate $\boldsymbol{\kappa} \circ \mathbf{v} - 2\boldsymbol{\kappa} \circ \mathbf{p}$. We solve $\mathbf{d}$ by a greedy algorithm. Initially, we set all the entries of $\mathbf{d}$ to zero. At each greedy step, we add one sample into the basis set of $\mathbf{x}$, which implies to set one component of $\mathbf{d}$ to 1. We choose the component as the one which minimizes the increase of $J(\mathbf{d})$. This process is iterated until $B$ entries of $\mathbf{d}$ have been set to 1.

Assuming each dimension of $\mathbf{x}$ is nonnegative, one can show immediately that $J(\mathbf{d})$ is a sub-modular function, i.e., $J(\mathbf{d}^1) + J(\mathbf{d}^2) \geq J(\mathbf{d}^1|\mathbf{d}^2) + J(\mathbf{d}^1\&\mathbf{d}^2)$, where $|$ and & are the bit-wise "or" and "and" operation. The following theorem guarantees the performance of greedy selection of $\mathbf{d}$:

**Theorem 5.34** *Let $\mathbf{d}^*$ denote the optimal solution of (5.6), and $\hat{\mathbf{d}}$ denote the approximation solution found by the greedy algorithm. We have*

$$J(\hat{\mathbf{d}}) \geq J(\mathbf{d}^*)(1 - 1/e),$$

*if $J(\mathbf{d})$ satisfies the following conditions: 1) $J(\mathbf{d}^1) \leq J(\mathbf{d}^2)$ if $\mathbf{d}^1 \subset \mathbf{d}^2$; 2) $J(\mathbf{d})$ is a submodular function, and 3) $J(\mathbf{0}) = 0$.*

After kernel learning, the kernel can be used in classification, or other semi-supervised and unsupervised tasks, such as dimension reduction, etc.

## 5.3 Experiments

In order to examine the efficacy of the proposed UMKL algorithm, we apply the proposed UMKL algorithm in two scenarios: (i) a pre-processing tool to find an appropriate kernel for classification task, and (ii) a nonlinear dimension reduction tool for machine learning and data mining tasks.

### 5.3.1 Experimental Testbed and Setup

We evaluate the performance of the proposed UMKL algorithms for binary classification tasks over a testbed of publicly available data sets as shown in Table 1[1] [2].

[1]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

[2]http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark

**Protocol 4** UMKL: Unsupervised Multiple Kernel Learning

**Input:** Unlabeled data $\mathbf{X}$, base kernels $\mathcal{K}_{base} = \{k_1, \ldots, k_m\}$, $\gamma$, $B$;
**Output:** Kernel weight $\boldsymbol{\mu}$, bases indicator matrix $\mathbf{D}$.

```
1: Initialize $[\mathbf{M}]_{ij} = \mathbf{x}_i^\top \mathbf{x}_i + \mathbf{x}_j^\top \mathbf{x}_j - 2\mathbf{x}_i^\top \mathbf{x}_j$, $\boldsymbol{\mu} = \mathbf{1}/m$, $\mathbf{P} = \mathbf{X}^\top \mathbf{X}$;
2: repeat
3:    $\mathbf{W} = \sum_{t=1}^m \sum_{i=1}^n \boldsymbol{\kappa}_{t,i} \boldsymbol{\kappa}_{t,i}^\top \circ \mathbf{d}_i \mathbf{d}_i^\top \circ \mathbf{P}$;
4:    $\boldsymbol{\mu} = \arg\min_{\boldsymbol{\mu}} \boldsymbol{\mu}^\top \mathbf{W} \boldsymbol{\mu} + \mathbf{u}^\top \boldsymbol{\mu}$;
5:    for each $\mathbf{x}_i$ do
6:       $\mathbf{U} = \boldsymbol{\kappa}\boldsymbol{\kappa}^\top \circ \mathbf{P}, \mathbf{v} = \boldsymbol{\kappa}(\mathbf{v} - 2\mathbf{p})$;
7:       Set $\mathcal{B}_i = \varnothing$;
8:       for $|\mathcal{B}_i| < B$ do
9:          $\mathbf{x}^* = \arg\min_{\mathbf{x}_j \in \mathcal{X} - \mathcal{B}_i} 2 \sum_{t:\mathbf{x}_t \in \mathcal{B}_i} U_{tj} + U_{jj} + v_j$;
10:         $\mathcal{B}_i = \mathcal{B}_i \cup \{\mathbf{x}^*\}$;
11:      end for
12:   end for
13: until convergence
```

Table 5.1: The statistics of the 12 binary-class data sets used in our experiments.

| Data Set | Breast | Diabetes | Waveform | Sonar | Liver | German |
|---|---|---|---|---|---|---|
| # instances | 683 | 768 | 400 | 208 | 345 | 1,000 |
| # dimensions | 10 | 8 | 21 | 60 | 6 | 24 |
| Data Set | Australian | Thyroid | Heart | Banana | Titanic | FlareSolar |
| # instances | 690 | 140 | 270 | 400 | 150 | 666 |
| # dimensions | 14 | 5 | 13 | 2 | 3 | 9 |

Following the settings of previous MKL studies [XJKL08], for each data set, we create the set of base kernels $\mathcal{K}$ as follows: (1) Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \dots, 2^{6}\}$) on all features and on each single feature; (2) polynomial kernels of degree 1 to 3 on all features and on each single feature. Each base kernel matrix is normalized to unit trace. The training instances are normalized to be of zero mean and unit variance, and the test instances are also normalized using the same mean and variance of the training data. To get stable results, for each data set, we repeat each algorithm 20 times and compute the average results of the 20 runs.

### 5.3.2 Experiment I:Unsupervised MKL versus Supervised MKL

The first experiment is to examine whether the proposed UMKL algorithm can produce good results comparing it with traditional supervised kernel learning (SKL) algorithms on the same set of training data.

Specifically, we have compared the following algorithms:

**AvgKernel**: The average combination of multiple kernels;

**MKL$^{\text{Level}}$**: The convex multiple kernel learning algorithm, that is, the target kernel class is $\mathcal{K}_{conv}$ defined in (5.1). We use the extended level method [XJKL08] to learn the kernel;

**LpMKL**: The MKL algorithm with $L_p$ norm regularization over the kernel weight [KBS+09]. We adopt their cutting plane algorithm with second order Taylor approximation of $L_p$;

**GMKL**: The Generalized MKL algorithm in [VB09]. The target kernel class is the Hadamard product of single Gaussian kernel defined on each dimension;

**KTA**: The Two-stage Kernel Learning by Kernel Target Alignment algorithm in [Cor09];

**UMKL**: The proposed Unsupervised MKL method with the greedy base set selection[3]; We first learn the kernel by UMKL and then apply the kernel for learning an SVM classifier.

For parameter settings, the regularization parameter $C$ in SVM for all the compared methods is determined by 5-fold cross validation on the training data over the range of $\{10^{-2}, 10^{-1}, \dots, 10^{2}\}$. For a fair comparison, the same set of base kernels was adopted by MKL$^{\text{Level}}$, LpMKL, KTA, and UMKL. For LpMKL, we examine $p = 2, 3, 4$ and report the best result.

[3]We also evaluated another variant of the UMKL algorithm by the sparse coding solution. We found that it is always slightly worse than the greedy algorithm for most cases.

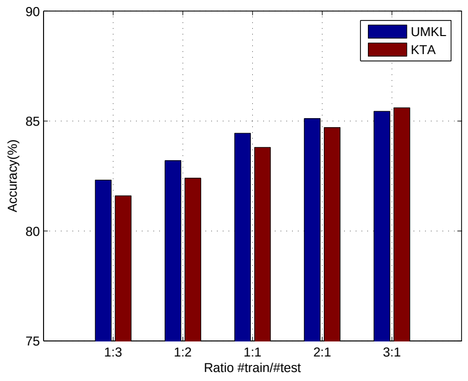
For each data set, we randomly sample 50% of all instances as training data, and use the rest as test data. The classification accuracy is shown in Table 2, from which we can draw two observations.

Table 5.2: The evaluation of classification performance by comparing with a number of different algorithms. Each element in the table shows the mean and standard deviation of classification accuracy (%). The relative ranking of different MKL algorithms on each data is shown in (). The last row shows the average rank score over all data sets achieved by each algorithm. The **bold** element indicates the best performance.

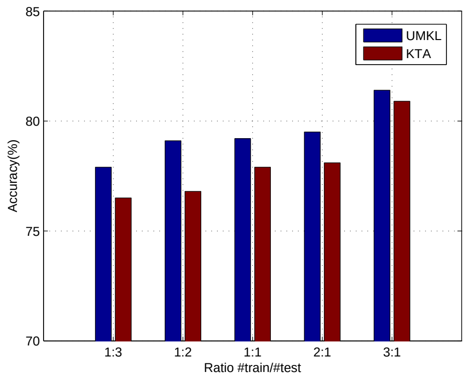
| Data Set | AvgKernel | $\text{MKL}^{\text{level}}$ | LpMKL | GMKL | KTA | UMKL |
|---|---|---|---|---|---|---|
| Breast | 95.5±1.0 | 96.5±0.8 | 96.2±0.7 | **97.0±1.0** | 96.5±0.8 | **97.0±0.6** |
| Diabetes | 65.4±2.0 | 75.8±2.5 | 72.6±2.5 | 66.4±2.5 | 75.2±3.3 | **77.1±2.3** |
| Titanic | 76.2±4.2 | 77.1±2.9 | 77.0±3.0 | 76.7±3.1 | 78.9±1.2 | **79.2±2.5** |
| Heart | 75.6±6.0 | 83.0±2.9 | 76.7±3.8 | 77.0±3.6 | 82.0±1.9 | **84.7±2.6** |
| German | 69.5±1.5 | 71.4±2.8 | **74.3±1.4** | 70.4±1.6 | 72.1±1.1 | **74.8±1.8** |
| Australian | 66.7±5.6 | 85.0±1.5 | 84.5±1.6 | 80.0±2.3 | **87.2±0** | 86.3±1.3 |
| Banana | 86.1±2.3 | 90.2±2.0 | 87.5±2.6 | 83.4±2.7 | **92.1±1.7** | 90.2±1.9 |
| Thyroid | 82.6±4.4 | 92.9±2.9 | 93.1±2.2 | 94.6±2.1 | **97.9±1.3** | 91.2±2.7 |
| FlareSolar | 64.8±1.6 | **67.6±2.0** | 64.8±1.8 | 65.3±1.8 | 66.7±2.3 | **67.8±1.8** |
| Waveform | 73.5±7.1 | 88.2±1.6 | **88.9±2.0** | 88.2±1.8 | 88.5±2.5 | 88.4±2.5 |
| Sonar | 59.1±9.7 | 78.3±3.5 | **84.8±3.2** | 78.8±4.6 | 81.3±2.9 | 81.0±2.9 |
| Liver | 57.6±2.3 | 62.3±4.5 | **69.4±2.9** | 63.6±2.6 | 68.7±1.5 | 68.3±4.3 |

First, UMKL is comparable with supervised kernel learning algorithms. Among the evaluated 12 data sets, UMKL reports 6 best results, which is the largest among all the evaluated methods. None of the compared 4 algorithms, i.e., MKL, LpMKL, GMKL, and KTA, can outperform the proposed UMKL algorithm significantly. The average rank of UMKL is comparable with KTA. These three algorithms are significantly better than MKL and GMKL. This result verifies the efficacy of UMKL clearly. When the half of the data have labels for training, SKL and UMKL, which does not use the label information, produce similar classification accuracy.

Second, among the SKL algorithms, KTA yield the best performance. The average rank of KTA among SKL algorithms is the best. Traditional MKL cannot result in the best accuracy on any evaluated data sets. The results of KTA on *thyroid* are statistically significant better than other SKL methods. Figure 1 illustrates the performance comparison of both UMKL and KTA algorithms by varying the fraction of training data instances on two datasets. For most situtations, the performance of UMKL is consistently better or comparable to that of the KTA algorithm, which again validates the efficacy of the unsupervised multiple kernel learning algorithm.



5.1.a: heart



5.1.b: titanic

Figure 5.1: Classification accuracy of UMKL and KTA by varying #train/#test.

### 5.3.3 Experiment II: UMKL for Dimensionality Reduction

Our second experiment is to examine if the proposed UKML algoithm is effective for an unsupervised dimension reduction task using kernel techniques. To this purpose, we apply the UMKL algorithm as a tool to learn an appropriate kernel function for Kernel PCA, which is a classical nonlinear dimensionality reduction technique that chooses the dominating principle component in the feature space. Typically, it is often non-trivial to select the appropriate kernel based on given data for kernel PCA.

Specifically, in this experiment, we compare the following different approaches of choosing kernels for kernel PCA towards dimension reduction tasks:

**Guassian**: We consider a single Guassian kernel, where the band-width parameter $\sigma$ is chosen empirically from the data (we simply fix $\sigma = 1$ in our experiments;

**Average**: The average kernel by combining the set of multiple kernels via a uniformly linear combination approach, where the set of multiple kernels is the same as the previous experiment;

**C-UKL**: The unsupervised kernel learning learning algorithm aiming to maximize margin of the data (cluster assumption on the data), proposed by [VJ06];

**UMKL**: The proposed UMKL algorithm to identify the appropriate linear combination of multiple kernels on the same set of multiple kernels.

Table 5.3: The classification accuracy of a 5-NN classifier on the projected data by KPCA with three different kinds of kerenls, i.e., a gaussian kernel with fixed sigma=1, the average combination of multiple kernels, and the kernel learned by UMKL on 9 benchmark data sets. The parameter $B$ of UMKL is fixed to 10. The **bold** element indicates the best performance.

| Data Set | Gaussian Kernel | Average Kernel | C-UKL | UMKL$^{\text{Sub}}$ |
|---|---|---|---|---|
| Breast | 95.1±0.8 | 92.7±2.7 | 94.8±1.0 | **95.2±1.3** |
| Diabetes | 64.6±2.4 | 65.1±2.3 | 64.2±2.4 | **66.3±2.5** |
| Heart | 56.2±3.5 | 61.5±5.8 | 55.7±3.4 | **72.4±5.9** |
| German | 63.9±1.7 | 64.3±1.9 | 63.4±1.8 | **64.9±2.2** |
| Australian | 57.7±2.3 | 55.9±2.6 | 69.4±2.9 | **72.9±3.8** |
| FlareSolar | 61.9±3.4 | 62.5±3.6 | 62.8±3.5 | **63.0±3.5** |
| Waveform | 59.4±3.5 | 63.2±4.1 | 63.9±5.8 | **76.9±3.9** |
| Sonar | 50.7±5.0 | 56.7±5.8 | 54.6±6.7 | **57.0±7.2** |
| Liver | 52.9±4.5 | 52.0±3.3 | **53.4±3.4** | **53.2±3.3** |

In this experiment, for each of the above approaches, we apply it to reduce data from the original dimensionality to 2 dimensions, and then adopt a 5-nearest neighbor classification scheme to evaluate the classification performance on the reduced data. For each data set, we randomly sample 50% of all instances as training data, and use the rest as test data. To obtain stable results, for each data set, we repeat the randomized sampling process 20 times, run each algorithm on these randomly sampled training/test data, and finally compute the average results of each algorithm over these 20 runs. We adopt 9 benchmark datasets as used in Experiment I [4].

Table 5.3.3 and Figure 5.3.3 summarize the experimental results of the kernel PCA by three kinds of different kernels for supervised classification tasks. From the results, we could see that the proposed UMKL algorithm is able to learn a better kernel that is considerably more effective than either a single gaussian kernel or an average kernel via a uniformly linear combination for most cases. The results are particularly impressive on several datasets including heart, australian, waveform, where UMKL significantly surpasses the other two approaches.

