Learning with multiple kernels : algorithms and applications

Hao, Xiao

2014

Hao, X. (2014). Learning with multiple kernels : algorithms and applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/61755>

<https://doi.org/10.32657/10356/61755>

Learning with Multiple Kernels: Algorithms and Applications

Hao Xia

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2014

Acknowledgments

Firstly, I would like to express my sincere appreciation to my supervisor Dr. Steven C.H. Hoi. He not only inspires me by his sharp insights into my research problem, but also helps me to form down-to-earth working attitude. He is not only a perfect supervisor for my research, but also a good friend in my life.

Secondly, I would also like to acknowledge my colleagues at Nanyang Technological University for their friendship and insightful discussions. Especially my group members: Dr. Jinfeng Zhuang, Dr. Peilin Zhao, Mr. Guangxia Li, Dr. Bin Li, Mr. Pengcheng Wu, Mr. Dayong Wang, Mr. Ning Chen, Mr. Boshen Zhang, Ms. Jing Lv, and Mr. Doyen Sahoo.

Finally, I must say thanks from bottom of my heart to my parents, grandparents, uncles, aunts and cousins. It is their support and love that make my life easy and focus.

Abstract

Multiple kernel learning (MKL) is a family of machine learning techniques by learning with multiple kernel functions/matrices. It has been demonstrated as a promising technique for solving many real-world applications, especially for its power of exploiting multiple kernels for fusing diverse information from multiple sources. Beyond the traditional MKL studies for classification, this thesis investigates a more general research problem of learning with multiple kernels for different applications. Specifically, this thesis has proposed a family of novel algorithms for learning with multiple kernels to tackle classification, hashing and similarity learning tasks, respectively. The following gives the main contributions of the proposed techniques.

Firstly, we aim to investigate novel learning with multiple kernels techniques for classification tasks. Specifically, this thesis presents a novel framework of Multiple Kernel Boosting (MKBoost) for classification tasks, which applies the idea of boosting techniques to learn kernel-based classifiers with multiple kernels for resolving classification problems. Conventional MKL methods often formulate the problem as an optimization task of learning the optimal combinations of both kernels and classifiers, which usually results in some forms of challenging optimization tasks that are often difficult to be solved. Different from the existing MKL methods, in this thesis, we investigate a boosting framework of multiple kernel learning for classification tasks, i.e., we adopt boosting to solve a variant of MKL problem, which avoids solving the complicated optimization tasks. Based on the proposed framework, we propose several variants of MKBoost algorithms and extensively examine their empirical performance on a number of benchmark datasets in comparisons to various state-of-the-art MKL algorithms on classification tasks. Experimental results show that the proposed method is more effective and efficient than the existing MKL techniques.

Secondly, we aim to investigate novel learning with multiple kernels techniques for hashing tasks in multimedia similarity search. Specifically, this thesis proposes a Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) scheme that significantly boosts the retrieval performance of existing hashing methods by making use of multiple kernels. Similarity search is a key challenge for multimedia retrieval applications where data are usually represented in high-dimensional space. Among various algorithms proposed for similarity search in high-dimensional space, Locality-Sensitive Hashing (LSH) is the most popular one, which recently has been extended to Kernelized Locality-Sensitive Hashing (KLSH) by exploiting kernel similarity for better retrieval efficacy. Typically, KLSH works only with a single kernel, which is often limited in real-world multimedia applications, where data may originate from multiple resources or can be represented in several different forms. For example, in content-based

multimedia retrieval, a variety of features can be extracted to represent contents of an image. So BMKLSH is proposed to overcome the limitation of regular KLSH. We conduct extensive experiments for large-scale content-based image retrieval, in which encouraging results show that the proposed method outperforms the state-of-the-art techniques.

Finally, we aim to investigate novel learning with multiple kernels techniques for learning similarity functions in multimedia retrieval. This thesis proposes a novel Online Multiple Kernel Similarity (OMKS) learning method, which learns a flexible nonlinear proximity function with multiple kernels to improve visual similarity search in CBIR. Recent years have witnessed a number of studies on distance metric learning to improve visual similarity search in CBIR. Despite their successes, most existing methods on distance metric learning are limited in two aspects. First, they usually assume the target proximity function follows the family of Mahalanobis distances, which limits their capacity of measuring similarity of complex patterns in real applications. Second, they often cannot effectively handle the similarity measure of multi-modal data that may originate from multiple resources. To overcome these limitations, this thesis investigates an online kernel similarity learning framework for learning kernel-based proximity functions, which goes beyond the conventional linear distance metric learning approaches. Based on the framework, OMKs is proposed. We evaluate the proposed technique for CBIR on a variety of image data sets, in which encouraging results show that OMKs outperforms the state-of-the-art techniques significantly.

Contents

Acknowledgments	i
Abstract	iii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Background and Motivations	1
1.1.1 Classification	2
1.1.2 Hashing	3
1.1.3 Similarity Learning	4
1.2 Problems and Research Scope	5
1.3 Approach and Methodology	5
1.4 Summary of Contributions	6
1.5 Relationship between the Proposed Algorithms	8
1.6 Thesis Organization	9
1.7 Notation	9
2 Literature Survey	13
2.1 Kernel Function	13
2.2 Learning with a Single Kernel	14
2.2.1 Typical Kernel Methods	14
2.2.2 Support Vector Machine	16
2.3 Multiple Kernel Learning	18
2.3.1 Efficiency	20
2.3.2 Effectiveness	24

2.3.3	Other Methods	28
2.4	Other Kernel Methods beyond MKL	29
2.5	Applications of Learning with Multiple Kernels	32
2.6	Summary	34
3	Multiple Kernel Boosting	35
3.1	Related Work	35
3.2	Multiple Kernel Boosting	37
3.2.1	Problem Formulation	37
3.2.2	Multiple Kernel Learning	38
3.2.3	Deterministic MKBoost Algorithms	39
3.2.4	Stochastic MKBoost Algorithms	42
3.2.5	Time and Space Complexity	45
3.3	Theoretical Analysis	46
3.4	Experiments	48
3.4.1	Experimental Testbed	49
3.4.2	Comparison Schemes	49
3.4.3	Experimental Setup	51
3.4.4	Comparison Results	53
3.4.5	Parameter Evaluation	61
3.5	Summary	66
4	Boosting Multi-Kernel Locality-Sensitive Hashing	67
4.1	Related Work	67
4.2	Preliminaries of Kernelized LSH	69
4.3	Learning to Combine Multiple Kernels for KLSH	71
4.3.1	Overview	71
4.3.2	WMKLSH by Weighted Bit Allocation	73
4.3.3	BMKLSH for Optimizing Bit Allocation	73
4.3.4	Time Complexity Analysis	74
4.4	Experiments	76
4.4.1	Experimental Testbed	76

4.4.2	Experimental Setup	77
4.4.3	Image Descriptors and Kernel Functions	78
4.4.4	Comparison Algorithms	79
4.4.5	Experimental Results	80
4.4.6	Parameter Sensitivity Evaluation	82
4.4.7	Analysis of Bit Allocation Weights	85
4.4.8	Evaluation of Time Efficiency	86
4.4.9	Evaluation of Qualitative Performance	87
4.5	Summary	88
5	Online Multiple Kernel Similarity	91
5.1	Related Work	91
5.1.1	Distance Metric Learning	91
5.1.2	Kernel-based Learning for Image Retrieval	92
5.1.3	Multiple Kernel Learning	93
5.2	Preliminaries	94
5.2.1	OMKL	94
5.2.2	OASIS	95
5.3	Online Multiple Kernel Similarity	96
5.3.1	Online Kernel Similarity	97
5.3.2	Online Multiple Kernel Similarity	100
5.4	Theoretical Analysis	103
5.5	Experiments	106
5.5.1	Experimental Testbed	106
5.5.2	Experimental Setup	107
5.5.3	Image Descriptors and Kernel Functions	109
5.5.4	Comparison Algorithms	110
5.5.5	Experimental Results	113
5.5.6	Evaluation with Varied-Size Training Data	116
5.5.7	Experiments Under Another Setup	117
5.5.8	Qualitative Comparison	118
5.6	Summary	120

6	Conclusions and Future Work	121
6.1	Conclusions	121
6.2	Limitations	123
6.3	Future Work	123
6.3.1	Improvement on MKBoost	123
6.3.2	Improvement on BMKLSH	124
6.3.3	Improvement on OMKS	124
6.3.4	Improvement on Regular MKL Algorithms	125
6.3.5	Solutions to Other Problems by Learning with Multiple Kernels	125
6.3.6	Other Applications of Learning with Multiple Kernels	125
A	Publication List	127
A.1	Publications Related to This Thesis	127
A.2	Other Publications	128
	References	129

List of Figures

1.1	The dependence structure of the topics in this thesis.	10
3.1	Evaluation of MKBoost classification accuracy with respect to boosting parameter T	63
3.2	Evaluation of MKBoost learning time cost with respect to boosting parameter T	63
3.3	Evaluation of MKBoost classification accuracy with respect to boosting sampling ratio.	64
3.4	Evaluation of MKBoost learning time cost with respect to boosting sampling ratio.	64
3.5	Evaluation of accuracy of stochastic MKBoost algorithms with respect to the sampling decay factor β	65
3.6	Evaluation of time cost of stochastic MKBoost algorithms with respect to the sampling decay factor β	66
4.1	The proposed framework of learning to combine multiple kernels for improving kernel LSH	72
4.2	Evaluation of average top- n precision of retrieval results by different algorithms.	81
4.3	Evaluation of parameter ρ and b on the “Holiday” data set	83
4.4	Evaluation of two parameters p and t used in the KLSH algorithm	84
4.5	Evaluation of the number of boosting rounds (T) in the proposed BMKLSH algorithm.	85
4.6	Visualization of bit allocation weights obtained by three different algorithms.	86
4.7	Evaluation of training and indexing time efficiency.	87

4.8	Comparison of qualitative retrieval performance on the “Holiday” dataset. This figure shows four examples. For each query, we show the top-3 retrieved images by three representative methods, i.e., KLSH-Best, MKLSH, and BMKLSH, respectively.	88
5.1	The system flow of the proposed online multiple kernel similarity learning scheme for image retrieval.	100
5.2	Top-n precision results on “Public” and “Caltech10” data set.	113
5.3	Evaluation of <i>RatioTrain</i> on both “Public” and “Caltech10” data set.	117
5.4	Qualitative comparison of image similarity search results on the “Caltech10” database by different algorithms. For each block, the first image is the query, and the results from the first line to the fourth line represents “OASIS-Best”, “OKS-Best”, “OMKS-U” and “OMKS”, respectively. The category names for the queries are as follows: 1 (roulette wheel), 2 (billard), 3 (skyscraper), 4 (bear), 5 (minotaur), 6 (laptop).	119

List of Tables

1.1	Some common notations used in the thesis.	11
2.1	Representative MKL algorithms.	20
2.2	Other tasks and applications of learning with multiple kernels.	30
3.1	Summary of the datasets (where C, n, D denote # classes, #examples, #features, respectively).	49
3.2	Comparison between two deterministic MKBoost algorithms and five different MKL algorithms.	54
3.3	Comparison between two deterministic MKBoost algorithms and Boosting based algorithms.	56
3.4	Comparison between two deterministic MKBoost algorithms and two recently published algorithms.	59
3.5	Comparison between the deterministic and stochastic MKBoost algorithms. . .	60
4.1	Summary of notation used in this chapter	70
4.2	Experimental results of mAP performance.	81
5.1	Experimental results of mAP performance.	114
5.2	Training time (seconds) of KITML versus OKS.	115
5.3	Mean average precision and precision at top 1, 10 and 50 on the Caltech object data sets.	118

Chapter 1

Introduction

In this chapter we will present a brief review to learning with multiple kernels and the main contribution of this thesis. The concepts introduced in this chapter will be explained in more details in the later chapters.

1.1 Background and Motivations

Kernel methods have been extensively studied in data mining and machine learning for their proven state-of-the-art performance in many real data analysis applications [SS02, STC04]. For any kernel method, a core element is the kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, which measures the similarity between two data examples $\mathbf{x}_i$ and $\mathbf{x}_j$. Many kernel methods usually adopt a single predefined kernel function, for example, a linear kernel or a gaussian kernel. However, in many real-world scenarios, it is usually not enough for employing a single predefined kernel function since real data may come from multiple diverse sources or could be given in terms of different kinds of representations. This has motivated the need of studying machine learning techniques for combining multiple kernels to achieve better capability and higher flexibility in solving real-world challenges. Such a learning problem is typically known as “Multiple Kernel Learning” (MKL) in machine learning and data mining areas [SRSS06]. Although MKL is the most widely known technique by learning with multiple kernels, the concept of learning with

multiple kernels can also help to improve many other tasks other than classification, such as hashing and similarity learning.

1.1.1 Classification

Recent years have witnessed a surge of studies that have attempted to explore a variety of techniques to resolve the Multiple Kernel Learning problems [RBCG08, SRSS06, XJKL08]. Similar to some well-known kernel methods, such as SVM [Vap98], MKL has been demonstrated as a promising technique with state-of-the-art performance for solving many real-world applications, especially for its power of exploiting multiple kernels in fusing diverse information from multiple sources. For instance, MKL techniques have been applied to resolve the challenges of Protein subcellular localization in Bioinformatics [OZ08], speech recognition in signal processing [LG09], and anomaly detection in data mining [DMSO10].

Despite being investigated actively in various domains, the regular MKL methods have some critical limitations. In particular, most existing MKL methods often formulate the problem as a complicated optimization task, typically a convex optimization task, e.g., a Semi-Definite Program (SDP) [LCB⁺04], and then attempt to resolve the optimization tasks by applying some existing optimization techniques. Despite their nice convex formulation by some existing studies, it is often a great challenge for solving such complicated optimization tasks. Recently, a surge of research efforts have attempted to improve the efficiency of the optimization task by various techniques [SRSS06, XJKL08]. For instance, the study in [SRSS06] formulated the MKL problem as a min-max optimization task, and proposed to find the saddle-point solution by solving a Semi-Infinite Linear Program (SILP). Although some encouraging progress has been made in this research direction, low efficiency and scalability issues remain the key limitations of the existing regular MKL methods towards a real large application.

In addition, despite engaging a complicate optimization process, the final classifier learned by the regular MKL methods is a single kernel-based classifier based on a kernel that is a linear

combination of multiple kernels. Such a single kernel-based classifier might not be effective enough to handle diverse patterns and complex decision boundaries in real applications.

1.1.2 Hashing

Similarity search, or Nearest Neighbor (NN) search plays a fundamental role in a variety of multimedia retrieval tasks [LSDJ06a, JB08, GB08], which has been extensively studied in multimedia and computer vision fields, especially for CBIR [SWS⁺00, LSDJ06b, HLLM06, RGZ⁺08]. Typically, images in a CBIR system are represented in a high-dimensional space, and the size of an image database can easily be over millions or even billions for large-scale real-world applications. These two aspects have made CBIR an open grand challenge although it has been extensively studied for decades.

A variety of data structures have been proposed for indexing and searching data points in a low-dimensional space [SH75, Cla88, Ben75, Rob81, Gut84, Yia93]. These approaches work well for low dimensional data. But, when the number of dimensions grows, they often become less efficient, a phenomenon known as the *curse of dimensionality*. Specifically, the time or space requirements of these approaches often grow exponentially with the dimensionality.

Since exact NN search is hard to scale for high-dimensional data, recent studies mainly focus on *approximation* approaches [IM98, GIM99, LJW⁺07, AMM09], which aim to remove the exponential dependence on dimensionality. Instead of finding the nearest point p to a query point q , approximate NN search allows to return any point within the distance of $(1 + \epsilon)$ times the distance from q to p . Recent studies have shown that by adopting the approximation, the complexity of NN search is reduced from exponential to polynomial in terms of its dependence on the dimensionality, when compared with those above mentioned speedup approaches such as, kd-tree [Ben75], R-tree [Gut84], VP-tree [Yia93], and so on. Several recent studies have successfully applied the random projection idea for approximate NN search over high-dimensional data. One of the most well-known techniques in this direction is Locality-Sensitive

Hashing (LSH) [IM98, GIM99, Cha02], which has been actively studied and successfully applied to many applications [IM98, DIIM04, AMM09].

One limitation of regular LSH is that they require explicit vector representation of data points. Kernelized LSH (KLSH) [KG09] addresses this limitation by employing kernel functions to capture similarity between data points without having to know their explicit vector representation. Despite the success, most existing KLSH techniques only adopt a single kernel function. This significantly limits its application to many real-world image retrieval tasks [ZMH⁺11, HL08], where images are often analyzed by a variety of feature descriptors and are measured by a wide class of diverse similarity functions.

1.1.3 Similarity Learning

The crux of similarity search is to find some proximity function that can effectively measure distance/similarity between images [AH00, LCW03, QMC05]. In a conventional CBIR system, given images represented in a vector space, the typical choices of such proximity functions are Euclidean distance and its variants, which are often not flexible enough to measure the proximity of images due to the nature of the fixed rigid functions.

In recent years, researchers have noticed the limitations of conventional rigid proximity functions in image similarity search. To address this issue, one group of active research studies are the Distance Metric Learning (DML) algorithms [HLLM06, SJHL06, LJJ08, HLC08], which usually learn to optimize the distance metric of proximity measure to improve image similarity search in CBIR. Despite the success of various DML algorithms to improve similarity search of CBIR, most existing DML algorithms are limited in two aspects. First, they typically assume the target proximity function follows the form of general Mahalanobis distances and restricts the DML task in finding an optimal linear distance metric, which may limit its capacity of measuring similarity of complex image patterns in real applications. Second, even though there are few existing works [CKTK10, JKDD12] for learning nonlinear similarity function with kernel, they usually do not handle the similarity with multiple kernels.

1.2 Problems and Research Scope

Generally, we point out three challenges and solve them by learning with multiple kernels in this thesis:

- Improve the efficiency and effectiveness of regular MKL algorithms for classification.
- Improve the state-of-the-art hashing algorithm (KLSH) by learning a combination of multiple kernels.
- Learn a flexible nonlinear proximity function with multiple kernels to overcome the limitations of existing similarity learning algorithms.

1.3 Approach and Methodology

For the first challenge, we investigate a new framework of multiple kernel learning, termed “Multiple Kernel Boosting” (MKBoost), which adapts the idea of boosting for learning classifiers with multiple kernels for classification tasks. The intuitive idea of multiple kernel boosting is to employ the boosting framework to learn an ensemble of multiple base kernel classifiers, each of them is learned from a single kernel. The combination weights for both the kernels and classifiers can be efficiently determined through the learning process of boosting. Consequently, we can efficiently learn a classifier with multiple kernels without resolving a complicated optimization task as required in a regular MKL approach. In addition, unlike the regular MKL methods that only learn a single kernel-based classifier, MKBoost learns an ensemble of multiple kernel-based classifiers, in which each individual classifier is trained on a single kernel and the final classifier is obtained by a weighted combination of all classifiers according to their performance.

For the second challenge, we propose a novel Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) framework, which improves KLSH by learning a combination of multiple kernels. The key challenge of multi-kernel LSH is to determine an optimal combination

of multiple kernels by determining appropriate bit size to each of the multiple kernels. To overcome the challenge, we propose a boosting scheme to greedily find a good solution in an efficient approach. Our extensive experiments show that BMKLSH significantly enhances the performance of KLSH in exploring the power of multiple kernels for CBIR.

For the third challenge, we propose a novel Online Multiple Kernel Similarity (OMKS) learning scheme, which ranks images by learning pairwise similarity of images that are represented in multiple modalities using multiple kernels. Unlike the conventional DML techniques, the target similarity function learned by OMKs can be any nonlinear function in some reproducing kernel Hilbert spaces induced by some predefined kernels. And different from some kernel-based DML algorithms, OMKs learns with multiple kernels. Thus, OMKs is able to learn a much more flexible and powerful proximity function to improve image similarity search in CBIR. The OMKs task is however very challenging because it must on one hand learn an optimal kernel-based similarity function for each kernel in each modality, and on the other hand determine an optimal combination of multiple kernels in building the final similarity function for similarity search with all modalities. To attack the challenges, we propose an online learning algorithm for OMKs, which learns both the optimal similarity function with each individual kernel and the optimal combination of multiple kernels in an effective and scalable online learning framework. In particular, we apply the online passive aggressive learning technique [CDK⁺06] to learn the kernel-based similarity function for each individual kernel, and the well-known Hedging online learning technique to learn the optimal combination weights of multiple kernels, from a sequence of triplet training data.

1.4 Summary of Contributions

A summary of contributions are given in forms of the contributions of each work as follows.

The contributions of MKBoost include:

- We present a novel framework of Multiple Kernel Boosting (MKBoost) for MKL, which offers a new approach to learning kernel based classifiers with multiple kernels efficiently.
- We propose two kinds of MKBoost approaches: deterministic and stochastic algorithms, which attempt to resolve the MKBoost problem to trade off between accuracy and efficiency.
- We conduct extensive experiments for empirically validating the performance of the proposed MKBoost algorithms by comparing with various state-of-the-art MKL algorithms.

The research output has been published in SDM2011 [XH11] and TKDE [XH13].

The main contributions of BMKLSH are:

- We propose a novel Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) framework, which improves KLSH by learning a combination of multiple kernels.
- Our extensive experiments show that BMKLSH significantly enhances the performance of KLSH in exploring the power of multiple kernels for CBIR.

This work has been published in SIGIR2012 [XWHJ12].

The key contributions of OMKS include:

- We propose a novel framework of learning kernel-based proximity functions with multiple kernels for visual similarity search. To the best of our knowledge, this is the first online learning work in CBIR that learns to rank images based on kernel-based similarity function using multiple kernels.
- We present an online learning algorithm for OMKS, which learns both the optimal kernel-based similarity function with an individual kernel and the optimal combination of multiple kernels.

- We conduct an extensive set of experiments to evaluate the performance of the proposed technique for CBIR on several image data sets.

The research output has been accepted by TPAMI [XHJZ13].

1.5 Relationship between the Proposed Algorithms

As you have seen, totally 3 algorithms are proposed in this thesis, they are MKBoost, BMKLSH, and OMKS respectively. Before proceed on, we would like to point out the relationships between them.

First, they have some properties in common:

- They are all based on learning with multiple kernels, i.e., they all need to learn to combine multiple kernels.
- Different from traditional MKL algorithms, MKBoost and BMKLSH are both boosting based.

Second, they are different in some aspects:

- MKBoost and BMKLSH are batch methods, while OMKS is online
- Those 3 proposed methods target different tasks, i.e., MKBoost, BMKLSH and OMKS are proposed to tackle classification, hashing and similarity learning tasks, respectively.

At last, it is worth to mention that we focus on the application to Content-Based Image Retrieval (CBIR) in this thesis. Both hashing and similarity learning can be applied to CBIR naturally, as they are both able to solve similarity search problem. But they are two extremes, hashing focus more on the efficiency, even at the expense of accuracy, while the purpose of similarity learning is quite the contrary.

1.6 Thesis Organization

Figure 1.1 shows the dependence structure of the topics in this thesis. The contributions of this thesis are highlighted in bold font.

The rest of this thesis is organized as follows. In Chapter 2 we conduct a literature survey on learning with multiple kernels. Chapter 3 proposes our Multiple Kernel Boosting (MKBoost) framework, which aims to improve the efficiency and effectiveness of regular MKL algorithms for classification. Chapter 4 presents a novel Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) framework, which improves the state-of-the-art hashing algorithm (KLSH) by learning a combination of multiple kernels. Chapter 5 introduces our Online Multiple Kernel Similarity (OMKS) learning scheme by learning a flexible nonlinear proximity function with multiple kernels to overcome the limitations of existing similarity learning algorithms. At last, Chapter 6 concludes the thesis and proposes some promising directions on the topic of learning with multiple kernels.

1.7 Notation

Matrices and vectors are represented in bold notation, e.g., $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{R}^{m \times n}$. For a symmetric square matrix $\mathbf{X}$, $\mathbf{X} \succeq \mathbf{0}$ means that $\mathbf{X}$ is positive semi-definite, while $\mathbf{X} \succ \mathbf{0}$ means positive definite.

In order to make the presentation more clear, we show some common notations in Table 1.1.

Figure 1.1: The dependence structure of the topics in this thesis.

Table 1.1: Some common notations used in the thesis.

Notation	Explanation
$\mathbb{R}$	domain of real number
$\mathcal{X}$	domain from which the samples come from
$\mathbf{x}$	a single point, typically belonging to $\mathbb{R}^d$
y	the label of $\mathbf{x}$, taking a value from $\{-1, +1\}$ in binary classification
$\mathbf{x}^\top$	transpose of $\mathbf{x}$
κ	kernel function
ϕ	the implicit mapping function of some kernel κ
$\mathbf{K}$	kernel matrix, the value of (i, j) entry is the evaluation $\kappa(\mathbf{x}_i, \mathbf{x}_j)$
n	number of training instances
m	number of different kernel functions
$f(\cdot)$	a decision function defined on $\mathcal{X}$
$\mathcal{H}$	hypothesis space
$\mathcal{H}_\kappa$	Hilbert space induced by some kernel κ

Chapter 2

Literature Survey

In this chapter we survey the algorithms and applications of learning with multiple kernels. The rest of this chapter is organized as follows. Section 2.1 introduces kernel function which is a basic concept in learning with multiple kernels. Section 2.2 introduces typical kernel methods, i.e., those methods learning with a single kernel. Then, we survey existing algorithms by learning with multiple kernels. We divide them into two categories: Multiple Kernel Learning (MKL) and other kernel methods beyond MKL, they are introduced in section 2.3 and section 2.4 respectively. After that, a brief introduction to the applications of learning with multiple kernels is given in section 2.5. Finally, section 2.6 summaries this chapter.

2.1 Kernel Function

Traditionally, machine learning algorithms have been very well developed for the linear case. But real world problems often require nonlinear methods to detect the kind of dependencies that lead to successful prediction. Kernel-based learning algorithms, such as support vector machine [Vap98], kernel logistic regression [ZH01] and kernel principle component analysis [SSM97], work by embedding the data into a Reproducing Kernel Hilbert Space (RKHS), and searching for linear relations in such a space. The embedding is performed implicitly by specifying the inner product between each pair of points, so that we never need to compute in the high-dimensional feature space explicitly. This approach has several advantages, the most

important one is that the inner product in the embedding space can often be computed more easily than the coordinates of the points.

Given an input set $\mathcal{X}$ and a dot product space $\mathcal{H}$, we consider a mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$. Given two points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, the function that returns the inner product between their projections in the feature space $\mathcal{H}$ is known as a kernel function.

Definition 2.1 *A kernel is a function κ , such that*

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \quad (\text{Eq. 2.1})$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, where ϕ is a mapping from $\mathcal{X}$ to a feature space $\mathcal{H}$.

Some kernels are widely used, such as:

- Linear Kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$. This kernel is the inner product in the input space, it is the most trivial one.
- Polynomial Kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^d$.
- Gaussian Kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$.

2.2 Learning with a Single Kernel

In this section we briefly introduce typical kernel methods at first. Then, we give more details on Support Vector Machine (SVM), because it is the most widely known and much of our work is based on it.

2.2.1 Typical Kernel Methods

Kernel methods have been extensively studied in data mining and machine learning for their proven state-of-the-art performance in many real data analysis applications [SS02, STC04]. The general idea of kernel methods is by mapping data from original space to feature space, it

can then do linear analysis in the feature space. So it provides a systematic way of transforming data into a high dimensional feature space to extract nonlinearity. All we need for kernel methods are kernel values, we do not need an explicit form of feature vector or feature space. In addition, another benefit of kernel methods is that they can operate on very general types of data, besides traditional vectorial data.

[SSM98] states that every algorithm that only uses scalar products can implicitly be executed in $\mathcal{H}$ by using kernels, i.e., one can very elegantly construct a nonlinear version of a linear algorithm. An exciting observation is that many linear methods use only the scalar products of data, and do not need the explicit form of the feature vector $\phi(\mathbf{x})$. Typical kernel methods derived from such linear methods include Kernel Support Vector Machine (KSVM) [MMR⁺01], Kernel Fisher Discriminant Analysis (KFDA) [MRW⁺99], Kernel Principal Component Analysis (KPCA) [SSM98], Kernel Canonical Correlation Analysis (KCCA) [MRB01], kernel k-means clustering [DGK04], and so on.

Next we'd like to introduce Representer theorem [SHS01, HSS08], which shows that solutions of a large class of optimization problems can be expressed as kernel expansions over the sample points.

Theorem 1 (*Representer theorem*) Denote by $\Omega : [0, \infty) \rightarrow \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by $c : (\mathcal{X} \times \mathbb{R}^2)^n \rightarrow \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the regularized risk functional

$$c((\mathbf{x}_1, y_1, f(\mathbf{x}_1), \dots, \mathbf{x}_n, y_n, f(\mathbf{x}_n))) + \Omega(\|f\|_{\mathcal{H}}^2)$$

admits a representation of the form

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

In fact, the solution to many kernel methods typically has the form indicated by representer theorem above. We can again observe that for kernel methods, we do not need to compute the coordinates of the points in the feature space, which may often be high dimensional (or even infinite). All we need is to evaluate kernel values, which is usually much more efficient.

2.2.2 Support Vector Machine

Now we proceed to introduce Support Vector Machine (SVM) [CV95, Vap98]. Among the large amount of works, SVM is one of the most successful work of machine learning research in recent years [CV95, Bur98, Mm06]. Refer to [Bur98] and [Mm06] for good surveys. The content here overlaps with [Mm06].

We restrict to binary classification. Among the infinite number of existing separating hyperplanes, support vector machine looks for the one that lies furthestmost 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. 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. Therefore, for every $i \in \{1, \dots, n\}$,

$$\mathbf{w}^\top \phi(\mathbf{x}_i) + b \geq +1 \quad \text{if} \quad y_i = +1 \quad (\text{Eq. 2.2})$$

$$\mathbf{w}^\top \phi(\mathbf{x}_i) + b \leq -1 \quad \text{if} \quad y_i = -1 \quad (\text{Eq. 2.3})$$

After the rescaling, the distance from the nearest point in those two classes to the hyperplane are both $\frac{1}{\|\mathbf{w}\|}$. Hence, the distance between the two classes is $\frac{2}{\|\mathbf{w}\|}$, which we call it margin. So as to maximize the margin, we obtain the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \geq 1, i = 1, \dots, n \end{aligned} \quad (\text{Eq. 2.4})$$

The square in the norm of $\mathbf{w}$ has been introduced to make the problem quadratic. Notice that due to convexity, the problem above has a unique global optimal solution. Suppose the optimal solution is $\mathbf{w}^*$ and b^* . Then the decision surface is $f(\mathbf{x}) = (\mathbf{w}^*)^\top \phi(\mathbf{x}) + b^* = 0$. Points $\phi(\mathbf{x}_i)$ which satisfy the equalities $y_i((\mathbf{w}^*)^\top \phi(\mathbf{x}_i) + b^*) = 1$ are called support vectors (SVs). The SVs often represent a small fraction of the samples. The hyperplane $f(\mathbf{x}) = 0$ is completely determined by the subset made up of the SVs. This implies that, for many applications, the

evaluation of the decision function $f(\mathbf{x})$ is computationally efficient, allowing the use of SVM on large datasets in realtime environments.