In addition to the fixed number of reduced dimensions, we also try to examine how the compared algorithms work when applying KPCA to obtain projected data of a varied numbers of dimensions. Figure 5.3.3 shows the experimental results of evaluating the classification performance on the data by applying KPCA to obtain low-dimensional

[4]Note that we do not use three of the datasets (i.e., Banana, Titanic, and THyroid) because of their original dimensionality is already very slow (e.g., 2,3,5 respectively)

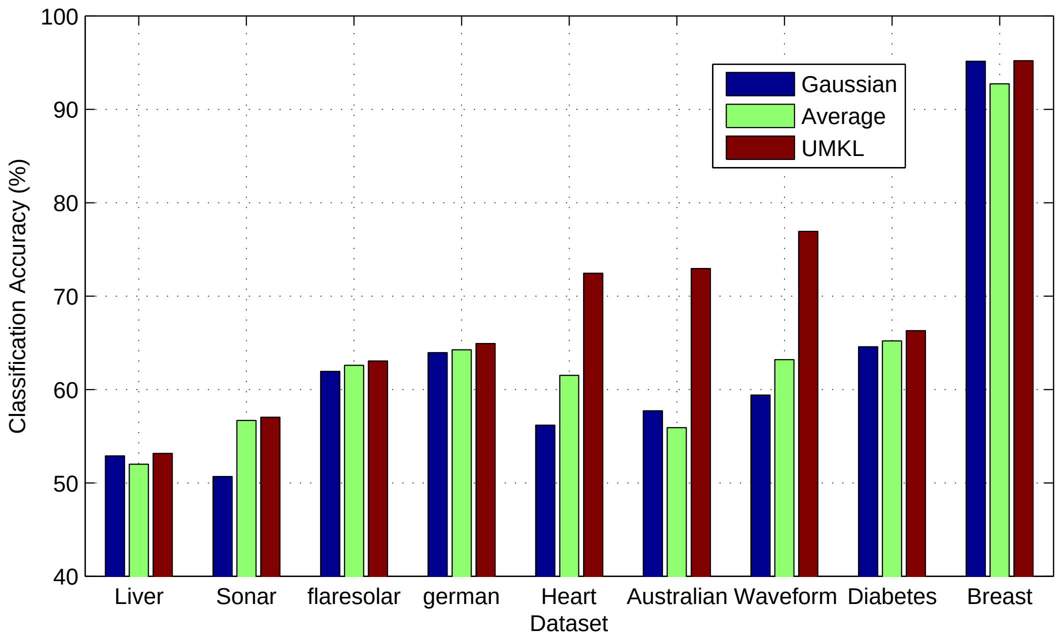


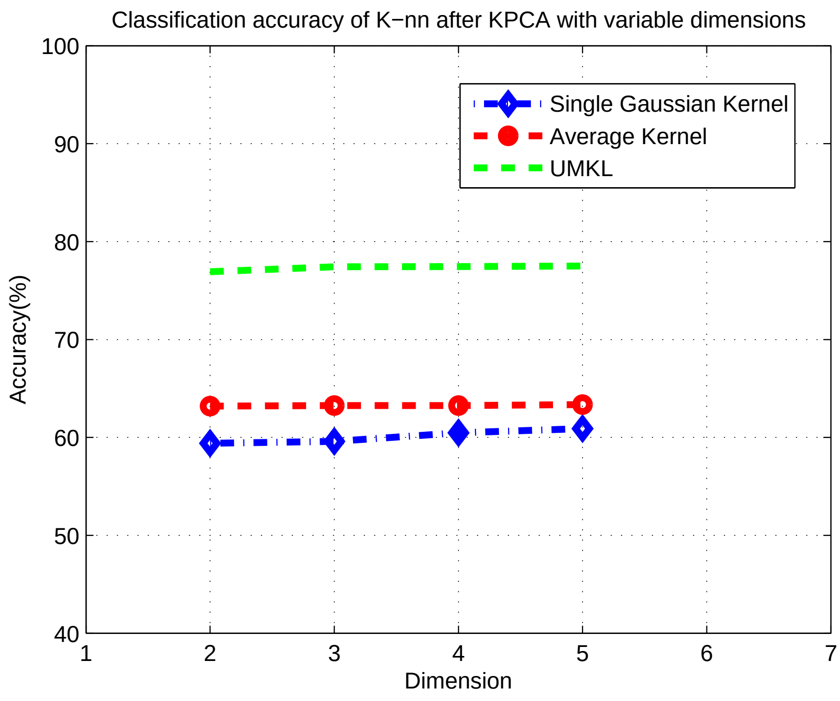
Figure 5.2: The classification accuracy of 5-NN after KPCA with Gaussian kernel with sigma=1,the average combination of multiple kernels, the kernel learned by UMKL on 9 benchmark data sets. The parameter $B$ of UMKL is fixed to 10.

data of varied numbers of dimensions. From the results, it is clear that UMKL performs consistently better than both the single gaussian kernel and the average kernel on all the cases of varied numbers of dimensions.

Finally, we also evaluate the classification performance by examining the effects of varied numbers of nearest neighbors, i.e., $k$, used in the $k$-NN classifiers. Figure 5.3.3 shows the detailed results of classification evaluation by varying the number of nearest neighbors on the heart dataset. The proposed UMKL algorithm consistently surpasses the other two baselines. Figure 5.3.3 gives most results on the other datasets. It is clear that the proposed UMKL algorithm always achieves the best performance for all the datasets.

## 5.4 Summary

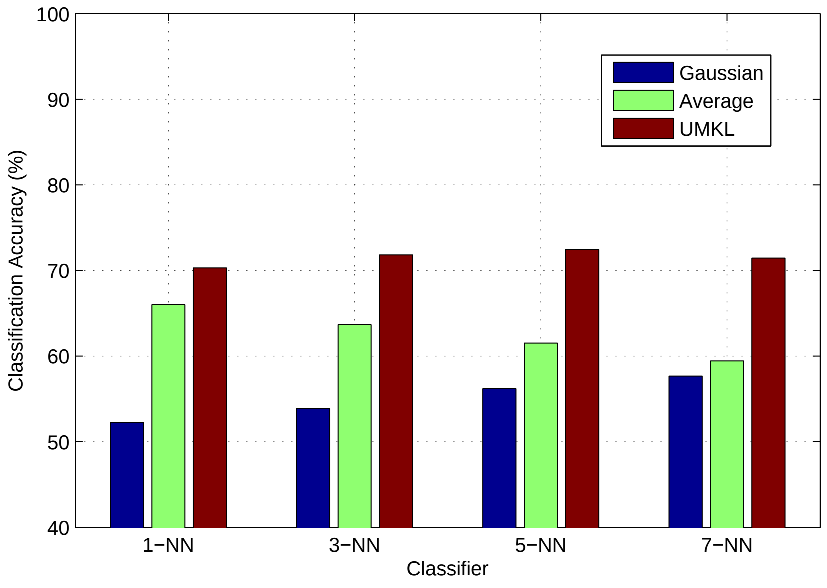
In this chapter we propose a novel unsupervised multiple kernel learning (UMKL) method, which is able to identify an appropriate linear combination of multiple kernels purely from



5.3.a: 5-NN classification after KPCA with varied number of dimensions

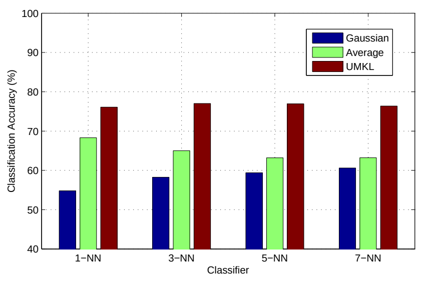
Figure 5.3: The classification accuracy of 5-NN after KPCA with Gaussian kernel with sigma=1,the average combination of multiple kernels, the kernel learned by UMKL with varied dimensions, on the waveform dataset.

unlabeled data without using class labels. In particular, the proposed UKML approach deems the kernel evaluation result of a data instance as its local coding, and adopts an efficient iterative algorithm to learn the kernel and the local set of basis simultaneously. We apply UKML for two machine learning tasks: (i) kernel learning for choosing appropriate kernels in a classifiction task and (ii) kernel selection in a kernel-based dimension reduction task based on the well-known Kernel PCA technique. Empirical results show that the classifier using the kernel learned by UMKL is comparable with the state-of-the-art supervised MKL algorithms, and the KPCA scheme using the kernel learned by UKML considerably outperforms the conventional approaches.

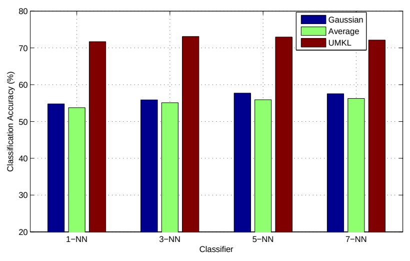


5.4.a: K-NN classification after KPCA with varied number of nearest neighbours

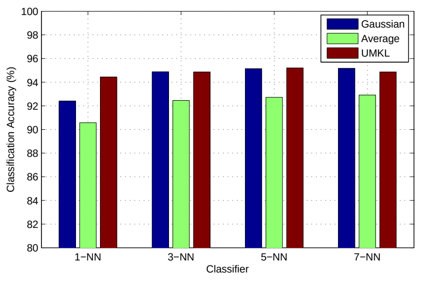
Figure 5.4: The classification accuracy of k-NN after KPCA with a gaussian kernel with sigma=1, the average combination of multiple kernels, the kernel learned by UMKL with varied number of nearest neighbours, on the heart dataset.



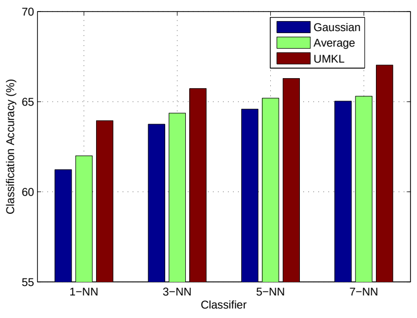
5.5.a: Waveform



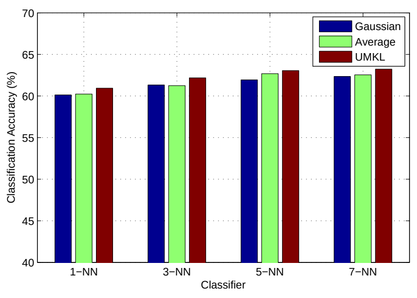
5.5.b: Australian



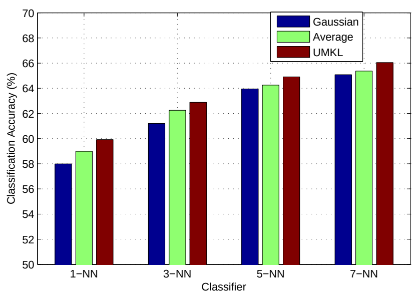
5.5.c: Breast



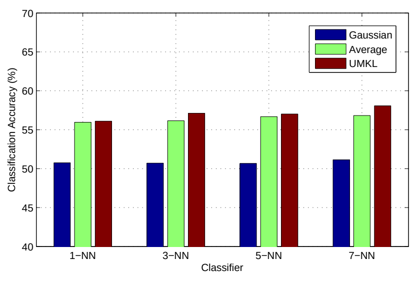
5.5.d: Diabetes



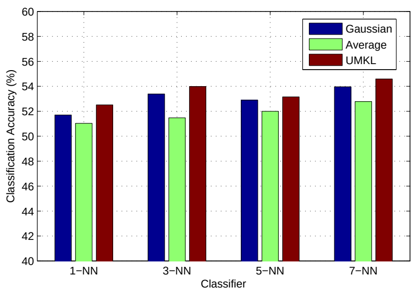
5.5.e: Flaresolar



5.5.f: German



5.5.g: Sonar



5.5.h: Liver

Figure 5.5: The classification accuracy of k-NN after KPCA with Gaussian kernel with sigma=1,the average combination of multiple kernels, the kernel learned by UMKL with varied number of nearest neighbours.

# Chapter 6

# Image Ranking by Non-Parametric Kernel Learning

Kernel methods have become one of the state-of-the-art methods in nowadays machine learning community and have been applied to a variety of applications. In this thesis, we study two important applications: image re-ranking with non-parametric kernel [ZH10] and social streng modeling with multiple kernel learning to rank [ZMH$^+$11].

Social image retrieval has become an emerging open problem in web image search. In this chapter, we address the challenge of text-based social image retrieval, which aims to identify a set of relevant social images related to a text-based query from a corpus of social images. Regular approaches for social image retrieval simply adopt typical text-based image retrieval techniques for retrieving the relevant social images based on the associated tags, which may suffer from noisy tags. In this chapter, we present a novel nonparametric kernel learning framework for social image ranking, which explores both textual and visual contents of social images for improving the ranking performance of social image retrieval tasks. Unlike existing methods that often adopt some fixed parametric kernel function, our framework learns a nonparametric kernel matrix that can effectively encode the information from both visual and textual domains. Although the proposed learning scheme is transductive, we suggest some solution to handle unseen data by warping the nonparametric kernel space to some input kernel function. Encouraging experimental results on real-world social image data exhibit the effectiveness of the proposed method.

Along with the popularity of various digital imaging devices and the advances of Internet technologies, more and more digital images and photos have been uploaded and shared on the World Wide Web. Unlike the situation of one decade ago, web images are playing a more and more important role nowadays. Web image search has become an active yet very challenging research problem.

One major difficulty for web image search is that most images are usually not annotated with proper tags, and many of them are even completely unlabeled. In addition, even for the annotated images, the associated tags could be noisy, irrelevant, and often incomplete for describing the contents of the images. This poses a challenge for typical approaches of web image search used by existing web search engines, which often simply apply regular text based retrieval techniques on web image search domains.

One possible direction to improve existing text-based web image search approaches is to explore the techniques of content-based image retrieval (CBIR) [SWS+00, SWS+00], which has been extensively studied in multimedia communities in the past decade. While CBIR techniques are able to retrieve images based on the low-level visual contents, one main drawback of CBIR lies in the difficulty of finding/drawing an appropriate query example for describing the user's search intention. This is probably why query-by-text image retrieval methods remain the most popular approach despite that their ordinary performance is far from perfect.

Instead of directly applying CBIR techniques, another alternative approach for improving the performance of text-based image retrieval is to investigate automatic image annotation techniques, which aims at annotating a web image with a set of relevant tags automatically. While automatic image annotation has been actively studied and shown to be able to improve the performance of text-based image retrieval in a small-scaled experimental testbed, it remains unknown for the effectiveness and scalability of the related techniques when being deployed in a massive scale environment, such as WWW images.

Recently, there is some emerging study for improving a text-based web image retrieval task by implicitly exploring the visual content without using query image examples [JB08]. They proposed a visual re-ranking method, termed "VisualRank", which is an extension of PageRank [BP98] by exploiting the visual information in the web image ranking task. Although encouraging results had been shown for improving regular text-based web image search, the VisualRank approach adopts a rigid visual similarity measure approach, which may suffer from the well-known challenge of semantic gap between low-level visual features and high-level semantic concepts [SWS+00]. This motivates our study in this chapter, which aims to improve VisualRank by a novel nonparametric kernel ranking scheme.

Besides, unlike previous studies on web image retrieval, our study considers *social images*, which are uploaded and shared by web users in social web sites. In contrast to generic web images, social images have rich user-generated contents, including tags provided by web users. Despite the good quality tags, noise remains an unavoidable issue for social image retrieval. Here our goal is to improve the performance of text-based social image retrieval by mining rich tag and visual contents through a novel nonparametric kernel based ranking approach.

In particular, our framework differs from the previous approaches that often assume some fixed kernel function for similarity measure. We propose to learn an optimized kernel from the rich annotated social images, aiming to bridge the semantic gap towards effective image similarity measure. To the best of our knowledge, this is the first kernel learning approach that learns to optimize a kernel for a web image ranking task.

As a summary, the key contributions of this chapter include:

- We propose a novel nonparametric kernel ranking framework that aims to learn an optimized kernel from both textual tag information and visual contents of social images;
- We present an efficient algorithm for nonparametric kernel ranking, which can efficiently learn nonparametric kernels for improving the VisualRank performance in a text-based social image retrieval task.
- We conduct an empirical study for comparing the proposed algorithm with the state-of-the-art method on a large-scale social image repository.

The rest of this section is organized as follows. Section 6.1 reviews some preliminaries of the VisualRank [JB08] framework, in which a similarity matrix plays a central role. Section 6.3 presents the proposed nonparametric kernel learning technique for learning kernels from social images, and studies its application for improving text-based social image retrieval tasks. Section 6.4 gives our empirical study.

## 6.1 Problem Setting

In general, a social image often contains rich contents, including user-generated tags, visual content, rating, etc. In our approach, we simplify the social image representation by considering only visual content and user-generated tags. Specifically, a social image is represented as $z_i = (x_i, t_i)$, where $x_i \in \mathcal{X}$ denotes the vector of its visual content, and $t_i \in \mathcal{T}$ denotes the vector of its associated tags. We further assume that the query space $\mathcal{Q}$ shares the same vocabulary space with the tag space $\mathcal{T}$, i.e., $\mathcal{T} := \mathcal{Q}$.

Consider a social image retrieval task, given some text-based query $q \in \mathcal{Q}$, our goal is to retrieve and rank the relevant social images from a social image repository $\mathcal{D} = \{z_i = (x_i, t_i), i = 1, \ldots, n\}$. Specifically, let us denoted $\mathcal{R}(q)$ and $\overline{\mathcal{R}}(q)$the set of relevant and irrelevant social images related to query $q$ respectively, the objective of the social image retrieval task is to find some ranking function $f : \mathcal{X} \times \mathcal{T} \to \mathcal{R}$ such that $f(z_i) > f(z_j)$ for any two images $z_i \in \mathcal{R}(q)$ and $z_j \in \overline{\mathcal{R}}(q)$.

Most approaches for web image retrieval by existing web search engines often formulate the ranking function $f$ by adopting some text-based retrieval model to rank the web images according to the associated tags. While these approaches are efficient by taking advantage of effective text indexing techniques, their ranking performance is not be always satisfied due to the noisy textual content that is common for WWW images. To address this challenge, the recent study [JB08] has proposed a visual re-ranking method for improving the web image search performance. Next we briefly review the Visual Ranking method.

## 6.2 VisualRank for Web Image Search

The key idea of VisualRank [JB08] was adapted from the intuition of PageRank [BP98], i.e., images similar to a relevant image are also relevant. Specifically, suppose we have a similarity matrix $K$, where $K_{ij}$ is the visual similarity between image $x_i$ and $x_j$. If the entry $K_{ij}$ has a large value, we deem that $x_i$ supports the importance of $x_j$ to a large extent. Therefore, by exploiting the visual information between pairs of images, we re-rank the images and hopefully improve the accuracy of retrieval results of text-based methods.

Let $VR \in \mathbb{R}^n$ be the ordering score of $n$ social images. A larger score indicates the image is more relevant, thus resulting a higher position in the ranking result. The VisualRank (VR) is iteratively defined as:

$$VR = K \times VR,$$

where $K$ is a normalized matrix such that $0 \leq K_{ij} \leq 1$ and each column sums to 1. When $K$ is symmetric, it becomes a doubly stochastic matrix [RC85]. From a random walk perspective, $K_{ij}$ measures the probability of $z_i$ randomly travels to $z_j$. Typically, a damping factor $\lambda_d$ is introduced to incorporate some prior ranking score $P$:

$$VR = \lambda_d K \times VR + (1 - \lambda_d)P, \text{where } P = \left[\frac{1}{n}\right]_{n\times 1}$$

The key to VisualRank is how to find a good similarity matrix $K$. The regular approach [JB08] is to simply calculate the similarity/distance matrix based on visual contents $\mathcal{X}$ (either local or global visual features) extracted from the images. Such an approach falls short when the visual similarity is not very effective, which is possible primarily due to the well-known semantic gap between low-level visual features and high-level semantic concepts. To address the limitation of the regular VisualRank approach, in this thesis, we present a novel nonparametric kernel ranking framework, which aims to learn a good visual kernel $K$ that best preserves the semantics by exploiting both visual and textual contents.

## 6.3 Nonparametric Kernel Ranking for Social Image Retrieval

In this section, we present a novel nonparametric kernel ranking method for social image retrieval. The key idea is to improve VisualRank by learning an optimized visual kernel.