Let us extend Eq. 2.4 to non-separable case by introducing slack variables:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, i = 1, \dots, n \end{aligned} \quad (\text{Eq. 2.5})$$

This is the most widely known SVM formulation. By introducing the Lagrangian multipliers α_i for each inequality constraint, and use the kernel evaluation $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ instead of $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, we arrive at the dual problem of (Eq. 2.5) (refer to [Bur98] for a detailed derivation):

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, i = 1, \dots, n. \end{aligned} \quad (\text{Eq. 2.6})$$

This problem is a quadratic program on α [FKM05]. Therefore, it has unique global optimal solution. From the KKT condition for optimality [FKM05], the primal variable $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i)$. So the decision function becomes $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) + b^*$.

Note that the potential high dimensional (possibly infinite) operations are replaced by a simple kernel evaluation. We do not even need to know the mapping function ϕ explicitly. One can just focus on selecting a proper kernel function κ to get good performance.

Next we extend SVM to the general learning problem in RKHS. We now consider the standard supervised learning task of which the solution exists in a RKHS:

$$\min_f L(f, \mathbf{y}) + C \|f\|_{\mathcal{H}}^2 \quad (\text{Eq. 2.7})$$

Here $L(f, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n l(y_i, f_i)$ is the empirical loss, $\|f\|_{\mathcal{H}}$ a regularization term, it is the norm of f in $\mathcal{H}$. This is widely known as the “loss + regularization” framework. It enjoys great

popularity due to both its theoretical generalization ability and good empirical performance. The good theoretical foundation of the “loss + regularization” framework has been established in statistical learning theory [Vap98].

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

2.3 Multiple Kernel Learning

We divide existing algorithms by learning with multiple kernels into two categories: Multiple Kernel Learning (MKL) and other kernel methods beyond MKL. In this section we survey MKL, and other kernel methods beyond MKL will be introduced in the next section. You can also refer to [GA11] for a good survey.

Finding appropriate representation via kernel trick leads to many powerful machine learning methods. But the choice of kernel must generally be made by user. It can be difficult to decide the use of one kernel instead of another.

Engineering the best kernel is not always trivial. Most kernel methods usually assume some predefined parametric kernels, e.g. a gaussian kernel, is given a priori, where the parameters of these kernel functions are usually determined empirically by cross validation [Sto74]. Several studies have been proposed to learn parametric, semi-parametric or nonparametric kernel functions/matrices from labeled and/or unlabeled data. Example techniques include cluster kernels [CWS02], diffusion kernels [KL02], marginalized kernels [KTI03], graph-based spectral kernel learning methods [ZKGL04, HLC06, BH02], nonparametric kernel learning [HJL07, ZTH09] and so on.

For many real-world scenarios, employing a single kernel function may not be enough since real data may come from multiple different sources or could be represented in different kinds of representations. This motivates the need of studying machine learning techniques for combining multiple kernels to achieve better capability and higher flexibility in solving real-world challenges. Such a learning problem is typically known as “Multiple Kernel Learning” (MKL) in machine learning and data mining [SRSS06].

The combination function is as follows:

$$\kappa_{\eta}(\mathbf{x}_i, \mathbf{x}_j) = f_{\eta}(\kappa_1(\mathbf{x}_i, \mathbf{x}_j), \dots, \kappa_m(\mathbf{x}_i, \mathbf{x}_j)) \quad (\text{Eq. 2.8})$$

Here, we do not restrict the type of f_{η} .

There are some works use fixed rules for kernel combination, without any training. It is reported that by training an SVM with the average of a set of predefined heterogeneous kernels outperforms the combination of multiple SVMs each equipped with a single kernel on a gene functional classification task in [PWCG01]. [BhN05] improves the performance for protein-protein interaction prediction task also by fixed rules for kernel combination.

But it is obvious that fixed rule combination is not optimal, this motivates MKL. MKL tries to optimize the combination of kernel functions/matrices so as to maximize a generalized performance measure. Typical measures used include: maximum margin, kernel target alignment [CSTEK01], Fisher discriminative analysis [YCJK07], and so on. In this section we focus on maximum margin classification errors which is the most prevalent.

The advantage of MKL is two-fold:

- Different kernels correspond to different projection functions, i.e., different feature spaces. Instead of selecting the one which performs best, MKL method permits us to use an optimal combination of them.
- Different kernels may use different representations from different input sources or modalities. So MKL can provide a method to combine information from multiple sources.

There are mainly two hot spots in MKL research. The first one is improving its efficiency. The second one is to improve the effectiveness of MKL. We classify a large amount of works in literature of MKL according to this, totally three categories are introduced: Efficiency, Effectiveness and Other Methods. We summarize representative MKL algorithms in Table 2.1.

Table 2.1: Representative MKL algorithms.

Goal	Combination Method	Representative Reference
Efficiency	Linear Function	[LCB ⁺ 02], [LCB ⁺ 04], [LDC ⁺ 04], [CG10]
	Linear Function with Nonnegative Coefficients	[LCB ⁺ 04], [QL05], [BLJ04], [SRS05], [SRSS06], [RBCG07], [RBCG08], [LG08], [LG09], [CR08], [XJKL08], [XJY ⁺ 10]
	Linear Function with Binary Coefficients	[XJY ⁺ 09]
Effectiveness	Linear Function	[CMR09a], [FYT09], [KBS ⁺ 09], [MKZ11], [Suz11], [CMR10a], [Suz11], [YXY ⁺ 11], [TS10], [SS12], [XTX13], [SGR08, SGR10], [JDR ⁺ 09], [ABTB ⁺ 11], [GN08], [SS10], [YMJ ⁺ 12], [ZO07], [HCK09], [JSJY08], [TCY09]
	Non-Linear Function	[CMR09b], [Bac08], [JZH11], [dDMM04], [GA08], [VR07], [VB09], [JVV12b]
Other Methods	Similarity Based Methods	[KSTC02], [LCB ⁺ 04], [QL09], [CMR10b],
	Boosting Methods	[BME02], [BZB04], [BFDR05], [CKS02]
	Bayesian Methods	[GR05a], [GZ06]

2.3.1 Efficiency

We further classify this category according to the type of f_η in Eq. 2.8 into Linear Function, Linear Function with Nonnegative Coefficients and Linear Function with Binary Coefficients.

2.3.1.1 Linear Function

[LCB⁺02, LCB⁺04] train a support vector machine and optimize the linear combination of some kernel matrices simultaneously. We denote the set of feasible kernel matrices as $\mathcal{K}$, i.e.,

$\mathcal{K}$ includes matrices $\mathbf{K}$ that satisfy the following condition:

$$\mathbf{K} = \sum_{l=1}^m \eta_l \mathbf{K}_l, \quad (\text{Eq. 2.9})$$

$$\mathbf{K} \succeq \mathbf{0}, \quad (\text{Eq. 2.10})$$

$$\text{trace}(\mathbf{K}) \leq c. \quad (\text{Eq. 2.11})$$

This started the new field of MKL which combines several kernel functions automatically. It is shown that this task can be solved by Semi-Definite Programming (SDP) techniques. This can give a powerful transductive algorithm, i.e., we can use the labeled training data to learn an embedding also applicable to the unlabeled data. In addition SDP is convex, so we can get the global optimal. Yet, the time complexity of SDP is so high that it is not practical for real world applications.

[LDC⁺04, LCB⁺04] consider classifying yeast proteins. Kernels based on different genomic data are combined using SDP formulation. The experiments show that in 9 out of 13 cases, it is better than SVM trained with each kernel.

[CG10] extends [LCB⁺02, LCB⁺04] to an inductive setting by constraining the kernel weights instead of the trace of the combined kernel matrix.

2.3.1.2 Linear Function with Nonnegative Coefficients

A milestone that makes MKL really practical is by restricting η to non-negative. That is $\mathcal{K}$ is the set of matrices $\mathbf{K}$ satisfying:

$$\mathbf{K} = \sum_{l=1}^m \eta_l \mathbf{K}_l, \quad (\text{Eq. 2.12})$$

$$\eta_l \geq 0, l \in \{1, \dots, m\} \quad (\text{Eq. 2.13})$$

$$\mathbf{K} \succeq \mathbf{0}, \quad (\text{Eq. 2.14})$$

$$\text{trace}(\mathbf{K}) \leq c. \quad (\text{Eq. 2.15})$$

So the optimization problem can be transformed into a Quadratically Constrained Quadratic Programming (QCQP) problem [LCB⁺04], which is a subset of Semi-Definite Programming (SDP). So the computational complexity can be significantly reduced.

[QL05] investigates multiple kernel learning for SV regression. They transform the optimization problems of SV regressions using both ϵ -sensitive loss function and automatic error control so that they can formulate it as SDP and QCQP problems.

Current toolboxes can only solve small scale SDP problems. The SDP and QCQP MKL are convex but also non-smooth. They can be cast as the minimization of a non-differentiable function subject to linear constraints. So the Sequential Minimal Optimization (SMO) techniques cannot be applied because the cost function is non-differentiable. Otherwise SMO may fail to converge or converge to an incorrect value. [BLJ04] proposes a novel dual formulation of the QCQP as a Second-Order Cone Programming (SOCP) problem. The advantage of the SOCP formulation is that they can design a Moreau-Yosida regularization which preserves the sparse SVM structure, therefore they can apply SMO techniques.

Another breakthrough is the arise of min-max formulation. It is proposed in [SRS05, SRSS06], and the min-max problem is converted into a Semi-Infinite Linear Program (SILP) problem. SILP can be solved more efficiently than SDP and QCQP. They propose a wrapper method and a chunking algorithm to solve the resulting SILP problem. The wrapper method uses a column generation [NS96] approach. The problem is divided into two subproblems: inner problem and outer problem. It can be solved by solving these two problems alternatively until convergence. The outer loop is given a fixed α and determines the optimal η . The inner loop trains a SVM, which is simple. So the wrapper algorithm can benefit from efficient SVM optimization routines [eFhCjL05]. So [SRS05, SRSS06] allow large-scale training. The chunking algorithm is more efficient than the wrapper method, since it optimizes α and η at the same time. The key idea of chunking algorithm is to freeze all but a small number of optimization variables (working set) and to solve a sequence of constant-size problems.

[RBCG07, RBCG08] propose SimpleMKL. SimpleMKL uses a projected gradient method to solve a different primal problem. The algorithm is similar with the algorithm used by [SRS05, SRSS06] to solve SILP formulation. Both algorithms optimize the kernel weights and the SVM classifiers alternatively. But different updating procedures are employed to update the kernel weights η : [SRS05, SRSS06] construct a cutting plane model and update η by linear programming; [RBCG07, RBCG08] use a projected gradient update and solving an LP. [LG08, LG09] add another regularization term to the objective function of SimpleMKL [RBCG07, RBCG08]:

$$\lambda \sum_{l=1}^m \left(\eta_l - \frac{1}{m} \right)^2 = \lambda \left(\sum_{l=1}^m \eta_l^2 - \frac{1}{m} \right) \quad (\text{Eq. 2.16})$$

So that the level of sparsity for the kernels can be controlled by λ . Positive values of λ encourage a uniform set of weight. Negative values lead to sparse combination.

[CR08] (HessianMKL) investigates the use of second order optimization approaches. They show that the hessian of the MKL can be computed and this information can be used to compute a better descent direction and the gradient. It is shown that HessianMKL outperforms SimpleMKL in terms of computational efficiency.

[XJKL08] (LevelMKL) proposes a hybrid method to combine SILP [SRS05, SRSS06] and SimpleMKL [RBCG07, RBCG08]. Despite the success of SILP and SimpleMKL, they have some shortcomings:

- the SILP method does not regularize the approximate solution of the cutting plane model.
- the SD (SimpleMKL) method only considers the current gradient.

LevelMKL exploits gradients of all the past iterations and regularizes the solution by projecting it to a level set to overcome these drawbacks. So LevelMKL can make sure that the new solution is close to the current one, and can reduce the objective function significantly.

We observe that the above mentioned algorithms: SILP, SD (SimpleMKL), HessianMKL, LevelMKL all belongs to the wrapper approach family. So we can summarize them together in Algorithm 1 [XJY⁺10].

Algorithm 1 Wrapper Approach for solving MKL

- 1: Initialize $\eta^0 \in \Delta$ and $t = 0$
 - 2: **repeat**
 - 3: Solve the dual of SVM with kernel $\kappa = \sum_{l=1}^m \eta_l^t \kappa_l$ and obtain optimal solution α^t
 - 4: Update kernel weights via cutting plane methods [SRS05, SRSS06], sub-gradient descent [RBCG07, RBCG08], second order descent [CR08], level set projection [XJKL08] or closed-form solution [XJY⁺10].
 - 5: Update $t = t + 1$
 - 6: **until** Convergence
-

Based on the equivalence between group-lasso and MKL, [XJY⁺10] formulates a closed-form solution for optimizing the kernel weights. This equivalence leads to an efficient algorithm for MKL, and is also able to generalize to L_p -MKL [KBS⁺09, MKZ11]. [XJY⁺10] can also be included in Algorithm 1.

2.3.1.3 Linear Function with Binary Coefficients

Another subcategory is restricting the combination coefficients of linear function to binary. It is reduced to kernel selection. We can discard those kernels whose coefficients are 0, while keep the others.

[XJY⁺09] develops an algorithm for non-monotonic feature selection that approximates the related combinatorial optimization problem by MKL. So as to achieve this goal, they define a kernel function:

$$\kappa_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^m \eta_l \kappa(\mathbf{x}_i[l], \mathbf{x}_j[l]) \quad (\text{Eq. 2.17})$$

where $\eta \in \{0, 1\}^m$. They further propose a strategy to derive a discrete solution for the relaxed problem (relax to continuous domain) with performance guarantee.

2.3.2 Effectiveness

We also further classify this category according to f_η into Linear Function and Non-Linear Function.

2.3.2.1 Linear Function

L_1 -norm of the kernel weights is most widely adopted by MKL methods. Its advantage is that it can lead to a sparse solution. But as argued in [KBL08], if the base kernels encode orthogonal information, the L_1 -norm may discard complementary information. So they study non-sparse multiple kernel learning by imposing an L_2 -norm constrain on the kernel weights. They also developed a Semi-Infinite Programming (SIP) formulation for non-sparse MKL. [CMR09a] analyzes the effect of L_2 -norm in MKL for regression problem. They conclude that using L_1 -norm improves the performance for a small number of kernels, but degrades the performance when combining a large number of kernels. L_2 -norm never decreases the performance, and for larger sets of candidate kernels it leads to significant increase. [FYT09] compares L_1 -norm and L_2 -norm in multimedia classification tasks. They conclude that L_2 -norm is preferred when the base kernels are complementary.

A natural extension to the L_2 -norm is the L_p -norm [KBS⁺09, MKZ11]. They generalize the MKL formulation for arbitrary L_p -norm ($p \geq 1$) by regularizing over the kernel coefficients or equivalently constraining them. [KBS⁺09, MKZ11] propose two alternative optimization strategies based on Newton descent and cutting planes. They apply L_p -norm MKL to real-world problems from computational biology, showing that non-sparse MKL achieves accuracies that go beyond the state-of-the-art. There are some works that provide theoretical analysis for L_p -norm ($p \geq 1$) regularized MKL, such as [Suz11, CMR10a], and so on.

[Suz11] gives a new generalization error bound of MKL for a general class of regularization. It shows a unified framework to derive the learning rate of MKL with arbitrary mixed-norm regularization. So this paper gives a theoretical tool to derive fast learning rates that is applicable to arbitrary mixed-norm regularization in a unified manner. It also shows a fast learning rate of L_p -norm MKL that is tightest among existing bounds.

As L_1 -norm emphasizes sparsity and L_p -norm ($p > 1$) can better keep complementary information among kernels, [YXY⁺11] proposes a generalized model incorporating a linear

combination of L_1 -norm and L_p -norm ($p > 1$) on the kernel weights. This type of constraint on the kernel weights is named elastic-net. It is obvious that previous MKL algorithms based on L_1 -norm and L_p -norm ($p > 1$) are special cases of it. [TS10] empirically investigates the best trade-off parameter for elastic-net regularization. They find that it depends not only on the sparsity of the true kernel-weight spectrum but also on the linear dependency among kernels and the number of samples. [SS12] investigates the learning rate of MKL with L_1 and elastic-net regularizations. Their analysis shows there is a trade-off between the sparsity and the smoothness when it comes to selecting which regularization to use.

[XTX13] presents a novel soft margin perspective for MKL. By introducing an additional slack variable to each quadratic constraint of MKL, which corresponds to one support vector machine model using a single base kernel, soft margin MKL can improve the effectiveness of MKL. Many existing MKL methods can be shown as special cases under the soft margin framework.

Most MKL algorithms do not take the base kernels' group structure into consideration. Sometimes, we may build a group of kernels from the same feature, even only 1 of them has non-zero weight, we have to extract this feature during the test phase. In such cases, as a trade-off between efficiency and effectiveness, we may want to keep or discard the whole group of kernels. [SGR08, SGR10] follow this idea and derive a MKL based on [RBCG07]. [JDR⁺09, ABTB⁺11] study the mixed-norm regularization of MKL. It is assumed that the given set of kernels are grouped into distinct components where each component is crucial for the learning task at hand. The formulation hence employs L_∞ regularization for promoting combinations at the component level and L_1 regularization for promoting sparsity among kernels in each component.

[GN08] considers the problem of automatically learning the kernel from general kernel classes. Given a set of candidate kernels, they apply existing finite MKL algorithms to compute the kernel weights, after that they add a new kernel to the candidate set, this process is repeated until converge.

[SS10] observes that the kernel weights obtained by existing MKL algorithms are not the sparsest. So they propose to use an additional log-based concave penalty term in the primal problem to induce sparsity in terms of groups of parameters. Their method is named Sparse Multiple Kernel Learning (SMKL). It is shown that for signal processing applications SMKL is able to obtain sparser solutions.

Directly applying MKL to noisy training data usually lead to suboptimal performance. [YMJ⁺12] addresses this challenge by casting it into a stochastic programming problem, and presenting a convex-concave formulation. They also present an accelerated mirror prox method for solving the related optimization problem.

There are some algorithms that extend MKL to multi-class and multi-label classification problems. [ZO07] provides a convenient and principled way for MKL with multi-class problems. They formulate it into a QCQP problem and further into two SILP problems. [HCK09] combines [ZO07] and the sparse kernel learning method of [WSB06] to obtain a sparse solution. [JSJY08, TCY09] provide extensions for multi-label settings.

2.3.2.2 Non-Linear Function

[CMR09b] learns kernels based on a polynomial combination of kernels. They analyze this problem in the case of regression and the kernel ridge regression problem.

[Bac08, JZH11] study hierarchical kernel learning approaches. [Bac08] considers learning a linear combination of exponential number of linear kernels. [JZH11] investigates a framework of Multi-Layer Multiple Kernel Learning (MLMKL) that aims to learn deep kernel machines. MLMKL framework offers higher flexibility than the regular MKL.

[dDMM04] proposes to combine kernels as follows:

$$\kappa_{\eta}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^m \eta_l(\mathbf{x}_i, \mathbf{x}_j) \kappa_l(\mathbf{x}_i, \mathbf{x}_j) \quad (\text{Eq. 2.18})$$

The weights are estimated heuristically. [GA08] proposes a localized multiple kernel learning algorithm which uses a gating model for selecting the appropriate kernel function locally. The optimization of localizing gating model and the kernel based classifier is done in a joint manner.

[VR07, VB09] propose GMKL. It extends the existing MKL formulations to learn general kernel combinations subject to general regularization. [JVV12b] develops a Spectral Projected Gradient (SPG) descent optimizer which takes into account second order information. It is demonstrated to be more efficient than GMKL which is optimized by projected gradient descent.

Generally, the non-linear combination MKL can lead to larger potential solution space, but it often leads to non-convex optimization problems. This usually means lower efficiency.

2.3.3 Other Methods

There are some other methods to build kernel machines using multiple kernels, we introduce some representative methods in this subsection. Some other methods, such as Bayesian [GR05a, GZ06] etc, are not included due to space limitation.

2.3.3.1 Similarity Based Methods

The most widely used similarity measure is the kernel alignment [CSTEK01, CSTE02].

Definition 2.2 *The (empirical) alignment of a kernel κ_1 with a kernel κ_2 with respect to the sample S is the quantity*

$$\hat{A}(S, \kappa_1, \kappa_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}} \quad (\text{Eq. 2.19})$$

where K_i is the kernel matrix for the sample S using kernel κ_i .

Researchers usually consider $K_2 = \mathbf{y}\mathbf{y}^\top$ as the optimal kernel, and try to optimize the alignment between a kernel κ with it.

[LCB⁺04] shows that kernel alignment can be used in their algorithm, leads to SDP and QCQP problems just as for SVM. [CMR10b] adopts a related but different definition of kernel alignment, and explores a two-stage technique and algorithm for learning kernels. There are some other works that adopt kernel alignment, such as [KSTC02] and [QL09].

Other similarity measures are also used, such as kernel distance [HCX08], FSM [NH08], KL divergence [YHC09], and so on.

2.3.3.2 Boosting Methods

[BME02, BZB04, BFDR05] adopt the idea of boosting for learning mixture-of-kernels models:

$$f(\mathbf{x}) = \sum_{i=1}^n \sum_{l=1}^m \alpha_{il} \kappa_l(\mathbf{x}_i, \mathbf{x}) + b \quad (\text{Eq. 2.20})$$

In [BME02], the authors address a kernel-based regression problem, and apply the idea of gradient boosting/column generation to optimize a heterogeneous kernel using a gradient descent algorithm in function space, which is named as the Multiple Additive Regression Kernels (MARK) algorithm. Unlike the MARK algorithm that is designed to mainly address a regression task, the authors in [BZB04] propose the Column Generation (CG) boosting algorithms from a family of kernels for both classification and regression tasks, which boost on kernel columns by employing the previous CG technique in [DBST02]. The similar idea is also extended to a semi-supervised learning setting in [BFDR05].

[CKS02] employs the idea of boosting for kernel design. In [CKS02], the authors formulate the kernel design problem as the construction of an accurate kernel from simple base kernels, and employ the boosting idea to perform the kernel construction process. Similar to the regular MKL method, the kernel design approach using boosting [CKS02] also learns a linear combination of kernels. It learns the single base kernel from empirical data in a transductive learning setting.

2.4 Other Kernel Methods beyond MKL

The idea of learning with multiple kernels has been adapted to solve many other machine learning problems, including transfer learning [DTX12], semi-supervised learning [WJHT10a, WHJT12], online learning [JHY10, HJZY13], learning of structured predictors [MSX⁺11],

multi-task learning [GKK11], Kernel Fisher Discriminant Analysis (KFDA) [KMB06, YJC08, YKMT09, YMB⁺10, YKMT12], clustering [ZKZ09, HCC12], dimension reduction [LLF08, LLF11], feature selection [XJY⁺09], etc. We summarize them on the left of Table 2.2.

Table 2.2: Other tasks and applications of learning with multiple kernels.

Other tasks	Applications
transfer learning [DTX12]	object recognition [HB07, VR07, MGL11]
semi-supervised learning [WJHT10a, WHJT12]	image retrieval [AHK ⁺ 10]
online learning [JHY10, HJZY13]	facial image [VB09, RSBP11]
learning of structured predictors [MSX ⁺ 11]	voice [DDKD08, LG09]
multi-task learning [GKK11]	traffic [SRB07, LTL ⁺ 12]
Kernel Fisher Discriminant Analysis [KMB06, YJC08, YKMT09, YMB ⁺ 10, YKMT12]	bioinformatics [LDC ⁺ 04, LCB ⁺ 04, SRSS06]
clustering [ZKZ09, HCC12]	medical [HHL ⁺ 11, GKN12]
dimension reduction [LLF08, LLF11]	anomaly detection [DMSO10]
feature selection [XJY ⁺ 09]	finance [XrLyZs10, YHL11]

[DTX12] proposes a Domain Transfer Multiple Kernel Learning (DTMKL) framework that can learn a kernel function and a classifier at the same time. The problem is formulated to minimize both the structural risk and the distribution mismatch between different domains. Experiments performed on 3 data sets show that DTMKL outperforms existing cross-domain and multiple kernel learning algorithms.

One of the challenging problems for large scale data mining is to design an algorithm that can work under the situation of little labeled data available and much noise exists. Though semi-supervised learning is one of the solutions, most existing methods can not incorporate multiple information sources. [WJHT10a, WHJT12] propose a Scalable Semi-Supervised Multiple Kernel Learning (S³MKL) method to deal with this problem. The kernels weights are regularized by group-lasso to enforce the sparsity in group level.

[JHY10, HJZY13] address a new research problem named Online Multiple Kernel Classification (OMKC), which learns a prediction function online based on multiple kernels. The target prediction function is a combination of multiple kernel classifiers different from typical

MKL algorithms, which mainly concern to learn the optimal kernel combination. The proposed algorithms for OMKC are in fact based on the fusion of two famous online learning algorithms: Perceptron and Hedge.

Obtaining good structured predictor requires much effort in feature/kernel design, which is typically via cross validation. It is known that learning of structured predictors is usually very slow, so [MSX⁺11] proposes a method that can learn structured predictors with multiple kernels online, i.e., both the kernel function and the target structured predictor are learned simultaneous in an online manner.

[GKK11] studies the problem of combining multi-task learning and multiple kernel learning. The proposed formulation trains coupled but nonidentical MKL models across multi-tasks. Particularly, the algorithm uses different kernel weights while regularizes them to be similar for each task.

The idea of learning with multiple kernels has also been adapted to improve the performance of Kernel Fisher Discriminant Analysis (KFDA). [KMB06, YJC08] replace the SVM criterion by Fisher Discriminant Analysis (FDA) and propose Multiple Kernel Fisher Discriminant Analysis (MKFDA). Similar to the development of MKL, as L_1 -norm is adopted in [KMB06, YJC08], it is then further extended to L_2 -norm [YKMT09] and L_p -norm [YMB⁺10, YKMT12].

[ZKZ09] develops a multiple kernel clustering algorithm by adopting the maximum margin clustering idea of [XNLS04] and a nonnegative linear combination of kernels. [HCC12] proposes a Multiple Kernel Fuzzy C-means (MKFC) algorithm, where fuzzy c-means is a popular soft clustering method. In fact, it is shown that multiple kernel k-means is a special case of MKFC.

Adopting multiple descriptors to better characterize the data has been a feasible way to improve performance in complex visual learning tasks. But the representation is typically high dimensional. [LLF08, LLF11] describe an approach that incorporates MKL and dimensionality reduction.

By restricting the combination coefficients of kernel combination to binary, it is possible to perform kernel selection. [XJY⁺09] develops an algorithm for non-monotonic feature selection by define the base kernels on each feature dimension.

2.5 Applications of Learning with Multiple Kernels

Learning with multiple kernels has been successfully applied in many real world applications, such as object recognition [HB07, VR07, MGL11], image retrieval [AHK⁺10], facial image [VB09, RSBP11], voice [DDKD08, LG09], traffic [SRB07, LTL⁺12], bioinformatics [LDC⁺04, LCB⁺04, SRSS06], medical [HHL⁺11, GKN12], anomaly detection [DMSO10], finance [XrLyZs10, YHL11], and so on. They are summarized on the right of Table 2.2.

[HB07] proposes a family of kernels between respective segmentation graphs of images, and select the most efficient one use multiple kernel learning. [VR07] investigates the problem of learning optimal descriptors for a given classification task by learning the optimal kernels as a combination of base kernels corresponding to base features which achieve different levels of trade-off. [MGL11] introduces a framework for multiple class object localization that incorporates different levels of contextual cues. To efficiently integrate different types of feature descriptions, they extend LMNN to support multiple kernels.

MKL for 1-class SVM is used in Pinview [AHK⁺10], a content-based image retrieval system that exploits implicit relevance feedback. But it is worth to mention that MKL for 1-class SVM is not specifically designed for an image retrieval task, image retrieval prefers ranking rather than classification.

[VB09] performs gender identification experiments from frontal facial images by combining kernels calculated on each individual feature. [RSBP11] introduces a new multi-resolution framework based on multiple kernel learning for facial landmark detection. Low resolution patches carry the global information and give a coarse but robust detection, while high resolution patches use local details, can refine this location.

[DDKD08] combines three different kernel functions by MKL and gets better results than the optimal linear score fusion obtained using logistic regression on speaker verification tasks. [LG09] describes a maximum-margin based scheme for learning kernel weights to combine derivative and parametric kernels for the speaker verification task.

[SRB07] presents a pedestrian detection method based on learning with multiple kernels, it can select and combine different kinds of image representations. [LTL⁺12] proposes a classification scheme for vehicle detection in wide area motion imagery by using GMKL, so as to learn the trade-off between HOG and Haar descriptors by constructing an optimal kernel with many base kernels.

[LDC⁺04, LCB⁺04] consider yeast proteins classification problems. Kernels built on different genomic data are combined using SDP formulation. The experiments show that in 9 out of 13 cases, it is better than SVM trained with each kernel. [SRSS06] successfully trains MKL on a 10 million splice site detection problem.

In order to recover the ability of movement affected by many diseases, rehabilitation is the only way to improve the situation for many clients. [HHL⁺11] proposes a robot with brain-computer interface for rehabilitation exercises, and use MKL to train the classifier. [GKN12] shows that using multiple kernel based classification, where the kernels are carefully selected for the different features, significantly increases the classification accuracy for medical images.

Learning with multiple kernels has also been applied to anomaly detection. [DMSO10] discusses an approach based on the theory of multiple kernel learning to detect potential safety anomalies in very large data bases of discrete and continuous data from world-wide aviation system.