### 6.3.1 Learning Semantics-Preserving Kernels

The goal of our task is to learn a visual kernel $K \in \mathbb{S}_+^{n\times n}$ for measuring similarity $k(z_i, z_j)$ between any two social images $z_i$ and $z_j$, which best preserves the semantics by exploring both textual and visual contents.

The first concern of our kernel learning task is how to effectively explore the textual contents of social images. One approach to this challenge is to optimize $K$ by maximizing the dependance between the two domains $\mathcal{X}$ and $T$. To this send, we suggest to employ the Hilbert-Schmidt Independence Criterion (HSIC) [GBSS05] for the dependence measure between $\mathcal{X}$ and $\mathcal{T}$, which is defined below.

**Definition 6.1 (Empirical HSIC)** *Let us denote by $Z = \{(x_1, t_1), \ldots, (x_n, t_n)\}$ a collection of $n$ independent observations drawn from the joint distribution $\mathcal{X}$ and $\mathcal{T}$, an estimator of HSIC is given by*

$$\mathrm{HSIC}(Z, \mathcal{X}, \mathcal{T}) = \frac{1}{(n-1)^2} tr\, KHK_tH,$$

*where $K$ and $K_t$ refers to the kernel matrices in visual space and textual space respectively, and $H = I - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top$.*

The HSIC measures the dependence between two random variables $x \in \mathcal{X}$ and $t \in \mathcal{T}$ by computing the square of the norm of the cross-covariance operator over the domain $\mathcal{X} \times \mathcal{T}$ in Hilbert Space. One can show the norm vanishes on the condition that $x$ and $t$ are independent. A large value of HSIC indicates a strong dependence with respect to the choice of the kernels. If we ignore the centering matrix $H$, it coincides with the concept of kernel target alignment [CSTEK01]. It has been successfully applied in some recent machine learning work [SSBG07].

In addition to the tag information, we expect the kernel $K$ should also encode the information of visual features. Combining the empirical HSIC and the visual feature concern, we formulate the semantics-preserving kernel learning problem as a nonparametric kernel learning task:

$$\begin{aligned} \max_K \quad & \mathrm{tr}\, KHK_tH \\ \text{s.t.} \quad & K_{ii} + K_{jj} - 2K_{ij} \le d_{ij}^2 \;\; \forall (i,j) \in \mathcal{N}, \\ & K \subset \mathbb{S}_+^{N\times N}, \end{aligned} \tag{6.1}$$

where $\mathbb{S}_+^{n\times n}$ denotes the space of positive semi-definite cones, and $\mathcal{N}$ is a set of all pairs $(i, j)$ where $x_i$ and $x_j$ are close to each other according to their visual distance, which is empirically obtained by checking if they are among each other's $k$ nearest neighbors based on some metric defined on the visual space $\mathcal{X}$.

**Remark.** Unlike a regular fixed kernel (e.g. RBF kernel) defined on visual space $X$, the above formulation finds an optimized kernel $K$ by (1) exploring the textual contents through $K_t$ and (2) encoding the visual information by enforcing the constraints according to the visual distance $d_{ij}$ in a unified framework that can effectively overcome the semantic gap challenge.

### 6.3.2 Efficient Kernel Learning Algorithm

The optimization of (6.1) is an SDP problem, which in general has to be solved in time complexity of $O(n^6)$ by a regular SDP solver. This intensive computation cost prohibits its application to large-scale real applications. To address this challenge, we propose an efficient algorithm that adopts a special SDP solver to explore the problem structures.

First of all, we notice that it may be too strong to enforce all distance constraints satisfied in the optimization problem (6.1). We reformulate the optimization by introducing slack variables $\xi$ (that are similar to the approach is soft margin SVM) as follows:

$$\begin{array}{ll} \min_{K,\xi} & -\text{tr } HKHK_t + \frac{C}{2}\sum \xi_{ij}^2 \\ \text{s.t.} & K \in \mathbb{S}_+^{n\times n}, \text{tr } KK \leq B, \\ & \text{tr } KA^{ij} \leq d_{ij}^2 + \xi_{ij},\ \forall (i,j) \in \mathcal{N} \end{array} \tag{6.2}$$

where $\xi$ penalizes the violation of the distance constraints, and $A^{ij}$ is a matrix of size $n \times n$ with only four nonzero elements, i.e., $A_{ii}^{ij} = A_{jj}^{ij} = 1$ and $A_{ij}^{ij} = A_{ji}^{ij} = -1$. In the above, we also introduce a normalization constraint $trKK \leq B$ to avoid $K$ being too large, where $B$ is a positive constant.

To solve the optimization, we introduce dual variables $\alpha$ for the slack variables $\xi$ and arrive at the partial Lagrangian (refer to [BV04] for standard techniques):

$$\begin{aligned} L(K;\alpha) &= -\text{tr } KHK_tH + \frac{C}{2}\sum \xi_{ij}^2 \\ &\quad + \sum \alpha_{ij}(\text{tr } KE^{ij} - d_{ij}^2 - \xi_{ij}) \end{aligned} \tag{6.3}$$

Taking derivatives of $L(K;\alpha)$ with respect to $\xi_{ij}$'s and set them to zero, we can rewrite the optimization in an equivalent form:

$$\begin{array}{ll} \max_\alpha \min_K & J(K,\alpha) \\ \text{s.t.} & K \in \mathbb{S}_+^{N\times N},\ \text{tr } KK \leq B \end{array} \tag{6.4}$$

where the objective function $J(K,\alpha)$ is formulated as:

$$\begin{aligned} J(K,\alpha) &= \text{tr } ((\sum_{ij} \alpha_{ij}A^{ij} - HK_tH)K) \\ &\quad - \sum_{ij} \alpha_{ij} d_{ij}^2 - \frac{1}{2C}\sum_{ij} \alpha_{ij}^2 \end{aligned} \tag{6.5}$$

Next we solve the above optimization using an iterative gradient projection approach, i.e.,(1) fixing $\alpha$ to update $K$, and then (2) fixing $K$ to update $\alpha$.

First of all, by fixing $\alpha$, we can find $K$ by reducing the optimization to the following

$$\max_K \mathrm{tr}\, AK \quad : \quad K \succeq \mathbf{0}, \ \ \mathrm{tr}\, KK \leq B, \tag{6.6}$$

where $A = \sum_{ij} \alpha_{ij} A^{ij} - HK_tH$. This optimization can be resolved by making use of the following theorem [ZTH09]:

**Theorem 6.35** *Consider the following optimization with a symmetric matrix $A$ and a positive constant $B$*

$$\max_K tr\, AK \quad : \quad K \succeq \mathbf{0}, \ \ tr\, KK \leq B, \tag{6.7}$$

*the optimal solution $K^*$ can be expressed as the following closed-form solution:*

$$K^* = A_+ \sqrt{\frac{B}{tr\,(A_+ A_+)}} \tag{6.8}$$

*where $A_+ = \pi_+(A)$ that projects $A$ onto the positive semi-definite space, i.e., $\pi_+(A) = V\Sigma^+V^\top$, where $\Sigma^+ = max(\Sigma, 0)$, $V$ is the matrix of $A$'s eigenvectors and $\Sigma$ is a diagonal matrix containing $A$'s eigenvalues.*

Second, when $K$ is fixed, we can update $\alpha$ by computing the gradient:

$$\alpha^{t+1} = \alpha^t + \eta(\mathrm{tr}\, KA^{ij} - \frac{1}{C}\alpha^t_{ij} - d^2_{ij})$$

where $\eta$ is a learning rate that may be changed at each step. Finally, we summarize the key steps of the proposed efficient algorithm in Algorithm 1.

**Protocol 5** Iterative Algorithm for Fast Kernel Learning.

**Input:** Training data set $Z$, kernel function $k_t$ on $\mathcal{T} \times \mathcal{T}$, pairwise constraint matrix $A$, parameters $C$ and $B$;

**Output:** $K^*$ and $\alpha$.

```
1: Construct the matrix $HK_tH$ by kernel $k_t$;
2: Initialize $\alpha$;
3: Set $A = HK_tH - \sum_{ij} \alpha^t_{ij} A^{ij}$;
4: Compute the closed-form solution of $K^*$ using (6.8);
5: Determine a learning rate $\eta$;
6: Update $\alpha^{t+1} \leftarrow \alpha^t + \eta(\mathrm{tr}\, K^* A^{ij} - \frac{1}{C}\alpha^t_{ij} - d^2_{ij})$;
7: Repeat steps 3-6 until convergence.
```

It can be shown that such an iterative gradient projection converges to the correct solution according to convex optimization theory [BX05].

### 6.3.3 Extension of Handling Unseen Data

The kernel learned above is purely nonparametric, which cannot handle unseen data directly. To address this limitation, we can extend the kernel $\widehat{K}$ to handle unseen data by some modification. Specially, we can adopt some kernel warping approach by warping the norm in Hilbert space by a regularizer depending on training data (both labeled and unlabeled) [SNB05].

Let $\mathcal{H}$ denote the original Hilbert space reproduced by kernel function $k(\cdot,\cdot)$, and $\widetilde{H}$ denote the deformed Hilbert space. In [SNB05], the authors assume the following relationship between the two Hilbert spaces:

$$\langle f, g\rangle_{\widetilde{\mathcal{H}}} = \langle f, g\rangle_H + f^\top G g$$

where $f(\cdot)$ and $g(\cdot)$ are two functions, $f = (f(x_1), ..., f(x_N))$ evaluates the function $f(\cdot)$ for both labeled and unlabeled data, and $G$ is the distance metric that captures the geometric relationship among all the data points. The deformation term $f^\top G g$ is introduced to assess the relationship between the functions $f(\cdot)$ and $g(\cdot)$ based on the observed data. Given an input kernel $k_x$, the explicit form of the new kernel function $\tilde{k}$ can be derived as below:

$$\tilde{k}(x, y) = k_x(x, y) - \kappa_y^\top (I + \widehat{L}K_x)^{-1}\widehat{L}\kappa_x, \tag{6.9}$$

where $K_x$ is the kernel matrix evaluated by the input kernel $k_x$, $L$ is chosen to be the graph Laplacian $\widehat{L} = \text{diag}\{\widehat{K}\cdot\mathbf{1}\} - \widehat{K}$ with respect to the learned transductive kernel $\widehat{K}$.

### 6.3.4 Implementation Issues: Further Speedup

Algorithm 1 makes the nonparametric kernel ranking method feasible for medium-scale data sets. However, in a real web-scale application, the size of training data $n$ can be very large, typically in millions or billions scale. To make our ranking scheme practical, we can further speed up our solution by the following strategies.

First, we notice that the main computation of the algorithm is the projection step by computing eigen-decomposition. Since the target kernel $K$ is often not full rank [ZTH09], it is possible to adopt some low rank matrix to approximate $\widehat{K} = VPV^\top$, where $V$ is of size $n \times m$, $m \ll n$. Specifically, let $L$ be the graph Laplacian of $\mathcal{X}$ using $n$ nearest neighbors, we choose $V$ to be the $m$ eigenvectors corresponding to the $m$ smallest eigenvalues of $L$. Using the low-rank representation of $K$, the main computation, i.e., the eigen-decomposition can be reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^3)$.

Second, the above kernel learning scheme in general learns a kernel matrix for all the social images from the database. In practice, we can speed up the solution by considering a query-dependent re-ranking approach, which is commonly adopted in web image retrieval, such as VisualRank [JB08], by considering that precision is often a more concern than recall in a web image retrieval task. Specifically, we employ a two-level social image retrieval scheme by first retrieving the relevant images using *tag* based ranking, and then applying the proposed non-parametric kernel ranking technique for re-ranking the subset of the social images.

## 6.4 Experiments

In this section, we empirically evaluate the performance of our nonparametric kernel ranking method for social image retrieval tasks.

### 6.4.1 Experimental Testbed

We create a large-scale social image data set with about 1,000,000 social images crawled from Flickr [1]. These social images contain rich information, including user-generated tags. In our text-based retrieval baseline, only associated tags are engaged for a social image retrieval task.

To form a benchmark testbed for performance evaluation, we pick a set of text based queries that covers a large range of generic WWW image search, including *animals*, *plants*, *humans*, *landmarks*, *natural sceneries*, and *human-made objects*. For the set of queries, we employ about 10 staffs to manually label the relevance of the top retrieved social images for each query to indicate if the retrieved image is relevant to the query.

We evaluate our non-parameter learning scheme in a query-dependent way. The list of 20 test queries are shown in Table 6.2.

### 6.4.2 Compared Methods and Experimental Setup

We adopt the VisualRank framework for social image retrieval and compare the following kernels:

- Raw: the text-based retrieval method. We simply compute the similarity between a query and the tag of some image. No re-ranking algorithm is performed. For single term query, there could be many images having the same score;
- Text: The similarity $k(x_i, x_j)$ is computed by linear kernel among the tags with tf-idf weighting;
- Linear: $k(x_i, x_j) = x_i^\top x_j$;

[1]http://www.flickr.com/

Table 6.1: The statistics of the images used in our experiments.

| | #Images | #Tags | #Tags/#Image |
|---|---|---|---|
| training | 25,819 | 557,160 | 21.58 |

Table 6.2: The statistics of the queries in the data set.

| TestQuery | #RelDoc | TestQuery | #RelDoc |
|---|---|---|---|
| eiffel tower | 919 | Barack Obama | 88 |
| great wall | 57 | Big Ben | 200 |
| red car | 477 | pyramid | 229 |
| airplane | 771 | Ferrari | 401 |
| lily | 452 | bicycle | 1475 |
| banana | 124 | sunflower | 798 |
| fruit orange | 266 | strawberry | 715 |
| singing | 945 | panda | 1512 |
| lion | 1308 | sheep | 487 |
| elephant | 739 | eagle | 664 |

- Polynomial: $k(x_i, x_j) = (1 + x_i^\top x_j)^d$;
- RBF: $k(x_i, x_j) = \exp\{-\|x_i - x_j\|^2/\sigma\}$;
- NPK: the proposed nonparametric kernel in this chapter.

The hyper-parameters of the above kernels are tuned by a set of validation queries. All experiments were conducted on a Windows PC with 3.4GHz 32bit CPU and 3GB RAM.

### 6.4.3 Feature Extraction and Evaluation Metric

For each social image in our data set, we must extract features from the tag and image domains.

For textual features, we use *normalized tf-idf* feature for each tag in the vocabulary. Assume each tag occurs at most one time for each image, the $j$-th dimension for $t_i$ is simplified as

$$t_{ij} = \frac{\delta_{ij} idf_j}{\sqrt{\sum_{j=1}^{d_1} (\delta_{ij} idf_j)^2}},$$

here $\delta_{ij}$ indicates whether the tag $t_{ij}$ appears in the tag $t_i$ of image $x_i$, $d_1 = |\mathcal{T}|$ is the vocabulary size. We filtered out non-English and extremely low frequent ones. The inverse document frequency is $idf_j = \log(r_j)$, where $r_j$ is the number of images that is tagged by $t_j$.

For visual features, we extract four kinds of effective global features that have been shown effective for previous CBIR studies [ZHLY08].

- **Grid Color Moment:** An image is partitioned into $3 \times 3$ grids. For each grid, we extract three kinds of color moments: color mean, color variance and color skewness in each color channel (R, G, and B), respectively. Thus, an 81-dimensional grid color moment vector is adopted for color features.
- **Local Binary Pattern:** The local binary pattern[OPH96] is defined as a gray-scale invariant texture measure, derived from a general definition of texture in a local neighborhood. We adopt a 59-dimensional LBP histogram vector.
- **Gabor Wavelets Texture:**To extract Gabor texture features, each image is first scaled to $64 \times 64$ pixels. The Gabor wavelet transform[LVB$^+$93] is then applied on the scaled image with 5 levels and 8 orientations, which results in 40 subimages. For each subimage, 3 moments are calculated: mean, variance and skewness. Thus, a 120-dimensional vector is used for Gabor texture features.
- **Edge:** An edge orientation histogram is extracted for each image. We first convert an image into a gray image, and then employ a Canny edge detector to obtain the edge map for computing the edge orientation histogram. The edge orientation histogram is quantized into 36 bins of 10 degrees each. An additional bin is used to count the number of pixels without edge information. Hence, a 37-dimensional vector is used for shape features.

In total, a 297-dimensional vector is used to represent the visual features for each image in the data sets.

Finally, for performance evaluation metric, we follow VisualRank[JB08] and employ the Top(n) accuracy. It is the percentage $\Pr(n)$ of relevant images ranked within the top $n$ positions by the retrieval model. Therefore, TOP $n$ evaluates the possibility that a user would locate the relevant materials on the first $n$- result page of a search engine.

### 6.4.4 Experiment I: Retrieval Accuracy

The detailed TopN retrieval accuracy is presented in Table 6.4.4, from which we can draw several observations.

First, among the four kinds of similarity measures, our nonparametric kernel yields the best performance. Especially for the top 3 and top 5 accuracy, NPK has a accuracy gain of greater than 5 percentage. For top 10-100 accuracy, it also has significant better

Table 6.3: The TopN re-rank accuracy (%) of different similarity measures plugged into VisualRank framework.

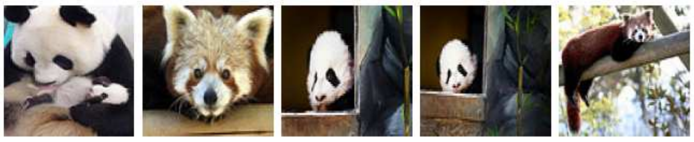
| | Raw | Text | Linear | Poly | RBF | NPK |
|---|---|---|---|---|---|---|
| Top3 | 51.7 | 61.7 | 50.0 | 50.0 | 65.0 | **70.0** |
| Top5 | 51.0 | 58.0 | 46.0 | 52.0 | 60.0 | **66.0** |
| Top10 | 46.0 | 59.0 | 47.0 | 52.0 | 58.5 | **63.0** |
| Top20 | 45.3 | 59.0 | 49.8 | 50.8 | 60.8 | **68.3** |
| Top50 | 48.4 | 55.8 | 48.8 | 50.6 | 59.0 | **65.0** |
| Top100 | 50.4 | 51.7 | 48.0 | 50.1 | 56.5 | **62.9** |

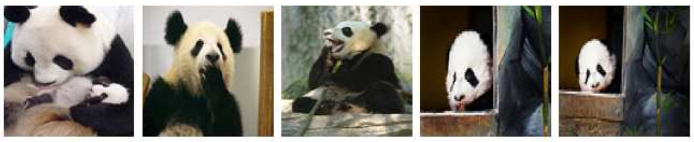
performance. We make the following conjectures about the merits of NPK: It is known that CBIR suffers from the semantic gap hindrance, that is, the visual computation among images may not reflect high-level concepts well. Parametric kernels defined on the image space also inherit this drawback. The proposed nonparametric kernel framework maximizes the dependency between image space and tag space. It provides a systematic method to incorporate the textual view and the visual view of social images such that the essential similarity between them are effectively captured. When the ranking scores are propagating with this similarity measure, the final scores are better than the ones obtained from off-the-shelf parametric kernels.

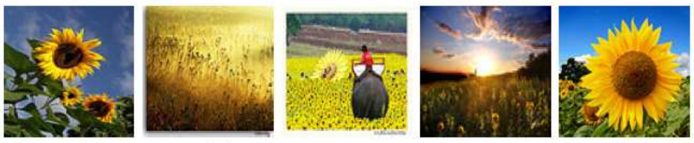
Second, with inappropriate kernels, the re-ranking may not necessarily yield better result comparing with raw text-based retrieval method. For example, the linear kernel deteriorates the accuracy at top 5 ranked images as high as 5 percentage. Both linear kernel and polynomial kernel hurts the top 3 accuracy. Moreover, in all cases, these two methods do not result in any significant better ranking accuracy. This phenomenon implies the importance of a good similarity measure among social images for the success of the visual rank framework. Since construction of this measure is left to users, people must pay attention to the kernel selection and be aware of the performance.

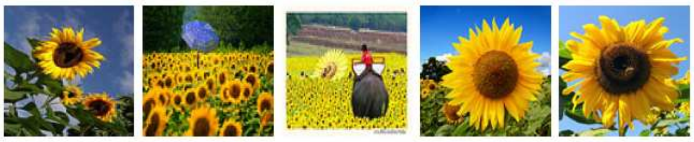
Last, both the Gaussian kernel and non-parametric kernel based re-ranking scheme outperforms raw text-based retrieval method under the top N accuracy measure, which implies that pure text-based retrieval may not be sufficient. We conjecture the possible limitations: 1) the annotated tags are incomplete. For example, the object *Ferrari* is essentially a *red car*, and a red car could be a Ferrari specifically. However, it is not possible to require people to label such ambiguous concepts accurately and completely. Therefore, textual information may not be good enough to replace visual contents in nature. 2) the annotated tags could be noisy. It is not practical to make users follow some universal and standard tag vocabulary. People may use different terms to describe the same object. Typos or completely unrelated tags could be added to the tag set. For example, people often add the descriptions of camera devices to the tag set.

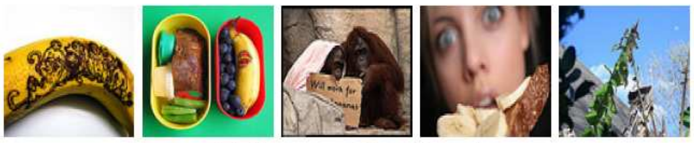
Finally, we also show some examples of qualitative ranking results in Figure 6.1. Comparing with the text-based baseline approach, the proposed NPK approach mostly can return more relevant images in the top retrieved examples. These results again validate the proposed NPK ranking approach is able to improve over the baseline.


(a) Panda: Text-based


(b) Panda: NPK-based


(c) sunflower: Text-based


(d) sunflower: NPK-based


(e) Banana: Text-based

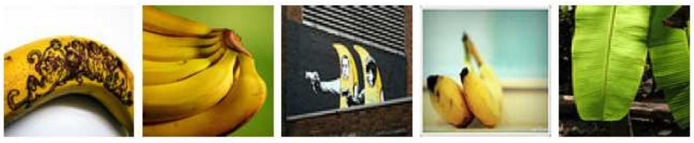

(f) Banana: NPK-based

Figure 6.1: Qualitative re-ranking performance by the proposed NPK learning method.

### 6.4.5 Experiment II: Computational Cost

The retrieval time is crucial for IR systems. In our non-parametric learning framework, we propose a fast solution avoiding general semi-definite programming solvers. In this section, we evaluate the time cost of the following solvers:

- **SDP**: We solve (6.5) using SDP solvers. However, the time complexity of SDP is in general $O(n^6)$, which prohibits the learning from real applications. Here we adopt the approximation method in Section 6.3.4 (see also [SSBG07]).
- **NPK**: The proposed method described in Algorithm 5 with the same approximation scheme.

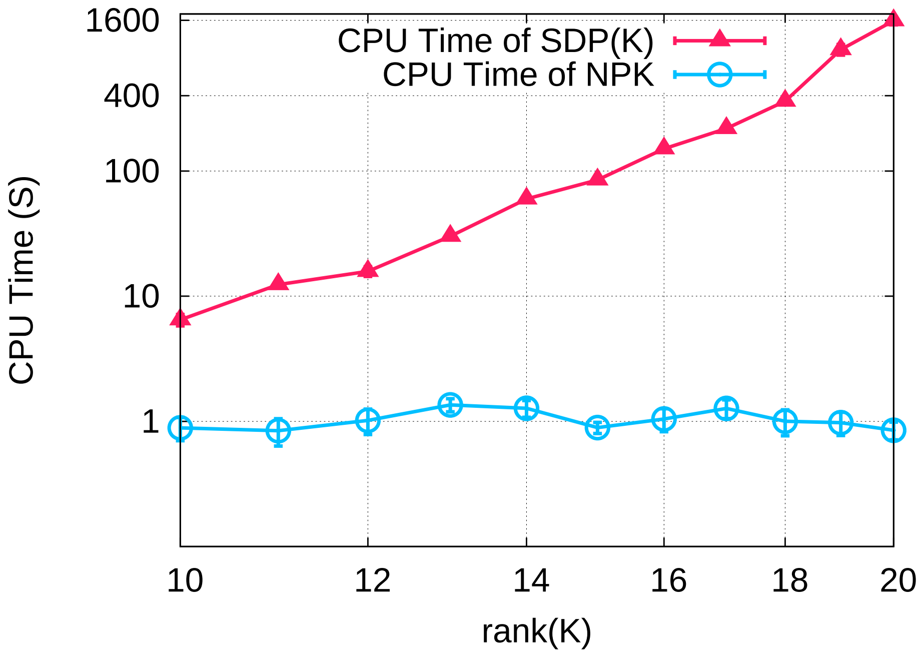


Figure 6.2: Log-scaled computation time of SDP and NPK on the query *Big Ben* which contains 200 images with the low-rank approximation scheme.

We plot the computational time of these two methods by Figure 6.4.5. We draw the following two observations: 1) The NPK method cost only around 1 second for a data set around 200 images, which is practical for online applications. More importantly, the cost does not crease with the rank significantly. For frequent queries, we could pre-compute the ranking result and cache it to speed up the retrieval time for real IR systems; 2) The complexity of SDP is verified by the exponential shaped curve. As the rank increases, it becomes infeasible easily.

## 6.5 Related Work

Our work is related to several branches of research topics: content-based annotation and retrieval, text-based social image retrieval, and nonparametric kernel learning.

Content-based methods for auto-image annotation and retrieval have been extensively studied in multimedia area. The earlier studies on CBIR have focused on investigating effective low-level features for image representation [SWS$^{+}$00]. While various features have been proposed, effective CBIR methods remain a challenging research problem till now, which is primarily due to the well-known semantic gap between the low-level features and high-level semantic concepts. To overcome the semantic gap challenge, much research efforts have been paid on improving the interactive image retrieval performance using relevance feedback techniques [RHOM98]. Our work is in general related to CBIR as we also exploit the visual contents of images, but is also different in that we focus on improving text-based image retrieval performance without explicit query image example while regular CBIR often focuses on solving example-based image retrieval tasks without text.

In addition to CBIR, multimedia researchers have actively explored content-based methods for auto-image annotation. By annotating an image with some machine learning model automatically, we are able to enable a large amount of unlabeled images indexed and searchable by existing text based image search engines. a variety of techniques have been proposed for auto-image annotation in recent years [DGLW06, WZZ08, LJY$^{+}$08, WHJ$^{+}$09]. In general, auto-image annotation can be viewed as an intermediate task for a generic web image retrieval task. Our work is however different from the annotation-based retrieval approach, in that our focus is not on improving the intermediate annotation performance, but for optimizing the retrieval performance directly.

The text-based image retrieval problem is about to select images related to a text query, which can be more clearly stated as: we rank the related images $\{x^{+}\}$ on top of unrelated ones $\{x^{-}\}$ for a query $q$ according to their relevance score. Thus our work closely related to learning to rank [Joa02, BSR$^{+}$05, BRL06, CQL$^{+}$07, GB08] and social image retrieval [LJH03, CHL$^{+}$04, JB08, GB08, HLZ$^{+}$04, DGLW06], the goal of which is to learn a model that outputs an order over a set of documents. For our proposed algorithm, we try to refining the results of some ranking model instead of the ranking framework, though it could be plugged into some kernel-based discriminative ranking model, like [GB08]. A lot of methods have been proposed for improving regular text-based web image retrieval performance, including re-ranking by relevance model [LJH03], clustering based re-ranking [CHL$^{+}$04], and the VisualRank approach by modifying PageRank [JB08]. Our work is mainly motivated by the VisualRank approach for overcoming its limitation through a kernel based learning approach, where a similarity matrix plays a central role for propagating the ranking scores among images. Moreover, the learned kernel can be plugged into other framework, for example, [GB08].

Our algorithm closely relates to the subject *kernel learning*[LCB+04], which is enjoying great popularity in machine learning community. Different from traditional parametric kernel combinations [LCB+04], researchers have proposed *nonparametric kernel learning* (NPK) algorithms, of which the output is just the kernel matrix [KSD06, HJL07, ZTH09]. It is known that each positive semi-definite kernel matrix $K$ (p.s.d.) corresponds to the evaluation of some underlying Mercer kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Therefore, people never need to find the explicit expression of $k$ in the transductive setting.

The most important merit of NPK is that it provides more flexibility to fit the available data since it does not require any parametric function. It often produces state-of-the-art empirical results for both classification and clustering. To the best of our knowledge, there is no existing studies in machine learning that have addressed web image retrieval by optimizing kernel in the ranking task. Our work focuses on adapting the recent advances of nonparametric kernel learning techniques with applications to the social image ranking task. However, due to the p.s.d. constraint over $K$, the learning often results in a semi-definite program [BV04], which has the time complexity of $O(N^6)$. Such heavy computation makes NPK prohibitive for large scale application. Very recently, Zhuang et. al. proposed an iterative algorithm which speeds up the kernel learning significantly. Based on their work, we derive an similar fast algorithm to solve our optimization problem. Together with some tricks, it works for a real data set consisting of more than 60,000 images.

## 6.6 Summary

In this chapter, we present a novel nonparametric kernel ranking approach to an emerging application of social image retrieval. Unlike the previous studies that may considerably suffer from the semantic gap by some rigid visual similarity function for web image re-ranking, our new solution aims to overcome the semantic gap issue by learning an effective kernel from mining both textual tags and visual images in a unified optimization framework. Encouraging experimental results on a large social image testbed validates the efficacy of the proposed method. Future work will explore other more effective visual features, such as SIFT [Low04], which poses a great challenge for being applied in a large-scale social image collection.

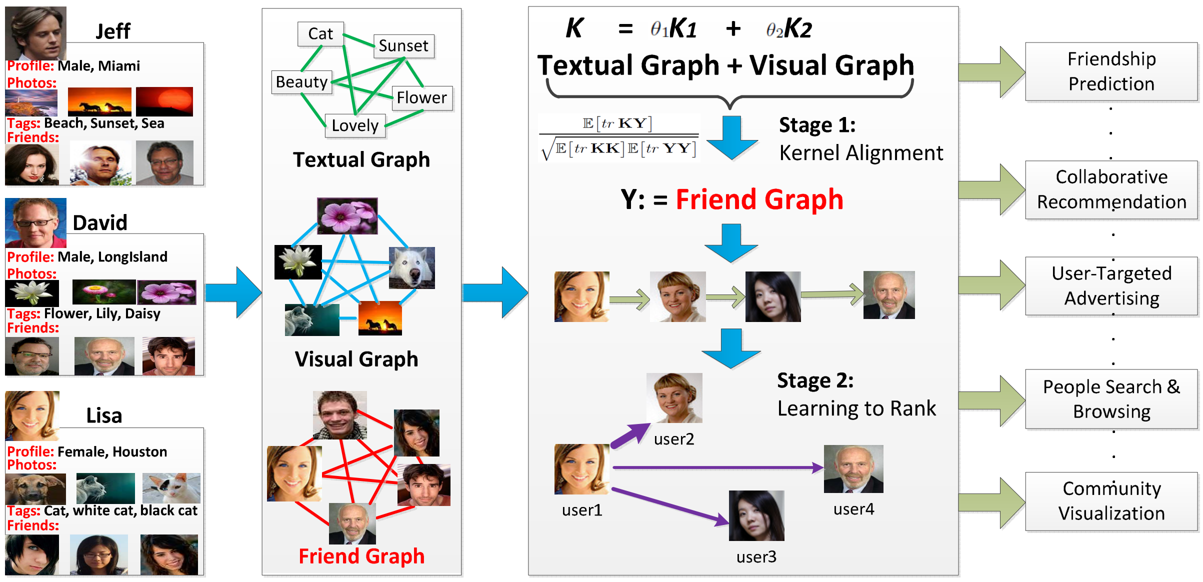
# Chapter 7

# Kernel-Based Social Strength Modeling

In this chapter, we apply kernel learning techniques to address the challenging problem of Social Strength Modeling (SSM) of users in a social media community. In particular, we take Flickr—the most popular online photo sharing community—as an example, in which users are sharing their experiences through substantial amounts of multimodal contents (e.g., photos, tags, geo-locations, friend lists) and social behaviors (e.g., commenting and joining interest groups). Such heterogeneous data in Flickr bring opportunities yet challenges to research community for SSM. One of the key issues in SSM is how to effectively explore the heterogeneous data and how to optimally combine them to measure the social strength. In this chapter, we present a kernel-based learning to rank framework for inferring the social strength of Flickr users. In the first stage, we employ kernel target alignment algorithm to integrate heterogeneous data into a holistic similarity space. With the learned kernel, in the second stage, we rectify the pair-wise learning to rank framework to estimate the social strength among Flickr users. By learning the social strength graph, we are able to conduct collaborative recommendation and collective classification on a large-scale data set crawled from Flickr. The promising results show that the learning-based approach is effective for SSM. Although focused on Flickr, our technique could be naturally applied to model social strength of users in other social media community, such as Facebook, Twitter, and so on.

## 7.1 Problem Setting

Social network mining (SNM), from communication networks, to friendship networks, to professional and organizational networks, has attracted a surge of interests in both industrial and academic communities. Most traditional studies focus on detecting *binary* relation ties between people (e.g., friends or not). Such a coarse indicator is not precise



$$\frac{\mathbb{E}[tr\,\mathbf{KY}]}{\sqrt{\mathbb{E}[tr\,\mathbf{KK}]\,\mathbb{E}[tr\,\mathbf{YY}]}}$$

Figure 7.1: Illustration of our kernel based learning to rank framework. The first panel shows the data of three users, based on which we have three graphs in the second panel (we omit other possible graphs for illustration purpose). In the third panel, we first learn the weight $\boldsymbol{\theta}$ by maximally aligning the combination to the friend graph. Then we adopt learning to rank framework with logistic loss to estimate the social strength.

enough to give insight about the strength of social relationship among people. Some recent work, e.g., the study in [XNR10], has attempted to address the problem of modeling the strength of social connections instead of simple binary linkage prediction. Inferring precise social strength can facilitate a variety of applications, including friendship linkage prediction, item recommendation, social search, and so on.

In this chapter, we investigate the challenging problem of Social Strength Modeling (SSM) of users in social media communities. In particular, we adopt Flickr, the most popular and largest online photo sharing portal, as the social media platform in our study. Flickr contains rich user-generated contents, including sharing photos, user-annotated tags, comments, etc. Analogues to other social networking sites, e.g., *Facebook* and *LinkedIn*, each Flickr user can add other users into contact list to indicate their friendship. Users can also create and join interest groups where users share photos of common interests. Besides the explicit mutual linkage relationship between users, the uploaded photos and their associated metadata (tags, comments, etc.) can also be leveraged to infer the similarity and implicit relationship between users. The versatile information available on Flickr poses a unique opportunity yet challenges for the research on SSM.

One key challenge of SSM is to effectively explore and combine heterogeneous data from multiple modalities to measure the social strength in a principled manner. Previous work on Flickr data mining has predominately focused on image-only or tag-only analysis. The other rich metadata has not been well exploited. To overcome the limitation of existing work, we present a novel framework for social strength modeling by unifying multi-modal heterogeneous data through a kernel-based machine learning approach. In particular, we suggest to use *kernel machine* [CV95] to compute user similarity between users in each modality of Flickr data, and further propose a two-stage learning scheme to measure social strength by kernel-based learning techniques.