Economic forecasting is an important research topic in management science. [XrLyZs10] proposes a forecasting method based on multiple kernel support vector regression. [YHL11] develops a two-stage multiple-kernel support vector regression algorithm by incorporating sequential minimal optimization and the gradient projection method. And the proposed algorithm has been applied to stock market forecasting.

2.6 Summary

This chapter gave a literature survey related to learning with multiple kernels. We introduced typical kernel methods that learned with a single kernel at first. After that, we surveyed existing algorithms by learning with multiple kernels and divided them into two groups: MKL and other kernel methods beyond MKL. And we also briefly introduced the applications of learning with multiple kernels.

In general, learning with multiple kernels is a generic idea that can be adapted to many areas and applied to many real world applications. So it is worthy of study.

Chapter 3

Multiple Kernel Boosting

This chapter addresses the first challenge posed in section 1.2, which is improving the efficiency and effectiveness of regular MKL algorithms for classification. The rest of this chapter is organized as follows. Section 3.1 reviews the related work. Section 3.2 formulates the proposed framework of multiple kernel boosting, gives two kinds of different approaches for solving the MKBoost problem, and briefly analyzes the time and space complexity. Section 3.3 gives the theoretical analysis. Section 3.4 discusses our empirical study. Section 3.5 summarizes this chapter.

3.1 Related Work

Kernel methods have been extensively studied in literature [STC04, CSTEK01, HTF09]. Most kernel methods usually assume some predefined parametric kernel, e.g., a gaussian kernel, is given a priori, where the parameters of the kernel function are usually determined empirically by cross validation. Several studies have been proposed to learn parametric, semi-parametric, or nonparametric kernel functions/matrices from labeled and/or unlabeled data. Example techniques include cluster kernels [CWS02], diffusion kernels [KL02], marginalized kernels [KTI03], spectral kernel learning [ZKGL04, HLC06, BH02], nonparametric kernel learning [HJL07], and so on.

Recently, Multiple Kernel Learning (MKL) [LCB⁺04, SRSS06, XJKL08, JHY10] has been actively studied, aiming to learn kernel based models by finding the optimal combination of multiple predefined kernels for classification tasks. Example algorithms include MKL by SDP [LCB⁺04], MKL by SILP [SRSS06], MKL by sub-gradient descent [RBCG08], MKL by the level based optimization [XJKL08], and MKL by the second-order Newton method [CR08]. Recently, some emerging works also attempt to improve the efficacy of regular MKL. For example, the ℓ_p -norm MKL method extended the regular ℓ_1 -norm MKL for arbitrary ℓ_p -norm MKL with $p > 1$. Besides, some recent studies [ZO07, JSJY08, TCY09] also attempt to address other issues of MKL, such as multi-class and multi-labeled classification. Unlike the existing MKL methods that have to solve complicated optimization tasks, our method employs a boosting approach, which is more efficient and scalable.

Moreover, our work is related to some existing works that adopt the idea of boosting for learning mixture-of-kernels models [BME02], [BZB04], [BFDR05]. In [BME02], the authors addressed a kernel-based regression problem, and applied the idea of gradient boosting/column generation to optimize a heterogeneous kernel using a gradient descent algorithm in function space, which was named as the Multiple Additive Regression Kernels (MARK) algorithm. Unlike the MARK algorithm that were designed to mainly address a regression task, the authors in [BZB04] proposed the column generation (CG) boosting algorithms from a family of kernels for both classification and regression tasks, which boost on kernel columns by employing the previous column generation (CG) technique in [DBST02]. The similar idea was also extended to a semi-supervised learning setting in [BFDR05]. Our work differs from these studies in that they all are based on the column generation idea, which usually engages a huge number of columns and is thus computationally very intensive.

Our work is also related to some existing work using the idea of boosting for kernel design [CKS02]. The authors in [CKS02] formulated the kernel design problem as the construction of an accurate kernel from simple base kernels, and employ the boosting idea to perform

the kernel construction process. Similar to the regular MKL method, the kernel design approach using boosting [CKS02] also learns a linear combination of kernels. However, their approach is limited in that it learns the single base kernel from empirical data in a transductive learning setting, while regular MKL and our approach learn from a set of predefined kernels in an inductive learning setting, which can be more general and flexible to handle multiple sources of data in real applications.

There are some works which adopt boosting technique to improve kernel methods, such as, Boost-SMO [PMD00], Boost-SVM [ZR08], AdaBoostSVM [SWS05, LWS08], AdaBoost with SVM [VSN08], etc. But they are all unable to deal with multiple kernels that may encode information originates from multiple resources.

Finally, it is worth to mention that [CMR12] presents an approach based on ensembles of kernel predictors, this work is published a little later than our proposed MKBoost algorithm [XH11], which can be seemed as a special case of it from a certain perspective.

3.2 Multiple Kernel Boosting

In this section, we present a novel framework of multiple kernel boosting (MKBoost), which adapts boosting techniques for multiple kernel learning. Before presenting our MKBoost algorithms, we will first introduce the problem and review the regular multiple kernel learning (MKL). To ease our presentation, we will restrict our discussion on a typical binary classification task. Similar algorithms in this work can be easily extended to multi-class classification tasks.

3.2.1 Problem Formulation

Consider a given set of training examples $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$ where $y_i \in \{-1, +1\}$, $i = 1, \dots, n$, and a collection of m kernel functions $\mathcal{K} = \{\kappa_j : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, j = 1, \dots, m\}$.

The goal of our multiple kernel boosting task is to learn a kernel based classifier f , which is an ensemble of kernel classifiers using the collection of m kernels trained from the given training data examples. Typically, we express such a kernel classifier as:

$$f(\mathbf{x}) = \sum_{t=1}^T \alpha_t f_t(\mathbf{x}) \quad (\text{Eq. 3.1})$$

where f_t is a hypothesis learned from a boosting trial, α_t is its associated weight in the final classifier, and T is the total number of boosting trials. The main challenge of multiple kernel boosting is to develop an effective boosting scheme to learn the optimal hypothesis f_t and its combination weight α_t at each boosting trial.

For instance, if we consider only a single kernel κ , f_t can be learned by applying any regular kernel classifier such as SVM from a subset of n_s training examples in a boosting trial, i.e.,

$$f_t(\mathbf{x}) = \sum_{i=1}^{n_s} \beta_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) \quad (\text{Eq. 3.2})$$

where β_i 's are the coefficients of an SVM model. Such an approach reduces to a regular boosting approach using a single kernel SVM as the base classifier.

In this chapter, we present two kinds of approaches to solving the MKBoost problem. Before presenting our algorithms, we first review some basics of regular MKL to better understand the motivation and difference of our work.

3.2.2 Multiple Kernel Learning

The goal of a regular MKL task is to identify the optimal combination of m kernels, denoted by $\theta = (\theta_1, \dots, \theta_m)$ by following maximum margin learning principle, which can be cast into the following optimization problem:

$$\min_{\theta \in \Delta} \min_{f \in \mathcal{H}_{K(\theta)}} \frac{1}{2} \|f\|_{\mathcal{H}_{K(\theta)}}^2 + C \sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i) \quad (\text{Eq. 3.3})$$

where $\Delta = \{\theta \in \mathbb{R}_+^m \mid \theta^\top \mathbf{e}_m = 1\}$, $K(\theta)(\cdot, \cdot) = \sum_{j=1}^m \theta_j \kappa_j(\cdot, \cdot)$, $\ell(f(\mathbf{x}_i), y_i) = \max(0, 1 - y_i f(\mathbf{x}_i))$.

In the above, we use notation $\mathbf{e}_m$ to represent a vector of m dimensions with all its elements being 1. The above formulation can also be turned into the following min-max optimization task:

$$\min_{\theta \in \Delta} \max_{\alpha \in \Xi} \left\{ \alpha^\top \mathbf{e}_n - \frac{1}{2} (\alpha \circ \mathbf{y})^\top \left(\sum_{j=1}^m \theta_j \mathbf{K}^j \right) (\alpha \circ \mathbf{y}) \right\}$$

where $\mathbf{K}^j \in \mathbb{R}^{n \times n}$ with $\mathbf{K}_{p,q}^j = \kappa_j(\mathbf{x}_p, \mathbf{x}_q)$, $\Xi = \{\alpha | \alpha \in [0, C]^n\}$, and $\circ$ defines the element-wise product between two vectors. Despite some encouraging results achieved recently [RBCG08, XJKL08], developing an efficient and scalable algorithm for solving the regular MKL problem remains an open challenge.

3.2.3 Deterministic MKBoost Algorithms

The general idea of MKBoost is to apply boosting techniques to learn a classifier using multiple kernels. To this purpose, we follow the typical procedure of a popular and successful boosting algorithm, i.e., Adaptive Boosting known as “Adaboost” [FS97].

In particular, we repeatedly learn some kernel classifiers with multiple kernels f_t through a series of boosting trials $t = 1, \dots, T$, where T denotes the total number of boosting trials. At each boosting trial, a distribution of weights D_t is engaged to indicate the importance of the training examples for learning. At each trial, we increase the weights of the wrongly classified examples and/or decrease the weights of those correctly classified examples in order to focus on those examples that are hard to be correctly classified.

During each boosting trial, we first sample a subset of n_s training examples according to distribution D_t , where n_s is specified by a predefined *boosting sampling ratio* that determines the proportion of training data to be sampled at each boosting trial. Once obtaining the subset of training data, the next key issue is how to learn the kernel based classifier f_t from these training data. The first approach is to learn one classifier f_t^j with each kernel κ_j from the set of m kernels using a regular kernel method, e.g., SVM used in our study. Based on the set of m

base classifiers, we can further measure the misclassification performance of each classifier f_t^j with kernel κ_j over distribution D_t of the whole collection of training data:

$$\epsilon_t^j = \epsilon(f_t^j) = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i) \quad (\text{Eq. 3.4})$$

where $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the condition is true, and 0 otherwise.

As a result, we can build the classifier f_t for the t -th boosting trial by choosing the best classifier with the smallest misclassification rate, i.e.,

$$f_t = \arg \min_{f_t^j, j \in \{1, \dots, m\}} \epsilon(f_t^j) \quad (\text{Eq. 3.5})$$

The next step follows the similar procedure of Adaboost by computing the misclassification rate ϵ_t for the combined classifier f_t over the distribution D_t on the whole collection of training data:

$$\epsilon_t = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t(\mathbf{x}_i) \neq y_i) \quad (\text{Eq. 3.6})$$

The last step of each boosting trial is to update the weight of each training example $D_{t+1}(i)$ as follows:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } f_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t} & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

where $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ and Z_t is a normalization factor to make D_{t+1} a distribution.

Finally, after finishing all T boosting trials, the algorithm outputs the final classifier as follows:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t f_t(\mathbf{x})\right) \quad (\text{Eq. 3.7})$$

We refer to the above MKBoost algorithm as “MKBoost-D1” for short. The details of the proposed algorithm are shown in Algorithm 2.

In the above algorithm, at each boosting trial, we simply choose the best classifier among the m kernel classifiers as the classifier and simply discard the other $m - 1$ classifiers. In some

Algorithm 2 The MKBoost algorithm (MKBoost-D1)

- 1: INPUT:
 - training data: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$
 - kernel functions: $\kappa_j(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, j = 1, \dots, m$
 - initial distribution $D_1(i) = 1/n, i = 1, \dots, n$
- 2: **for** $t = 1, \dots, T$ **do**
- 3: sample n_s examples using distribution D_t
- 4: **for** $j = 1, \dots, m$ **do**
- 5: train weak classifier with kernel κ_j :
 $f_t^j : \mathcal{X} \rightarrow \{-1, +1\}$
- 6: compute the training error over D_t :
 $\epsilon_t^j = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i)$
- 7: **end for**
- 8: select the best classifier with the minimal error rate

$$f_t = \arg \min_{f_t^j} \epsilon_t^j = \arg \min_{f_t^j} \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i),$$

- 9: choose $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$, where $\epsilon_t = \min_{j \in \{1, \dots, m\}} \epsilon_t^j$
- 10: Update $D_{t+1}(i)$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } f_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t} & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

Z_t is a normalization factor to make D_t a distribution.

- 11: **end for**
 - 12: OUTPUT: $f(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t f_t(\mathbf{x})\right)$
-

situation, it is possible that other kernel classifiers may make complementary contribution in improving the performance. Thus, we propose another way to build the classifier by combining all these m classifiers, each of which is assigned with a weight. Specifically, we suggest to build the classifier at the t -th boosting trial:

$$f_t(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^m \alpha_t^j f_t^j(\mathbf{x})\right) \quad (\text{Eq. 3.8})$$

where the weight α_t^j is computed based on the misclassification rate ϵ_t^j , i.e.,

$$\alpha_t^j = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t^j}{\epsilon_t^j}\right) \quad (\text{Eq. 3.9})$$

Algorithm 3 The MKBoost Algorithm (MKBoost-D2)

```

1: INPUT:
    • training data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ 
    • kernel functions:  $\kappa_j(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, j = 1, \dots, m$ 
    • initial distribution  $D_1(i) = 1/n, i = 1, \dots, n$ 
2: for  $t = 1, \dots, T$  do
3:   sample a set of  $n_s$  examples using distribution  $D_t$ 
4:   for  $j = 1, \dots, m$  do
5:     train weak classifier with kernel  $\kappa_j$ :
        $f_t^j : \mathcal{X} \rightarrow \{-1, +1\}$ 
6:     compute the training error over  $D_t$ :
        $\epsilon_t^j = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i)$ 
7:     choose  $\alpha_t^j = \frac{1}{2} \ln(\frac{1-\epsilon_t^j}{\epsilon_t^j})$ 
8:   end for
9:   combine the  $m$  classifiers:
        $f_t(\mathbf{x}) = \text{sign}(\sum_{j=1}^m \alpha_t^j f_t^j(\mathbf{x}))$ 
10:  compute the training error over  $D_t$ :
        $\epsilon_t = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t(\mathbf{x}_i) \neq y_i)$ 
11:  choose  $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ 
12:  Update  $D_{t+1}(i)$ :
       
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } f_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t} & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

        $Z_t$  is a normalization factor to make  $D_t$  a distribution.
13: end for
14: OUTPUT:  $f(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t f_t(\mathbf{x}))$ 

```

We refer to this MKBoost algorithm as “MKBoost-D2” for short. The other part of this algorithm is same to MKBoost-D1. Algorithm 3 gives the details of the proposed MKBoost-D2 algorithm.

3.2.4 Stochastic MKBoost Algorithms

One important limitation for the above two MKBoost algorithms is that they need to repeatedly train a set of m kernel classifiers at each boosting trial. This could be computationally intensive

Algorithm 4 The MKBoost algorithm (MKBoost-S1)

```

1: INPUT:
    • training data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ 
    • kernel functions:  $\kappa_j(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, j = 1, \dots, m$ 
    • init. data distribution  $D_1(i) = 1/n, i = 1, \dots, n$ 
    • init. kernel sampling prob.  $S_1(j) = 1, j = 1, \dots, m$ 
    • sampling decay rate  $0 < \beta < 1$ 
2: for  $t = 1, \dots, T$  do
3:   Sample  $n_s$  examples using distribution  $D_t$ 
4:   Sample a set of kernels  $\mathcal{K}_t$  by sampling probability  $S_t$ 
5:   for  $\kappa_j \in \mathcal{K}_t$  do
6:     Train weak classifier with kernel  $\kappa_j$ :
        $f_t^j : \mathcal{X} \rightarrow \{-1, +1\}$ 
7:     Compute the training error over  $D_t$ :
        $\epsilon_t^j = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i)$ 
8:     Update  $S_{t+1}(j) \leftarrow S_t(j) \beta^{\epsilon_t^j}$ 
9:   end for
10:  Update  $S_{t+1}(j) \leftarrow S_{t+1}(j) / Z^S, Z^S = \max(S_{t+1})$ 
11:   $f_t = \arg \min_{\kappa_j \in \mathcal{K}_t} \epsilon_t^j = \arg \min_{\kappa_j \in \mathcal{K}_t} \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i)$ 
12:  Compute the training error over  $D_t$ :
      $\epsilon_t = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t(\mathbf{x}_i) \neq y_i)$ 
13:   $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ 
14:  Update  $D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i f_t(\mathbf{x}_i))$ 
15: end for
16: OUTPUT:  $f(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t f_t(\mathbf{x}))$ 
    
```

if the number of kernels m is large in some applications. To address this limitation, in this section, we propose a *stochastic* learning approach for MKBoost, which aims to avoid the need of training all the m kernel classifiers. To ease our presentation, we refer to the new algorithms as *stochastic* MKBoost algorithms, and the previous two algorithms as *deterministic* MKBoost algorithms.

The intuitive idea of a stochastic MKBoost approach is that we could try to avoid unnecessary costs of training classifiers with some kernels that have relatively poor classification

Algorithm 5 The MKBoost algorithm (MKBoost-S2)

```

1: INPUT:
    • training data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ 
    • kernel functions:  $\kappa_j(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, j = 1, \dots, m$ 
    • init. data distribution  $D_1(i) = 1/n, i = 1, \dots, n$ 
    • init. kernel sampling prob.  $S_1(j) = 1, j = 1, \dots, m$ 
    • sampling decay rate  $0 < \beta < 1$ 
2: for  $t = 1, \dots, T$  do
3:   Sample  $n_s$  examples using distribution  $D_t$ 
4:   Sample a set of kernels  $\mathcal{K}_t$  by sampling probability  $S_t$ 
5:   for  $\kappa_j \in \mathcal{K}_t$  do
6:     Train weak classifier with kernel  $\kappa_j$ :
        $f_t^j : \mathcal{X} \rightarrow \{-1, +1\}$ 
7:     Compute the training error over  $D_t$ :
        $\epsilon_t^j = \sum_{i=1}^n D_t(i) \mathbb{I}(f_t^j(\mathbf{x}_i) \neq y_i)$ 
8:     Choose  $\alpha_t^j = \frac{1}{2} \ln(\frac{1-\epsilon_t^j}{\epsilon_t^j})$ 
9:     Update  $S_{t+1}(j) \leftarrow S_t(j) \beta^{\epsilon_t^j}$ 
10:  end for
11:  Update  $S_{t+1}(j) \leftarrow S_{t+1}(j) / Z^S, Z^S = \max(S_{t+1})$ 
12:   $h_t(\mathbf{x}) = \text{sign}(\sum_{\kappa_j \in \mathcal{K}_t} \alpha_t^j f_t^j(\mathbf{x}))$ 
13:  Compute the training error over  $D_t$ :
        $\epsilon_t = \sum_{i=1}^n D_t(i) \mathbb{I}(h_t(\mathbf{x}_i) \neq y_i)$ 
14:  Choose  $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ 
15:  Update  $D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$ 
16: end for
17: OUTPUT:  $f(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}))$ 

```

performance for the classification task. To this purpose, we introduce a variable $S_t(j)$ as the kernel sampling probability, which indicates how likely a kernel κ_j will be sampled at the t -th boosting trial. At the beginning of the MKBoost algorithm, all $S_1(j)$ values are set to 1, which means that all m kernels will be definitely selected at the first boosting trial.

For each boosting trial, we sample a subset of kernels according to the kernel sampling probability S_t . The proposed MKBoost algorithm will train kernel classifiers only for those

selected kernels. At the end of each boosting trial, we update the kernel sampling probability according to its classification performance:

$$S_{t+1}(j) \leftarrow S_t(j)\beta^{e_t^j} \quad (\text{Eq. 3.10})$$

where $\beta \in (0, 1)$ is a constant parameter introduced as a sampling decay factor for updating the kernel sampling probability, and e_t^j is the misclassification rate of the kernel classifier with a selected kernel κ_j . The above formula indicates the larger the misclassification rate, the more decay penalty will be applied to the kernel to reduce the chance of being sampled in the next trial. Finally, at the end of a boosting trial, we conduct a normalization step by ensuring all kernel sampling weights are in $[0, 1]$. This normalization step could affect the sampling weights of those kernels that are not selected.

By incorporating the above kernel sampling scheme, we suggest two stochastic MKBoost algorithms, MKBoost-S1 and MKBoost-S2, which are corresponding to the previous two deterministic MKBoost algorithms, respectively. Specifically, at each boosting trial, MKBoost-S1 choose the best classifier among a set of classifiers as the classifier of this trial and discard the others, while the classifier of MKBoost-S2 is a combination of them. The details of these two algorithms are shown in Algorithm 4 and 5.

3.2.5 Time and Space Complexity

Compared with the regular MKL methods, the MKBoost algorithms enjoy significant advantages in their high efficiency and scalability performance.

Time Complexity. We first denote by $\mathcal{C}(n_s)$ the time complexity of training a base kernel classifier from a collection of n_s examples, where n_s is specified by the boosting sampling ratio. As we adopt SVM based on the SMO solver in the LIBSVM library [cCL01], the empirical complexity of $\mathcal{C}(n_s)$ is about $O(n_s^{1.4})$ to $O(n_s^{2.3})$, depending on different datasets and parameter settings. Since all the proposed MKBoost algorithms follow the similar framework, they in

general share the same worst case time complexity, i.e., $O(T \times m \times (n + \mathcal{C}(n_s)))$, where T is the total number of boosting trials, m is the total number of kernels, and n is total number of training examples. For a large-scale application, we could choose a small value of n_s ($n_s \ll n$), making the algorithms efficient and scalable to large classification tasks. Finally, comparing the deterministic and stochastic variants of the MKBoost algorithms, despite sharing the same worst case time complexity, the stochastic algorithms are empirically more efficient than the deterministic ones because the number of sampled kernels is usually smaller than m especially in the later boosting stages.

Space Complexity. We denote by $\mathcal{S}(n_s)$ the space complexity of a base kernel classifier trained from a set of n_s examples, which is typically equal to or smaller than $O(n_s)$. For the proposed MKBoost algorithms, MKBoost-D1 and MKBoost-S1 share the same worst case space complexity, i.e., $O(T \times \mathcal{S}(n_s))$, because both of them choose the best kernel classifier at each boosting trial. In contrast, MKBoost-D2 and MKBoost-S2 have higher space complexity. Specifically, both of them have the worst case space complexity of $O(T \times m \times \mathcal{S}(n_s))$, but the empirical space complexity of MKBoost-S2 is lower than that of MKBoost-D2 because MKBoost-S2 usually samples a number of kernels that is smaller than m at each boosting trial.

3.3 Theoretical Analysis

Similar to the theoretical analysis of Adaboost, we have the following theorem for MKBoost-D1:

Theorem 2 Assume $\epsilon_t^j = 1/2 - \gamma_t^j$, $\gamma_t^j \in (0, 1/2)$, $\forall j \in \{1, \dots, m\}$ and $t \in \{1, \dots, T\}$, which implies the weak classifiers perform better than random guess. For the proposed algorithm, we have the empirical risk, i.e., $R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}[y_i \neq f(\mathbf{x}_i)]$ of the final hypothesis f bounded by the following inequality

$$R_{emp}(f) \leq \exp(-2 \sum_{t=1}^T \gamma_t^2)$$

where $\gamma_t = \max_{j \in \{1, \dots, m\}} \gamma_t^j$. If further assume $\gamma_t \geq \gamma > 0, \forall t$, we have

$$R_{emp}(f) \leq \exp(-2\gamma^2 T).$$

Proof: Firstly, according to the relation between D_{t+1} and D_t , we have

$$\begin{aligned} D_{T+1}(i) &= D_T(i) \frac{\exp(-\alpha_T y_i f_T(\mathbf{x}_i))}{Z_T} \\ &= \dots \\ &= D_1(i) \frac{\exp(-y_i \sum_{t=1}^T \alpha_t f_t(\mathbf{x}_i))}{\prod_{t=1}^T Z_t} \\ &= D_1(i) \frac{\exp(-y_i f(\mathbf{x}_i))}{\prod_{t=1}^T Z_t} \end{aligned}$$

Sum the above inequality over $i = 1, \dots, n$ gives

$$1 = \sum_{i=1}^n D_{T+1}(i) = \sum_{i=1}^n D_1(i) \frac{\exp(-y_i f(\mathbf{x}_i))}{\prod_{t=1}^T Z_t} = \frac{1}{n} \sum_{i=1}^n \frac{\exp(-y_i f(\mathbf{x}_i))}{\prod_{t=1}^T Z_t}$$

where we used the facts D_{T+1} is a distribution, and $D_1(i) = 1/n$. This equation indicates

$$\frac{1}{n} \sum_{i=1}^n \exp(-y_i f(\mathbf{x}_i)) = \prod_{t=1}^T Z_t$$

Using the above inequality, we can bound the empirical risk of the final classifier as

$$\begin{aligned} R_{emp}(f) &= \frac{1}{n} \sum_{i=1}^n \mathbb{I}[y_i \neq f(\mathbf{x}_i)] \\ &= \frac{1}{n} \sum_{i=1}^n \mathbb{I}[y_i f(\mathbf{x}_i) \leq 0] \\ &\leq \frac{1}{n} \sum_{i=1}^n \exp(-y_i f(\mathbf{x}_i)) \\ &= \prod_{t=1}^T Z_t \end{aligned}$$

To bound $R_{emp}(f)$, we will provide an upper bound of Z_t as follows

$$\begin{aligned}
Z_t &= \sum_{i=1}^n D_t(i) \exp(-\alpha_t y_i f_t(\mathbf{x}_i)) \\
&= \sum_{i=1}^n D_t(i) \exp(-\alpha_t) \mathbb{I}[y_i = f_t(\mathbf{x}_i)] + \sum_{i=1}^n D_t(i) \exp(\alpha_t) \mathbb{I}[y_i \neq f_t(\mathbf{x}_i)] \\
&= (1 - \epsilon_t) \sqrt{\epsilon_t / (1 - \epsilon_t)} + \epsilon_t \sqrt{(1 - \epsilon_t) / \epsilon_t} = 2\sqrt{\epsilon_t(1 - \epsilon_t)} \\
&= 2\sqrt{(1/2 - \gamma_t)(1/2 + \gamma_t)} \\
&= \sqrt{1 - 4\gamma_t^2} \\
&\leq \sqrt{\exp(-4\gamma_t^2)} \\
&= \exp(-2\gamma_t^2),
\end{aligned}$$

where we used the fact $\gamma_t = \max_j \gamma_t^j = \max_j (1/2 - \epsilon_t^j) = 1/2 - \min_j \epsilon_t^j = 1/2 - \epsilon_t$.

Combining the above two inequalities concludes the first part of the theorem. Given the first part, the second one of the theorem is easy to be verified.

This theorem can be directly applied to MKBoost-S1 which also choose the best classifier as the classifier of each boosting trial. Furthermore, for MKBoost-D2 and MKBoost-S2, if we treat the combined classifier in each boosting trial as a single classifier, a slightly modified version of this theorem is also applicable to them. We omit the details here due to space limitation.

3.4 Experiments

In this section, we conduct an extensive set of experiments to examine the classification performance of the proposed MKBoost algorithms by comparing with a number of state-of-the-art techniques in literature. All the source code and datasets have been made available¹.

¹<http://www.cais.ntu.edu.sg/chhoi/mkboost/>

3.4.1 Experimental Testbed

We evaluate the performance of MKBoost algorithms for both binary and multiclass classification tasks. We randomly choose a number of publicly available datasets from web machine learning repositories². TABLE 3.1 shows the summary of the datasets in our experiments. Note that the sizes of these datasets are not very large because we want to compare our algorithms with other existing MKL algorithms, most of them are usually not scalable and only applicable to relatively small-sized datasets. In general, our algorithms are efficient and scalable for large applications.

Table 3.1: Summary of the datasets (where C, n, D denote # classes, #examples, #features, respectively).

Abbr.	name	source	C	n	D
D1	a1a	UCI	2	1605	123
D2	a3a	UCI	2	3185	123
D3	a4a	UCI	2	4781	123
D4	balance-scale	UCI	2	576	4
D5	german.numer	Statlog	2	1000	24
D6	glass	UCI	2	214	10
D7	ionosphere	UCI	2	351	34
D8	monks1	UCI	2	556	6
D9	monks2	UCI	2	1033	6
D10	satimage	Statlog	6	4435	36
D11	segment	Statlog	7	2310	19
D12	sonar	UCI	2	208	60
D13	svmguide3	CWH03a	2	1243	21
D14	svmguide4	CWH03a	6	300	10
D15	vehicle	Statlog	4	846	18
D16	wdbc	UCI	2	569	30

3.4.2 Comparison Schemes

In our experiments, we compare the proposed MKBoost algorithms against several state-of-the-art MKL algorithms using different optimization techniques. Besides, we also compare with

²<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

some boosting based kernel methods. In addition, we compare with two recently published algorithms which are published later than our proposed MKBoost algorithms.

The regular MKL algorithms in our comparisons include:

- MKL-SD: An MKL algorithm solved by a Subgradient Descent (SD) method [RBCG08].
- MKL-SILP: An MKL algorithm solved by Semi-Infinite Linear Program (SILP) [SRSS06].
- MKL-Level: An MKL algorithm solved by extended level optimization method [XJKL08].
- MKL-Hessian: An MKL algorithm solved by a second order Newton update optimization method [CR08].
- Lp-MKL: An extended MKL algorithm that generalizes the regular ℓ_1 -norm MKL method to arbitrary ℓ_p -norm MKL [KBS⁺09].

The existing boosting based algorithms include:

- Boost-Exp: An algorithm uses boosting paradigm to perform the kernel construction process [CKS02], with ExpLoss: $\exp(-yf(\mathbf{x}))$.
- Boost-Log: An algorithm uses boosting paradigm to perform the kernel construction process [CKS02], with Logloss: $\log(1 + \exp(-yf(\mathbf{x})))$.
- Boost-CG-L1: An algorithm uses column-generation boosting methods to learn mixture of kernels [BZB04], with L1 norm.
- Boost-CG-L2: An algorithm uses column-generation boosting methods to learn mixture of kernels [BZB04], with L2 norm.
- Boost-SVM: An algorithm uses Adaboost to improve SVM learning accuracy [ZR08].

Those two recently published algorithms include:

- EKP: An algorithm solves learning with multiple kernels problem based on ensembles of kernel predictors [CMR12].
- SMMKL: An algorithm which presents a novel soft margin perspective for MKL [XTX13].

3.4.3 Experimental Setup

For the setup of our experiments, we follow the typical approach used in the previous MKL studies in literature. In particular, for each dataset, we create a set of 17 base kernels, i.e.,

- Gaussian kernels with 14 different widths ($2^{-6}, 2^{-5}, \dots, 2^7$) on all features.
- Polynomial kernels of degree 1 to 3 on all features.