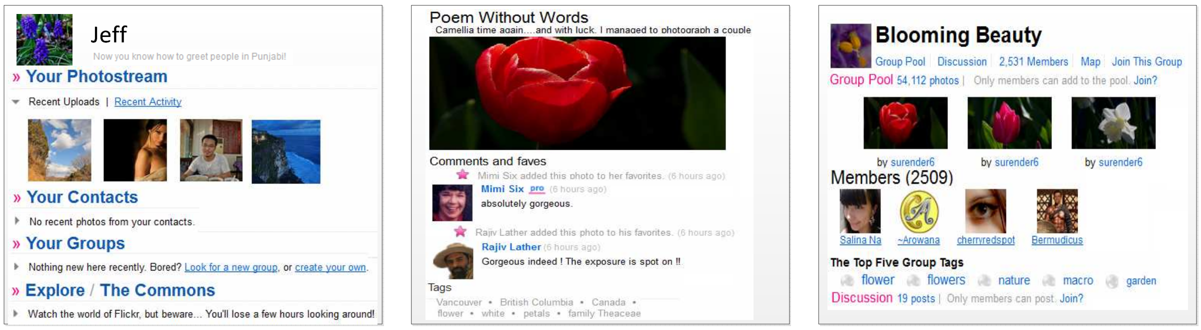
Specifically, we assume the final proximity graph of users is a linear combination of multiple proximity graphs derived from all the modalities in Flickr. Each proximity graph is computed by defining a kernel function on each modality. At the first learning stage, we propose to combine the multiple proximity graphs by learning the optimal combination by following the kernel target alignment principle in kernel learning theory [CMR10b, CSTEK01]. At the second learning stage, we propose a kernel-based learning to rank the social strength of users with the optimal kernel learned from the previous stage. The proposed two-stage learning approach is able to model the social strength of users by exploring multi-modal heterogeneous contents in social media communities in a systematic and comprehensive way. It is worth noticing that although we take Flickr as a particular example of social media community (as Flickr opens a public way to access its rich metadata, which are more comprehensive and formal than other social sites), the proposed learning approach can be applied to any kind of community such as Facebook and LinkedIn, in which users are also associated with multi-modal metadata.

By learning the continuous social strength between users, our technique can facilitate a number of real-world social media applications. In particular, we apply our technique to devise interest item recommendation and collective classification social tasks. To summarize, this chapter makes the following major contributions:

- We study the problem of modeling social network strength of users in a social media community. To the best of our knowledge, this is the first work focused on this particular topic in multimedia research community.

- We propose a novel two-stage kernel-based learning framework for social strength modeling, which effectively integrates heterogeneous data by optimally combining multiple kernels, and learns to rank social strength of users by a kernel-based learning to rank approach.

- We conduct extensive experiments to evaluate the performance of our technique on a large-scale real-world dataset, and found encouraging insights and knowledge about social connections in social multimedia platform like Flickr.


Jeff

Now you know how to greet people in Punjabi!

» Your Photostream

Recent Uploads | Recent Activity

» Your Contacts

No recent photos from your contacts.

» Your Groups

Nothing new here recently. Bored? Look for a new group, or create your own.

» Explore / The Commons

Watch the world of Flickr, but beware... You'll lose a few hours looking around!

Poem Without Words

Camellia time again....and with luck. I managed to photograph a couple

Comments and faves

Mimi Six added this photo to her favorites. (6 hours ago)

Mimi Six pro (6 hours ago)

absolutely gorgeous.

Rajiv Lather added this photo to his favorites. (6 hours ago)

Rajiv Lather (6 hours ago)

Gorgeous indeed ! The exposure is spot on !!

Tags

Vancouver • British Columbia • Canada • flower • white • petals • family Theaceae

Blooming Beauty

Group Pool | Discussion | 2,531 Members | Map | Join This Group

Group Pool 54,112 photos | Only members can add to the pool. Join?

by surender6 by surender6 by surender6

Members (2509)

Salina Na ~Arowana cherryredspot Bermudicus

The Top Five Group Tags

flower flowers nature macro garden

Discussion 19 posts | Only members can post. Join?

Figure 7.2: Illustration of the elements (both photos and the rich metadata) of a typical Flickr user (Left), a Flickr image (Middle), and Flickr interest group (Right).

The rest of the section is organized as follows. Section 7.1 formulates the problem of social strength modeling on Flickr data. Section 7.2 presents the proposed framework. Section 7.5 discusses our experiments. Section 7.6 reviews related works.

In this section, we give the problem definition of social strength modeling. We commence by summarizing the list of frequently used symbols in Table 1. For math notations used in this chapter, we use upper case letter to denote a variable, curlycue upper case letter to denote a space, and bold case letter denote a matrix or a vector.

Table 7.1: List of frequently used symbols.

| Symbol | Description |
|---|---|
| $\mathbb{N}_u$ | $\{1, \ldots, u\}$, a set of integers up to $u$ |
| $N_u$ | $\lvert\mathbb{N}_u\rvert$, the cardinality of $\mathbb{N}_u$ |
| $u_i$ | the $i$-th Flickr user |
| $\mathcal{U}$ | $\{u_i : i \in \mathbb{N}_u\}$, a collection of $N_u$ users |
| $X, T, D, L, \mathcal{C}$ | attributes of a Flickr photo: picture, tags, date, location, and comments, respectively |
| $\mathcal{P}, \mathcal{G}, \mathcal{N}$ | a collection of Flickr photos, groups, and friends, respectively |
| $\#$ | the number of some object |
| $K, \mathbf{K}$ | a similarity / kernel function and matrix |

First of all, we formally define the problem of social strength modeling with Flickr data below.

**Definition 7.1 (Social Strength Modeling)** *Given a collection of Flickr users $\mathcal{U}$, the goal of a* social strength modeling *problem is to learn a function $f : \mathcal{U} \times \mathcal{U} \to \mathbb{R}_+$ such that $f(u_i, u_j)$ measures the strength of social relationship between user $u_i$ and user $u_j$.*

In the above definition, a basic element is a Flickr user $u_i \in \mathcal{U}$, which can be defined below. to be

**Definition 7.2 (Flickr user)** *Each* Flickr user *$u_i \in \mathcal{U}$ is modeled as a three-dimensional tuple $u_i := [\mathcal{P}, \mathcal{N}, \mathcal{G}]$, where $\mathcal{P} = \{p_i : i \in \mathbb{N}_z\}$ is a collection of Flickr photos uploaded by user $u_i$, $\mathcal{N} \subset \mathcal{U}$ is the collection of users who appear in the contact list of user $u_i$, $\mathcal{G}$ is the collection of interest groups joined by user $u_i$.*

In the above definition, $\mathcal{N}$ and $\mathcal{G}$ indicate the explicit social relationship and organizations of Flickr users, and $\mathcal{P}$ may be beneficial to disclosing some implicit relationship between users. We discuss each of them below. The left picture in Figure 2 is an intuitive example of Flickr user. *Your Photostream, Your Contacts,* and *Your Groups* correspond to $\mathcal{P}$, $\mathcal{N}$, and $\mathcal{G}$, respectively.

The contact list $\mathcal{N}$ indicates the binary social ties between users. Note that the friend relationship given by $\mathcal{N}$ is often asymmetric, which can be naturally handled by our proposed learning to rank framework in the next section. We employ $\mathcal{N}$ as the training data of known social strength for building our model.

The set $\mathcal{G}$ is the interest groups the user $u_i$ has joined. For each $g \in \mathcal{G}$, it is created and self-organized by registered Flickr users. Users belonging to the same group tend to share photos of common interests. The right picture in Figure 2 show a popular interest group.

The photos $\mathcal{P}$ are useful to find implicit social relationship since they contain rich context information that expresses user interest and the communication between users. We formally define a Flickr photo below.

**Definition 7.3 (Flickr image)** *A* Flickr photo *is defined as 5-dimensional tuple $P := [X, T, D, L, \mathcal{C}]$, where*

- *$X \in \mathcal{X} \subset \mathbb{R}^{d_x}$ is a visual picture under some fixed-length feature representation, $d_x$ is the dimension of images;*
- *$T \in \mathcal{T} \subset \mathbb{R}^{d_t}$ is a vector representing the tags associated with $X$, $d_t$ is the size of tag vocabulary;*
- *$D$ is the date that $X$ is created;*
- *$L := [latitude, longitude] \in \mathbb{R}_+ \times \mathbb{R}_+$ is the location where $P$ is created;*
- *$\mathcal{C} := \{[U, C]_i \in \mathcal{U} \times \mathbb{R}^{d_t} : i \in \mathbb{N}_c\}$ is a collection of comments of $P$, where the first component $U$ is the user who posts the comment and the second component $C$ is the content of the comment.*

The rich context information $(T, D, L, \mathcal{C})$ besides the uploaded photo $X$ encode social communication information between users. The middle picture shows an example of Flickr users. In this chapter, we incorporate them into a unified discriminative framework to model the social strength.

## 7.2 Kernel-based Social Strength Modeling

In this section, we propose a kernel-based learning framework for social strength modeling.

### 7.2.1 Motivations

Referring to Figure 2 which illustrates the data we are dealing with here. The first challenge is how to combine such heterogeneous data effectively. Note that we can compute the similarity $K$ under different modality, which is akin to a kernel function in kernel machines including support vector machine and kernelized logistic regression. This inspires us the multiple kernel learning (MKL) scheme to integrate the multiple modalities, which has been actively studied in recent years since the pioneering work [LCB$^+$04]. Though not always yielding better results than single kernel chosen by cross-validation[Cor09], it is indeed useful in combining heterogeneous data sources (e.g., in bioinformatics[YFD$^+$10]). Therefore, we make use of the-state-of-the-art MKL algorithm [CMR10b] to weight each modality.

Second, we notice the social strength essentially ranks the degree of affinity between a pair of people. Therefore, we adopt the pair-wise learning to rank framework to further rectify the social strength based on the learned $K$ in the first stage. Comparing with generative models, such a discriminative model often enjoys better generalization ability [Vap98]. Moreover, it avoids latent variable assumption and parametric-form assumption of generative model functions, which makes the learning more compact and precise.

We first discuss how to measure similarity of Flickr users by defining a variety of kernel functions $\{K\}$ on different modalities of Flickr data. We then present a kernel learning technique to determine the optimal combination of multiple kernels following the *kernel target alignment* (KTA) principle [CMR10b, CSTEK01]. Finally, based on the learned kernel, we formulate the social strength modeling problem into a pair-wise kernel-based learning to rank task, which can be efficiently solved in an iterative manner as IVM [ZH01].

### 7.2.2 Kernels for Measuring User Similarity

For machine learning algorithms, the kernel trick is an important technique for mapping observations from a general set into a much higher- and possibly infinite-dimensional inner product space without ever having to compute the mapping explicitly. Typically, every kernel $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ on a set $S$ essentially defines the way of measuring similarity between data instances in the set $\mathcal{S}$. In this section, we present a series of candidate kernel functions in different modalities for measuring similarity of Flickr users, which will be further explored with other kernel methods for inferring social strength.

#### 7.2.2.1 Similarity in Visual Space

For visual feature representation, we adopt the bag-of-(visual) word model. Specifically, we first extract the SIFT descriptors [Low04]. All these descriptors are split into $d_x$ groups by the k-Mean clustering process. Given a photo, we assign each of its SIFT descriptors to a nearest cluster center. Then each image is converted into a fixed length vector $\mathbf{x} \in \mathbb{R}^{d_x}$, where $d_x$ is the size of visual dictionary. The $i$-th component of this vector counts the frequency of SIFT features assigned to cluster $i$. We measure image similarity on $\mathcal{X}$ by a Gaussian kernel:

$$s(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}.$$

where $\sigma$ is a kernel parameter. For a specific user, we employ the centroid of its images to represent its visual contents, i.e.,

$$\bar{u} = \sum_{\mathbf{x}_i \in u} \mathbf{x}_i / |u|,$$

where $|u|$ is the number of photos belonging to user $u$. Thus, the similarity between two users can be measured by:

$$K^1(u_i, u_j) := e^{-\|\bar{u_i} - \bar{u_j}\|^2/\sigma^2}.$$

It is difficult to obtain an optimal bandwidth parameter $\sigma$. We set it to be the average Euclidean distance empirically.

#### 7.2.2.2 Similarity in Textual Space

Note each uploaded photo can be annotated with a set of possible tags by users. We adopt the bag-of-word model to represent the textual information[BYRN99]. In particular, we collect all the tags and build a tag dictionary of size $d_t$. All the tags of an entity is converted into a fixed length vector in $\mathbb{R}^{d_t}$ by the traditional *tf-idf* weighing method. Here the inverse document frequency is the number of users containing that tag. We employ the normalized linear kernel which is widely used for text classification:

$$K^2(u_i, u_j) := \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\sqrt{\mathbf{x}_i^\top \mathbf{x}_i}\sqrt{\mathbf{x}_j^\top \mathbf{x}_j}}$$

Comparing with the visual kernel $K^1$, the tag-based kernel carries more semantic information.

#### 7.2.2.3 Similarity by Mutual Comments

The interaction between users reflects the social connection between each other. For example, if two users often post comments on each other's photos, probably they have strong social ties. In general, people who are friends in real life would communicate more frequently. We can collect the mutual comment information between users to construct an symmetric link graph, in which each vertex represents a user and the edge weight is the number of comments between two users. Thus we have

$$K^3(u_i, u_j) = \#\text{Comments between } u_i \text{ and } u_j.$$

When $i = j$, the kernel value is how many times $u_i$ has ever posted to itself.

#### 7.2.2.4 Similarity by Common Interest Groups

The Flickr users can create or join in *Flickr interest groups*, which consists of a collection of users who share common interest or taste on photos of specific styles. Such interest groups can help users to find people or photos of their interest. Researchers have proposed techniques (eg., [NGP08b, NAP$^+$09]) to analyze the structure and themes of the groups due to their importance in organizing Flickr users. Intuitively, people among which strong social connections exist are more likely to join the same interest groups as they may affect each other and share similar interest. Therefore, we can use the number of common interest groups to measure the similarity between people:

$$K_u^4(u_i, u_j) = \#\text{Groups both } u_i \text{ and } u_j \text{ joined.}$$

When $i = j$, the kernel value is the number of groups $u_i$ has joined.

#### 7.2.2.5 Similarity by Common Friends

Each Flickr user has a contact list, which can be deemed as "friends". When two users share many common friends, it is reasonable to infer that they have strong social connection to each other. Naturally, we can count this number as the kernel value to quantify this modality:

$$K^5(u_i, u_j) = \#\text{Friends belong to both } u_i\text{and}u_j.$$

#### 7.2.2.6 Similarity via Geo-tags

The social photos are often associated with geo-tags, which states the latitude and longitude where the photos are created. If two users have been traveled to the same place frequently, it is reasonable to conclude they are similar to each other. Similar to the representation in textual space, we use the bag of geo-tag to compute the similarity between users.

$$K^6(u_i, u_j) = \#\text{Locations where both } u_i \text{ and } u_j \text{ taken photos.}$$

At the diagonal position of $K^6$, the value is the number of places the user has been to. The raw geo-tags are in form of longitude and latitude pairs. We discrete it by dividing the whole globe into grids. Each location is represented by the lattice it belongs to.

#### 7.2.2.7 Similarity via Favorite Photos

One important function provided by Flickr is that users can mark their favorite photos (fave). Assuming the number of faves correlates to the social strength, we define

$$K^7(u_i, u_j) = \#\text{Photos both } u_i \text{ and } u_j \text{ favriate.}$$

The rationale underlying this assumption is that users who like the same photos tend to be friend. Apparently, this may not be true in real case. Through the KTA algorithm we learn the weights of these kernels, which provides us insight about which modalities are indeed informative for detecting social strength in multimedia communities.

The similarity measures $K_u^{1\sim7}$ are not necessarily positive semi-definite (p.s.d.). We can force these measures to be p.s.d. to construct kernels by adding a properly scaled identity matrix to the corresponding similarity matrix.

## 7.3 Optimal Combination of Multiple Kernels

In the above, we define various kernel functions for similarity measures in different modalities. The next challenge is to find the best way of combining these modalities, which plays a key role in social strength modeling. Specially, our goal is to determine a linear combination of multiple kernels in fusing all modalities for similarity measure:

$$K(u_i, u_j; \boldsymbol{\theta}) = \sum_{t=1}^{N_k} \theta_t K^t(u_i, u_j), \tag{7.1}$$

where $K^t$ is the kernel defined under the $t$-th view of the users, $N_k$ is the total number of views.

One naive way is to manually fix weights of different modalities, which however highly relies on domain knowledge and cannot find the optimal combination due to the heuristic manner. In this section, we present a kernel-based learning technique to find the optimal combination of multiple kernels by following the principle of kernel target alignment [CSTEK01].

Specially, given the target labels $\mathbf{Y}$, we can adopt the *kernel alignment* [CSTEK01] to measure the quality of kernel $\mathbf{K}$ with respect to the target label matrix $\mathbf{Y}$. The formal definition of kernel alignment is defined below:

**Definition 7.4 (Kernel Alignment)** *Let $\mathbf{K} \in \mathbb{R}^{N_u \times N_u}$ and $\mathbf{Y} \in \mathbb{R}^{N_u \times N_u}$ be two kernel matrices such that $\|\mathbf{K}\|_F \neq 0$ and $\|\mathbf{Y}\|_f \neq 0$. Then, the* alignment *between $\mathbf{K}$ and $\mathbf{Y}$ is defined by*

$$\rho(\mathbf{K}, \mathbf{Y}) = \frac{\mathbb{E}\big[tr\,\mathbf{KY}\big]}{\sqrt{\mathbb{E}\big[tr\,\mathbf{KK}\big]\mathbb{E}\big[tr\,\mathbf{YY}\big]}},$$

Here the expectation is taken over samples on which the kernel is evaluated. Note that the kernel matrices usually have to be centered [CMR10b], i.e., $\mathbb{E}[K_{ij}] = 0$. This centering step can be simply computed from

$$[\mathbf{K}]_{ij} := K_{ij} - \frac{1}{N_u}\sum_{i=1}^{N_u} K_{ij} - \frac{1}{N_u}\sum_{j=1}^{m} K_{ij} + \frac{1}{N_u^2}\sum_{i,j=1}^{N_u} K_{ij}.$$

Given the target graph represented by matrix $\mathbf{Y}$, we maximize the alignment $\rho$ over $\mathbf{K}$ to solve the kernel. The matrix $\mathbf{Y}$ is observed from the Flickr platform. For example, in a friend prediction task, $\mathbf{Y}$ is the mutual contact graph constructed from the profile of each Flickr users. Suppose we have $N_k$ modalities $\mathbf{K}^1, \ldots, \mathbf{K}^{N_k}$. We assume the target kernel is in form of $\mathbf{K} = \sum_{t=1}^{N_k} \theta_t \mathbf{K}^t$, where $0 \leq \theta_t \leq 1, \sum_t \|\theta_t\|_2 = 1$. Thus the target variable is reduced from $\mathbb{R}^{N_u \times N_u}$ to $\mathcal{M} := \big\{\|\boldsymbol{\theta}\|_2 = 1 \wedge \boldsymbol{\mu} \geq 0\big\}$. The following theorem guarantees the optimal solution can be computed immediately[CMR10b]:

**Theorem 7.36** *The solution $\boldsymbol{\theta}^*$ of the optimization problem*

$$\max_{\boldsymbol{\theta} \in \mathcal{M}} \rho(\mathbf{K}, \mathbf{Y}) \;:\; \mathbf{K} = \sum_{t=1}^{N_k} \theta_t \mathbf{K}^t$$

*is given by $\boldsymbol{\theta}^* = \boldsymbol{\theta}^*/\|\boldsymbol{\theta}^*\|$, where $\boldsymbol{\theta}^*$ is the solution of the following quadratic program:*

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \geq \mathbf{0}} \boldsymbol{\theta}^\top \mathbf{M} \boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \mathbf{a},$$

*where $\mathbf{a}$ is the vector $(\langle \mathbf{K}^1, \mathbf{Y}\rangle_F, \ldots, \langle \mathbf{K}^{N_k}, \mathbf{Y}\rangle_F)^\top$ and $\mathbf{M}$ is the matrix $M_{kl} := \langle \mathbf{K}^k, \mathbf{K}^l\rangle_F$, for $k, l \in [1, N_k]$.*

## 7.4 Kernel-based Learning to Rank

### 7.4.1 Framework

Let $z$ be a $\{0,1\}$-valued latent variable to indicate whether there is a connection between two users. The social connection strength inference is about to estimate the probability $P(z_{ij}=1|u_i,u_j)$. To this purpose, we build a logistic normal model based on training pairs. The connection between Flickr users come from the mutual contact $\mathcal{N}$ in Flickr user profile, i.e., $z_{ij}=1$ if and only if $u_j$ is in the contact list of $u_i$.

$$\min_{\mathbf{w}} \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{u_i,u_j\in\mathcal{U}} l(y_{ij}, \mathbf{w}^\top(\Phi(u_i)-\Phi(u_j))), \tag{7.2}$$

where we use a linear model $f(u_i,u_j)=\mathbf{w}^\top(\Phi(u_i)-\Phi(u_j))$ parameterized by $\mathbf{w}$ to predict the social strength between $u_i$ and $u_j$.

The label $y_{ij}=1-z_{ij}$ takes value in $\{0,1\}$ indicating the connection between $u_i$ and $u_j$. When $u_i$ has $u_j$ as a friend, we set $y_{ij}=0$. The function $\Phi(\cdot):\mathcal{U}\to\mathbb{R}^d$ maps a user $U$ to a feature representation under some specific view. The function $l(\cdot,\cdot):\mathbb{R}\times\mathbb{R}\to\mathbb{R}_+$ measures the loss of prediction. We adopt the logistic loss $l(y_1,y_1)=\ln(1+e^{-y_1y_2})$ instead of hinge loss in traditional SVM-based ranking model, as it can predict the probability of classification result. Thus we can estimate the social strength by

$$P(z=1|u_i,u_j)=e^{f(u_i,u_j)}/(1+e^{f(u_i,u_j)}).$$

According to the representor theorem[SHS01], the solution of the above problem can be expressed in form of $f(\hat{u}_i)=\sum_j \alpha_j K(\hat{u}_i,\hat{u}_j)$, where $\hat{u}_i := [u_{i1},u_{i2}]$ is an ordered pair of users, $f(\hat{u}_i)$ predicts the scoring indicating the strength between this pair of users. Thus the dual objective can be written as

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^{N_u}} \frac{1}{N_p}\mathbf{1}^\top \ln\left(\mathbf{1}+e^{-\mathbf{y}\circ(\mathbf{K}\boldsymbol{\alpha})}\right)+\frac{C}{2}\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}, \tag{7.3}$$

where $\mathbf{K}\in\mathbb{S}^{N_u^2\times N_u^2}$ is the kernel matrix, evaluated from a kernel function $K:(\mathcal{U}\times\mathcal{U})\times(\mathcal{U}\times\mathcal{U})\to\mathbb{R}$ defined on the user pairs.

The kernel of two pairs $\hat{u}_i$ and $\hat{u}_j$ is computed as

$$\begin{aligned} K(\hat{u}_i,\hat{u}_j) &= (\Phi(u_{i1})-\Phi(u_{i2}))^\top(\Phi(u_{j1})-\Phi(u_{j2})) \\ &= K_u(u_{i1},u_{j1})+K_u(u_{i2},u_{j2}) \\ &\quad -K_u(u_{i1},u_{j2})-K_u(u_{i2},u_{j1}). \end{aligned}$$

The dual objective (3) can be solved efficiently as an im- port vector machine (IVM)[ZH01].

### 7.4.2 Training Pair Selection

When a user $u_j$ appears in the contact list of $u_i$, it is likely that $u_j$ is a friend of $u_i$ in real life or that the content uploaded by $u_j$ is of the interest of $u_i$. Therefore, we can treat $(u_i, u_j)$ as a positive pair directly when training the model. However, it is not so trivial for the case of negative pairs. It is a common case that $u_i$ has not noticed his friend $u_j$ has registered. Thus $u_i$ does not add $u_j$ into his contact list. If we simply treat $(u_i, u_j)$ as negative training pairs, the learned model would fail to predict such latent friends effectively. Actually one of our important applications is to reveal such potential friends. It calls for elaborate strategies for sampling negative pairs.

Moreover, there are computational necessity for training pair sampling. Suppose each user has $N_k$ friends on average, we would have $O(N_k \times (N_u - N_k) \times N_u)$ training pairs over all. Such a large scale makes it prohibitive to apply learning to rank algorithm directly. On the other hand, some of the training pairs are not helpful for learning the models. We propose a two-stage scheme for sampling pairs.

First, we choose the pairs of users that are least likely to be friends to construct negative training pairs. To this end, for each user $u_i$, we sort the values $K^t(u_i, u_j)$, $1 \leq t \leq N_k$ in descending order. The users with small similarity to $u_i$ are excluded from the potential friend list of $u_i$. Since we have multiple $N_k$ kernels defined on users, it is unclear to adopt which one for the sampling purpose. Here we employ a very conservative scheme, that is, we choose the top $N_e$ users that appear in the top $N$ ($N > N_e$) users of $u_i$ under all the similarity measures $K^t$. For constructing positive pairs, we use $K^3$, i.e., the kernel counting the mutual comments, to determine the most frequently communicate with $u_i$ as positive pairs.

Second, we adopt the active sample selection scheme in IVM [ZH01] during the training phase. Due to the logistic loss function of (7.2), the solution is non-sparse. However, many of the samples are not necessary to yield a useful solution. We can actively select a pair to expand the set of support vectors at each iteration. In this way, the learning can be significantly sped up while the efficacy of the learned classifier is preserved [ZH01].

### 7.4.3 Applications

The proposed social strength modeling framework can benefit a variety of applications, including not limited to:

**Friendship Prediction.** In Flickr each user can add other registered users into his/her contact list. However, when users are known to each other in real life, or they share quite similar interest in photo styles, the friendship link may not exist explicitly due to the limited searching and browsing functionality. Our framework can predict the implicit links between Flickr users by leveraging the various content and context information.

Table 7.2: The statistics about our crawled Flick dataset.

| #user | #group | #image | #tag | #contact |
|---|---|---|---|---|
| 4,919 | 109,205 | 5,001,601 | 116,372 | 81,447 |

**Collaborative Recommendation.** It is very useful to improve user experience by recommending proper objects to users. For example, the interest groups and favorite photos. Such item recommendation tasks can be benefited from the modeled strength as the popularity of these terms are correlated with the social strength among people. Affinity propagation algorithms can be designed based on the learned social strength graph.

**User-Targeted Advertising.** Similar to the case of object recommendation, we can provide user-targeted advertising to a connected component consisting of similar users, so that the advertisements are relevant to user interest through a propagation among similar users.

**User Search & Browse.** We can rank the user search results according to their social strengths with the one who conducted the query. The user is more likely to find out the targets of his interest. This technique can compensate the lack of informative words matching with the query. Therefore, we expect the results based on traditional simple keyword matching can be greatly improved.

**Community Visualization.** The applications of visualizing people's local social network could be improved by scaling/shading links according to the estimated relationship strengths.

## 7.5 Experiments

In this section, we evaluated the proposed social strength modeling techniques on a real data set crawled from Flickr. We start from a random user as seed and expand the crawling according to its friend list in a bread first search manner. We stopped at 4,919 users as we consider such a scale is enough to work with to conduct reliable conclusions. Besides the photos, all the associated metadata, including the ones defined in Section 2, are downloaded in XML format. We evaluate the effect of social strength modeling on recommendation tasks.

### 7.5.1 Friend Recommendation

The essence of SSM is to infer the acquaintance between two people. Therefore, the most direct criterion of SSL is to measure the friend recommendation accuracy. It is also an

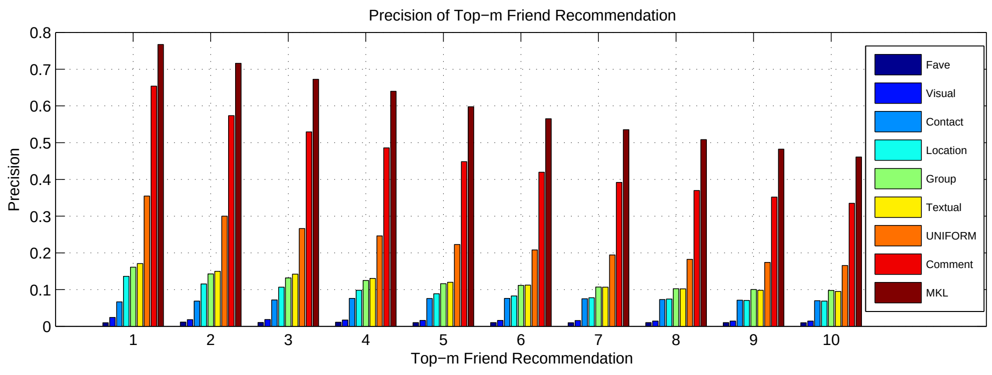


Figure 7.3: The average top-10 friend recommendation accuracy of single kernel and combined multiple kernels.

Table 7.3: The base kernels and the weights by KTA in the friend and group recommendation task.

| Kernel | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Name | Visual | Textual | Comment | Group | Contact | Location | Fave |
| Friend | .0042 | .1007 | .6488 | .0637 | .1042 | .0499 | .0285 |
| Group | .0183 | .1687 | .5065 | N/A | .2030 | .0397 | .0299 |

important application for social network sites (including Flickr, though it is not initially designed for social purpose).

We randomly choose 4,000 users for training purpose. The target kernel matrix $\mathbf{Y}$ in Section is the friend indicating matrix. The $i$-th row of $\mathbf{Y}$ is a binary vector indicating whether a user is in the contact list of user $u_i$. We adopt the learning to rank to infer the social strength based on kernels. For each kernel matrix $\mathbf{K}$, we normalize it by dividing each row with the maximal value at that row. Then we use $\mathbf{K} = (\mathbf{K} + \mathbf{K}^\top)/2$ to make it symmetric. For the ones not satisfying the positive semi-definite (p.s.d.) property (this can be examined by eigen-decomposition), we add $\delta\mathbf{I}$ to make them p.s.d. The valuated kernels include:

- *Single*: A single kernel defined in Section 7.2.2. By examining the kernel one-by-one, one can observe the effect of each modality clearly;
- *Uniform*: Each kernel is assigned the same kernel weight. This is the baseline showing the result of simple combination, which serves as baseline method;
- *KTA*: The kernel target alignment method introduced in Section 7.5.1.

Given a test user, we sort the values in descending order and extract the top users as recommended friends. The learned weights of the kernels are presented in Table 3. The self-explanatory name of kernel implies its definition in Section 7.2.2. The top-10 friend recommendation results are plotted in Figure 3. We draw several observations from the results.

First, KTA-MKLR yields the best performance among all the evaluated kernels. Its top-1 accuracy approaches 80%, and top-10 accuracy approaches 50%. Such relatively high accuracy confirms that inferring social strength on multimedia site is possible. It could open a new perspective for social network mining. For $m$ from 1 to 10, KTA-MKLR significantly outperforms other kernels. Specially, the naive unform combination only reports half of the accuracy of MKL. This fact verifies the efficacy of the proposed method. Moreover, it implies that the underlying single kernels are complementary. It coincides with previously conclusion that MKL is effective at concatenating heterogeneous data.

Second, The *Comment* kernel ($K^3$) reports the second best result. Recall the definition of $K^3$, it's the number of mutual comments among Flickr users. Its good results show that the *mutual communication* is still the most informative modality when inferring social strength. This fact is consistent with the large volume of works on social network analysis, most of which are mutual communication based. Moreover, note that $K^3$ is also better than uniform combination. It means some of the modality is actually noisy. Therefore, learning the modality weight in a principle manner is crucial. Careless combination can actually hurt performance.

Third, we can analyze the contribution of each modality from the results of every single kernel. Specifically:

- The *Textual* kernel ($K^2$) reports the second best accuracy among all the single kernels, though it's significantly worse than $K^3$. Thus we can conclude that the photos of friends exhibit stronger semantically similarity. However,

- The *Visual* kernel ($K^1$) reports very poor performance. This could be possibly explained by the semantic gap. Besides, we use the mean of photos to measure visual similarity due to the difficulty of dealing with large scale photos. This also could lose much information;

- The *Fave* kernel ($K^7$) produces the worst performance among the 7 base kernels. With $m$ varying from 1 to 10, its accuracy is most zero consistently. This fact means it is not useful when inferring social strength;

- The *Group* kernel ($K^4$) is quite similar to *Textual* kernel. Their performance can be explained in the same manner as the group membership indicate the semantics of users' photos;

- The *Location* kernel ($K^6$) is helpful as friends have higher probability residing or taking pictures at the same place.

### 7.5.2 Group Recommendation

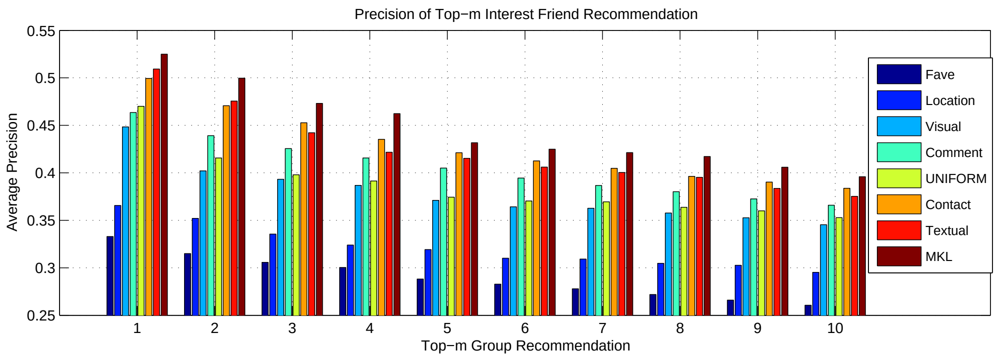


Figure 7.4: The average top-10 group recommendation accuracy of single kernel and combined multiple kernels.

In this section, we evaluate the interest group recommendation task based on social strength modeling. For the user $u_i$ and group $g_k$, the recommendation score is

$$s(g_k, u_i) = \sum_j f(u_i, u_j)\delta(u_j, g_k),$$

where $\delta(u_j, g_k) \in \{0, 1\}$ indicates whether $u_j$ belongs to $g_k$, $f$ is the social strength predicted by MKLR method. We use the *Group* kernel as the target kernel matrix $\mathbf{Y}$ for KTA. We filter out the 1,000 most popular groups for prediction (actually the number of groups is much larger than users). We use the average recommendation accuracy to measure performance, which is plotted in Figure 4. The weight of the 6 base kernels are presented in Table 3.

We draw several observations from the results. First, KTA-MKLR outperforms the other candidate kernels as in the case of friend recommendation. The top-10 accuracy is about 40%. It is practical to recommend interest groups to users. This fact confirms the efficacy of MKL again. The *uniform* combination only ranks the 4-th position among the 8 methods. It is worse than two base single kernels. Therefor, careless kernel combination can hurt the performance as some kernels are noisy.

For the single kernels, the *Textual* kernel performs the best among the 6 base kernels. We conclude semantic information is most important for predicting group membership. This is in contrast to the case of friend recommendation, where mutual communication record is more informative. Each interest group has relatively focus themes. The user annotated tags express the sematic of the photos. Therefore, it could be more useful. Content based analysis is necessary for social network mining.

Unlike the case of friend recommendation, the *Contact* kernel is almost as good as Textual kernel. it is quite effective. We conjecture that user-user relationship is correlated with user-group membership. Both communication and content info contributes to social strength modeling.

The accuracy of *Fave* and *Location* kernel is far from satisfying. It is a bit surprising that Fave kernel is even worse than Location kernel. Intuitively, the location info is not for group recommendation as it is too coarse, while the Fave kernel carries the taste of photos of users, which is related to group membership.

The kernel weight learned in Table 4 is quite different from the results Table 3. Our framework omits the latent variable modeling process such that it can be task dependent.