To implement the existing MKL algorithms, we adopt the SimpleMKL toolbox [RBCG08] that includes the implementations of both SILP and SD algorithms, the LevelMKL toolbox [XJKL08] that implements MKL-Level, and the MKL-Hessian toolbox provided by the author of [CR08]. For running the above existing MKL algorithms, we adopt their default settings suggested by the three toolboxes, which employ the duality gap as the stopping criterion, and stop the optimization process when the duality gap is less than a threshold (0.01) or the max number (500) of optimization iterations is reached. For Lp-MKL, as no code available, we implemented it by ourselves, and set $p = 2$ for the best tradeoff of accuracy and efficiency.

For Boost-Exp and Boost-Log [CKS02], we implemented them by ourselves as no code is available. We use linear kernel as base kernel, adopt Matlab built-in function “eig” to compute the generalized eigenvector, and set the number of boosting rounds to 30 as suggested in [CKS02] (the algorithm involves the computation of generalized eigenvector [HJ85] which is very time-consuming, this parameter setting is empirically a good tradeoff between efficacy and efficiency), . For Boost-CG-L1 and Boost-CG-L2 [BZB04], as no code is available too, we implemented them by ourselves, where we adopt MOSEK³ to solve the restricted optimization

³<http://www.mosek.com/>

problems instead of the commercial package ILOG CPLEX 8.0 used by the authors [BZB04]. For Boost-SVM [ZR08], 10-fold cross validation is adopted to select the best kernel, other settings are the same as in our MKBoost algorithms.

For EKP we implement it by ourselves as no code is available. The training of EKP includes two stages. For the first stage, base hypotheses are learned by using SVM. For the second stage, the combination weights of the kernels are obtained by solving standard SVM augmented with an explicit constraint $\mu \geq 0$ (μ is the combination weights), the parameter for this stage is determined by cross validation. For SMMKL, we experiment with the hinge loss soft margin MKL. Because based on the experimental results in [XTX13], its performance is comparable with the square hinge loss soft margin MKL, but it is more efficient. By following the setting in [XTX13], the parameter θ is set to $\frac{1}{vm}$, where v is a ratio parameter determined by cross validation and m is the number of different kernels.

For the implementation of our MKBoost algorithms, by default, we set the total number of boosting trials T to 100, the boosting sampling ratio to 0.2, and the sampling decay factor β to 2^{-5} for stochastic MKBoost algorithms. For SVM, we adopt the popular LIBSVM toolbox [cCL01] as the SVM solver. For the multiclass classification tasks, we adopt a one-to-one approach to training a set of binary classifiers. Finally, all the experiments were running in Matlab on a Linux machine with 3GHz Intel CPU and 16GB RAM.

For all the experiments, we repeat each algorithm 20 times on every dataset. Similar to the previous studies, for each run, 50% of the examples were randomly sampled as training data and the remaining were used for test. The training data were normalized to have zero mean and unit variance, and the test data were then normalized using the mean and variance of the training data. The penalty cost parameter C was fixed to 50 for SVM and the compared algorithms. To avoid unstable results for stochastic MKBoost algorithms, we run 10 times under each setting and report average performances. Finally, we report both classification accuracy and time cost for performance evaluation.

3.4.4 Comparison Results

We first compare the performances between two deterministic MKBoost algorithms (MKBoost-D1 and MKBoost-D2) and the five existing MKL algorithms (MKL-SD, MKL-SILP, MKL-Level, MKL-Hessian and Lp-MKL). Then we compare two deterministic MKBoost algorithms with boosting based algorithms (Boost-Exp, Boost-Log, Boost-CG-L1, Boost-CG-L2, Boost-SVM). After that we compare two deterministic MKBoost algorithms with those two recently published algorithms (EKP and SMMKL). Further, we will compare both the accuracy and efficiency performances between the deterministic and stochastic MKBoost algorithms. Finally, we will evaluate the effects of various parameters that may influence the performance of the MKBoost algorithms. To examine statistical significance of the comparisons, for the experimental results reported below, we highlight the best result in each group in bold font by conducting student t -tests with the significance level $\alpha = 0.05$.

3.4.4.1 MKBoost vs. Regular MKL

TABLE 3.2 shows the results of comparisons between the two deterministic MKBoost algorithms and the five existing MKL algorithms.

Several observations can be drawn from the results. First of all, in terms of classification accuracy performance, among the four ℓ_1 -norm MKL algorithms (MKL-SD, MKL-SILP, MKL-Level, and MKL-Hessian), we found that their accuracy performances are generally comparable, in which no one algorithm significantly performs better than the others. This is consistent to the previous MKL studies as all these MKL algorithms are essentially based on the same formulation, but adopt different optimization techniques to speed up the optimization process. Further, comparing the four regular MKL algorithms and the two proposed MKBoost algorithms, we found the MKBoost algorithms considerably outperform the four regular MKL algorithms in most cases. For example, for dataset D6(“glass”), the four regular MKL algorithms achieved the accuracy of no more than 95%, while the proposed deterministic MKBoost

Table 3.2: Comparison between two deterministic MKBoost algorithms and five different MKL algorithms.

Dataset	Metric	MKBoost-D1	MKBoost-D2	MKL-SD	MKL-SILP	MKL-Level	MKL-Hessian	Lp-MKL
D1	accuracy	0.8122	0.8093	0.7547	0.7547	0.7544	0.7544	0.7607
	std	± 0.0077	± 0.0086	± 0.0009	± 0.0009	± 0.0000	± 0.0000	± 0.0066
	time	4.8513	4.9239	34.5300	15.6220	30.8875	15.169	19.865
	std	± 0.0750	± 0.0592	± 3.5283	± 1.9550	± 3.6841	± 1.6931	± 6.3624
D2	accuracy	0.8137	0.8122	0.7576	0.7576	0.7575	0.7575	0.7597
	std	± 0.0051	± 0.0048	± 0.0003	± 0.0003	± 0.0000	± 0.0000	± 0.0040
	time	21.5289	21.9411	113.2365	62.0065	101.0525	60.216	85.2905
	std	± 0.4417	± 0.5168	± 8.0158	± 4.7652	± 7.2299	± 4.0650	± 29.6062
D3	accuracy	0.8110	0.8103	0.7515	0.7515	0.7515	0.7515	0.7561
	std	± 0.0052	± 0.0037	± 0.0001	± 0.0001	± 0.0000	± 0.0000	± 0.0033
	time	60.6319	62.5635	258.1065	157.279	238.038	152.84	268.5895
	std	± 2.4952	± 3.6807	± 16.9583	± 9.4822	± 17.7089	± 3.7433	± 88.5061
D4	accuracy	0.9858	0.9907	0.9464	0.946	0.9726	0.9642	0.9809
	std	± 0.0079	± 0.0057	± 0.0099	± 0.0132	± 0.0102	± 0.0115	± 0.0078
	time	0.7658	0.8683	2.952	1.3615	5.175	1.5255	4.1735
	std	± 0.0262	± 0.0240	± 0.6294	± 0.5663	± 0.4705	± 0.2693	± 0.8449
D5	accuracy	0.7361	0.7263	0.7001	0.7003	0.7	0.7	0.7065
	std	± 0.0117	± 0.0094	± 0.0004	± 0.0025	± 0.0000	± 0.0000	± 0.0055
	time	4.6168	4.5845	51.8245	46.996	8.4915	24.1665	12.4300
	std	± 0.1372	± 0.1533	± 19.2827	± 25.6543	± 1.7317	± 4.9634	± 4.3739
D6	accuracy	0.9885	0.9890	0.9462	0.9448	0.9458	0.9491	0.9646
	std	± 0.0140	± 0.0121	± 0.0181	± 0.0166	± 0.0181	± 0.0180	± 0.0170
	time	0.6058	0.4497	1.812	2.4105	3.57	0.9425	1.4880
	std	± 0.0964	± 0.0067	± 1.2291	± 0.8126	± 4.6531	± 0.1290	± 0.5612
D7	accuracy	0.9426	0.9453	0.9306	0.9249	0.93	0.9329	0.9337
	std	± 0.0139	± 0.0103	± 0.0155	± 0.0255	± 0.0183	± 0.0177	± 0.0200
	time	0.697	0.6048	4.376	3.922	4.0235	1.486	2.683
	std	± 0.0849	± 0.0170	± 1.3413	± 1.5076	± 0.8049	± 0.5212	± 0.8026
D8	accuracy	0.9522	0.8829	0.791	0.7932	0.7835	0.8259	0.8556
	std	± 0.0154	± 0.0199	± 0.0251	± 0.0311	± 0.0232	± 0.0266	± 0.0221
	time	1.163	1.0960	8.6205	5.051	6.527	2.9815	5.0955
	std	± 0.1033	± 0.0142	± 2.5738	± 1.2678	± 3.4252	± 0.6179	± 0.8415
D9	accuracy	0.9590	0.9132	0.6628	0.6628	0.6628	0.6628	0.7421
	std	± 0.0143	± 0.0167	± 0.0000	± 0.0000	± 0.0000	± 0.0000	± 0.0147
	time	3.1505	3.2468	73.3835	17.5325	27.1455	18.228	13.125
	std	± 0.0469	± 0.0383	± 10.4151	± 11.0115	± 7.3326	± 1.6739	± 5.6322
D10	accuracy	0.9090	0.9032	0.8646	0.8579	0.8648	0.8321	0.8793
	std	± 0.0070	± 0.0050	± 0.0048	± 0.0143	± 0.0050	± 0.0045	± 0.0061
	time	36.0281	39.8875	361.6830	119.3535	303.527	205.3645	239.864
	std	± 0.4904	± 0.6097	± 19.8077	± 7.1526	± 8.4926	± 6.8991	± 13.8356
D11	accuracy	0.9663	0.9675	0.9096	0.9076	0.9101	0.9065	0.9406
	std	± 0.0030	± 0.0026	± 0.0075	± 0.0115	± 0.0087	± 0.0101	± 0.0064
	time	16.7547	17.5432	171.386	91.966	177.194	62.3781	134.704
	std	± 0.3175	± 0.3060	± 14.4990	± 6.5391	± 15.6756	± 4.5846	± 6.9403
D12	accuracy	0.8183	0.8021	0.7752	0.7791	0.7684	0.7587	0.8024
	std	± 0.0396	± 0.0338	± 0.0423	± 0.0427	± 0.0407	± 0.0365	± 0.0406
	time	0.5722	0.4547	2.135	1.9425	4.4125	0.7485	1.0120
	std	± 0.0891	± 0.0045	± 0.9856	± 0.6026	± 4.3765	± 0.2909	± 0.2790
D13	accuracy	0.8158	0.7923	0.7744	0.7746	0.7733	0.7617	0.7816
	std	± 0.0067	± 0.0079	± 0.0038	± 0.0040	± 0.0036	± 0.0000	± 0.0057
	time	5.2625	5.2193	62.118	25.4405	33.147	52.0675	16.7775
	std	± 0.1788	± 0.1737	± 20.8911	± 7.4684	± 3.8329	± 5.0793	± 4.2652
D14	accuracy	0.6493	0.6329	0.5513	0.553	0.5587	0.554	0.5856
	std	± 0.0268	± 0.0273	± 0.0285	± 0.0288	± 0.0272	± 0.0249	± 0.0337
	time	5.2695	5.4672	12.245	15.4545	12.955	6.291	9.4140
	std	± 0.0314	± 0.0741	± 3.1804	± 2.1441	± 2.3191	± 0.1907	± 1.0471
D15	accuracy	0.7877	0.7848	0.6737	0.6733	0.673	0.6374	0.7371
	std	± 0.0085	± 0.0115	± 0.0191	± 0.0199	± 0.0186	± 0.0150	± 0.0140
	time	4.1932	4.5257	28.208	21.383	25.9065	16.559	20.429
	std	± 0.0654	± 0.0524	± 4.6233	± 2.0378	± 1.9193	± 2.7623	± 2.6937
D16	accuracy	0.9663	0.9741	0.9595	0.9595	0.9597	0.9482	0.9692
	std	± 0.0107	± 0.0074	± 0.0094	± 0.0094	± 0.0097	± 0.0091	± 0.0079
	time	0.8688	0.8285	2.514	1.576	4.9735	1.3365	5.7715
	std	± 0.0663	± 0.0361	± 0.5658	± 0.7922	± 1.0263	± 0.3829	± 1.4157

algorithms are able to attain a much higher accuracy of over 98% for both MKBoost-D1 and MKBoost-D2 algorithms. Moreover, by examining the performance of the state-of-the-art Lp-MKL algorithm, we found that it often performs better than the four regular ℓ_1 -norm MKL algorithms, but still performs considerably worse than our MKBoost algorithms. Finally, by comparing the two deterministic MKBoost algorithms themselves, we found that their classification accuracy performances are generally comparable, in which MKBoost-D1 outperformed MKBoost-D2 in some datasets (D1, D2, D3, D5, D8, D9, D10, D12, D13, D14, D15) but failed to beat MKBoost-D2 in the other datasets.

In addition to the clear gain of classification accuracy, we also found that the MKBoost algorithms enjoy a significant advantage in higher learning efficiency over the regular MKL algorithms. In particular, by examining the time costs of learning the classifiers, we found that the two proposed MKBoost algorithms are often significantly more efficient than the five existing MKL algorithms. Typically, both MKBoost algorithms are several times (about 2~10 times depending on datasets) faster than the existing MKL algorithms. For example, for the D5(“german.number”) dataset, the time costs of both MKBoost algorithms are only about one tenth of the costs by MKL-SD and MKL-SILP, and about half, one-sixth and one-third of the costs by MKL-Level, MKL-Hessian and Lp-MKL, respectively. Finally, for the two proposed MKBoost algorithms, their time costs are comparable as they essentially share the same time complexity in theory.

The above encouraging results show that the two MKBoost algorithms are not only more accurate than the regular MKL algorithms, but also are able to learn the models considerably more efficiently.

3.4.4.2 MKBoost vs. Other Boosting Algorithms

TABLE 3.3 shows the results of comparisons between the two deterministic MKBoost algorithms and the existing boosting based algorithms. Several observations can be drawn from the results.

Table 3.3: Comparison between two deterministic MKBoost algorithms and Boosting based algorithms.

Dataset	Metric	MKBoost-D1	MKBoost-D2	Boost-Exp	Boost-Log	Boost-CG-L1	Boost-CG-L2	Boost-SVM
D1	accuracy	0.8122	0.8093	0.7855	0.7830	0.7095	0.7942	0.8008
	std	± 0.0077	± 0.0086	± 0.0052	± 0.0055	± 0.0043	± 0.0054	± 0.0085
	time	4.8513	4.9239	265.6700	276.3300	361.04	1271.32	7.7455
D2	std	± 0.0750	± 0.0592	± 15.7931	± 18.4387	± 26.2137	± 134.2567	± 0.5961
	accuracy	0.8137	0.8122	0.7594	0.7368	0.7682	0.7922	0.8031
	std	± 0.0051	± 0.0048	± 0.0049	± 0.0048	± 0.0050	± 0.0054	± 0.0100
D3	time	21.5289	21.9411	2487.6000	2152.6600	815.91	6734.44	32.2918
	std	± 0.4417	± 0.5168	± 56.7813	± 63.4569	± 61.2565	± 531.2546	± 3.3735
	accuracy	0.8110	0.8103	0.7251	0.769	0.7931	0.7912	0.8065
D4	std	± 0.0052	± 0.0037	± 0.0037	± 0.0041	± 0.0047	± 0.0043	± 0.0107
	time	60.6319	62.5635	7830.15	7968.52	1801.96	32456.2655	71.8097
	std	± 2.4952	± 3.6807	± 153.1465	± 187.3217	± 126.1425	± 3013.2650	± 5.0836
D5	accuracy	0.9858	0.9907	0.941	0.941	0.8472	0.9811	0.9817
	std	± 0.0079	± 0.0057	± 0.0097	± 0.0135	± 0.0096	± 0.0157	± 0.0099
	time	0.7658	0.8683	7.38	7.85	1.73	256.5200	0.6973
D6	std	± 0.0262	± 0.0240	± 1.2564	± 1.3466	± 0.2977	± 34.6522	± 0.0253
	accuracy	0.7361	0.7263	0.672	0.648	0.692	0.7180	0.7141
	std	± 0.0117	± 0.0094	± 0.0050	± 0.0050	± 0.0051	± 0.0053	± 0.0149
D7	time	4.6168	4.5845	48.91	47.21	53.33	467.0900	5.4105
	std	± 0.1372	± 0.1533	± 5.1546	± 6.1354	± 6.1365	± 45.3166	± 0.3827
	accuracy	0.9885	0.9890	0.9657	0.9611	0.9623	0.9806	0.9729
D8	std	± 0.0140	± 0.0121	± 0.0168	± 0.0146	± 0.0140	± 0.0121	± 0.0237
	time	0.6058	0.4497	1.51	1.51	0.92	120.4900	0.1725
	std	± 0.0964	± 0.0067	± 0.3456	± 0.3689	± 0.1366	± 21.2364	± 0.0035
D9	accuracy	0.9426	0.9453	0.859	0.8419	0.8914	0.9371	0.9365
	std	± 0.0139	± 0.0103	± 0.0136	± 0.0157	± 0.0204	± 0.0200	± 0.0185
	time	0.697	0.6048	1.87	1.89	1.96	154.62	0.3699
D10	std	± 0.0849	± 0.0170	± 0.6431	± 0.7613	± 0.7865	± 30.0132	± 0.0033
	accuracy	0.9522	0.8829	0.6703	0.6835	0.5899	0.9101	0.9251
	std	± 0.0154	± 0.0199	± 0.0244	± 0.0244	± 0.0214	± 0.0135	± 0.0277
D11	time	1.163	1.096	6.63	6.8	8.95	204.67	0.7738
	std	± 0.1033	± 0.0142	± 0.9736	± 0.9513	± 2.6354	± 25.3643	± 0.0202
	accuracy	0.9590	0.9132	0.6628	0.6628	0.6705	0.9496	0.9469
D12	std	± 0.0143	± 0.0167	± 0.0113	± 0.0138	± 0.0127	± 0.0143	± 0.0220
	time	3.1505	3.2468	59.5300	61.5500	47.91	399.3	3.5175
	std	± 0.0469	± 0.0383	± 9.1466	± 12.3685	± 8.1666	± 15.3147	± 0.0484
D13	accuracy	0.9090	0.9032	0.8597	0.8592	0.7333	0.8798	0.901
	std	± 0.0070	± 0.0050	± 0.0057	± 0.0051	± 0.0052	± 0.0054	± 0.0056
	time	36.0281	39.8875	3551.8400	3607.8900	440.61	16901.86	87.1784
D14	std	± 0.4904	± 0.6097	± 156.4560	± 179.1655	± 13.5655	± 435.2456	± 4.0017
	accuracy	0.9663	0.9675	0.9255	0.9152	0.5905	0.9455	0.9597
	std	± 0.0030	± 0.0026	± 0.0097	± 0.0091	± 0.0097	± 0.0026	± 0.0065
D15	time	16.7547	17.5432	235.66	226.17	58.46	7299.5	17.7602
	std	± 0.3175	± 0.3060	± 12.3646	± 11.9764	± 4.1655	± 268.1564	± 0.2537
	accuracy	0.8183	0.8021	0.7443	0.7799	0.7961	0.7961	0.8049
D16	std	± 0.0396	± 0.0338	± 0.0313	± 0.0325	± 0.0403	± 0.0343	± 0.0363
	time	0.5722	0.4547	1.56	1.56	2.2	38.15	0.1887
	std	± 0.0891	± 0.0045	± 0.3146	± 0.3568	± 1.0012	± 7.3247	± 0.0009
D17	accuracy	0.8158	0.7923	0.7681	0.7375	0.5539	0.8148	0.7844
	std	± 0.0067	± 0.0079	± 0.0047	± 0.0043	± 0.0032	± 0.0060	± 0.0141
	time	5.2625	5.2193	103.86	110.96	159.9	574.4	5.6725
D18	std	± 0.1788	± 0.1737	± 10.3654	± 12.1467	± 16.6554	± 49.8135	± 0.1725
	accuracy	0.6493	0.6329	0.6121	0.6174	0.2819	0.5772	0.6266
	std	± 0.0268	± 0.0273	± 0.0370	± 0.0356	± 0.0179	± 0.0219	± 0.0318
D19	time	5.2695	5.4672	11.29	11.22	16.27	1209.91	5.983
	std	± 0.0314	± 0.0741	± 1.4366	± 1.2547	± 2.3665	± 156.7213	± 0.0287
	accuracy	0.7877	0.7848	0.7214	0.7275	0.6256	0.7299	0.7779
D20	std	± 0.0085	± 0.0115	± 0.0144	± 0.0144	± 0.0152	± 0.0047	± 0.0208
	time	4.1932	4.5257	20.46	20.62	27.01	905.01	4.8038
	std	± 0.0654	± 0.0524	± 2.6888	± 2.7832	± 4.3654	± 101.3590	± 0.0412
D21	accuracy	0.9663	0.9741	0.8838	0.9331	0.9542	0.9554	0.9611
	std	± 0.0107	± 0.0074	± 0.0097	± 0.0095	± 0.0077	± 0.0067	± 0.0095
	time	0.8688	0.8285	8.07	7.63	3.57	146.77	0.7467
D22	std	± 0.0663	± 0.0361	± 1.9875	± 1.7825	± 0.8765	± 21.5791	± 0.0248

First of all, in terms of classification accuracy, among all the algorithms, the proposed two MKBoost algorithms achieve the best for most cases, and Boost-SVM tends to perform better than the other existing algorithms. In particular, comparing with Boost-Exp and Boost-Log, MKBoost-D1 and MKBoost-D2 obtain significantly better results on all the datasets; further, comparing with Boost-CG-L1 and Boost-CG-L2, MKBoost-D1 and MKBoost-D2 achieve much better results on most datasets, except a few datasets (e.g., D6, D9, D13) where Boost-CG-L2 is comparable; Boost-SVM usually performs better than the other 4 boosting based algorithms, but worse than MKBoost-D1 and MKBoost-D2 for most datasets.

Second, in terms of time efficiency, the proposed MKBoost algorithms are clearly more efficient than the former four boosting based algorithms, and Boost-CG-L2 is the most inefficient algorithm among those four compared algorithms. Specifically, the time costs by Boost-Exp and Boost-Log are often a number of times higher than those by the proposed MKBoost algorithms. For example, on several datasets (D1, D2, D3 and D10), the time costs by Boost-Exp and Boost-Log are almost 100 times of MKBoost-D1 and MKBoost-D2. By examining the other two boosting algorithms, Boost-CG-L1 is in general comparable to Boost-Exp and Boost-Log, while Boost-CG-L2 is the most computationally intensive one. For example, for some large dataset such as D3, the proposed MKBoost-D1 and MKBoost-D2 algorithms took about 1 minute, in contrast, Boost-CG-L2 took almost 9 hours. When compared with Boost-SVM, MKBoost tends to be more efficient on larger datasets (which take more running time), but less efficient on those smaller scale datasets (D4, D6, D7, D8, D12 and D16). This is because SVM is solved by LIBSVM, whose empirical time complexity is about $O(n^{1.4}) \sim O(n^{2.3})$, cross validation in Boost-SVM is time-consuming on larger datasets. Thus, compared with Boost-SVM, MKBoost is more scalable for large dataset.

In a nutshell, the above observations clearly indicate the proposed MKBoost algorithms are empirically more effective, efficient and scalable than the existing boosting based algorithms.

3.4.4.3 MKBoost vs. Two Recently Published Algorithms

TABLE 3.4 shows the results of comparisons between the two deterministic MKBoost algorithms and the two recently published algorithms.

At first, by comparing the results of SMMKL with regular MKL algorithms shown in Table 3.2, it is obvious that SMMKL is not only more effective but also more efficient. Its effectiveness is mainly because of the novel soft margin prospective. The training of SMMKL is very efficient because it adopts analytical updating rule.

Next, several observations can be drawn from the results in TABLE 3.4:

In terms of accuracy, it is hard to say which is the best among MKBoost, EKP and SMMKL. The models of MKBoost and EKP are both built up of a set of base classifiers, they are different only in the approach to combining them, so it is reasonable that they perform generally comparable. It is a little surprising to find out that SMMKL can achieve comparable performance with MKBoost and EKP, this again demonstrates the effect of the novel soft margin prospective.

In terms of efficiency, EKP is the most efficient on most datasets. While SMMKL is generally slower than its two opponents, except two datasets (D12 and D14) on which it spends less time than MKBoost and EKP due to fast convergence. Though in MKBoost the base learners are trained with a sampled set of training instances which are less than that of EKP, but MKBoost adopts boosting to combine these base learners while EKP solves an optimization problem only once, so the difference in efficiency between them is as expected.

The above observations show that although SMMKL can improve upon regular MKL algorithms significantly, ensemble methods such as MKBoost and EKP are still very promising.

3.4.4.4 Deterministic vs. Stochastic MKBoost

TABLE 3.5 shows the results of comparisons between the deterministic MKBoost algorithms (MKBoost-D1 and MKBoost-D2) and the stochastic MKBoost algorithms (MKBoost-S1 and MKBoost-S2). Several observations can be drawn from the results.

Table 3.4: Comparison between two deterministic MKBoost algorithms and two recently published algorithms.

Dataset	Metric	MKBoost-D1	MKBoost-D2	EKP	SMMKL
D1	accuracy	0.8122	0.8093	0.8180	0.8240
	std	± 0.0077	± 0.0086	± 0.0072	$\pm \mathbf{0.0074}$
	time	4.8513	4.9239	0.6455	7.4885
	std	± 0.0750	± 0.0592	$\pm \mathbf{0.1464}$	± 4.7943
D2	accuracy	0.8137	0.8122	0.8241	0.8290
	std	± 0.0051	± 0.0048	± 0.0068	$\pm \mathbf{0.0056}$
	time	21.5289	21.9411	2.6780	33.6150
	std	± 0.4417	± 0.5168	$\pm \mathbf{0.4721}$	± 19.4102
D3	accuracy	0.811	0.8103	0.8264	0.8319
	std	± 0.0052	± 0.0037	± 0.0037	$\pm \mathbf{0.0055}$
	time	60.6319	62.5635	4.9940	89.789
	std	± 2.4952	± 3.6807	$\pm \mathbf{0.2704}$	± 39.1319
D4	accuracy	0.9858	0.9907	0.9826	0.9854
	std	± 0.0079	$\pm \mathbf{0.0057}$	± 0.0071	± 0.0065
	time	0.7658	0.8683	0.4430	1.323
	std	± 0.0262	± 0.0240	$\pm \mathbf{0.2444}$	± 0.2515
D5	accuracy	0.7361	0.7263	0.742	0.7626
	std	± 0.0117	± 0.0094	± 0.0119	$\pm \mathbf{0.0140}$
	time	4.6168	4.5845	0.2750	7.763
	std	± 0.1372	± 0.1533	$\pm \mathbf{0.0705}$	± 4.3902
D6	accuracy	0.9885	0.9890	0.9792	0.9693
	std	$\pm \mathbf{0.0140}$	$\pm \mathbf{0.0121}$	± 0.0169	± 0.0159
	time	0.6058	0.4497	0.0485	0.1585
	std	± 0.0964	± 0.0067	$\pm \mathbf{0.0318}$	± 0.1536
D7	accuracy	0.9426	0.9453	0.9411	0.9517
	std	$\pm \mathbf{0.0139}$	$\pm \mathbf{0.0103}$	± 0.0120	$\pm \mathbf{0.0166}$
	time	0.697	0.6048	0.0905	0.8795
	std	± 0.0849	± 0.0170	$\pm \mathbf{0.0312}$	± 0.4825
D8	accuracy	0.9522	0.8829	0.9011	0.8993
	std	$\pm \mathbf{0.0154}$	± 0.0199	± 0.0203	± 0.0197
	time	1.163	1.096	0.3505	2.291
	std	± 0.1033	± 0.0142	$\pm \mathbf{0.1380}$	± 0.3307
D9	accuracy	0.9590	0.9132	0.9045	0.9180
	std	$\pm \mathbf{0.0143}$	± 0.0167	± 0.0179	± 0.0198
	time	3.1505	3.2468	0.3450	16.4685
	std	± 0.0469	± 0.0383	$\pm \mathbf{0.0462}$	± 8.2543
D10	accuracy	0.9090	0.9032	0.8973	0.8998
	std	$\pm \mathbf{0.0070}$	± 0.0050	± 0.0039	± 0.0056
	time	36.0281	39.8875	7.2170	220.8475
	std	± 0.4904	± 0.6097	$\pm \mathbf{0.6950}$	± 78.9768
D11	accuracy	0.9663	0.9675	0.9571	0.959
	std	$\pm \mathbf{0.0030}$	$\pm \mathbf{0.0026}$	± 0.0035	± 0.0035
	time	16.7547	17.5432	3.6555	18.5125
	std	± 0.3175	± 0.3060	$\pm \mathbf{0.6620}$	± 7.3322
D12	accuracy	0.8183	0.8021	0.8015	0.8087
	std	$\pm \mathbf{0.0396}$	$\pm \mathbf{0.0338}$	$\pm \mathbf{0.0235}$	$\pm \mathbf{0.0401}$
	time	0.5722	0.4547	0.0548	0.0370
	std	± 0.0891	± 0.0045	± 0.0156	$\pm \mathbf{0.0150}$
D13	accuracy	0.8158	0.7923	0.8287	0.8242
	std	± 0.0067	± 0.0079	$\pm \mathbf{0.0100}$	$\pm \mathbf{0.0098}$
	time	5.2625	5.2193	0.3935	7.363
	std	± 0.1788	± 0.1737	$\pm \mathbf{0.1013}$	± 1.5165
D14	accuracy	0.6493	0.6329	0.6826	0.605
	std	± 0.0268	± 0.0273	$\pm \mathbf{0.0354}$	± 0.0315
	time	5.2695	5.4672	0.3645	0.3195
	std	± 0.0314	± 0.0741	$\pm \mathbf{0.0430}$	$\pm \mathbf{0.1980}$
D15	accuracy	0.7877	0.7848	0.7960	0.7551
	std	± 0.0085	± 0.0115	$\pm \mathbf{0.0141}$	± 0.0148
	time	4.1932	4.5257	0.8310	8.249
	std	± 0.0654	± 0.0524	$\pm \mathbf{0.1831}$	± 1.6121
D16	accuracy	0.9663	0.9741	0.9783	0.9750
	std	± 0.0107	± 0.0074	$\pm \mathbf{0.0055}$	$\pm \mathbf{0.0060}$
	time	0.8688	0.8285	0.3445	1.927
	std	± 0.0663	± 0.0361	$\pm \mathbf{0.1434}$	± 1.8508

Table 3.5: Comparison between the deterministic and stochastic MKBoost algorithms.