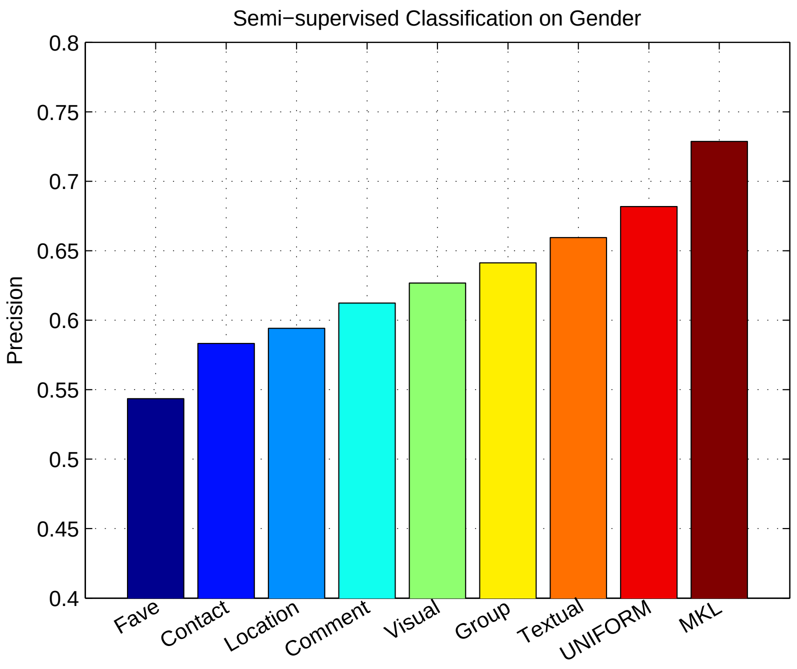
### 7.5.3 Semi-supervised Classification

A prior similarity matrix defined over input sample is crucial for semi-supervised learning (e.g., [ZGL03, ZKGL04, ZBL+05]) as it characterizes the clustering or manifold structure of the data. Such unsupervised information is helpful for classification purpose as similar input data often share similar class labels. The learned social strength graph in this chapter can play an important role for semi-supervised learning (SSL). We plugged it into [ZGL03], which is one of the most influential SSL algorithms, to predict the *gender* and *location* of Flickr users. For the first task, we predict a user is male or female. For the second one, we first count the most frequently photographed location in our dataset. Then we predict whether a user photographes here. The target matrix $\mathbf{Y}$ is set the contact graph for both tasks. We set the ratio of supervised sample to be 0.7. The free parameters in the SSL algorithm is tuned by cross-validation. The results are plotted in Figure 5.

For both tasks, KTA-MKLR performs best. It is better than uniform combination. This verifies that the modalities are complementary for the evaluated classification tasks. Regrading the single kernels, *Textual* kernel performs the best, even rivals MKL for the location prediction task. Recall this also holds for the friend and group recommendation task. We can conclude that the tag-based kernel is most effective for SSM. The most possible reason is the semantic information carried by tags is most meaningful for characterizing the users. On the other hand, among all the 4 tasks, the *Fave* kernel performs worst. Recall that it is defined by the count of photos users like. It is dominated by the quality of photos, instead of social relationship. So it is not very useful for SSM.

## 7.6 Discussion and Related Work

Our work in this chapter is closely related to the following research topics.



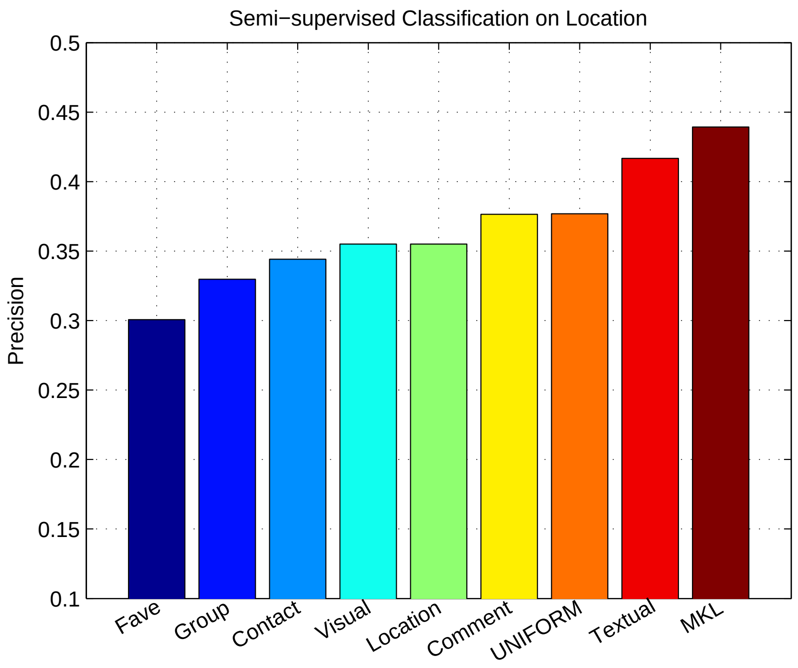


Figure 7.5: The evaluation of inferred social strength graph for semi-supervised classification. The left figure is the accuracy of *gender* prediction, the right figure is the accuracy of *location*.

### 7.6.1 Data Mining with Flickr Data

There exists rich research on mining Flickr data. In [NGP08a], the author conducted probabilistic latent semantic analysis on the Flickr interest groups, where each group is abstracted to be a collection of tags annotated to the images belonging to this group. This work is group-oriented and it is unclear how strong the relationship between users is. Due to the subjective and noisy process in group generating and expanding, Negoescu et. al. [NAP$^+$09][NGP08b] proposed methods to hierarchically organize groups and model users and groups equally. These works are also *group-oriented*.

There are works focus on how to make use of the metadata of Flickr images to facilitate other applications. For example, geo-tag analysis [KNA$^+$07][SMvZ09], automatic tag annotation[WHJ$^+$09], concept / tag modeling[WHY$^+$08].These techniques are related to our work as they analyze both Flickr images and their annotated tags. However, all these studies are *image-orientated*, which cannot solve our *user-orientated* problem.

Our discriminative framework predicts the social strength between two arbitrary input users. It incorporates the rich content and context information on Flickr effectively, whereas previous work cannot make use of multiple modalities effectively.

### 7.6.2 Social Strength Modeling

Our task in this work is to infer the social strength among Flickr users. This connects to social strength modeling or link predictions, where quite a few representative works exists, e.g., [AA01, TWAK03, SR08, GK09, XNR10, LHK10, CMHW10]. However, few existing works make use of the rich content+context data of Flickr images to infer the social relationship between Flickr users. Moreover, our motivation here is to help users find out who exhibits similar interest in photo sharing. Thus our technique should be content driven, instead of mutual communication-based on traditional works.

The unique characteristics of Flickr data provides more flexibility in social network mining. The idea of using Flickr data to infer social relationships has been recently proposed [WT09, PSG08, YJL09]. Wu et. al. [WT09] tried to reveal the closeness of people by face detection techniques. However, this work is severely limited by the range of useable images. Singla et. al. [PSG08] identified the social relationships between individuals in consumer photos with the principled Markov logic networks. Due to the diversity of the images in Flickr, this rule based method is not applicable for our task here. Yu et. al. [YJL09] intended to recommend a user's images to a known interest group based on supervised classification where the initial group membership is deemed as label information. As we aforementioned, the initial group membership is subjective, incomplete, and noisy, which may not be reliable enough to serve as supervised information.

Very recently, Xiang et. al. proposed a hybrid generative-discriminative model which assumes a latent variable measuring the social strength computed from the user profile similarities[XNR10]. The interactive activities are results of such latent variables. However, the profile data of Flickr is not so complete since it is not initially designed to be a social network site. It is not sufficient to infer the social strength solely based on the profile similarities. In addition, separating the interaction graph from profile similarity graph increases modeling complexity.

### 7.6.3 Multiple Kernel Learning to Rank

Our main technique is built on two topics, *kernel learning* and *learning to rank*. The most crucial element of a kernel method is *kernel*, which is in general a function that defines an inner product between any two examples in some induced Hilbert space [CST00]. Recent years have witnessed the active research of learning effective kernels automatically from data [LCB$^+$04]. The most popular example technique for kernel learning is *Multiple Kernel Learning* (MKL) [LCB$^+$04, SRSS06, RBCG08], which aims at learning a linear (or convex) combination of a set of predefined kernels in order to identify a good target kernel for the applications. Besides the work of improving the efficiency of MKL, a number of extended MKL techniques have been proposed to improve the regular linear MKL method [VB09, CCR09, KBS$^+$09, CMR10b]. Here we adopt the two-stage method of Cortes et. al.[CMR10b] as it is computationally simple and works well both theoretically and practically. We learn the weight of each modality by maximizing the inner product between the combined kernel and the target kernel, which can be task dependent to make our solution more flexible.

Learning to rank is an active research topic in machine learning and information retrieval (IR) community (refer to [Liu10]). It provides a principled and effective paradigm for IR applications. Here we are aware that the social strength essentially ranks the degree of affinity among people. If we can rank such relationship properly, we solved the SSM problem. Thus, we borrow the key idea of learning to rank to use and rectify the kernels learned in the first stage.

Instead of detecting binary social ties in traditional social network mining works, we study how to infer the continuous social strength in multimedia communities. The learned graph is more delicate and informative and can be applied in a lot of important applications, including friend prediction, item recommendation, visualization, etc. Our key idea consists of three components: 1) leverage the multiple data sources to compute multiple similarity functions, which can be enforced to satisfy positive semi-definite property to construct valid kernels; 2) employ the kernel target alignment algorithm to learn the weight of each modality and use the weighted summation of base kernels as the ideal kernel; 3) devise the learning to rank framework based on the learned kernel to infer the social strength. Our techniques are are purely discriminative and do not require any assumption of the underlying parametric statistical models that governs the social strength. Empirical results verifies its efficacy. We open this work can call more attention to SSM on multimedia communities.

# Chapter 8

# Conclusions and Future Work

This chapter sets out the conclusion of this thesis and discusses some future directions.

## 8.1 Thesis Summary

In this thesis, we first introduce kernel machines and statistical learning theories. Based on such fundamental knowledge, we conduct a thorough survey on both multiple kernel learning and non-parametric kernel learning algorithms.

We present a family of simple non-parametric kernel learning algorithms (SimpleNPKL). Comparing with parametric kernels (e.g., Gaussian kernel), the non-parametric kernels have more flexibility to fit the diversity of the data. Thus it can yield competitive performance in the transductive inference setting. SiimpleNPKL reduces the time complexity from $O(n^{6.5})$ in traditional semi-definite programming to $O(cn^3)$. Moreover, we generalize SimpleNPKL to applications including clustering, classification, and data embedding. Extensive experiments are conducted on public available data sets and verify both the effectiveness and efficiency of our techniques. We further explore two novel directions towards effective KL. The first strategy is deep MKL, inspired by recent development in deep architectures. The other is unsupervised kernel learning, which shifts the focus of KL from binary classification to discovering the structure of the data. Experiments on public data sets show these two directions are promising.

We study two important applications of NPKL and MKL: image re-ranking and social strength modeling. Empirical results verify the effectiveness of KL algorithms and bridge the gap between KL and real applications.

Below we address several promising directions towards kernel learning.

## 8.2 Generalization Risk of Kernel Learning

In kernel learning, the candidate kernel class increases the model complexity as it is fixed in traditional kernel based machine, like SVM. Therefore, bounding the risk of kernel learning method is crucial for providing insights of which factors affect the generalization ability of kernel machines. We believe bridging the gap between theories and algorithms in kernel methods is very important.

In section 3.3 we derive the risk bounds of more generalized MKL algorithms ($L_p$-norm MKL[KBS+09], polynomial MKL[CCR09], and generalized MKL[VB09]). Later on new works ([CMR10a],[HST11]) come out and beat our results. We believe more delicate complexity measures like local Rademacher complexity over kernel class can be explored. Refer to [KB11] for a successful application in the case of $L_p$-norm MKL.

## 8.3 Deep Kernel Learning

Recently deep learning achieves good results on several pattern recognition tasks, where a multi-layer feature mapping function plays a central role. In our opinion, deep learning is nothing more than traditional neural networks except that it employs an unsupervised pretraining process to initialize the parameters. The researchers from this area argue that kernel machines are local template based methods in the sense that a new test point has to be compared with each support vector by a kernel function.

Recall that a kernel is the inner product after a feature mapping function defined on the input space. Therefore, we can do kernel imbedding to make a kernel deep, i.e., plug the mapped feature space into a new feature mapping function. With such deep kernels, kernel machines like SVM are still "local template based". It is interesting to study what really matters for performance: the local mechanism or the quality of the kernel. In chapter 5.1 we proposed a general framework for deep MKL and an algorithm for the two-layer case, where the main contribution has been published in [ZHT11]. We believe deep MKL is promising where two key problems are worthy of study: 1) how to generate the mapping function between layers, which in turn specifies how to embed kernels into the next layer and generate new base kernels; 2) how to combine the base kernels into a powerful single kernel.

## 8.4 Unsupervised Kernel Learning

Currently kernel learning algorithms, e.g., MKL or kernel target alignment, are supervised. However, we think kernel itself is the prior knowledge related to the marginal distribution $p(\mathbf{x})$ only. If a kernel is good enough for discovering the structure of the data, such as manifold or clustering structure, then it should be good for classification or regression.

Recent advances in *local coordinate coding* inspires pure unsupervised kernel learning[YZG09, YZ10, WYY+10], i.e., for a given point, using the kernel evaluation with anchor points as its coding. It also relates to previous works on random projections[BBV06]. Since in real world unlabeled data is cheap to obtain, we believe this direction can make kernel learning more effective and practical. Our preliminary result shows that 1) unsupervised MKL rivals supervised MKL for classification; 2) unsupervised MKL achieves best results on data embedding.

Note that the unsupervised essence of our work implies we can explore novel heuristics towards kernel learning beyond the traditional SVM-based MKL. Thus it is also promising in designing data-dependent kernel learning algorithms. We believe this would open a new perspective towards KL.

We think the above directions are promising to boost kernel methods and kernel learning. We hope to *make kernel machines fully automated*, that is, people need not pay much effort on model selection. Given a training sample, all one has to do is to pass the data to the kernel machines.

**Publication List**:

- Jinfeng Zhuang, Jialei Wang, and Steven C.H. Hoi (2011). Unsupervised Multiple Kernel Learning. *ACML (submitted).*
- Jinfeng Zhuang, Tao Mei, Steven C.H. Hoi, Xian-sheng Hua, and Shipeng Li (2012). Flickr Community Discovery by Low-Rank Matrix Recovery. *WSDM (submitted).*
- Jinfeng Zhuang, Tao Mei, Steven C.H. Hoi, Yingqing Xu, and Shipeng Li (2011). When Recommendation Meets Mobile: Contextual and Personalized Recommendation On The Go. *UbiComp.*
- Jinfeng Zhuang, Tao Mei, Steven C.H. Hoi, and Xian-sheng Hua (2011). Social Strength Modeling of Flickr Users. *ACM Multimedia.*
- Jinfeng Zhuang, Ivor W. Tsang, and Steven C.H. Hoi (2011). Two-layer Multiple Kernel Learning. *AISTATS.*
- Jinfeng Zhuang, Ivor W. Tsang, and Steven C.H. Hoi (2011). A Family of Simple Non-Parametric Kernel Learning from Pairwise Constraints. *JMLR.*
- Jinfeng Zhuang, Steven C.H. Hoi (2011). A Two-view Model-free Learning Approach for Social Tag Ranking. *WSDM.*
- Jinfeng Zhuang, and Steven C.H. Hoi (2011). Social Image Tag Ranking by Two-view Learning. Book chapter in *Social Media Modeling and Computing.*
- Jinfeng Zhuang, Steven C.H. Hoi (2010). Nonparametric Kernel Ranking Approach for Social Image Retrieval. *CIVR.*
- Jinfeng Zhuang, Steven C.H. Hoi, Aixin Sun, Rong Jin (2010). Entry Selection Techniques for Blog Profiling and Classification. *Information Science.* (In Revision)
- Jinfeng Zhuang, Ivor W. Tsang, Steven C.H. Hoi (2009). SimpleNPKL : Simple Non-Parametric Kernel Learning. *ICML.*
- Jinfeng Zhuang, Steven C. H. Hoi, Aixin Sun, Rong Jin (2008). Representative entry selection for profiling blogs. *CIKM.*
- Jinfeng Zhuang, Steven C. H. Hoi, Aixin Sun (2008). On profiling blogs with representative entries. *2nd Workshop on Analytics for Noisy Unstructured Text Data.*

# References

[AA01] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *SOCIAL NETWORKS*, 25:211–230, 2001.

[AB99] M. Anthony and P.L. Bartlett. *Neural Networks Learning: Theoretical Foundations.* Cambridge University Press, 1999.

[AHMP06] Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Pontil. A dc-programming algorithm for kernel selection. In *Proceedings of the International Conference on Machine Learning*, pages 41–48, 2006.

[AHO97] Farid Alizadeh, Jean-Pierre A. Haeberly, and Michael L. Overton. Complementarity and nondegeneracy in semidefinite programming. *Math. Program.*, 77:111–128, 1997.

[AHP05] Andreas Argyriou, Mark Herbster, and Massimiliano Pontil. Combining graph laplacians for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2005.

[AMP05] Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil. Learning convex combinations of continuously parameterized basic kernels. In *COLT*, pages 338–352, 2005.

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68, 1950.

[BBL02] Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48(1-3):85–113, 2002.

[BBL03] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Advanced Lectures on Machine Learning*, pages 169–207, 2003.

[BBM02] Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Localized rademacher complexities. In *COLT*, pages 44–58, 2002.

[BBV06] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, 2006.

[BBV08] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *STOC*, pages 671–680, 2008.

[Ben09] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[BH02] Olivier Bousquet and Daniel J. L. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems*, pages 399–406, 2002.

[BH08] F. Bach and Z. Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems*, pages 49–56, 2008.

[BKD09] Matyas A. Sustik Brian Kulis and Inderjit S. Dhillon. Low-rank kernel learning with bregman matrix divergences. 10, 2009.

[BKL06] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the International Conference on Machine Learning*, pages 97–104, 2006.

[BLJ04] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the International Conference on Machine Learning*, 2004.

[BM02] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[BPH06] Kristin P. Bennett and Emilio Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

[BRL06] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems*, pages 193–200, 2006.

[BS96] J. F. Bonnans and Alexander Shapiro. Optimization problems with perturbations, a guided tour. *SIAM REVIEW*, 40:228–264, 1996.

[BSR$^+$05] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning*, pages 89–96, 2005.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[BX05] Stephen Boyd and Lin Xiao. Least-squaures covariance matrix adjustment. *SIAM Journal of Matrix Anal. Appl.*, 27(2):532–546, 2005.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999.

[CCR09] Mehryar Mohri Corinna Cortes and Afshin Rostamizadeh. Learning nonlinear combinations of kernels. In *NIPS*, 2009.

[CHL$^+$04] Deng Cai, Xiaofei He, Zhiwei Li, Wei-Ying Ma, and Ji-Rong Wen. Hierarchical clustering of www image search results using visual, textual and link information. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 952–959, New York, NY, USA, 2004. ACM.

[CKM07] Corinna Cortes, Leonid Kontorovich, and Mehryar Mohri. Learning languages with rational kernels. In *COLT*, pages 349–364, 2007.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[CMHW10] Munmun De Choudhury, Winter A. Mason, Jake M. Hofman, and Duncan J. Watts. Inferring relevant social networks from interpersonal communication. In *WWW*, pages 301–310, 2010.

[CMR10a] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Generalization bounds for learning kernels. In *ICML*, pages 247–254, 2010.

[CMR10b] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Two-stage learning kernel algorithms. In *ICML*, pages 239–246, 2010.

[Cor09] Corinna Cortes. Invited talk: Can learning kernels help performance? In *ICML*, page 161, 2009.

[CQL$^+$07] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning*, pages 129–136, 2007.

[CS09] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *NIPS*, pages 342–350, 2009.

[CS10] Youngmin Cho and Lawrence K. Saul. Large-margin classification in infinite neural networks. *Neural Comput.*, 22(10):2678–2697, 2010.

[CST00] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, New York, NY, USA, 2000.

[CSTEK01] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems*, pages 367–373, 2001.

[CSWB06] Ronan Collobert, Fabian H. Sinz, Jason Weston, and Léon Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[CWS02] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 585–592, 2002.

[CY08] Jianhui Chen and Jieping Ye. Training svm with indefinite kernels. In *Proceedings of the International Conference on Machine Learning*, pages 136–143, 2008.

[DGLW06] Ritendra Datta, Weina Ge, Jia Li, and James Ze Wang. Toward bridging the annotation-retrieval gap in image search by a generative modeling approach. In *ACM Multimedia*, pages 977–986, 2006.

[DTXM09] Lixin Duan, Ivor Wai-Hung Tsang, Dong Xu, and Stephen J. Maybank. Domain transfer svm for video concept detection. In *CVPR*, pages 1375–1381, 2009.

[DY07] Guang Dai and Dit-Yan Yeung. Kernel selection forl semi-supervised kernel machines. In *Proceedings of the International Conference on Machine Learning*, pages 185–192, 2007.

[EYP07] Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its applications. In *COLT*, pages 157–171, 2007.

[FDBR04] Glenn Fung, Murat Dundar, Jinbo Bi, and R. Bharat Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *Proceedings of the International Conference on Machine Learning*, 2004.

[FLW09] Yu-Hong Dai Cristian Sminchisescu Fuxin Li, Yunshan Fu and Jue Wang. Kernel learning by unconstrained optimization. In *AISTATS*, 2009.

[GA08] Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, pages 352–359, 2008.

[Gär03] Thomas Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003.

[GB08] David Grangier and Samy Bengio. A discriminative kernel-based approach to rank images from text queries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(8):1371–1384, 2008.

[GBSS05] Arthur Gretton, Olivier Bousquet, Alex J. Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *ALT*, pages 63–77, 2005.

[GD07] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. 8:725–760, 2007.

[GFKS02] Thomas Gartner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. Multi-instance kernels. In *Proceedings of the International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 2002.

[GK09] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *CHI*, pages 211–220, 2009.

[GN08] Peter Vincent Gehler and Sebastian Nowozin. Infinite kernel learning. In *TECHNICAL REPORT NO. TR-178,Max Planck Institute for Biological Cybernetics*, 2008.

[Hau99] David Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

[HJ08] Steven C. H. Hoi and Rong Jin. Active kernel learning. In *Proceedings of the International Conference on Machine Learning*, pages 400–407, 2008.

[HJL07] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of the International Conference on Machine Learning*, pages 361–368, 2007.

[HLC06] Steven C. H. Hoi, Michael R. Lyu, and Edward Y. Chang. Learning the unified kernel machines for classification. In *KDD*, pages 187–196, 2006.

[HLZ$^+$04] Jingrui He, Mingjing Li, HongJiang Zhang, Hanghang Tong, and Changshui Zhang. Manifold-ranking based image retrieval. In *ACM Multimedia*, pages 9–16, 2004.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[HR00] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[HS06] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[HSS08] Thomas Hofmann, Bernhard Scholkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

[HST11] Zakria Hussain and John Shawe-Taylor. A note on improved loss bounds for multiple kernel learning. *CoRR*, abs/1106.6258, 2011.

[JB08] Yushi Jing and Shumeet Baluja. Visualrank: Applying pagerank to large-scale image search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1877–1890, 2008.

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[Joa08] Thorsten Joachims. Svm-light, 2008.

[JS06] Tony Jebara and Vlad Shchogolev. B-matching for spectral clustering. In *ECML*, pages 679–686, 2006.

[JZ08] Rie Johnson and Tong Zhang. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288, 2008.

[KB11] Marius Kloft and Gilles Blanchard. The local rademacher complexity of lp-norm multiple kernel learning. In *arXiv:1103.0790v1*, 2011.

[KBS$^+$09] Marius Kloft, Ulf Brefeld, Soren Sonnenburg, Pavel Laskov, Klaus-Robert Muller, and Alexander Zien. Efficient and accurate $l_p$-norm multiple kernel learning. In *NIPS*, 2009.

[KK06] John E. Mitchell Kartik Krishnan. A unifying framework for several cutting plane methods for semidefinite programming. *Optimization Methods and Software*, 21(1):57–74, 2006.

[KL02] Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the International Conference on Machine Learning*, pages 315–322, 2002.

[KMB06] Seung-Jean Kim, Alessandro Magnani, and Stephen P. Boyd. Optimal kernel selection in kernel fisher discriminant analysis. In *Proceedings of the International Conference on Machine Learning*, pages 465–472, 2006.

[KNA$^+$07] Lyndon S. Kennedy, Mor Naaman, Shane Ahern, Rahul Nair, and Tye Rattenbury. How flickr helps us make sense of the world: context and content in community-contributed media collections. In *ACM Multimedia*, pages 631–640, 2007.

[KP00] V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. *Progress in Probability*, 47:443–458, 2000.

[KRB10] Marius Kloft, Ulrich Rückert, and Peter L. Bartlett. A unifying view of multiple kernel learning. In *ECML/PKDD (2)*, pages 66–81, 2010.

[KSD06] Brian Kulis, Mátyás Sustik, and Inderjit S. Dhillon. Learning low-rank kernel matrices. In *Proceedings of the International Conference on Machine Learning*, pages 505–512, 2006.

[KT03] James Kwok and Ivor Tsang. Learning with idealized kernels. In *Proceedings of the International Conference on Machine Learning*, pages 400–407, 2003.

[LBC$^+$04] Gert R. G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[LBRN06] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2006.

[LCB$^+$02] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semi-definite programming. In *Proceedings of the International Conference on Machine Learning*, pages 323–330, 2002.

[LCB$^+$04] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[Ld07] Ronny Luss and Alexandre d'Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems*, 2007.

[LEC$^+$07] Hugo Larochelle, Dumitru Erhan, Aaron C. Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007.

[LHK10] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, pages 641–650, 2010.

[Liu10] Tie-Yan Liu. Learning to rank for information retrieval. In *SIGIR*, page 904, 2010.

[LJH03] Wei-Hao Lin, Rong Jin, and Alexander Hauptmann. Web image retrieval re-ranking with relevance model. In *WI '03: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, page 242, Washington, DC, USA, 2003. IEEE Computer Society.

[LJN06] Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Nonstationary kernel combination. In *Proceedings of the International Conference on Machine Learning*, pages 553–560, 2006.

[LJY$^+$08] Xianming Liu, Rongrong Ji, Hongxun Yao, Pengfei Xu, Xiaoshuai Sun, and Tianqiang Liu. Cross-media manifold learning for image retrieval & annotation. In *Multimedia Information Retrieval*, pages 141–148, 2008.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[LSST$^+$02] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[LSY98] R. B. Lehoucq, D. C. Sorensen, and C. Yang. Arpack users guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. Technical report, 1998.

[LVB$^+$93] Martin Lades, Jan C. Vorbrüggen, Joachim M. Buhmann, Jörg Lange, Christoph von der Malsburg, Rolf P. Würtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Trans. Computers*, 42(3):300–311, 1993.

[LVBL98] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Applications*, 284:193–228, 1998.

[LYW07] Fuxin Li, Jian Yang, and Jue Wang. A transductive framework of distance metric learning by spectral dimensionality reduction. In *Proceedings of the International Conference on Machine Learning*, pages 513–520, 2007.

[Mik99] G.; Weston-J.; Scholkopf B.; Mullers K.R. Mika, S.; Ratsch. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, 1999.

[MKZ11] S. Sonnenburg M. Kloft, U. Brefeld and A. Zien. Lp-norm multiple kernel learning. 2011.

[Mm06] Javier M. Moguerza and Alberto munoz. Support vector machines with applications. *Statistical Science*, 21(3):322–336, 2006.

[MP05] Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

[MT11a] Qi Mao and Ivor W. Tsang. Multiple template learning for structured prediction. *CoRR*, abs/1103.0890, 2011.

[MT11b] Qi Mao and Ivor W. Tsang. Parameter-free spectral kernel learning. 2011.

[NAP$^+$09] Radu Andrei Negoescu, Brett Adams, Dinh Q. Phung, Svetha Venkatesh, and Daniel Gatica-Perez. Flickr hypergroups. In *ACM Multimedia*, pages 813–816, 2009.

[Nes05] Yu. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

[NGP08a] Radu Andrei Negoescu and Daniel Gatica-Perez. Analyzing flickr groups. In *CIVR*, pages 417–426, 2008.

[NGP08b] Radu Andrei Negoescu and Daniel Gatica-Perez. Topickr: flickr groups and users reloaded. In *ACM Multimedia*, pages 857–860, 2008.

[NN94] Y. Nesterov and A. Nemirovski. *Interior-point Polynomial Algorithms in Convex Programming.* Society for Industrial and Applied Mathematics, 1994.

[OBL05] Stephane Boucheron Olivier Bousquet and Gabor Lugosi. Theory of classification : a survey of some recent advances. 9, 2005.

[OCK08] Vikas Sindhwani Olivier Chapelle and Sathiya S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2008.

[OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[OZ08] Cheng Soon Ong and Alexander Zien. An automated combination of kernels for predicting protein subcellular localization. In *WABI*, pages 186–197, 2008.

[Pat95] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. Technical Report MSRR-604, Carnegie Mellon University, August 1995.

[PBM06] M.I. Jordan P.L. Bartlett and J.D. McAuliffe. Convexity, classification, and risk bounds. *J. of the American Statistical Association*, 473:138–156, 2006.

[PSG08] J Luo P Singla, H Kautz and A Gallagher. Discovery of social relationships in consumer photo collections using markov logic. In *CVPR workshop*, 2008.

[RBCG07] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, pages 775–782, 2007.

[RBCG08] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[RC85] R.A.Horn and C.A.Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[RCM02] Samy Bengio Ronan Collobert and Johnny Mariethoz. Torch: a modular machine learning software library. In *Technical Report IDIAP-RR 02-46, IDIAP*, 2002.

[RHOM98] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool in interactive content-based image retrieval. *IEEE Trans. CSVT*, 8(5):644–655, September 1998.

[rMMR$^+$01] Klaus robert Mller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12:181–201, 2001.

[SARK08] Hichem Sahbi, Jean-Yves Audibert, Jaonary Rabarisoa, and Renaud Keriven. Robust matching and recognition using context-dependent kernels. In *Proceedings of the International Conference on Machine Learning*, pages 856–863, 2008.

[SBD06] Nathan Srebro and Shai Ben-David. Learning bounds for support vector machines with learned kernels. In *COLT*, pages 169–183, 2006.

[SHS01] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *COLT/EuroCOLT*, pages 416–426, 2001.

[SJ07] B. Shaw and T. Jebara. Minimum volume embedding. In *AISTATS*, 2007.

[SJ09] Blake Shaw and Tony Jebara. Structure preserving embedding. In *Proceedings of the International Conference on Machine Learning*, page 118, 2009.

[SK08] Kilho Shin and Tetsuji Kuboyama. A generalization of haussler's convolution kernel: mapping kernel. In *Proceedings of the International Conference on Machine Learning*, pages 944–951, 2008.

[SMvZ09] Pavel Serdyukov, Vanessa Murdock, and Roelof van Zwol. Placing flickr photos on a map. In *SIGIR*, pages 484–491, 2009.

[SNB05] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, pages 824–831, 2005.

[SR08] Parag Singla and Matthew Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW*, pages 655–664, 2008.

[SRS05] Sören Sonnenburg, Gunnar Rätsch, and Christin Schäfer. A general and efficient multiple kernel learning algorithm. In *NIPS*, 2005.

[SRSS06] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[SS02] B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press., 2002.

[SSBG07] Le Song, Alex J. Smola, Karsten M. Borgwardt, and Arthur Gretton. Colored maximum variance unfolding. In *Advances in Neural Information Processing Systems*, 2007.

[SSM98] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[Stu99] J.F. Sturm. Using sedumi: a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

[SWS$^+$00] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[SWY$^+$09] Ju Sun, Xiao Wu, Shuicheng Yan, Loong Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, pages 2004–2011, 2009.

[TN04] Koji Tsuda and William Stafford Noble. Learning kernels from biological networks by maximizing entropy. In *ISMB/ECCB (Supplement of Bioinformatics)*, pages 326–333, 2004.

[TRW05] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

[TWAK03] Benjamin Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *NIPS*, 2003.

[TWT10] Mingkui Tan, Li Wang, and Ivor W. Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *ICML*, pages 1047–1054, 2010.

[Vap98] V. N. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[VB09] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, page 134, 2009.

[VGVZ09] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *ICCV*, pages 606–613, 2009.

[VJ06] Hamed Valizadegan and Rong Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *NIPS*, pages 1417–1424, 2006.

[WHJ$^+$09] Lei Wu, Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Nenghai Yu. Distance metric learning from uncertain side information with application to automated photo tagging. In *ACM Multimedia*, pages 135–144, 2009.

[WHY$^+$08] Lei Wu, Xian-Sheng Hua, Nenghai Yu, Wei-Ying Ma, and Shipeng Li. Flickr distance. In *ACM Multimedia*, pages 31–40, 2008.

[WS08] Kilian Q. Weinberger and Lawrence K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the International Conference on Machine Learning*, pages 1160–1167, 2008.

[WSS04] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the International Conference on Machine Learning*, 2004.

[WT09] Peng Wu and Daniel Tretter. Close & closer: social cluster and closeness from photo collections. In *ACM Multimedia*, pages 709–712, 2009.

[WYY$^+$10] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas S. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[WZZ08] Changhu Wang, Lei Zhang, and Hong-Jiang Zhang. Learning to reduce the semantic gap in web image retrieval and annotation. In *Proceedings of ACM Special Interest Group on Information Retrieval*, pages 355–362, 2008.

[XJKL08] Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, pages 1825–1832, 2008.

[XNJR02] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, 2002.

[XNR10] Rongjing Xiang, Jennifer Neville, and Monica Rogati. Modeling relationship strength in online social networks. In *WWW*, pages 981–990, 2010.

[YC09] Yiming Ying and Colin Campbell. Generalization bounds for learning the kernel. In *COLT*, 2009.

[YC10] Yiming Ying and Colin Campbell. Rademacher chaos complexity for learning the kernel problem. In *TECHNICAL REPORT, http://secamlocal.ex.ac.uk/people/staff/yy267/KLbound-version3.pdf*, 2010.

[YFD$^+$10] Shi Yu, Tillmann Falck, Anneleen Daemen, Léon-Charles Tranchevent, Johan A. K. Suykens, Bart De Moor, and Yves Moreau. L2-norm multiple kernel learning and its application to biomedical data fusion. *BMC Bioinformatics*, 11:309, 2010.

[YFL09] James T. Kwok Zhi-Hua Zhou Yu-Feng Li, Ivor W. Tsang. Tighter and convex maximum margin clustering. In *AISTATS*, 2009.

[YJL09] Jie Yu, Dhiraj Joshi, and Jiebo Luo. Connecting people in photo-sharing sites by photo content and user annotations. In *ICME*, pages 1464–1467, 2009.

[YYG09] C. Campbell Y. Ying and M. Girolami. Analysis of svm with indefinite kernels. In *NIPS*, 2009.

[YZ10] Kai Yu and Tong Zhang. Improved local coordinate coding using local tangents. In *ICML*, pages 1215–1222, 2010.

[YZG09] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *NIPS*, 2009.

[ZA05] Tong Zhang and Rie Ando. Analysis of spectral kernel design based semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2005.

[ZBL$^+$05] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schlkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, 2005.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, pages 912–919, 2003.

[ZH01] Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2001.

[ZH10] Jinfeng Zhuang and Steven C. H. Hoi. Non-parametric kernel ranking approach for social image retrieval. In *CIVR*, pages 26–33, 2010.

[Zha04] Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

[ZHLY08] Jianke Zhu, Steven C. H. Hoi, Michael R. Lyu, and Shuicheng Yan. Near-duplicate keyframe retrieval by nonrigid image matching. In *ACM Multimedia*, pages 41–50, 2008.

[ZHT11] Jinfeng Zhuang, Steven Hoi, and Ivor Tsang. Two-layer multiple kernel learning. In *AISTATS*, 2011.

[Zhu05] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[ZKGL04] Xiaojin Zhu, Jaz S. Kandola, Zoubin Ghahramani, and John D. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2004.

[ZMH$^+$11] Jinfeng Zhuang, Tao Mei, Steven C.H. Hoi, Xian sheng Hua, and Shipeng Li. Modeling social strength in social media community via kernel-based learning. In *ACM Multimedia*, 2011.

[ZO07] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*, pages 1191–1198, 2007.

[ZTH09] Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. Simplenpkl: simple non-parametric kernel learning. In *Proceedings of the International Conference on Machine Learning*, page 160, 2009.