Dataset	Metric	MKBoost-D1	MKBoost-D2	MKBoost-S1	MKBoost-S2
D1	accuracy	0.8122	0.8093	0.8126	0.8134
	std	± 0.0077	± 0.0086	± 0.0072	± 0.0069
	time	4.8513	4.9239	2.2384	2.2776
	std	± 0.0750	± 0.0592	± 0.0396	± 0.0384
D2	accuracy	0.8137	0.8122	0.8132	0.8164
	std	± 0.0051	± 0.0048	± 0.0052	± 0.0041
	time	21.5289	21.9411	9.4543	9.4108
	std	± 0.4417	± 0.5168	± 0.3263	± 0.3138
D3	accuracy	0.811	0.8103	0.8118	0.8156
	std	± 0.0052	± 0.0037	± 0.0055	± 0.0046
	time	60.6319	62.5635	26.0512	25.8940
	std	± 2.4952	± 3.6807	± 1.4618	± 1.4806
D4	accuracy	0.9858	0.9907	0.9878	0.9886
	std	± 0.0079	± 0.0057	± 0.0067	± 0.0061
	time	0.7658	0.8683	0.435	0.3975
	std	± 0.0262	± 0.0240	± 0.0119	± 0.0192
D5	accuracy	0.7361	0.7263	0.7355	0.7359
	std	± 0.0117	± 0.0094	± 0.0106	± 0.0079
	time	4.6168	4.5845	2.0623	2.0320
	std	± 0.1372	± 0.1533	± 0.0746	± 0.0563
D6	accuracy	0.9885	0.9890	0.989	0.9879
	std	± 0.0140	± 0.0121	± 0.0138	± 0.0137
	time	0.6058	0.4497	0.2732	0.2727
	std	± 0.0964	± 0.0067	± 0.0088	± 0.0078
D7	accuracy	0.9426	0.9453	0.9469	0.943
	std	± 0.0139	± 0.0103	± 0.0130	± 0.0152
	time	0.697	0.6048	0.3547	0.3527
	std	± 0.0849	± 0.0170	± 0.0085	± 0.0106
D8	accuracy	0.9522	0.8829	0.943	0.8998
	std	± 0.0154	± 0.0199	± 0.0153	± 0.0221
	time	1.163	1.096	0.5347	0.5423
	std	± 0.1033	± 0.0142	± 0.0122	± 0.0132
D9	accuracy	0.9590	0.9132	0.9519	0.9308
	std	± 0.0143	± 0.0167	± 0.0156	± 0.0189
	time	3.1505	3.2468	1.6099	1.6198
	std	± 0.0469	± 0.0383	± 0.0233	± 0.0183
D10	accuracy	0.9090	0.9032	0.9085	0.9070
	std	± 0.0070	± 0.0050	± 0.0070	± 0.0056
	time	36.0281	39.8875	20.6196	20.4699
	std	± 0.4904	± 0.6097	± 0.3701	± 0.3793
D11	accuracy	0.9663	0.9675	0.9665	0.9677
	std	± 0.0030	± 0.0026	± 0.0034	± 0.0028
	time	16.7547	17.5432	9.366	9.2138
	std	± 0.3175	± 0.3060	± 0.1516	± 0.1477
D12	accuracy	0.8183	0.8021	0.8186	0.8076
	std	± 0.0396	± 0.0338	± 0.0356	± 0.0338
	time	0.5722	0.4547	0.2388	0.2370
	std	± 0.0891	± 0.0045	± 0.0051	± 0.0039
D13	accuracy	0.8158	0.7923	0.8132	0.8065
	std	± 0.0067	± 0.0079	± 0.0066	± 0.0080
	time	5.2625	5.2193	2.4548	2.4192
	std	± 0.1788	± 0.1737	± 0.0892	± 0.0663
D14	accuracy	0.6493	0.6329	0.6416	0.6335
	std	± 0.0268	± 0.0273	± 0.0270	± 0.0274
	time	5.2695	5.4672	2.8047	2.8174
	std	± 0.0314	± 0.0741	± 0.0382	± 0.0257
D15	accuracy	0.7877	0.7848	0.786	0.7853
	std	± 0.0085	± 0.0115	± 0.0114	± 0.0110
	time	4.1932	4.5257	2.6175	2.6172
	std	± 0.0654	± 0.0524	± 0.0357	± 0.0333
D16	accuracy	0.9663	0.9741	0.9673	0.9701
	std	± 0.0107	± 0.0074	± 0.0102	± 0.0087
	time	0.8688	0.8285	0.554	0.5508
	std	± 0.0663	± 0.0361	± 0.0251	± 0.0242

First of all, it is clear to see that the stochastic MKBoost algorithms are more efficient than the deterministic MKBoost algorithms. The exact value of time efficiency speedup achieved by the stochastic algorithms depends on different datasets. Overall, the time cost of the two stochastic MKBoost algorithms is about half of the cost taken by the two deterministic algorithms as observed from this set of experiments.

In addition to the time efficiency issue, another concern is how accurate can the stochastic MKBoost algorithms perform in comparison to the deterministic algorithms. By further examining the classification accuracy results in details, it is a bit surprising to find that the two stochastic algorithms perform very well, which are in general fairly comparable to the deterministic algorithms. The difference of the accuracy values between a deterministic algorithm and a stochastic algorithm is often no more than 1%. Besides, for a few cases, we even observed that a stochastic algorithm slightly outperformed a deterministic algorithm. Finally, by comparing the two different stochastic algorithms, we found that both of their accuracy and efficiency performances are quite comparable, which is similar to the previous comparison of the two deterministic algorithms.

The above observations show that the proposed stochastic MKBoost algorithms can effectively speed up the deterministic algorithms by maintaining comparable classification accuracy.

3.4.5 Parameter Evaluation

We notice that there are a few parameters that could affect the performance (both accuracy and efficiency) of the proposed MKBoost algorithms. These parameters include the total number of boosting trials T , the sampling ratio of training data at each boosting trial, and the sampling decay factor β in the stochastic MKBoost algorithms. To examine the influence of these parameters, we conduct a set of experiments to evaluate their impacts on both accuracy and efficiency performance in the classification tasks.

3.4.5.1 Evaluation of Boosting Parameter T

The first set of experiments is to examine the influence of the total number of boosting trials T for the MKBoost algorithms. In our previous experiments, we simply fix T to 100. In this set of experiments, we examine the experimental results by varying the parameter T from 20 to 200. Fig. 3.1 and Fig. 3.2 show the evaluation results for the impact of the boosting parameter T on the classification accuracy and learning time cost, respectively. Some observations can be drawn from the evaluation results.

First of all, we observed that, in terms of classification accuracy performance, increasing the total number of boosting trials T in general is able to boost the accuracy performance of all the proposed MKBoost algorithms consistently. Such observation is particularly more evident when the total number of boosting trials T is small (e.g. $T < 50$) at the beginning. The improvements of classification accuracy performance usually become very small when the parameter T is large (e.g. $T > 200$).

On the other hand, we found that increasing the T value leads to a linear increase of the time cost for training the models by all the proposed MKBoost algorithms. This is not surprising since all the MKBoost algorithms have a linear time complexity with respect to the total number of boosting trials T .

The above empirical observations indicate that choosing an appropriate boosting parameter T is essentially a tradeoff between classification accuracy and efficiency performances. In practice, it is not difficult to choose an appropriate T value that usually falls in between 50 and 200, which sometimes also depends on the empirical requirements of efficiency and accuracy in a real-life application.

3.4.5.2 Evaluation of Boosting Sampling Ratio

Another boosting parameter that may affect the MKBoost algorithms is the boosting sampling ratio, which determines the proportion of training data examples sampled from the whole collection of training data at each boosting trial. Fig. 3.3 and Fig. 3.4 respectively show the

Figure 3.1: Evaluation of MKBoost classification accuracy with respect to boosting parameter T .

Figure 3.2: Evaluation of MKBoost learning time cost with respect to boosting parameter T .

evaluations of accuracy and efficiency performance with respect to the boosting sampling ratio by varying its value from 0.05 to 0.5.

From the experimental results, we found that the MKBoost algorithms with a large boosting sampling ratio value usually produced better classification accuracy performance. This is especially more evident when the boosting sampling ratio is small. For example, on the dataset “Monks1”, the MKBoost-S1 algorithm has the accuracy of less than 93%, which was boosted to above 95% when the sampling ratio is increased to 0.3. Despite the improvement when increasing the boosting sampling ratio, we found that the classification accuracy tends to saturate when the value is large enough (e.g. larger than 0.4). All these observations are not difficult to interpret because when the sampling ratio is too small, the base kernel classifiers trained at the boosting process may suffer from insufficient training examples. On the other hand, employing

Figure 3.3: Evaluation of MKBoost classification accuracy with respect to boosting sampling ratio.

Figure 3.4: Evaluation of MKBoost learning time cost with respect to boosting sampling ratio.

a too large sampling ratio may lead to sample too many training data examples for some large dataset, which may be somewhat redundant for building the base classifiers at the boosting trials.

In addition to the impact on the accuracy, the boosting sampling ratio parameter also affects the efficiency of the MKBoost algorithms. Similar to the observation from the evaluation of parameter T , increasing the boosting sampling ratio also leads to the increase of the time cost needed by the MKBoost algorithms. In particular, the relationship between the sampling ratio and the learning time cost of the MKBoost algorithms fully depends on the underlying algorithms used in training the kernel classifiers. As we adopt SVM for training the base kernel classifiers, the time cost of an MKBoost algorithm thus depends on the time cost of the SVM algorithm (e.g., the empirical time complexity of the SMO algorithm implemented in the

LIBSVM package [cCL01] is about $O(n^{1.4})$ to $O(n^{2.3})$.

3.4.5.3 Evaluation of the Sampling Decay Factor β for Stochastic MKBoost Algorithms

The last set of experiments is to examine the effect of the sampling decay factor β for the two stochastic MKBoost algorithms (MKBoost-S1 and MKBoost-S2).

Fig. 3.5 and Fig. 3.6 show the evaluation of sampling decay factor in terms of both accuracy and time cost, respectively. From the results, we found that when setting β to 1 in the extreme case, the two stochastic algorithms converge to the two deterministic algorithms, respectively. This is not surprising as $\beta = 1$ indicates that there is no decay at all; as a result the stochastic algorithm reduces to the deterministic one. Further, we observed that when decreasing the value of β , the time cost can be consistently decreased since only some kernels will be sampled for training due to the decay effect.

Figure 3.5: Evaluation of accuracy of stochastic MKBoost algorithms with respect to the sampling decay factor β .

However, there is not a consistent trend for the impact on the accuracy by varying the β value. But we empirically found that the performance of MKBoost-D1 is in general better than MKBoost-D2 for those datasets where there is a single best kernel classifier that significantly dominates among all the kernels (e.g., Monks1, Sonar); for such datasets, MKBoost-S1 usually favors larger β , while MKBoost-S2 prefers smaller β . However, for those datasets where no such a single best kernel classifier exists, the situation is reversed. In a real-world application, it

Figure 3.6: Evaluation of time cost of stochastic MKBoost algorithms with respect to the sampling decay factor β .

is an open question to determine which algorithm is more preferred. To address this challenge, we could try to check if there exists a single best kernel classifier for the application; or if such prior knowledge is not available, we could adopt cross validation to address the algorithm selection issue.

3.5 Summary

This chapter presented a framework of Multiple Kernel Boosting (MKBoost) for classification, which applies boosting techniques to learn classification models with multiple kernels. As a trade-off between accuracy and efficiency, we first proposed two deterministic MKBoost algorithms using all kernels at each boosting trial; we then presented two stochastic MKBoost algorithms that randomly sample a subset of kernels at each boosting trial. We found MKBoost empirically achieved better accuracy than the regular MKL methods, and performed much more efficiently than the state-of-the-art MKL methods. Besides, we also found the stochastic MKBoost algorithms considerably improved the efficiency of the deterministic algorithms by maintaining comparable accuracy.

As you can see, the proposed MKBoost framework can address the first challenge which is improving the efficiency and effectiveness of regular MKL algorithms for classification to some extent.

Chapter 4

Boosting Multi-Kernel Locality-Sensitive Hashing

This chapter addresses the second challenge posed in section 1.2, which is improving the state-of-the-art hashing algorithm (KLSH) by learning a combination of multiple kernels. The rest of this chapter is organized as follows. Section 4.1 reviews related work. Section 4.2 gives some preliminaries of KLSH, and section 4.3 presents the proposed BMKLSH method. Section 4.4 discusses our experiments. Finally Section 4.5 summarizes this chapter.

4.1 Related Work

This section reviews related work in approximate nearest neighbor (NN) search with the focus on image retrieval applications.

In literature, developing efficient techniques for indexing high dimensional images has been studied extensively in information retrieval, multimedia, and database communities [BBK01, CZPC02, DIO⁺09]. Spatial data structure approaches (e.g., kd-tree [Ben75, Rob81] or metric tree [Uhl91]) were used to handle the NN search problem; however, they scale poorly with data dimensionality. In practice, if the number of dimensions is large enough, kd-tree and other similar data structures require an expensive inspection in the data set, thereby perform no better than an exhaustive linear search that simply compares a query to every data point in the database.

Instead of solving the exact similarity search for high dimensional indexing, recent years have witnessed active studies of approximate high-dimensional indexing techniques [IM98, GIM99, LJW⁺07, BKK96, COST05, DCL08]. The key of most techniques is to exploit random projection to tackle the curse of dimensionality issue, such as Locality-Sensitive Hashing (LSH) [IM98], a very well-known and highly successful technique in this area. In general, we can group most of existing approaches into two major categories: linear projection methods and kernel-based methods. Below we briefly review the first category and then focus on discussing the second category.

For the first category, one of the most notable techniques is LSH [IM98], which utilizes a family of locality sensitive hashing functions which map similar items to same bucket with high probability, and dissimilar items to same bucket with low probability. With such a hash function, one can easily search within the bucket a query point belongs to in order to find nearest neighbors of the query. LSH efficiently solves the approximate similarity search problem and achieves query time in the worst case $O(dn^{1/\epsilon})$. Gionis et al. [GIM99] improved the techniques and achieved significant query time $O(dn^{1/(1+\epsilon)})$. Recently, many studies have attempted to improve upon the regular LSH technique. For example, [LJW⁺07] introduced multi-probe LSH methods that reduce the space requirement of the basic LSH method. Tao et al. [TYSK09] proposed the locality-sensitive B-tree technique that mainly concerns the performance improvement of I/O disk access. Besides, there are many other existing works that accommodate LSH functions in tackling different issues, including Hamming distance [IM98], inner products [Cha02], ℓ_p -norms [DIIM04], normalized partial matching [GD07], learned Mahalanobis metrics [PJG08], and so on. Last but not least, there are also many recent studies of learning compact binary codes inspired by the idea of LSH [ZWCL10, HCRB11, WKC10, JB11].

Our study is more related to the second category of kernel-based methods. In particular, kernel-based LSH (KLSH) [KG09] was recently proposed to overcome the limitation of the regular LSH technique that often assumes the data come from a multidimensional vector

space and the underlying embedding of the data must be explicitly known and computable. KLSH provides a powerful framework to explore arbitrary kernel/similarity functions where their underlying embedding only needs to be known implicitly. In [RL09], the authors proposed a variant of KLSH which aims to force the expected Hamming distance between the binary codes of two vectors is related to the value of a shift-invariant kernel. In [HLC10], the authors proposed an improved algorithm for learning binary codes with kernel to speed up the KLSH technique. Despite being studied actively, most existing kernel-based hashing methods [KG09, RL09, HLC10, MSY10] only consider a single kernel, which cannot fully explore the potential of multiple kernel functions in a real CBIR application.

Very recently, some emerging study has attempted to apply KLSH by exploring multiple kernels. In [WJHT10b], they proposed a simple solution named MKLSH by applying the existing KLSH algorithm for each specific kernel individually, and then combining the outputs of these KLSH processes. Our work however differs from their method in several aspects. First of all, their naive approach to combining multiple kernels simply treats each kernel equally, which fails to fully explore the power of combining multiple diverse kernels in KLSH. Second, their technique is essentially unsupervised, which does not fully explore the data characteristics and thus cannot achieve the optimal indexing performance. In contrast, our technique is in general supervised and is able to learn the optimal combination of multiple kernels from training data to maximize the indexing performance. But it is worth to note that supervised method is not always better than unsupervised method in practical applications, especially for some applications which lack enough training data and require high generalization ability and efficiency.

4.2 Preliminaries of Kernelized LSH

In this section, we review the algorithm of Kernelized Locality-Sensitive Hashing (KLSH).

Symbol	Meaning
d	number of dimensions of input data
n	number of image examples in database
$\mathbf{X}$	$d \times n$ matrix of n image examples
$\mathbf{q}$	query image example
$\mathbf{x}_i$	i -th image example in database
$\mathbf{K}$	kernel matrix
n_q	number of training query examples
m	number of different kernel functions
κ_l	l -th kernel function
b	length of a hash key
p	number of data points sampled in KLSH
t	number of indices selected in KLSH
D_t	distribution at the t -th round
T	number of boosting rounds

Table 4.1: Summary of notation used in this chapter

Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the collection of data points to be searched. Given a query $\mathbf{q} \in \mathbb{R}^d$, to efficiently find the k nearest neighbors, LSH projects each data point into a low-dimensional binary space, referred to as the *hash* key. The hash keys are constructed by applying b binary-valued hash functions $h_1, \dots, h_b$ to the data points in $\mathbf{X}$. KLSH generalizes LSH by introducing a kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ to map a data point $\mathbf{x}_i$ to a functional space through a nonlinear feature mapping $\phi(\mathbf{x}_i)$ that satisfies the condition $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi^\top(\mathbf{x}_i)\phi(\mathbf{x}_j)$. To build the hash function, KLSH first randomly selects a subset of p data points from $\mathbf{X}$, denoted as $S = \{\mathbf{x}_1^s, \dots, \mathbf{x}_p^s\}$, and forms a kernel matrix $\mathbf{K}$ over the sampled data points; it then generates b random vectors $\mathbf{e}_S^1, \dots, \mathbf{e}_S^b$ and computes a hashing function for each random vector $\mathbf{e}_S^k$ as

$$h_k(\phi(\mathbf{x})) = \text{sign} \left(\sum_{j=1}^p w_j^k \kappa(\mathbf{x}, \mathbf{x}_j^s) \right),$$

where $\mathbf{w}^k = (w_1^k, \dots, w_p^k)^\top$ is given by $\mathbf{w}^k = \mathbf{K}^{-1/2} \mathbf{e}_S^k$, $\mathbf{e}_S^k$ is generated by selecting t indices at random from $[1, \dots, p]$ and setting them to 1 while others are set to 0. Algorithm 6 outlines the key steps of KLSH, where b is a critical parameter that determines the length of hash key to be constructed in KLSH.

Algorithm 6 KLSH

INPUT:

- an image database: $\{\mathbf{x}_i | i = 1, \dots, n\}, \mathbf{x}_i \in \mathbb{R}^d$
- length of the hash key: b
- a kernel function: $\kappa(\cdot, \cdot)$

```

1: randomly select  $p$  data points  $S = \{\mathbf{x}_j^s | j = 1, \dots, p\}$ 
2:  $\mathbf{K} = [\kappa(\mathbf{x}_i^s, \mathbf{x}_j^s)]_{p \times p}$ 
3: for  $k = 1, \dots, b$  do
4:   form  $\mathbf{e}_S^k$ : select  $t$  indices at random from  $[1, \dots, p]$ 
5:   form  $\mathbf{w}^k = \mathbf{K}^{-1/2} \mathbf{e}_S^k$ 
6:   construct hash function:
        $h_k(\phi(\mathbf{x})) = \text{sign}(\sum_{j=1}^p w_j^k \kappa(\mathbf{x}, \mathbf{x}_j^s))$ 
7: end for
    
```

 OUTPUT: a set of b hash functions $\mathcal{H} = \{h_k | k = 1, \dots, b\}$

4.3 Learning to Combine Multiple Kernels for KLSH

4.3.1 Overview

In this section, we propose a framework of learning the combination of multiple kernels for Kernelized Locality-Sensitive Hashing, which aims to boost KLSH by making use of a combination of multiple kernels. One key question is how to determine the weights for kernel combination. A straightforward approach is to assign equal weight to each kernel function, and apply KLSH with the uniformly combined kernel function. Such an approach might not fully explore the power of multiple kernels.

To address this limitation, we propose a scheme to assign each kernel a different number of bits so as to reflect the importance of the kernel. The key challenge of this scheme is thus how to optimize the bit size allocation with respect to different kernel functions. Figure 4.1 illustrates the proposed framework, which consists of two key steps: (i) bit allocation optimization, and (ii) the multi-kernel hashing as shown in Algorithm 7.

During the first step which is the bit allocation optimization, after the weights for kernel combination are determined, the bits allocation vector $[b_1, \dots, b_m]$ can be obtained accordingly. Specifically, b_l (the bit size for kernel $\kappa_l, l = 1, \dots, m$) is allocated proportional to its

Figure 4.1: The proposed framework of learning to combine multiple kernels for improving kernel LSH

Algorithm 7 The Multi-Kernel Hashing scheme

INPUT:

- an image database: $\{\mathbf{x}_i | i = 1, \dots, n\}, \mathbf{x}_i \in \mathbb{R}^d$
- a set of m different kernels: $\{\kappa_l | l = 1, \dots, m\}$
- bits allocation vector $[b_1, \dots, b_m]$

```

1:  $k = 0$ 
2: randomly select  $p$  data points  $S = \{\mathbf{x}_j^s | j = 1, \dots, p\}$ 
3: for  $l = 1, \dots, m$  do
4:    $\mathbf{K}_l = [\kappa_l(\mathbf{x}_i^s, \mathbf{x}_j^s)]_{p \times p}$ 
5:   for  $r = 1, \dots, b_l$  do
6:      $k = k + 1$ 
7:     form  $\mathbf{e}_S^k$ : select  $t$  indices at random from  $[1, \dots, p]$ 
8:     form  $\mathbf{w}^k = \mathbf{K}_l^{-1/2} \mathbf{e}_S^k$ 
9:     construct hash function:
        $h_k(\phi(\mathbf{x})) = \text{sign}(\sum_{j=1}^p w_j^k \kappa_l(\mathbf{x}, \mathbf{x}_j^s))$ 
10:   end for
11: end for
    
```

OUTPUT: a set of b hash functions $\mathcal{H} = \{h_k | k = 1, \dots, b\}$

importance weight. In practice, proportional allocation can be implemented by rounding the cumulative bit sizes for a set of kernels $(\{\kappa_1, \dots, \kappa_l\}, l = 1, \dots, m)$ based on the cumulative weights of them.

Unlike the regular KLSH that adopts a single kernel, BMKLSH employs a set of m kernels for the hashing scheme. Besides, a key difference between BMKLSH and some existing Multi-Kernel LSH (MKLSH) [WJHT10b] is the bit allocation optimization step to find the parameter $[b_1, \dots, b_m]$ that determines the allocations of bit sizes for a set of m kernels. Unlike the existing MKLSH approach [WJHT10b] that simply assigns the same number of bits to each kernel, leading to a uniform combination of multiple kernels, we develop two supervised learning al-

gorithms (WMKLSH and BMKLSH), described below, that effectively learn the importance of individual kernels, and consequentially determine the appropriate number of bits for each kernel.

4.3.2 WMKLSH by Weighted Bit Allocation

We first propose a Weighted Multi-Kernel Locality-Sensitive Hashing (WMKLSH) scheme by a supervised learning approach to determine the allocation of bit size, where a kernel is assigned a larger size of bits if it better captures the similarity between data points.

In order to learn the importance weights of different kernels for the retrieval tasks, we assume a small training data set is available for our learning task. The training set consists of a small set of queries and their relevance judgements, which usually can be easily collected in a real-life CBIR system via the relevance feedback mechanism [RHOM98, HLJ06].

For the WMKLSH algorithm, we begin by testing the retrieval performance of KLSH with a set of m kernels on the given training set. After that, we can obtain the retrieval performance in terms of mean Average Precision (mAP). As a result, we can compute the weights based on the retrieval performance of each kernel, i.e., $\alpha_l = e^{mAP_l}$, $l = 1, \dots, m$. Finally, the bit sizes of the kernels are allocated proportionally according to their importance weights.

4.3.3 BMKLSH for Optimizing Bit Allocation

To further improve the above learning scheme, we propose a boosting scheme, referred to as BMKLSH, to learn the optimal allocation of bit sizes, which adopts the similar idea of boosting algorithms for classification [FS97]. Let us denote by n_q the number of queries, $AP_l(i)$, $l \in [m]$, $i \in [n_q]$, be the Average Precision performance of applying the l -th kernel $\kappa_l(\cdot, \cdot)$ to retrieve vectors for the i -th query, and $AP(i)$, $i \in [n_q]$, be the Average Precision performance of applying the multi-kernel hashing scheme for the i -th query. In order to find

the optimal allocation of bit sizes, we cast it into the following optimization problem:

$$\max_{b_1, \dots, b_m \in [b]} \sum_{i=1}^{n_q} AP(i) \text{ subject to } \sum_l b_l = b \quad (\text{Eq. 4.1})$$

Since each b_l is an integer variable, the above problem is essentially an integer programming task, which is NP-hard. To find an efficient solution, we approximate the above optimization problem by introducing a set of combination weights $\alpha_l, l \in [m]$, each of which represents the importance of each kernel so as to determine each bit size b_l .

To efficiently solve the problem, we adopt a boosting based strategy. Following the similar procedure of Adaboost algorithm [FS97], we introduce a distribution of weights D_t to indicate the retrieval difficulty of the instances in the training data set. At each boosting round, we measure the retrieval performance (e.g. average precision) of each kernel based on the existing KLSH algorithm (shown in Algorithm 6), and select the best kernel with the largest weighted Average Precision (wAP) performance over the current distribution D_t , which is defined as:

$$wAP_l = \sum_{i=1}^{n_q} D_t(i) AP_l(i) \quad (\text{Eq. 4.2})$$

At the end of each boosting round, an importance weight is computed for the selected kernel based on its retrieval performance, and the weights of each poorly retrieved query example will be increased such that the next selected kernel will focus more on those hardly retrieved examples. The boosting procedure will be repeated T times. Finally, we allocate a bit size to each kernel based on its cumulative weight in all the T boosting rounds. The details of the proposed algorithm are given in Algorithm 8.

4.3.4 Time Complexity Analysis

First of all, assume the length of hash key b is fixed, both WMKLSH and BMKLSH algorithms have the same querying time cost as that of KLSH and MKLSH. Next we focus on discussing training and indexing time cost.

Algorithm 8 BMKLSH: Boosting Multi-Kernel LSH algorithm for optimizing the bit size allocation

INPUT:

- an image database: $\{\mathbf{x}_i | i = 1, \dots, n\}$, $\mathbf{x}_i \in \mathbb{R}^d$
- length of the hash key: b
- a set of m different kernels, $\{\kappa_l | l = 1, \dots, m\}$
- a set of n_q training instances $\{\mathbf{x}_i^t | i = 1, \dots, n_q\}$ with additional feedback info (positive and negative lists)
- number of boosting rounds: T
- initial distribution: $D_1(i) = 1/n_q, i = 1, \dots, n_q$

```

1: for  $t = 1, \dots, T$  do
2:   for  $l = 1, \dots, m$  do
3:     obtain  $AP_l(i), l = 1, \dots, m, i = 1, \dots, n_q$  by testing on the training set based on
       Algorithm 6
4:      $wAP_l = \sum_{i=1}^{n_q} D_t(i) AP_l(i)$ 
5:      $\alpha_t^l = e^{wAP_l}$ 
6:   end for
7:    $\alpha_t^l \leftarrow \frac{\alpha_t^l}{\sum_{i=1}^m \alpha_t^i}, l = 1, \dots, m$ 
8:    $l^* = \arg \max_{l \in \{1, 2, \dots, m\}} \alpha_t^l$ 
9:    $\alpha_t \leftarrow \alpha_t^{l^*}, AP_t \leftarrow AP_{l^*}, wAP_t \leftarrow wAP_{l^*}$ 
10:  update the distribution of the training instances:
    
```

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } AP_t(i) \geq wAP_t \\ e^{\alpha_t} & \text{if } AP_t(i) < wAP_t \end{cases}$$

Z_t is a normalization factor to make D_{t+1} a distribution

```

11: end for
12:  $\alpha_l = \sum_{t=1}^T \alpha_t^l, l = 1, \dots, m$ 
13: allocate bits to  $m$  kernels  $[b_1, \dots, b_m]$  based on the weights  $\alpha_l$ 
14: Multi-Kernel Hashing( $[b_1, \dots, b_m]$ )
    
```

As supervised methods, WMKLSH and BMKLSH algorithms need to test the performance for all training instances with all kinds of kernels once. So as to test the performance of each kernel, we have to employ Algorithm 6 to build a set of hash functions for each kernel κ_l , $l = 1, \dots, m$. It is obvious that computing $\mathbf{K}^{-1/2}$ is the most time consuming process in Algorithm 6, the time complexity of which is lower bounded by $O(n^2 \log(n))$. For a fixed l and i , the time complexity for computing $AP_l(i)$ is $O(n)$, where n is the number of image

examples in database. So for each kernel κ_l the time complexity for testing the performance of all training instances is $O(nn_q)$, where n_q is the number of training instances, which is usually much smaller than n . For BMKLSH, at each boosting round, it only updates the distribution of training instances and computes the weight for each round based on the distribution. The updating step is in general linear with n_q . Thus, the main time cost consumed mostly falls into the process of validating the performance of training instances.

4.4 Experiments

We conduct extensive experiments to examine the efficacy of the proposed algorithms for scalable content-based image retrieval.

4.4.1 Experimental Testbed

We perform experiments on two well-known public image databases which have been widely used for benchmark image retrieval tasks. Besides, we downloaded 1,000,000 social images from Flickr to form a background class and combine it with the second database to form a large scale data set. We briefly introduce some details of the two data sets below.

The first data set is the INRIA Holidays data set¹, which has been widely used for benchmark evaluation of image retrieval performance [JDS08]. It contains 500 image groups, each of which represents a distinct scene or object. The first image of each group is the query image and the correct retrieval results are the other images of the group. There are 1491 images in total, including 500 queries and 991 corresponding relevant images.

The second data set is a large-scale data set, which consists of both the ImageCLEF database² and the collection of 1,000,000 images crawled from Flickr (named as “Flickr1M”). ImageCLEF is a medical image database, which was also used in [YJM⁺10]. For all the categories,

¹<http://lear.inrialpes.fr/~jegou/data.php>

²<http://www.imageclef.org>

we randomly select 10% images as the query pool, the other images as database pool. For the Flickr photos, we treat all of them as the background noisy photos, which are mainly used to test the scalability of our algorithms. We denote this data set as the “ImageCLEF + Flickr1M” data set or “ImageCLEFFlickr” for short.

4.4.2 Experimental Setup

We present our experimental results by showing the percentage of database items searched with the hashing function instead of measuring the exact search time in order to avoid the unfairness from different implementations of the codes. To achieve this purpose, we have to set a parameter for the LSH search, i.e., ρ , which is used to control the fraction of nearest neighbors to be linearly scanned.

For performance metric, we evaluate the retrieval performance based on mean Average Precision (mAP) and top- n ($n = 1, 2, \dots, 5$) retrieval accuracy. The Average Precision (AP) value is the area under precision-recall curve for a query. The mAP value is calculated based on the average AP value of all the queries. The precision value is the ratio of relevant examples over the total retrieved examples, while recall is the ratio of the relevant examples retrieved over the total relevant examples in the database.

Our objective is to achieve fast and accurate image retrieval. As LSH can only return a portion of images, only the top returned images (Hit Items) are needed to be evaluated. Based on this concern, the mAP performance reported in our experiments is based on the mAP over all the returned items (except for the experiments of parameter sensitivity evaluation, where we adopt the top-10 mAP performance to enable a fair evaluation of different parameter settings). It is worth mentioning that in some previous works they usually reported mAP values for the whole data set. As we know, the more the image examples retrieved the higher the mAP value obtained. Thus, it is not totally fair to directly compare the exact value of our results with the results of the previous works. In our experiments, we have implemented all the

compared methods including our algorithms under the same evaluation criterion to enable fair comparisons. Nonetheless, from the experimental results, we found that the mAP results we achieved are higher than some of the others's reported results, although we are actually based on a criterion that generates relatively lower mAP values.

As the proposed algorithms need some training data, we split the original query set into two parts, each time we select one part for query, and the other part for training. The final result is obtained by computing the average of the results over the two splits. Finally, we conduct the evaluations of all the algorithms 10 times and average the results over these 10 runs to obtain a stable result.

4.4.3 Image Descriptors and Kernel Functions

4.4.3.1 Image Descriptors

We adopt both global and local feature descriptors for representing images in our experiments. We have some preprocessing by resizing all images to 500×500 pixels while keeping the aspect ratio unchanged. For global features, we extract five kinds of features, including (1) Color histogram and color moments, (2) Edge direction histogram, (3) Gabor wavelets transform, (4) Local Binary Pattern, and (5) GIST. For local features, we extract the bag of visual words features based on two types of descriptors: SIFT and SURF. In particular, for SIFT: we adopt the Hessian-Affine interest region detector with threshold 500 and the SIFT descriptor; for SURF: we use SURF detector with threshold 500 and SURF descriptor. For the clustering, we adopt a forest of 16 kd-trees and search 2048 neighbor to speed up the clustering task. Finally, we use TF-IDF to generate the bag of visual words to represent the local features. In total, with different vocabulary sizes, we extracted four kinds of local features, including SIFT200, SIFT1000, SURF200 and SURF1000.

4.4.3.2 Kernel Functions

From the above, we represent each image by 9 types of different features. We then build kernel functions for these 9 types of features. Before we compute the kernels, we normalize the feature vectors to zero mean (each dimension) and unit length (each point). We adopt the RBF kernel:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\gamma^{-1}d(\mathbf{x}, \mathbf{x}')) \quad (\text{Eq. 4.3})$$

where $d(\cdot, \cdot)$ is the distance and γ is selected as the mean of the pairwise distance where the distance is computed using L2 distance. In total, we have a set of 9 kernels for our retrieval tasks. Finally, we note that all kernel matrices are normalized to unit trace to balance different kernels.

4.4.4 Comparison Algorithms

To extensively examine the efficacy of the proposed algorithms, we have implemented several different solutions for adopting KLSH with multiple kernels. In particular, we have implemented the following algorithms:

- **KLSH-Uniform:** a baseline method that uniformly combines the m kernels, i.e., $\kappa = \sum_{l=1}^m \frac{1}{m} \kappa_l$, and adopts this combined kernel for KLSH.
- **KLSH-Best:** We test the retrieval performance of all kernels, evaluate their mAP values on the training set, and then select the best kernel (with the highest mAP value). We adopt this best kernel for KLSH.
- **KLSH-Weight:** We evaluate the mAP performance of all kernels on the training set, calculate the weight of each kernel w.r.t. their mAP values: $\alpha_l = e^{mAP_l}$ (the same weight function as WMKLSH and BMKLSH), and finally normalize the weights (sum to be 1). Finally, we adopt the weighted combination of the m kernels: $\kappa = \sum_{l=1}^m \alpha_l \kappa_l$ for KLSH.

- MKLSH [WJHT10b]: an existing KLSH approach that uses multiple kernels by a uniform bit size allocation.
- WMKLSH: the proposed algorithm by using a weighted bit size allocation for multiple kernels, as described in Section 4.3.2.
- BMKLSH: the proposed BMKLSH algorithm by optimizing bit size allocation via boosting as shown in Algorithm 8.

4.4.5 Experimental Results

We now present the performance evaluation results on the data sets. We measure the performance in terms of top- n ($n = 1, 2, \dots, 5$) precision and the mAP value of all returned Hit items. For this experiment, we fix the parameters as follows: $\rho = 0.1$, $b = 300$, $p = 300$, $t = 30$, and $T = 20$. We will evaluate the sensitivity of these parameters in the subsequent section. We summarize the experimental results of mAP performance of the compared algorithms on the two data sets in Table 4.2, and illustrate the details of the top- n precision results in Figure 4.2. Below we discuss the empirical observations from these results.

To examine statistical significance of the comparisons, for the experimental results reported below, we highlight the best result in each group in bold font by conducting student t-tests with the significance level $\alpha = 0.05$.

4.4.5.1 On the “Holiday” Data Set

From the experimental result shown in Table 4.2 and Figure 4.2, we can draw several observations. First of all, by comparing the three different KLSH algorithms with different kernels, it seems a bit surprising to find that KLSH-Uniform, a simple uniform combination of all kernels, outperformed KLSH-Best, which is based on the best kernel chosen from the training set. But when thinking further, it is not difficult to explain the result as KLSH-best only explores a single kernel, while KLSH-Uniform jointly exploits multiple kernels. This result is further

Table 4.2: Experimental results of mAP performance.

Algorithm	Metric	Holiday	ImageCLEFFlickr
KLSH-Uniform	mean	0.58506	0.16902
	std	± 0.00258	± 0.00100
KLSH-Best	mean	0.50361	0.09813
	std	± 0.00364	± 0.00018
KLSH-Weight	mean	0.59986	0.17823
	std	± 0.00321	± 0.00156
MKLSH	mean	0.58994	0.16761
	std	± 0.00110	± 0.00086
WKLSH	mean	0.60562	0.17621
	std	± 0.00037	± 0.00064
BMKLSH	mean	0.66867	0.20460
	std	$\pm$ 0.00337	$\pm$ 0.00400

Figure 4.2: Evaluation of average top- n precision of retrieval results by different algorithms.

verified when we examine the result of KLSH-Weight, which outperform both KLSH-Best and KLSH-Uniform. These observations show that it is very important to explore the power of multiple kernels for KLSH in some real-world applications.

Second, by comparing the proposed WMKLSH and BMKLSH algorithms against the KLSH and MKLSH algorithms, we found that the proposed algorithms generally perform better than the KLSH and MKLSH algorithms, and the proposed BMKLSH algorithm attained the best result among all the compared algorithms, which was significantly better than the other algorithms. These promising results showed that the proposed BMKLSH technique is more effective to explore the power of multiple kernels for enhancing the image retrieval performance.

4.4.5.2 On the “ImageCLEF+Flickr1M” Data Set

The experimental results of this data set are shown in the last column of Table 4.2 and the right of Figure 4.2. By examining the results of the three KLSH algorithms, i.e., KLSH-Uniform, KLSH-Best, KLSH-Weight, we found that the situation is the same as that on the “Holiday” data set. Further, by comparing the proposed WMKLSH and BMKLSH algorithms with the KLSH and MKLSH algorithms, we found the similar observation where the proposed BMKLSH achieved the best result among all the compared schemes. This result indicates that by the proposed boosting scheme, the BMKLSH algorithm is to effectively identify the best kernel and achieve a good tradeoff between the best kernel and the optimal combination of multiple kernels.

4.4.6 Parameter Sensitivity Evaluation

In this section, we aim to examine the parameter sensitivity of the proposed BMKLSH scheme for image retrieval tasks. Specifically, there are several important parameters, including (1) ρ , the parameter that controls the fraction of nearest neighbors to be linearly scanned, (2) $\mathbf{b}$, the bit length of hash key, (3) the parameter $\mathbf{t}$ used in KLSH, which is to choose t indices for forming

vector $\mathbf{e}_S$, (4) the parameter $\mathbf{p}$ used in KLSH, which is to choose a subset of p examples for computing the kernel matrix in KLSH, and (5) the number of rounds $\mathbf{T}$ used in the boosting algorithm.

For the rest of the experiments, when varying one of the parameters for evaluation, the others will remain fixed at the following default settings: $\rho = 0.1$, $b = 300$, $t = 30$, $p = 300$, and $T = 20$. We adopt the top-10 mAP performance for evaluation in this section.

4.4.6.1 Evaluation of ρ and b

Figure 4.3: Evaluation of parameter ρ and b on the “Holiday” data set

The ρ and b are two key parameters for the LSH algorithm. Figure 4.3 (a) and (b) show the evaluation of two parameters ρ and b , respectively. From the experimental results, it is not difficult to see that increasing the value of ρ in general leads to increase of the top-10 mAP performance. This is not difficult to understand as the larger the ρ value, the more examples in the database will be inspected, thus more relevant image examples can be likely retrieved. Similarly, we also observe that increasing the value of hash key length b also leads to the increase of the top-10 mAP performance. This is easy to understand as the larger the hash key length, the more information can be encoded, which thus could lead to more accurate results.

Finally, similar to the previous observations, for all different values of these two parameters, BMKLSH consistently outperformed the other algorithms.

4.4.6.2 Evaluation of parameter p and t in KLSH

Figure 4.4: Evaluation of two parameters p and t used in the KLSH algorithm

The parameter p and t are two parameters in the KLSH algorithm. Figure 4.4 (a) and (b) show the evaluation of two parameters p and t , respectively. From the results, we can see that increasing the p value in general leads to better performance of all the algorithms. This is reasonable as more examples are sampled we are able to obtain a more accurate estimate of the distribution for KLSH. However, when p is large enough (e.g., $p > 200$), the improvement of increasing p becomes not significant. In practice, to trade off the performance and efficacy, we can choose any value between 200 and 400 for this situation. On the other hand, for the parameter t , we found that increasing the value of t does not always lead to improvement of the performance. In some cases, a large t value could slightly degrade the performance. Nonetheless, all the algorithms are generally not very sensitive to this parameter.

4.4.6.3 Evaluation of the number of boosting rounds T

We now examine how the number of boosting rounds affects the performance of the proposed BMKLSH algorithm. Figure 4.5 shows the evaluation results. From the results, we can see that the performance of BMKLSH algorithm in general increases with respect to the increase of T . The performance of the BMKLSH algorithm becomes saturated when T is sufficiently large, e.g., $T \geq 20$.

Figure 4.5: Evaluation of the number of boosting rounds (T) in the proposed BMKLSH algorithm.

4.4.7 Analysis of Bit Allocation Weights

Figure 4.6 illustrates the bit allocation weights by three algorithms on the two data sets. The x, y, z-axis denotes the index of kernel used, the algorithm, and the weight assigned to each kernel, respectively. From this figure, we can see that MKLSH assigns equal weights to all the kernels, WMKLSH assigns different weights according to their performance and the weights are non-zero, while the weights assigned by BMKLSH are sparse, i.e., it focuses on those kernels which are more beneficial to the retrieval tasks. Specifically, for the “Holiday” dataset, the weights learned by BMKLSH are mainly assigned to Color, GIST, SIFT1000 and SURF1000;

Figure 4.6: Visualization of bit allocation weights obtained by three different algorithms.

while for the “ImageCLEFFlickr” dataset, BMKLSH allocates all the weights to only GIST and SURF1000. This is reasonable as most of the medical images in the “ImageCLEF” dataset are gray-level images and contain rich texture contents, which thus favor GIST features instead of color features. Moreover, it is interesting to observe that for the “Holiday” data set, the weight of Color is less than SIFT200 and SURF200 assigned with WMKLSH, but BMKLSH filters SIFT200 and SURF200 while keeps Color. This is also quite reasonable as SIFT1000 and SURF1000 are somewhat redundant with SIFT200 and SURF200, but they are complementary to Color. These observations indicate that BMKLSH can learn an effective and sparse combination of multiple kernels.

4.4.8 Evaluation of Time Efficiency

Finally, we evaluate the efficiency of all the six algorithms on the “Holiday” data set. The experiments were running in Matlab on a Linux machine with 3GHz Intel CPU and 16GB RAM. As we analyzed before, all the compared algorithms share the same querying time given a fixed bit size b . In our experiments, typically for $b = 300$, the average retrieval time per query is about 0.65ms for all the compared algorithms. In the following, we focus on the evaluation of training and indexing time efficiency.

Figure 4.7 shows the evaluation results of the total amount of training and indexing time cost on the holiday data set, which were averaged over 10 runs. Among all the algorithms, it is

Figure 4.7: Evaluation of training and indexing time efficiency.

not surprising that WMKLSH and BMKLSH took more time cost for training and indexing because of the nature of their supervised learning processes. Such additional overhead is however acceptable since the training process typically is done in an offline manner. To further examine if the training and indexing process is scalable, we examine the relationship of their time cost with respect to the bit size b , and found that they generally follow a linear relationship. Besides, we also vary the number of boosting rounds T for the BMKLSH algorithm. From the results, we can see that the time cost only increases slightly w.r.t. the increase of T , which is almost neglected.

4.4.9 Evaluation of Qualitative Performance

Finally, we illustrate the qualitative retrieval performance by randomly choosing some query images from the database. Figure 4.8 shows the retrieval results by different algorithms. This

figure includes four retrieval cases of different queries, each of which shows the top-3 retrieved results by three representative algorithms: KLSH-Best, MKLSH, and BMKLSH. From the qualitative results, we can see that the proposed BMKLSH algorithm in general is able to return more relevant results than the other algorithms.

Figure 4.8: Comparison of qualitative retrieval performance on the “Holiday” dataset. This figure shows four examples. For each query, we show the top-3 retrieved images by three representative methods, i.e., KLSH-Best, MKLSH, and BMKLSH, respectively.

4.5 Summary

This chapter investigated a framework of Multi-Kernel Locality-Sensitive Hashing by exploring multiple kernels for efficient and scalable content-based image retrieval. We first gave some preliminaries of kernel LSH (KLSH). We proposed two new algorithms: (i) WMKLSH that combines multiple kernels via a simple weighted combination, and (ii) BMKLSH that employs a boosting-like scheme to optimize the bit allocation of multiple kernels for KLSH. We

have conducted an extensive set of experiments to evaluate the performance of the proposed algorithms, in which the encouraging results showed that the proposed BMKLSH algorithm using the boosting approach is able to considerably surpass a number of baseline methods.

So, we can conclude that by learning a combination of multiple kernels, BMKLSH is able to improve on the state-of-the-art hashing algorithm (KLSH), i.e., the second challenge is addressed in this chapter.

Chapter 5

Online Multiple Kernel Similarity

This chapter addresses the third challenge posed in section 1.2: learn a flexible nonlinear proximity function with multiple kernels to overcome the limitations of existing similarity learning algorithms. The rest of this chapter is organized as follows. Section 5.1 reviews related work. Section 5.2 gives some preliminaries of related techniques in literature. Section 5.3 introduces the problem definition and presents the proposed online learning algorithm for OMKS, followed by theoretical analysis in section 5.4. Section 5.5 discusses the experimental results, and section 5.6 summarizes this chapter.

5.1 Related Work

This section reviews related work which can be generally grouped into three major categories as follows.

5.1.1 Distance Metric Learning

Distance Metric Learning (DML) from side information has been actively studied in CBIR for several years. In general, most DML works aim to learn an optimal distance metric in the family of Mahalanobis distances, which can be viewed as an equivalent problem of learning an optimal linear projection of original data into a new space, where Euclidean distance is adopted to measure proximity between objects.

In literature, a variety of DML techniques have been proposed in both machine learning [BHHSW05, XNJR02, WBS06] and multimedia communities [HLLM06, SJHL06, LJJ08, WHJ⁺09, YJM⁺10]. Some well-known techniques include Relevant Component Analysis (RCA) [BHHSW05], Discriminative Component Analysis (DCA) [HLLM06] using the idea of Fisher's Linear Discriminant Analysis, Large Margin Nearest Neighbor (LMNN) [WBS06], Metric Learning by Collapsing Classes [GR05b], learning globally-consistent local distance functions [FSSM07], Regularized Distance Metric Learning [SJHL06] and Laplacian Regularized Metric Learning (LRML) [HLC08], and so on.

In general, the DML task is cast as a Semi-Definite Programming (SDP) problem due to the impose of Positive Semi-Definite (PSD) constraint on the solution, which is computationally intensive, especially when data is of high dimensionality. Some recent work has attempted to resolve the challenge, such as ITML [DKJ⁺07] and OASIS [CSSB10]. ITML uses LogDet divergence which can enforce positive semi-definiteness automatically to bypass the PSD constraint. OASIS drops the PSD constraint and resolves the DML task by an online learning algorithm to maximize the large margin criterion. OASIS is in general a linear metric learning method. Our technique is partially inspired to overcome the limitations of OASIS by studying kernel-based learning techniques.

5.1.2 Kernel-based Learning for Image Retrieval

Kernel-based learning techniques are not new for image retrieval. Our technique differs from the existing kernel-based learning techniques proposed for image retrieval in literature. For example, kernel SVM algorithms have been proposed for active learning in CBIR [TC01], which however address different types of problems as ours.

In literature, several kernel-based distance metric learning algorithms [CY07, YC07] were proposed for learning similarity functions in CBIR. Also, a family of metric learning algorithms including LMNN and NCA have been shown to be able to be kernelized by KPCA

trick [CKTK10]. Some recent work also reveals the connections between metric and kernel learning in [JKDD12], which may naturally provide kernelization for a larger class of metric learning methods. Our techniques differ from these approaches in two key aspects. First, they are designed to learn with a single kernel while the proposed algorithm learns with multiple kernels; and second, they usually run in a batch learning approach, which does not scale to large-scale applications. In contrast, we present online learning algorithms for learning a similarity function with multiple kernels.

We also note that our work is very different from existing kernel learning studies, such as KernelBoost [HBHW06] and SimpleNPKL [ZTH09], which mainly aim to learn a kernel function/matrix consistent with given constraints. Unlike these studies, the proposed technique learns a kernel-based similarity function, instead of a kernel function, from constraints.

Finally, some recent studies also proposed online learning techniques to learn kernel-based similarity function for text-based image search. For example, PAMIR [GB08] proposed to learn a discriminative model for the image retrieval from text queries using online learning techniques. Our study differs from PAMIR because PAMIR is specially designed for text-based queries and thus cannot be directly applied to the CBIR task.

5.1.3 Multiple Kernel Learning

Our work is also closely related to Multiple Kernel Learning (MKL) studies [LCB⁺04, SRSS06], which aim to find the optimal combination of multiple kernels for learning classifiers towards a given classification task. Exemplar algorithms include the convex optimization [LCB⁺04], the Semi-Infinite Linear Program (SILP) approach [SRSS06], the subgradient descent approach [RBCG08], and the level method [XJKL08]. In addition, several recent studies [ZO07, JSJY08] address multiple kernel learning for multi-class and multi-labeled data, and some other works aim at improving its efficiency and generality [VB09, VsAV10, JVV12a].

Despite sharing the common goal of finding the optimal combination of multiple kernels, our technique differs significantly from the existing MKL studies in two key aspects. First,

we aim to learn kernel-based proximity functions for image ranking tasks while conventional MKL studies often address classification tasks. Second, the training data used by conventional MKL studies are in the regular form of single data instances with class label, while the training data in our problem are in the form of triplet instances.

We should note that this work was also inspired by our previous work on Online Multiple Kernel Learning (OMKL) [JHY10]. The OMKL technique was proposed to learn classifiers by finding an optimal combination of multiple kernels for classification tasks, while the goal of this work is to learn the image similarity function from triplets for retrieval tasks in CBIR. In particular, special care is needed in the design of algorithms to handle the triple constraints. Finally, although MKL has successfully been applied to several computer vision applications [VR07, VGVZ09], to the best of our knowledge, this is the first online learning work that learns a similarity function using multiple kernels for CBIR.

5.2 Preliminaries

To better motivate our work, we introduce some preliminaries of two closely related techniques: (1) online multiple kernel learning, and (2) large scale online learning of image similarity through ranking.

5.2.1 OMKL

We briefly review the recent work on Online Multiple Kernel Learning (OMKL) [JHY10]. This technique in general aims to address a Multiple Kernel Learning (MKL) problem.

Specifically, given a set of training examples $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$ where $y_i \in \{-1, +1\}, i = 1, \dots, n$, and a collection of m kernel functions $\mathcal{K} = \{\kappa_i : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, i = 1, \dots, m\}$, the goal of an MKL task is to identify the optimal combination of the kernel matrices, denoted by $\theta = (\theta_1, \dots, \theta_m)$, which minimizes the margin-based classification error. This

can be formulated as the following optimization task:

$$\min_{\theta \in \Delta} \min_{f \in \mathcal{H}_{K(\theta)}} \frac{1}{2} \|f\|_{\mathcal{H}_{K(\theta)}}^2 + C \sum_{i=1}^n l(f(\mathbf{x}_i), y_i) \quad (\text{Eq. 5.1})$$

where $K(\theta)(\cdot, \cdot) = \sum_{i=1}^m \theta_i \kappa_i(\cdot, \cdot)$, $l(f(\mathbf{x}_i), y_i) = \max(0, 1 - y_i f(\mathbf{x}_i))$, and Δ is defined by

$$\Delta = \{\theta \in \mathbb{R}_+^m \mid \theta^\top \mathbf{e}_m = 1\}. \quad (\text{Eq. 5.2})$$

The OMKL technique simplifies the MKL problem by learning the following classification function with multiple kernels:

$$f(\mathbf{x}) = \sum_{i=1}^m \theta_i \text{sign}(f_i(\mathbf{x})) \quad (\text{Eq. 5.3})$$

The OMKL algorithm iteratively updates the prediction function by the perceptron algorithm, i.e.,

$$f_{t+1,i}(\mathbf{x}) = f_{t,i}(\mathbf{x}) + z_i(t) y_t \kappa_i(\mathbf{x}_t, \mathbf{x}) \quad (\text{Eq. 5.4})$$

and learns the combination weights by applying the Hedging online learning technique, i.e.,

$$\theta_i(t+1) = \theta_i(t) \beta^{z_i(t)} \quad (\text{Eq. 5.5})$$

where $\beta \in (0, 1)$ is a discount weight parameter, which is employed to penalize the kernel classifier that performs incorrect prediction at each learning step, and $z_i(t)$ indicates if the i -th kernel classifier makes a mistake on the prediction of the example $\mathbf{x}_t$.

5.2.2 OASIS

Below we briefly introduce another related work of large scale online learning of image similarity through ranking [CSSB10]. Specially, the goal of this problem is to learn a similarity function $S(\mathbf{p}_i, \mathbf{p}_j)$ that assigns higher similarity scores to pairs of more relevant images, i.e.,

$$S(\mathbf{p}_i, \mathbf{p}_i^+) > S(\mathbf{p}_i, \mathbf{p}_i^-), \quad \forall \mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^- \in P \quad (\text{Eq. 5.6})$$

$$\text{such that } r(\mathbf{p}_i, \mathbf{p}_i^+) > r(\mathbf{p}_i, \mathbf{p}_i^-) \quad (\text{Eq. 5.7})$$

where $r(\cdot, \cdot)$ reflects the relevance between two images, $\mathbf{p}_i$, $\mathbf{p}_i^+$ and $\mathbf{p}_i^-$ are all images, while $\mathbf{p}_i^+$ is more relevant to $\mathbf{p}_i$ than $\mathbf{p}_i^-$.

Consider a parametric similarity function that has a bi-linear form, $S_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_j) = \mathbf{p}_i^\top \mathbf{W} \mathbf{p}_j$ where $\mathbf{W} \in \mathbb{R}^{d \times d}$, the goal is to find a parametric similarity function S such that all triplets obey the following constraints:

$$S_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^+) > S_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^-) + 1 \quad (\text{Eq. 5.8})$$

One can define the following hinge loss function for the triplet:

$$l_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^-) = \max\{0, 1 - S_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^+) + S_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^-)\} \quad (\text{Eq. 5.9})$$

As a result, the *batch* optimization problem of this task can be formulated as:

$$\mathbf{W} = \arg \min_{\mathbf{W}} \|\mathbf{W}\|_F^2 + C \sum_i l_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^-) \quad (\text{Eq. 5.10})$$

The *online* optimization problem is formulated as:

$$\mathbf{W}_i = \arg \min_{\mathbf{W}} \frac{1}{2} \|\mathbf{W} - \mathbf{W}_{i-1}\|_F^2 + C l_{\mathbf{W}}(\mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^-) \quad (\text{Eq. 5.11})$$

By initializing $\mathbf{W}_0 = \mathbf{I}$, the online solution of updating $\mathbf{W}$ is given as:

$$\mathbf{W}_i = \mathbf{W}_{i-1} + \tau_i \mathbf{V}_i \quad (\text{Eq. 5.12})$$

where $\tau_i = \min\{C, \frac{l_{\mathbf{W}_{i-1}}(\mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^-)}{\|\mathbf{V}_i\|_F^2}\}$ and $\mathbf{V}_i = \mathbf{p}_i(\mathbf{p}_i^+ - \mathbf{p}_i^-)^\top$.

5.3 Online Multiple Kernel Similarity

We first present our framework for online kernel similarity learning, and then extend it to online multiple kernel similarity learning.

5.3.1 Online Kernel Similarity

Our challenge is how to extend the linear similarity function used in OASIS to its kernel version. To this end, for a given kernel $\kappa(\cdot, \cdot)$ and the corresponding Hilbert space $\mathcal{H}$, we introduce a linear operator $\mathbf{L} : \mathcal{H} \mapsto \mathcal{H}$ that maps a function $f \in \mathcal{H}$ to another function $\mathbf{L}[f] \in \mathcal{H}$. Given the linear operator $\mathbf{L}$, we define the similarity function $S_{\mathbf{L}}(\mathbf{p}, \mathbf{q})$ as

$$S_{\mathbf{L}}(\mathbf{q}, \mathbf{p}) = \langle \kappa(\mathbf{q}, \cdot), \mathbf{L}[\kappa(\mathbf{p}, \cdot)] \rangle_{\mathcal{H}} \quad (\text{Eq. 5.13})$$

$\mathbf{q} \in \mathcal{X}$ is a query image and $\mathbf{p} \in \mathcal{X}$ is an image in database to be retrieved. Compared to the similarity function $S_{\mathbf{W}}(\mathbf{p}, \mathbf{q}) = \mathbf{p}^{\top} \mathbf{W} \mathbf{q}$, we observe that the linear operator $\mathbf{L}$ plays the same role as matrix $\mathbf{W}$. Let $\mathcal{L}$ be the space that includes all the linear operators in $\mathcal{H}$, i.e.,

$$\mathcal{L} = \{\mathbf{L} : \mathcal{H} \mapsto \mathcal{H}, \mathbf{L} \text{ is a linear operator}\} \quad (\text{Eq. 5.14})$$

Following the framework of OASIS, we develop a framework of kernel similarity learning based on the linear operator. It searches for the optimal linear operator by minimizing the rank loss, i.e.,

$$\mathbf{L}_* = \arg \min_{\mathbf{L} \in \mathcal{L}} \|\mathbf{L}\|_{HS}^2 + C \sum_i \ell_{\mathbf{L}}(\mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^-) \quad (\text{Eq. 5.15})$$

$\|\cdot\|_{HS}$ is the Hilbert Schmidt norm, $\ell_{\mathbf{L}}(\mathbf{p}_i, \mathbf{p}_i^+, \mathbf{p}_i^-) = \max(0, 1 - S_{\mathbf{L}}(\mathbf{p}_i, \mathbf{p}_i^+) + S_{\mathbf{L}}(\mathbf{p}_i, \mathbf{p}_i^-))$.

Next, we develop an online learning algorithm for efficiently solving (Eq. 5.15) based on the online Passive Aggressive (PA) learning [CDK⁺06]. Similar to the OASIS algorithm, in the proposed online learning algorithm, at each trial t , given triplet $\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-$, we solve the following simple optimization problem

$$\mathbf{L}_t = \arg \min_{\mathbf{L} \in \mathcal{L}} \frac{1}{2} \|\mathbf{L} - \mathbf{L}_{t-1}\|_{HS}^2 + C \ell_{\mathbf{L}}(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-) \quad (\text{Eq. 5.16})$$

where we initialize $\mathbf{L}_0$ to be an identity operator at the beginning of online learning. The following proposition gives the closed-form solution to the above optimization.

Proposition 1 *The optimal solution to the optimization problem in (Eq. 5.16) can be expressed as:*

$$\mathbf{L}_t = \mathbf{L}_{t-1} + \tau_t \mathbf{Z}_t \quad (\text{Eq. 5.17})$$

where $\mathbf{Z}_t \in \mathcal{L}$ is a rank one linear operator and is given by $\mathbf{Z}_t[h](\cdot) = \kappa(\mathbf{p}_t, \cdot) (h(\mathbf{p}_t^+) - h(\mathbf{p}_t^-))$ for any $h \in \mathcal{H}$. The coefficient τ_t in (Eq. 5.17) is calculated as

$$\tau_t = \min \left\{ C, \frac{\max\{0, 1 - S_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^+) + S_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^-)\}}{\kappa(\mathbf{p}_t, \mathbf{p}_t)(\kappa(\mathbf{p}_t^+, \mathbf{p}_t^+) - 2\kappa(\mathbf{p}_t^+, \mathbf{p}_t^-) + \kappa(\mathbf{p}_t^-, \mathbf{p}_t^-))} \right\} \quad (\text{Eq. 5.18})$$

Proof: In the absence of ambiguity, to simplify the notation, τ_t is denoted as τ during this proof.

We rewrite the problem into a constrained form, i.e.,

$$\begin{aligned} \min_{\mathbf{L} \in \mathcal{L}, \xi \geq 0} \quad & \frac{1}{2} \|\mathbf{L} - \mathbf{L}_{t-1}\|_{HS}^2 + C\xi \\ \text{s. t.} \quad & 1 - \langle \kappa(\mathbf{p}_t, \cdot), \mathbf{L}[\kappa(\mathbf{p}_t^+, \cdot) - \kappa(\mathbf{p}_t^-, \cdot)] \rangle_{\mathcal{H}} \leq \xi \end{aligned}$$

We define the Lagrangian as

$$\begin{aligned} g(\mathbf{L}, \xi, \tau, \lambda) = \quad & \frac{1}{2} \|\mathbf{L} - \mathbf{L}_{t-1}\|_{HS}^2 + C\xi - \lambda\xi \\ & + \tau \left(1 - \xi - \text{tr}(\mathbf{L}\mathbf{Z}_t^\dagger) \right) \end{aligned} \quad (\text{Eq. 5.19})$$

where $\tau \geq 0$ and $\lambda \geq 0$ are Lagrangian multipliers, and $\mathbf{Z}_t : \mathcal{H} \mapsto \mathcal{H}$ is a rank one linear operator defined as

$$\mathbf{Z}_t[h](\cdot) = \kappa(\mathbf{p}_t, \cdot) \langle \kappa(\mathbf{p}_t^+, \cdot) - \kappa(\mathbf{p}_t^-, \cdot), h \rangle_{\mathcal{H}}$$

and $\mathbf{Z}_t^\dagger$ is the adjoint of $\mathbf{Z}_t$. By setting $\frac{\partial g(\mathbf{L}, \xi, \tau, \lambda)}{\partial \mathbf{L}} = 0$, we have the following

$$\frac{\partial g(\mathbf{L}, \xi, \tau, \lambda)}{\partial \mathbf{L}} = \mathbf{L} - \mathbf{L}_{t-1} - \tau \mathbf{Z}_t = 0$$

and therefore $\mathbf{L} = \mathbf{L}_{t-1} + \tau \mathbf{Z}_t$. Next, by setting $\frac{\partial g(\mathbf{L}, \xi, \tau, \lambda)}{\partial \xi} = 0$, we have

$$C - \tau - \lambda = 0$$

Since $\lambda \geq 0$, we have $\tau \leq C$. Thus, we have the following:

$$\begin{aligned} g(\tau) &= \frac{1}{2}\tau^2 \|\mathbf{Z}_t\|_{HS}^2 + \tau(1 - \text{tr}(\mathbf{L}\mathbf{Z}_t^\dagger)) \\ &= -\frac{1}{2}\tau^2 \|\mathbf{Z}_t\|_{HS}^2 + \tau(1 - \text{tr}(\mathbf{L}_{t-1}\mathbf{Z}_t^\dagger)) \end{aligned}$$

Further, by setting $\frac{\partial g(\tau)}{\partial \tau} = 0$, we have

$$\begin{aligned} \frac{\partial g(\tau)}{\partial \tau} &= -\tau \|\mathbf{Z}_t\|_{HS}^2 + 1 - \text{tr}(\mathbf{L}\mathbf{Z}_t^\dagger) \\ &= -\tau \|\mathbf{Z}_t\|_{HS}^2 + \ell_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-) \\ &= 0 \end{aligned}$$

Thus, we have

$$\tau = \frac{\ell_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)}{\|\mathbf{Z}_t\|_{HS}^2}$$

Combining the fact that $\tau \leq C$, we prove the proposition by using the fact

$$\begin{aligned} \|\mathbf{Z}_t\|_{HS}^2 &= \|\kappa(\mathbf{p}_t, \cdot)\|_{HS}^2 \|\kappa(\mathbf{p}_t^+, \cdot) - \kappa(\mathbf{p}_t^-, \cdot)\|_{HS}^2 \\ &= \kappa(\mathbf{p}_t, \mathbf{p}_t) (\kappa(\mathbf{p}_t^+, \mathbf{p}_t^+) + \kappa(\mathbf{p}_t^-, \mathbf{p}_t^-) - 2\kappa(\mathbf{p}_t^+, \mathbf{p}_t^-)) \end{aligned}$$

Using the above result, we can rewrite $S_{\mathbf{L}_t}(\mathbf{q}, \mathbf{p})$ as

$$\begin{aligned} S_{\mathbf{L}_t}(\mathbf{q}, \mathbf{p}) &= \langle \kappa(\mathbf{q}, \cdot), \mathbf{L}_t[\kappa(\mathbf{p}, \cdot)] \rangle_{\mathcal{H}} \\ &= \kappa(\mathbf{q}, \mathbf{p}) + \sum_{l=1}^t \tau_l \kappa(\mathbf{q}, \mathbf{p}_l) (\kappa(\mathbf{p}_l^+, \mathbf{p}) - \kappa(\mathbf{p}_l^-, \mathbf{p})) \end{aligned} \quad (\text{Eq. 5.20})$$

Similar to the function learned by support vector machines (SVM), we slightly abuse the concept of Support Vectors (SV) to define each triplet $(\mathbf{p}_l, \mathbf{p}_l^+, \mathbf{p}_l^-)$ of nonzero coefficient $\tau_l > 0$ as a support vector for the learned linear operator. Thus, during the whole training process, we should only keep trace of the support vectors and their coefficients. Algorithm 9 summarizes the proposed algorithm for Online Kernel Similarity (OKS).

Algorithm 9 Online Kernel Similarity (OKS)

 INPUT: parameter C , training triplets: $(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)$, an input kernel $\kappa(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$

- 1: Initialization: $\mathbf{L}_0 = \mathbf{I}$
- 2: **for** $t = 1, 2, \dots, T$ **do**
- 3: Receive a training triplet: $(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)$
- 4: $\tau_t = \min \left\{ C, \frac{\max\{0, 1 - S_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^+) + S_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^-)\}}{\kappa(\mathbf{p}_t, \mathbf{p}_t) (\kappa(\mathbf{p}_t^+, \mathbf{p}_t^+) - 2\kappa(\mathbf{p}_t^+, \mathbf{p}_t^-) + \kappa(\mathbf{p}_t^-, \mathbf{p}_t^-))} \right\}$
- 5: $\mathbf{L}_t = \mathbf{L}_{t-1} + \tau_t \mathbf{Z}_t$
- 6: **end for**

 OUTPUT: $S(\mathbf{p}, \mathbf{q}) = \langle \kappa(\mathbf{p}, \cdot), \mathbf{L}_T[\kappa(\mathbf{q}, \cdot)] \rangle$

5.3.2 Online Multiple Kernel Similarity

We now extend the above online kernel similarity learning problem to the setting of learning with multiple kernels, i.e., the Online Multiple Kernel Similarity (OMKS) learning task.

Figure 5.1 shows the system flow of the proposed OMKS scheme.

Figure 5.1: The system flow of the proposed online multiple kernel similarity learning scheme for image retrieval.

Let $\mathcal{K} = \{\kappa_i : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, i = 1, \dots, m\}$ be a collection of m kernel functions. Our goal is to identify the optimal combination of the m kernels, denoted by $\theta = (\theta_1, \dots, \theta_m)$, and consequentially learn the combined kernel similarity function that can be used effectively for

image similarity search, i.e.,

$$f(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^m \theta_i S_i(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^m \theta_i \langle \kappa_i(\mathbf{q}, \cdot), \mathbf{L}_i[\kappa_i(\mathbf{p}, \cdot)] \rangle_{\mathcal{H}_{\kappa_i}} \quad (\text{Eq. 5.21})$$

where $\mathbf{L}_i \in \mathcal{L}_i = \{\mathbf{L} : \mathcal{H}_{\kappa_i} \mapsto \mathcal{H}_{\kappa_i}, \mathbf{L} \text{ is a linear operator}\}$ and $S_i(\mathbf{q}, \mathbf{p}) = \langle \kappa_i(\mathbf{q}, \cdot), \mathbf{L}_i[\kappa_i(\mathbf{p}, \cdot)] \rangle_{\mathcal{H}_{\kappa_i}}$ is the similarity function based on the linear operator $\mathbf{L}_i$. To simultaneously learn both the combination weights $\{\theta_i\}_{i=1}^m$ and the linear operators $\{\mathbf{L}_i\}_{i=1}^m$, we cast multiple kernel similarity learning into the following optimization problem

$$\min_{\theta \in \Delta} \min_{\{\mathbf{L}_i\}_{i=1}^m} \frac{1}{2} \sum_{i=1}^m \theta_i \|\mathbf{L}_i\|_{HS}^2 + C \sum_{t=1}^T \ell(f(\mathbf{p}_t, \mathbf{p}_t^+) - f(\mathbf{p}_t, \mathbf{p}_t^-)) \quad (\text{Eq. 5.22})$$

where $f(\mathbf{p}, \mathbf{q})$ is given in (Eq. 5.21), Δ is defined in (Eq. 5.2) and $\ell(z)$ is the hinge loss.

Remark. At the first glance, the formulation of multiple kernel similarity learning in (Eq. 5.22) may be very different from that for multiple kernel learning in (Eq. 5.1). This difference is in fact superficial. According to [JHY10], the problem in (Eq. 5.1) is equivalent to the following optimization problem

$$\min_{\theta \in \Delta} \min_{\{f_i\}_{i=1}^m} \frac{1}{2} \sum_{i=1}^m \theta_i \|f_i\|_{HS}^2 + C \sum_{t=1}^T \ell(f(\mathbf{x}_t), y_t) \quad (\text{Eq. 5.23})$$

where $f(\mathbf{x}) = \sum_{i=1}^m \theta_i f_i(\mathbf{x})$. By comparing (Eq. 5.23) to (Eq. 5.22), we can find that multiple kernel similarity learning is almost identical to multiple kernel learning except that the kernel prediction functions f_i in (Eq. 5.1) is replaced with the linear operator $\mathbf{L}_i$ in (Eq. 5.22), and the loss functions are somewhat different.

There are two sets of target variables to be learned in the OMKS task, i.e., the combination weights of multiple kernels, and the set of linear operators with respect to each of different kernels. Following the idea of the online multiple kernel learning [JHY10], we apply the Hedging algorithm to online learn the combination weights of multiple kernels, and then apply the online kernel similarity learning algorithm to learn the similarity function of each individual kernel.

Specifically, for each of the m kernels, e.g., κ_i , on every online learning trial, we apply Proposition 1 to find the optimal coefficient for learning the similarity function with respect to kernel κ_i , and then apply the Hedging algorithm to update the combination weights as follows:

$$\theta_i(t) = \theta_i(t-1)\beta^{z_i(t)}$$

where $\beta \in (0, 1)$ is a discounting parameter, and $z_i(t)$ equals to 1 when $S_{\mathbf{L}_{t-1,i}}(\mathbf{p}_t, \mathbf{p}_t^+) - S_{\mathbf{L}_{t-1,i}}(\mathbf{p}_t, \mathbf{p}_t^-) \leq 0$, and 0 otherwise.

Algorithm 10 summarizes the proposed algorithm for Online Multiple Kernel Similarity (OMKS). Finally, we also analyze the theoretical bounds of the two proposed OKS and OMKS algorithms in section 5.4.

Algorithm 10 Online Multiple Kernel Similarity (OMKS)

INPUT:

- Kernels: $\kappa_i(\cdot, \cdot) : \chi \times \chi \rightarrow \mathbb{R}, i = 1, \dots, m$
- Combination weights $\theta_i(0) = 1, i = 1, \dots, m$
- Discount weight $\beta \in (0, 1)$

```

1: Initialize  $\mathbf{L}_{0,i} = \mathbf{I}, i \in [m]$ 
2: for  $t = 1, 2, \dots, T$  do
3:   Receive a training triplet:  $(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)$ 
4:   for  $i = 1, 2, \dots, m$  do
5:     Compute  $\tau_{t,i}$  in (Eq. 5.18) using  $\mathbf{L}_{t-1,i}$  for  $\mathbf{L}_{t-1}$ 
6:     Update  $\mathbf{L}_{t,i}$  by (Eq. 5.17) with  $\mathbf{Z}_{t,i}$  given by  $\mathbf{Z}_{t,i}[h](\cdot) =$ 
        $\kappa_i(\mathbf{p}_t, \cdot) (\langle \kappa_i(\mathbf{p}_t^+, \cdot), h \rangle_{\mathcal{H}_{\kappa_i}} - \langle \kappa_i(\mathbf{p}_t^-, \cdot), h \rangle_{\mathcal{H}_{\kappa_i}})$ 
7:     if  $S_{\mathbf{L}_{t-1,i}}(\mathbf{p}_t, \mathbf{p}_t^+) - S_{\mathbf{L}_{t-1,i}}(\mathbf{p}_t, \mathbf{p}_t^-) \leq 0$  then
8:       Set  $z_i(t) = 1$ 
9:     else
10:      Set  $z_i(t) = 0$ 
11:    end if
12:    Update  $\theta_i(t) = \theta_i(t-1)\beta^{z_i(t)}$ 
13:  end for
14: end for
    
```

 OUTPUT: $f(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^m \theta_i(T) \langle \kappa_i(\mathbf{q}, \cdot), \mathbf{L}_{T,i}[\kappa_i(\mathbf{p}, \cdot)] \rangle_{\mathcal{H}_{\kappa_i}}$

Remark. It is not difficult to analyze the time complexity of the proposed OKS and OMKS algorithms. Specifically, for each training step, the time mainly spends on calculating τ_t whose time complexity is $\mathcal{O}(|SV|)$, so the time complexity of OKS is $\mathcal{O}(T|SV|)$, where T is the number of training instances (i.e., the number of training steps for the proposed algorithms which are in an online manner), $|SV|$ denotes the size of support vectors of the similarity function; and the time complexity of OMKS is $\mathcal{O}(T|SV|m)$, where m denotes the number of different kernels. By assuming small $|SV|$ and m values, both algorithms are generally linear w.r.t. T , i.e., the number of training instances, making the proposed learning scheme efficient and scalable for large applications.

5.4 Theoretical Analysis

First of all, we analyze the mistake bound of the proposed Online Kernel Similarity (OKS) learning algorithm as shown in Algorithm 9 in the following theorem.

Theorem 5.1 *Let $(\mathbf{p}_1, \mathbf{p}_1^+, \mathbf{p}_1^-), \dots, (\mathbf{p}_T, \mathbf{p}_T^+, \mathbf{p}_T^-)$ be a sequence of triplets, where $\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^- \in \mathbb{R}^n$, and assume $\|\mathbf{Z}_t\|_{HS}^2 = \kappa(\mathbf{p}_t, \mathbf{p}_t)(\kappa(\mathbf{p}_t^+, \mathbf{p}_t^+) - 2\kappa(\mathbf{p}_t^+, \mathbf{p}_t^-) + \kappa(\mathbf{p}_t^-, \mathbf{p}_t^-)) \leq R$ for all t . Then the number of prediction mistakes M made by OKS on this sequence of examples is bounded by:*

$$M \leq \min_{\mathbf{L}} \left\{ \frac{1}{\min(1/R, C)} \left[\|\mathbf{L} - \mathbf{I}\|_{HS}^2 + 2C \sum_{t=1}^T l_{\mathbf{L}}(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-) \right] \right\}$$

Proof:

$$\begin{aligned} \Delta_t &= \|\mathbf{L}_{t-1} - \mathbf{L}\|_{HS}^2 - \|\mathbf{L}_t - \mathbf{L}\|_{HS}^2 \\ &= \|\mathbf{L}_{t-1} - \mathbf{L}\|_{HS}^2 - \|\mathbf{L}_{t-1} - \mathbf{L} + \tau_t \mathbf{Z}_t\|_{HS}^2 \\ &= -2\tau_t [(S_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^+) - S_{\mathbf{L}_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^-)) \\ &\quad - (S_{\mathbf{L}_t}(\mathbf{p}_t, \mathbf{p}_t^+) - S_{\mathbf{L}_t}(\mathbf{p}_t, \mathbf{p}_t^-))] - \tau_t^2 \|\mathbf{Z}_t\|_{HS}^2 \\ &\geq \tau_t (2\ell_t - \tau_t \|\mathbf{Z}_t\|_{HS}^2 - 2\ell_t^*) \end{aligned}$$

where $\ell_t = \ell_{L_{t-1}}(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)$ and $\ell_t^* = \ell_L(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)$, thus

$$\sum_{t=1}^T \tau_t (2\ell_t - \tau_t \|\mathbf{Z}_t\|_{HS}^2 - 2\ell_t^*) \leq \sum \Delta_t \leq \|\mathbf{L} - \mathbf{I}\|_{HS}^2 \quad (\text{Eq. 5.24})$$

Since $\tau_t = \min(\ell_t / \|\mathbf{Z}_t\|_{HS}^2, C)$, $\tau_t \|\mathbf{Z}_t\|_{HS}^2 \leq \ell_t$ and $\tau_t \leq C$.

$$\sum_{t=1}^T \tau_t (2\ell_t - \tau_t \|\mathbf{Z}_t\|_{HS}^2 - 2\ell_t^*) \geq \sum_{t=1}^T (\tau_t \ell_t - 2C\ell_t^*) \quad (\text{Eq. 5.25})$$

Combining Equation (Eq. 5.24) and (Eq. 5.25), we have

$$\sum_{t=1}^T \tau_t \ell_t \leq \|\mathbf{L} - \mathbf{I}\|_{HS}^2 + 2C \sum_{t=1}^T \ell_t^* \quad (\text{Eq. 5.26})$$

If the algorithm makes a mistake on round t then $\ell_t \geq 1$. In addition, according to the assumption, we have $\tau_t = \min(\ell_t / \|\mathbf{Z}_t\|_{HS}^2, C) \geq \min(1/R, C)$. Thus, we have:

$$\sum_{t=1}^T \tau_t \ell_t \geq \min(1/R, C) M$$

Plugging the above result into the previous equation will result in the conclusion stated in the theorem.

Secondly, we analyze the mistake bound of the proposed Online Multiple Kernel Similarity (OMKS) learning algorithm in Algorithm 10. For the convenience of discussions, we define the following notations:

$$\begin{aligned} \theta_t &= \sum_{i=1}^m \theta_i(t), \quad q_i(t) = \frac{\theta_i(t)}{\theta_t} \\ z_i(t) &= I(S_{\mathbf{L}_{t-1,i}}(\mathbf{p}_t, \mathbf{p}_t^+) - S_{\mathbf{L}_{t-1,i}}(\mathbf{p}_t, \mathbf{p}_t^-) \leq 0) \end{aligned}$$

where $I(x)$ is an indicator function that outputs 1 when x is true and 0 otherwise. Here, $q_i(t)$ essentially defines the mixture of kernel similarity functions, and $z_i(t)$ indicates if training example $(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-)$ is misclassified by the i th kernel similarity function at trial t . Finally, we define the optimal margin error $g(\kappa_i, l, \mathcal{L})$ for the kernel $\kappa_i(\cdot, \cdot)$ with respect to a collection

of training examples $\mathcal{L} = \{(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-), t = 1, \dots, T\}$ satisfying $\kappa_i(\mathbf{p}_t, \mathbf{p}_t)(\kappa_i(\mathbf{p}_t^+, \mathbf{p}_t^+) - 2\kappa_i(\mathbf{p}_t^+, \mathbf{p}_t^-) + \kappa_i(\mathbf{p}_t^-, \mathbf{p}_t^-)) \leq R_i$ as

$$g(\kappa_i, l, \mathcal{L}) = \min_{\mathbf{L}} \left\{ \frac{\left[\|\mathbf{L} - \mathbf{I}\|_{HS}^2 + 2C \sum_{t=1}^T \ell_{\mathbf{L}}(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-) \right]}{\min(1/R_i, C)} \right\}$$

Theorem 5.2 *After receiving a sequence of T training examples, denoted by $\mathcal{L} = \{(\mathbf{p}_t, \mathbf{p}_t^+, \mathbf{p}_t^-), t = 1, \dots, T\}$ satisfying $\kappa_i(\mathbf{p}_t, \mathbf{p}_t)(\kappa_i(\mathbf{p}_t^+, \mathbf{p}_t^+) - 2\kappa_i(\mathbf{p}_t^+, \mathbf{p}_t^-) + \kappa_i(\mathbf{p}_t^-, \mathbf{p}_t^-)) \leq R_i$, the number of mistakes M made by running the algorithm in Algorithm 10, denoted by*

$$M = \sum_{t=1}^T I \left(\sum_{i=1}^m q_i(t-1) z_i(t) \geq 0.5 \right)$$

is bounded as follows

$$M \leq \frac{2 \ln(1/\beta)}{1 - \beta} \min_{1 \leq i \leq m} \sum_{t=1}^T z_i(t) + \frac{2 \ln m}{1 - \beta} \quad (\text{Eq. 5.27})$$

$$\leq \frac{2 \ln(1/\beta)}{1 - \beta} \min_{1 \leq i \leq m} g(\kappa_i, l, \mathcal{L}) + \frac{2 \ln m}{1 - \beta} \quad (\text{Eq. 5.28})$$

By choosing the value of β as follows:

$$\beta = \frac{\sqrt{T}}{\sqrt{T} + \sqrt{\ln m}},$$

we have

$$M \leq 2 \left(\left(1 + \sqrt{\frac{\ln m}{T}} \right) \min_{1 \leq i \leq m} g(\kappa_i, l, \mathcal{L}) + \ln m + \sqrt{T \ln m} \right)$$

The proof to the above theorem essentially combines the results of the passive aggressive learning algorithm [CDK⁺06] and the Hedge learning algorithm. We omit the details of our proof due to space limitation.

5.5 Experiments

In this section, we conduct an extensive set of experiments to evaluate the efficacy of the proposed algorithms for visual similarity search in CBIR.

5.5.1 Experimental Testbed

We adopt five publicly available image data sets¹. These five data sets have been widely used for the benchmark of image retrieval, classification and recognition tasks.

The first testbed is the “Indoor” database², which was used for the research of recognizing indoor scenes [QT09]. This data set consists of 67 indoor categories, and a total of 15620 images. The number of images contained in different categories are diverse, but each category contains at least 100 images. It is further divided into five subsets: *store*, *home*, *public spaces*, *leisure*, *working place*. We evaluate the performance of different algorithms individually on a randomly picked subset: *public spaces* (we name it “Public”, it is also used to evaluate the effect of parameters) as well as the whole indoor collection.

The second testbed is the “Caltech256” database³, which has been widely adopted for object recognition and image retrieval tasks [GHP07, CSSB10]. This database contains 256 object categories (excluding the background category) and a total of 30607 images. Following the similar experiments as the previous work [CSSB10], we pick 10, 20 or 50 out of the 256 classes to form three subsets (the same sets as used in [CSSB10]), which are named as “Caltech10”, “Caltech20”, and “Caltech50”, respectively.

The third testbed is the “Corel5000” database [HLLM06]. The image testbed consists of real-world photos from COREL image CDs. It has 50 categories, with each category contains exactly 100 images that are randomly selected from relevant examples in the COREL image CDs.

¹The data sets used in our experiments are all available in our project website:<http://OMKS.stevenhoi.org/>

²<http://web.mit.edu/torralba/www/indoor.html>

³http://www.vision.caltech.edu/Image_Datasets/Caltech256/

The fourth testbed is the “ImageCLEF” database⁴, which was also used in [YJM⁺10]. It is a medical image data set. We also combine “ImageCLEF” with a collection of 100,000 social photos crawled from Flickr, this larger set is named “ImageCLEF+”. For the Flickr photos, we treat all of them as the background noisy photos, which are mainly used to test the scalability of our algorithms.

The fifth testbed is the “Oxford Buildings” database⁵, which was first used in [PCI⁺07]. It consists of 5062 images collected from Flickr by searching for particular Oxford landmarks. The query set contains 55 queries from 11 different landmarks. We name it “Oxford” for short.

5.5.2 Experimental Setup

For each data set (except “Oxford” for which no categorical information is available), we randomly select a subset from each class to make sure that all classes have the same number of images as the one has least images in the original data set. This can avoid the performance being dominated by some single class of large number of images. Based on the data set, we then randomly select 50% examples from each class to form a training set, 10% examples to form a validation set, 10% examples to form a query set, and the rest 30% examples to form the test set for retrieval evaluation. The validation set is mainly used to determine the best parameters and the best cases of the compared algorithms. The final results are averages over 5 splits. We measure both mean and standard deviation of the results, and highlight the best case by performing student t-tests with the significance level $\alpha = 0.05$.

We need to generate side information in the forms of triplet training instances for learning the similarity functions by OASIS and the proposed algorithms, and also pairwise training data instances for the kernelized ITML algorithm (KITML) [JKDD12]. In our approach, we generate the side information by sampling triplet constraints from the images in the training set according to their ground truth class labels. Specifically, we generate all positive pairs

⁴<http://imageclef.org/>

⁵<http://www.robots.ox.ac.uk/vgg/data/oxbuildings/index.html>

(two images belong to the same class), and for each positive pair we randomly select another image from another class to form a triplet. Then two pairwise constraints $(\mathbf{p}, \mathbf{p}^+, +1)$ and $(\mathbf{p}, \mathbf{p}^-, -1)$ can be derived from the triplet $(\mathbf{p}, \mathbf{p}^+, \mathbf{p}^-)$. After that we randomly sample 20% (i.e., $RatioTrain = 20\%$) of all training instances to form the training instance set in our experiments to speed up the experiments (as we can see in section 5.5.6, the performance improves along with the number of training instances increases and then arrives at a saturated value).

For the “Oxford” data set, we randomly split the query set into 5 portions. Then for each split, we test the algorithms 5 times, each time select a different portion as query set, one portion as validation set, the others are left as training set. This can make sure that the average of these 5 runs is the evaluation of the whole query set. The same as the other data sets, the final results are averaged over 5 splits. The side information can be generated easily from the ground truth, we use all positives, and for each positive we randomly select a negative.

We evaluated the performance of all algorithms using some standard performance metrics for ranking in multimedia retrieval. Specifically, for each query image in the query set, all the test images are ranked according to their similarities to the query image, which returns a set of top n similar images for the query. We can measure the precision at top n returned images by computing the number of top- n images of the same class label as the query image. We also adopt the standard mean Average Precision (mAP) to evaluate the retrieval result. In particular, the Average Precision (AP) value is the area under precision-recall curve for a query. The mAP value is calculated based on the average AP value of all the queries. The precision value is the ratio of relevant examples over the total retrieved examples, while recall is the ratio of the relevant examples retrieved over the total relevant examples in the database.

Finally, all of the experiments were run in MATLAB environment on a Linux machine with 3GHz Intel CPU and 16GB RAM.

5.5.3 Image Descriptors and Kernel Functions

Here we describe how to extract features from images by different descriptors, and how to compute different kernel functions based on different kinds of features.

5.5.3.1 Image Descriptors

We adopt both global and local feature descriptors to extract features for representing images in our experiments. We have done some preprocessing of resizing all the images to the scale of 500×500 pixels while keeping the aspect ratio unchanged.

For global features, we extract five kinds of features to describe images, including (1) color histogram and color moments (81 dimensions), (2) edge direction histogram (37 dimensions), (3) Gabor wavelets transform (120 dimensions), (4) Local Binary Pattern (59 dimensions), and (5) GIST features (512 dimensions). These global features have been widely adopted in other previous CBIR studies.

For local features, we extract the bag-of-visual-words features using two types of descriptors. The first is the SIFT descriptor, in which we adopt the Hessian-Affine interest region detector with threshold 500; and the second is the SURF descriptor, in which we use the SURF detector with threshold 500. For the clustering step, we adopt a forest of 16 kd-trees and search 2048 neighbors to speed up the clustering task. The training data of the visual codebook for each data set are the descriptors generated from the data set itself. Finally, we adopt the TF-IDF weighing scheme to generate the final bag-of-visual-words for describing the local features. By choosing different descriptors (SIFT/SURF) and vocabulary sizes (200/1000), we totally extracted four kinds of local features: SIFT200, SIFT1000, SURF200 and SURF1000. For the “Oxford” data set, we use larger vocabulary sizes (20,000 and 100,000) instead, because it prefers larger code book size.

We apply PCA to all kinds of features and keep the first 50 dimensions (if the original dimension is less than 50, we keep all dimensions) to improve the efficiency of the experi-

ment. For those features whose dimension is larger than 10,000, Singular Value Decomposition (SVD) is performed to keep the first 1,000 dimensions. After dimension reduction, we normalize all feature vectors to unit length.

5.5.3.2 Kernel Functions

In the above, we represent each image in our database by a total of 9 types of different features. Based on these features, we can build a series of kernel functions on these features. To facilitate the learning tasks, we normalize all the kernel values to the range of $[0,1]$.

We adopt 4 kernel functions to build kernels on each kind of feature, which thus results in a total of 36 different kernels. The 4 kernels used in our approach are described as follows:

- RBF kernel: $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{d(\mathbf{x}, \mathbf{x}')}{\gamma\sigma^2})$, where $d(\cdot, \cdot)$ is the Euclidean distance, the kernel parameter γ is selected as the mean of the pairwise distance, σ is used to control the bandwidth, we select $\sigma \in \{2^{-1}, 2^0, 2^1\}$.
- Kernel using cosine similarity: $\kappa(\mathbf{x}, \mathbf{x}') = \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}$. We normalize the kernel $\kappa(\mathbf{x}, \mathbf{x}') = 0.5 \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2} + 0.5$ to the range of $[0,1]$.

5.5.4 Comparison Algorithms

To extensively examine the efficacy of the proposed algorithms, we have implemented the following algorithms:

- Eucl.-Best: We test the retrieval performance of all kinds of features on the validation set by ranking with Euclidean distance, and then select the best feature of the highest mAP. We report the result of this feature by ranking with Euclidean distance.
- RCA-Best: We train the RCA [BHHSW03] model on the training set for all the features and test the retrieval performance of all kinds of features on the validation set by RCA, and then select the best feature of the highest mAP value. We report the result of this feature by RCA.

- **LMNN-Best:** We train the LMNN [WBS06] model on the training set for all the features and test the retrieval performance of all kinds of features on the validation set by LMNN, and then select the best feature of the highest mAP value. We report the result of this feature by LMNN.
- **OASIS-Best:** We train the OASIS [CSSB10] model on the training set for all the features and test the retrieval performance of all kinds of features on the validation set by OASIS, and then select the best feature of the highest mAP value. We report the result of this feature by OASIS.
- **KRCA-Best:** We train the KRCA [TmCK05] model on the training set for all the kernels and test the retrieval performance of all kinds of kernels on the validation set by KRCA, and then select the best kernel of the highest mAP value. We report the result of this kernel by KRCA.
- **KITML-Best:** We train the kernelized ITML (KITML) [JKDD12] model on the training set for all the kernels, then test the retrieval performance of all kinds of kernels by KITML on the validation set, and finally select the best kernel of the highest mAP value. We report the result of this scheme using KITML with this kernel.
- **Eucl.-Con:** We concatenate all kinds of features together at first. Then we report the result of this feature by ranking with Euclidean distance.
- **RCA-Con:** We concatenate all kinds of features together at first. Then we report the result of this feature by RCA [BHHSW03].
- **LMNN-Con:** We concatenate all kinds of features together at first. Then we report the result of this feature by LMNN [WBS06].
- **OASIS-Con:** We concatenate all kinds of features together at first. Then we report the result of this feature by OASIS [CSSB10].

- KRCA-Con: We concatenate all kinds of features together at first. Then we train the KRCA [TmCK05] model on the training set for all 4 kernels of this feature and test the retrieval performance of theses 4 kinds of kernels on the validation set by KRCA, and select the best kernel of the highest mAP value. We report the result of this kernel by KRCA.
- KITML-Con: We concatenate all kinds of features together at first. Then we train the KITML [JKDD12] model on the training set for all 4 kernels of this feature, then test the retrieval performance of theses 4 kinds of kernels by KITML on the validation set, and finally select the best kernel of the highest mAP value. We report the result by KITML using this kernel.
- OKS-Best: We train the OKS model by Algorithm 9 on the training set for all the kernels and test the retrieval performance of all kinds of kernels on the validation set by OKS, and then select the best kernel of the highest mAP value. We report the result of this kernel by OKS.
- OKS-Avg: We build an average kernel at first by $\kappa(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^m \frac{1}{m} \kappa_i(\mathbf{x}, \mathbf{x}')$. Then we report the result of this kernel by Algorithm 9.
- OMKS-U: Use $f(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^m \frac{1}{m} S_i(\mathbf{q}, \mathbf{p})$ instead of $f(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^m \theta_i S_i(\mathbf{q}, \mathbf{p})$ for OMKS.
- OMKS-W: Use $f(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^m \theta_i S_i(\mathbf{q}, \mathbf{p})$ for OMKS, but the weight is computed as $\theta_i = e^{mAP_i}$. mAP_i is obtained by training the OKS model on the training set and test it on the validation set.
- OMKS: The proposed OMKS algorithm as shown in Algorithm 10.

5.5.5 Experimental Results

We now present the experimental results of performance evaluations on the data sets. We measure the performance in terms of top- n ($n = 1, 2, \dots, 5$) precision and the mAP values. We summarize in TABLE 5.1 the experimental results, measured by mAP, of the compared algorithms on all data sets. Fig. 5.2 illustrate the details of the top- n precision results on two sampled data sets. In TABLE 5.1, we highlight the best result in each group in bold font by conducting student t-tests with the significance level $\alpha = 0.05$. It is worth to note that, as we focus on the comparison between different algorithms for similarity learning when designing the experimental protocol, so we have not collected those features and kernels that may lead to the best performance, so the results cannot be directly compared with those reported by the others. We draw several empirical observations from the experimental results as follows.

Figure 5.2: Top- n precision results on “Public” and “Caltech10” data set.

First of all, by comparing the linear methods based on the best feature, we notice that

Table 5.1: Experimental results of mAP performance.

Alg.	Metric	Public	Indoor	Caltech10	Caltech20	Caltech50	Corel5000	ImageCLEF	ImageCLEF+	Oxford
Eucl.-Best	mAP std	0.1628 ± 0.0017	0.0439 ± 0.0007	0.2220 ± 0.0068	0.1668 ± 0.0017	0.0982 ± 0.0028	0.1833 ± 0.0047	0.4125 ± 0.0131	0.1224 ± 0.0105	0.4360 ± 0.0000
RCA-Best	mAP std	0.1595 ± 0.0073	0.0435 ± 0.0006	0.2203 ± 0.0085	0.1656 ± 0.0011	0.0995 ± 0.0031	0.1831 ± 0.0040	0.4563 ± 0.0130	0.1413 ± 0.0129	-
LMNN-Best	mAP std	0.1618 ± 0.0096	0.0447 ± 0.0005	0.2281 ± 0.0040	0.1639 ± 0.0085	0.1021 ± 0.0029	0.1958 ± 0.0072	0.4840 ± 0.0092	0.1367 ± 0.0124	-
OASIS-Best	mAP std	0.1681 ± 0.0047	0.0462 ± 0.0009	0.2365 ± 0.0061	0.1777 ± 0.0062	0.0997 ± 0.0023	0.1841 ± 0.0056	0.4530 ± 0.0138	0.1388 ± 0.0121	0.6078 ± 0.0422
KRCA-Best	mAP std	0.1657 ± 0.0040	0.0446 ± 0.0007	0.2451 ± 0.0052	0.1855 ± 0.0038	0.1114 ± 0.0016	0.2292 ± 0.0052	0.5520 ± 0.0165	0.1222 ± 0.0133	-
KITML-Best	mAP std	0.1754 ± 0.0048	0.0460 ± 0.0022	0.2513 ± 0.0047	0.1774 ± 0.0079	0.0993 ± 0.0030	0.1909 ± 0.0075	0.5434 ± 0.0108	0.1962 ± 0.0164	-
Eucl.-Con	mAP std	0.1921 ± 0.0033	0.0568 ± 0.0019	0.2052 ± 0.0093	0.1562 ± 0.0038	0.1013 ± 0.0032	0.2542 ± 0.0066	0.3919 ± 0.0184	0.1118 ± 0.0092	0.2032 ± 0.0000
RCA-Con	mAP std	0.1916 ± 0.0039	0.0566 ± 0.0019	0.2137 ± 0.0088	0.1727 ± 0.0046	0.1061 ± 0.0029	0.2606 ± 0.0059	0.4974 ± 0.0174	0.1509 ± 0.0128	-
LMNN-Con	mAP std	0.2009 ± 0.0049	0.0593 ± 0.0020	0.2265 ± 0.0103	0.1709 ± 0.0048	0.1114 ± 0.0034	0.2808 ± 0.0080	0.5251 ± 0.0150	0.1553 ± 0.0185	-
OASIS-Con	mAP std	0.2004 ± 0.0090	0.0581 ± 0.0028	0.2249 ± 0.0088	0.1659 ± 0.0055	0.1023 ± 0.0031	0.2518 ± 0.0103	0.4751 ± 0.0166	0.1540 ± 0.0114	0.4580 ± 0.0277
KRCA-Con	mAP std	0.1958 ± 0.0037	0.0589 ± 0.0021	0.241 ± 0.0153	0.1971 ± 0.0055	0.1226 ± 0.0021	0.3376 ± 0.0091	0.5762 ± 0.0222	0.1624 ± 0.0098	-
KITML-Con	mAP std	0.2049 ± 0.0090	0.0572 ± 0.0055	0.2468 ± 0.0164	0.1891 ± 0.0096	0.1044 ± 0.0034	0.2757 ± 0.0073	0.5597 ± 0.0215	0.1919 ± 0.0104	-
OKS-Best	mAP std	0.1897 ± 0.0044	0.0513 ± 0.0014	0.2653 ± 0.0088	0.1972 ± 0.0099	0.1192 ± 0.0058	0.2407 ± 0.0091	0.5785 ± 0.0099	0.2804 ± 0.0114	0.6698 ± 0.0142
OKS-Avg	mAP std	0.2152 ± 0.0088	0.0677 ± 0.0024	0.2601 ± 0.0160	0.2335 ± 0.0041	0.1433 ± 0.0049	0.3542 ± 0.0060	0.5036 ± 0.0150	0.2802 ± 0.0104	0.2312 ± 0.0051
OMKS-U	mAP std	0.218 ± 0.0085	0.0678 ± 0.0023	0.252 ± 0.0088	0.224 ± 0.0056	0.1382 ± 0.0035	0.3633 ± 0.0085	0.5766 ± 0.0316	0.3794 ± 0.0234	0.2187 ± 0.0192
OMKS-W	mAP std	0.2187 ± 0.0084	0.068 ± 0.0024	0.2559 ± 0.0084	0.2255 ± 0.0056	0.1389 ± 0.0035	0.3654 ± 0.0085	0.5966 ± 0.0289	0.3991 ± 0.0214	0.2709 ± 0.2709
OMKS	mAP std	0.2453 ± 0.0082	0.0794 ± 0.0022	0.3253 ± 0.0035	0.2690 ± 0.0089	0.1592 ± 0.0045	0.3858 ± 0.0091	0.6681 ± 0.0176	0.4442 ± 0.0084	0.7411 ± 0.0151

Note: KITML algorithms are too computationally intensive to run on large data sets, so we report the results by a low rank scheme proposed in [JKDD12] instead; some algorithms cannot run on “Oxford” dataset without class labels.

RCA-Best and LMNN-Best are not guaranteed to outperform Eucl.-Best, while OASIS-Best can achieve consistent improvements over Eucl.-Best.

Second, by comparing the linear methods based on the best feature (RCA-Best and OASIS-Best) with the kernel-based methods using the best kernel (KRCA-Best and OKS-Best), we observed that the kernel methods can improve the performance of the linear methods significantly. OKS-Best consistently outperforms all the other methods based on either single feature or single kernel. We also got the results of full rank KIMTL-Best (KITMLFR-Best) on three small scale data sets “Public”, “Caltech10” and “Caltech20”, their mAP values are

0.1911, 0.2754, 0.1989 respectively. It seems that KITMLFR-Best performs slightly better than OKS-Best (though their difference is not statistically significant). We believe this result is fairly encouraging since OKS-Best is an online learning method while KITMLFR-Best is a batch learning method. Despite their comparable performance, we emphasize OKS-Best is empirically more attractive due to its significant advantage in efficiency and scalability over KITMLFR-Best. In particular, as the time efficiency evaluation shown in TABLE 5.2, the running time of KITMLFR-Best is at least 700 times of that of OKS-Best, and the gain becomes more significant when the data set size increases. Because of the extremely high computational cost, KITMLFR-Best simply cannot run on large data sets as it will take several months to run these data sets on the same machine, so we report the results by a low rank scheme proposed in [JKDD12] instead. The dimension of KITML is set to 1/5 that of the original dimension and no more than 200. It can be observed that though this low rank scheme can improve the efficiency, it causes some loss in performance, such as for “Oxford” data set when the dimension is reduced from about 5,000 to 200, KITML failed to beat the baseline, so they were not included in TABLE 5.1. These promising results show that the proposed OKS algorithm is able to learn the similarity function more effectively and efficiently than the state-of-the-art techniques.

Table 5.2: Training time (seconds) of KITML versus OKS.

	Public	Caltech10	Caltech20
KITMLFR-Best	887.76	63.54	888.80
KITML-Best	15.07	2.34	15.16
OKS-Best	0.59	0.09	0.43

Third, we found that the methods based on the concatenated feature do not always outperform those based on the best feature. For example, consider the “Euclidean” distance, the feature concatenation approach outperforms the best feature only on dataset “Public”, “Indoor”, “Caltech50”, “Corel5000”, but fails on the other datasets. This observation implies that the feature concatenation is not optimal for combining different kinds of features.

Fourth, OKS-Avg outperforms OKS-Best on dataset “Public”, “Indoor”, “Caltech20”, “Caltech50”, “Corel5000”, but fails on the other data sets. In general, it is hard to conclude which is always better than the others. We believe whether or not the average kernel outperforms the best kernel should depend on the properties of the underlying data set and individual kernels. If the best kernel significantly outperforms the other kernels or there are a number of very poor kernels for the given data set, OKS-Best would be more likely to outperform OKS-Avg on such data set.

Fifth, OMKS-U and OMKS-W outperform OKS-Best in most cases, but fail on “Caltech10” and “Oxford”. This is primarily because some kernels have very poor performance, which in turn reduces the effectiveness of the uniform and the simple weighted combination. This result again motivates the importance of studying more advanced kernel combination approaches.

Finally, by examining the results of the proposed algorithm, we found that it consistently outperforms the other algorithms on all the data sets. This promising result shows that OMKS is able to learn an effective similarity function with multiple kernels by learning the optimal combination weights.

5.5.6 Evaluation with Varied-Size Training Data

In the previous experiments, we fixed training data by setting the parameter of *RatioTrain* to 20%. In this section, we evaluate the impact of varied amounts of training triplets for the proposed algorithms, as well as OASIS-Best which also adopts the same amounts of triplets as input.

Fig. 5.3 shows the evaluation results under varied values of *RatioTrain* used for building the similarity functions on two sampled data sets. From the results, we observed that all the algorithms in comparison share the similar performance trend as the number of training triplets increases. In particular, the larger the value of *RatioTrain*, the better the retrieval performance

can be achieved by the learning algorithms. Moreover, when $RatioTrain$ is large enough, e.g., over 40%, the improvements by most of the learning algorithms tend to become smaller, which is mainly attributed to sufficiently large amount of training data. Finally, similar to the previous experiments, for all the cases under varied values of $RatioTrain$, the proposed OMKS algorithm can perform significantly better than the other competing algorithms.

Figure 5.3: Evaluation of $RatioTrain$ on both "Public" and "Caltech10" data set.

5.5.7 Experiments Under Another Setup

Table 5.3 shows the experimental results obtained by following exactly the same settings as the previous work of OASIS [CSSB10], where "OASIS-Ori" is our implementation of OASIS based on the original features used in [CSSB10], and the others are the same as those described in Section 5.5.4 using our own features.

First of all, by comparing "OASIS-Ori" with the results published in [CSSB10], they are very similar, where the only slight differences were caused due to the randomization issues, i.e., different random splits, random generated training triplets and cross validation ranges.

Second, as our implemented algorithms adopt different features, the results of "OASIS-Best" are different from those of "OASIS-Ori". In general, our single best feature tends to

perform worse than the original features used in [CSSB10]; we conjecture this may be because they have adopted a well-design feature for this particular data set.

Table 5.3: Mean average precision and precision at top 1, 10 and 50 on the Caltech object data sets.

Caltech10	OMKS	OKS-Best	OASIS-Best	OASIS-Ori
Mean avg prec.	40±1.7	35±1.3	24±1.1	31±1.1
Top 1 prec.	59±1.8	49±3.3	35±3.4	45±2.1
Top 10 prec.	49±2.5	41±1.2	29±1.4	38±1.8
Top 50 prec.	27±1.3	25±1.0	20±0.5	23±0.3
Caltech20	OMKS	OKS-Best	OASIS-Best	OASIS-Ori
Mean avg prec.	31±0.6	25±0.7	20±0.5	19±0.7
Top 1 prec.	49±2.1	39±0.9	29±1.1	28±2.5
Top 10 prec.	39±0.8	31±1.0	24±0.9	23±1.1
Top 50 prec.	22±0.3	18±0.6	16±0.4	15±0.7
Caltech50	OMKS	OKS-Best	OASIS-Best	OASIS-Ori
Mean avg prec.	20±0.5	15±0.6	10±0.3	12±0.4
Top 1 prec.	36±1.5	26±1.5	16±1.5	20±0.6
Top 10 prec.	27±0.7	20±0.9	13±0.5	16±0.5
Top 50 prec.	15±0.2	12±0.3	9±0.1	10±0.3

Third, by comparing the results of “OMKS”, “OKS-Best”, “OASIS-Best” and “Eucl.-Best”, it is again to validate that our proposed algorithms “OMKS” and “OKS-Best” are significantly more effective by following another different experimental setup.

Finally, no matter which kinds of features used by OASIS, our algorithms can always make consistent improvements over the results of OASIS by following the same experimental setup in [CSSB10] .

5.5.8 Qualitative Comparison

In the last experiment, we sample several query images, and compare the top ranked images retrieved by different methods. Fig. 5.4 shows the qualitative comparisons of six different query examples obtained by four different algorithms, including “OASIS-Best”, “OKS-Best”, “OMKS-U” and “OMKS”. From the visual results, we observe that in general, “OKS-Best”

retrieves more relevant images than “OASIS-Best”, as illustrated by the results for the first two queries. This result implies the importance of introducing nonlinear similarity functions in ranking. But at the same time, we notice that for all the other 4 queries, the results by “OKS-Best” are not so perfect, which often returns irrelevant images similar to OASIS-Best. The result of query 3 and 4 indicates that “OMKS-U” tends to perform better than “OKS-Best”, validating the importance of incorporating multiple kernels built from diverse modalities. On the other hand, “OMKS-U” does not outperform “OKS-Best”. For example, for query 5, “OKS-Best” obtained 3 relevant images out of 4, while “OMKS-U” only obtained 1. Overall, OMKS overcomes the limitations of OMKS-U and is able to always find relevant images for all the queries, showing the significance of appropriately weighing individual kernels.

Figure 5.4: Qualitative comparison of image similarity search results on the “Caltech10” database by different algorithms. For each block, the first image is the query, and the results from the first line to the fourth line represents “OASIS-Best”, “OKS-Best”, “OMKS-U” and “OMKS”, respectively. The category names for the queries are as follows: 1 (roulette wheel), 2 (billard), 3 (skyscraper), 4 (bear), 5 (minotaur), 6 (laptop).

5.6 Summary

This chapter addressed a fundamental problem of learning similarity functions for ranking images towards image similarity search. To overcome the limitations of conventional distance metric learning techniques, we proposed a novel Online Multiple Kernel Similarity (OMKS) learning scheme that can effectively improve image similarity search by learning nonlinear proximity functions beyond conventional linear distance metric learning framework. By exploring the power of multiple kernels in combining multi-modal data, OMKs learns a much more flexible and powerful kernel-based proximity function to improve image similarity search in CBIR. We developed an efficient online learning algorithm and extensively evaluated the proposed algorithms for image similarity search on a number of public image databases. Our empirical results showed that OMKs significantly surpasses the state-of-the-art linear and nonlinear metric learning techniques for image similarity search.

In general, we can conclude that the third challenge which is learning a flexible nonlinear proximity function with multiple kernels to overcome the limitations of existing similarity learning algorithms, can be addressed by OMKs proposed in this chapter.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Multiple kernel learning (MKL) is a family of machine learning techniques by learning with multiple diverse kernel functions/matrices. It has been demonstrated as a promising technique for solving many real-world applications, especially for its power of exploiting multiple kernels for fusing diverse information from multiple sources. Beyond the traditional MKL studies, we investigate a more general problem which is learning with multiple kernels in this thesis. In the previous chapters, we conducted a survey on learning with multiple kernels and try to address three challenges:

- Improve the efficiency and effectiveness of regular MKL algorithms for classification.
- Improve the state-of-the-art hashing algorithm (KLSH) by learning a combination of multiple kernels.
- Learn a flexible nonlinear proximity function with multiple kernels to overcome the limitations of existing similarity learning algorithms.

For the first challenge we presented a framework of Multiple Kernel Boosting (MKBoost) for classification, which applies boosting techniques to learn classification models with multiple kernels. As a trade-off between accuracy and efficiency, we first proposed two deterministic

MKBoost algorithms using all kernels at each boosting trial; we then presented two stochastic MKBoost algorithms that randomly sample a subset of kernels at each boosting trial. We found MKBoost empirically achieved better accuracy than the regular MKL methods, and performed much more efficiently than the state-of-the-art MKL methods. Besides, we also found the stochastic MKBoost algorithms considerably improved the efficiency of the deterministic algorithms by maintaining comparable accuracy.

For the second challenge, we investigated a framework of Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) by exploring multiple kernels for efficient and scalable content-based image retrieval. We first gave some preliminaries of kernel LSH (KLSH). We proposed two new algorithms: (i) WMKLSH that combines multiple kernels via a simple weighted combination, and (ii) BMKLSH that employs a boosting-like scheme to optimize the bit allocation of multiple kernels for KLSH. We have conducted an extensive set of experiments to evaluate the performance of the proposed algorithms, in which the encouraging results showed that the proposed BMKLSH algorithm using the boosting approach is able to considerably surpass a number of baseline methods.

For the third challenge, we addressed a fundamental problem of learning similarity functions for ranking images towards image similarity search. To overcome the limitations of conventional distance metric learning techniques, we proposed a novel Online Multiple Kernel Similarity (OMKS) learning scheme that can effectively improve image similarity search by learning nonlinear proximity functions beyond conventional linear distance metric learning framework. By exploring the power of multiple kernels in combining multi-modal data, OMKs learns a much more flexible and powerful kernel-based proximity function to improve image similarity search in CBIR. We developed an efficient online learning algorithm and extensively evaluated the proposed algorithms for image similarity search on a number of public image databases. Our empirical results showed that OMKs significantly surpasses the state-of-the-art linear and nonlinear metric learning techniques for image similarity search.

6.2 Limitations

Although our proposed algorithms can address those three challenges to some extent, they also have some limitations. Next we will discuss their limitations one by one.

For MKBoost, it is limited in the following aspects:

- As the model of MKBoost is built up of a set of kernel classifiers, while traditional MKL algorithms only consist of a single classifier, so it may take more time during testing.
- The theoretical analysis is not directly related to the set of multiple kernels used.

BMKLSH also has some limitations:

- The current greedy solution is based on boosting technique, which is not guaranteed to be optimal.
- We have not provided theoretical analysis for the greedy solution.

OMKS also has some disadvantages. Such as, the time complexity of OMKs increases linearly with the size of support vectors. This will be a problem when we dealing with very large scale problems.

6.3 Future Work

Learning with multiple kernels is a generic method, we are sure that it is a promising direction. Some potential directions for our future research are listed in this section.

6.3.1 Improvement on MKBoost

We can improve the Multiple Kernel Boosting framework as follows:

- Improve the theoretical analysis of the MKBoost algorithms.

- Extend MKBoost to solve other problems other than classification, such as, regression, clustering, dimension reduction and feature selection etc.
- Apply MKBoost to some real-world applications. It is shown that MKBoost is both more efficient and effective than the state-of-the-art multiple kernel learning algorithms, so apply MKBoost to real-world applications may help to build more practical application systems.

6.3.2 Improvement on BMKLSH

We plan to improve BMKLSH in the following directions:

- Apply our BMKLSH technique to tackle other problems, such as search-based image annotation.
- Propose a strict formulation so as to get the optimal solution. Currently, we use a boosting scheme to greedily find a good solution, which is not guaranteed to be optimal.

6.3.3 Improvement on OMKS

OMKS can be improved in the following ways:

- Propose a budget scheme to overcome the problem led by increasing number of active support vectors.
- Investigate a projection scheme so as to solve the problem in linear space. No matter how good the budget scheme is designed, the budget size cannot be too small so as to maintain the performance. This problem becomes more serious when solving very large scale problems, so it will be interesting if we can reduce the problem to linear space.

6.3.4 Improvement on Regular MKL Algorithms

It can be seen from the survey (Chapter 2) that improving the regular MKL algorithms has been widely investigated and is still a hot spot. Researchers try to improve the efficiency by formulating MKL into optimization problems that can be solved easier, developing advanced solutions for existing optimization problems and so on. More effectiveness is obtained by using non-linear combinations of base kernels, considering more complicated frameworks and so on. So there are some opportunities that we can contribute in this aspect.

6.3.5 Solutions to Other Problems by Learning with Multiple Kernels

Learning with multiple kernels has already be adopted to solve some problems other than classification, such as regression, clustering, dimension reduction, feature selection, and so on. It is easy to observe that learning with multiple kernels is a generic idea. So in the future we plan to solve some other problems by learning with multiple kernels. For example, ordinal regression, multiple-instance learning, etc.

6.3.6 Other Applications of Learning with Multiple Kernels

In this thesis, we focus on the application to content-based image retrieval. But in the future, we plan to try other applications, such as video surveillance, video retrieval, and so on.

Appendix A

Publication List

A.1 Publications Related to This Thesis

- **Hao Xia**, Steven C.H. Hoi. MKBoost: A Framework of Multiple Kernel Boosting. The Eleventh SIAM International Conference on Data Mining (SDM2011), Mesa, Arizona, USA, pp. 199-210, 2012.
- **Hao Xia**, Steven C.H. Hoi, Pengcheng Wu, Rong Jin. Boosting Multi-Kernel Locality-Sensitive Hashing for Scalable Image Retrieval. The 35th Annual International ACM SIGIR Conference (SIGIR2012), Portland, Oregon, USA, 12-16 August, 2012.
- **Hao Xia**, Steven C.H. Hoi. MKBoost: A Framework of Multiple Kernel Boosting. IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 25 no. 7 pp. 1574-1586, July 2013.
- **Hao Xia**, Steven C.H. Hoi, Rong Jin, Peilin Zhao. Online Multiple Kernel Similarity Learning for Visual Similarity Search. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2013. To appear.

A.2 Other Publications

- Pengcheng Wu, Steven C.H. Hoi, **Hao Xia**, Peilin Zhao, Dayong Wang. Online Multi-modal Deep Similarity Learning with Application to Image Retrieval. The 21th International Conference on Multimedia (MM2013), Barcelona, Catalunya, Spain, 2013. To Appear.
- **Hao Xia**, Pengcheng Wu, Steven C.H. Hoi. Online Multi-modal Distance Learning for Scalable Multimedia Retrieval. ACM International Conference on Web Search and Data Mining (WSDM2013), Rome, 4-8 Feb, 2013.
- Pengcheng Wu, Steven C. H. Hoi, Peilin Zhao, **Hao Xia**, Zhi-Yong Liu, Chunyan Miao. Online Multi-modal Distance Metric Learning with Application to Image Retrieval. IEEE Transactions on Knowledge and Data Engineering (TKDE). (submitted).
- Pengcheng Wu, Steven C. H. Hoi, Peilin Zhao, **Hao Xia**. A Family of Scalable Learning to Rank Algorithms for Image Retrieval and Annotation. IEEE Transactions on Image Processing (TIP). (submitted).

References

- [ABTB⁺11] Jonathan Aflalo, Aharon Ben-Tal, Chiranjib Bhattacharyya, Jagarlapudi Saketha Nath, and Sankaran Raman. Variable sparsity kernel learning. *Journal of Machine Learning Research*, 12:565–592, 2011.
- [AH00] Selim Aksoy and Robert M. Haralick. Probabilistic vs. geometric similarity measures for image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2357–2362, 2000.
- [AHK⁺10] Peter Auer, Zakria Hussain, Samuel Kaski, Arto Klami, Jussi Kujala, Jorma Laaksonen, Alex Po Leung, Kitsuchart Pasupa, and John Shawe-Taylor. Pin-view: Implicit feedback in content-based image retrieval. *Journal of Machine Learning Research - Proceedings Track*, 11:51–57, 2010.
- [AMM09] Sunil Arya, Theodoris Malamatos, and David M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57(1):1–54, 2009.
- [Bac08] Francis Bach. Exploring large feature spaces with hierarchical multiple kernel learning. *Computing Research Repository*, abs/0809.1:105–112, 2008.
- [BBK01] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.
- [Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.
- [BFDR05] Jinbo Bi, Glenn Fung, Murat Dundar, and R. Bharat Rao. Semi-supervised mixture of kernels via lpboost methods. In *ICDM*, pages 569–572, 2005.

REFERENCES

- [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.
- [BHHSW03] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning distance functions using equivalence relations. In *Proceedings of International Conference on Machine Learning*, pages 11–18, 2003.
- [BHHSW05] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *JMLR*, 6:937–965, 2005.
- [BhN05] Asa Ben-hur and William Stafford Noble. Kernel methods for predicting protein-protein interactions. In *Intelligent Systems in Molecular Biology*, volume 21, pages 38–46, 2005.
- [BKK96] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. In *VLDB*, pages 28–39, San Francisco, CA, USA, 1996.
- [BLJ04] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of International Conference on Machine Learning*, Banff, Alberta, Canada, 2004.
- [BME02] Kristin P. Bennett, Michinari Momma, and Mark J. Embrechts. Mark: A boosting algorithm for heterogeneous kernel models. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 24–31, 2002.
- [Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [BZB04] Jinbo Bi, Tong Zhang, and Kristin P. Bennett. Column-generation boosting methods for mixture of kernels. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 521–526, 2004.

REFERENCES

- [cCL01] Chih chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines, 2001.
- [CDK⁺06] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.
- [CG10] Domenico Conforti and Rosita Guido. Kernel based support vector machine via semidefinite programming: Application to medical diagnosis. *Computers & OR*, 37(8):1389–1394, 2010.
- [Cha02] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.
- [CKS02] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems*, pages 537–544, 2002.
- [CKTK10] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10-12):1570–1579, 2010.
- [Cla88] Kenneth L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17(4):830–847, 1988.
- [CMR09a] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. In *Uncertainty in Artificial Intelligence*, 2009.
- [CMR09b] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems*, pages 396–404. 2009.
- [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.

REFERENCES

- [CMR12] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Ensembles of kernel predictors. *CoRR*, abs/1202.3712, 2012.
- [COST05] Bin Cui, Beng Chin Ooi, Jianwen Su, and Kian-Lee Tan. Indexing high-dimensional data for efficient in-memory similarity search. *IEEE Trans. on Knowl. and Data Eng.*, 17(3):339–353, 2005.
- [CR08] Olivier Chapelle and Alain Rakotomamonjy. Second order optimization of kernel parameters. *NIPS Workshop*, 2008.
- [CSSB10] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.
- [CSTE02] Nello Cristianini, John Shawe-Taylor, and André Elisseeff. On kernel-target alignment. *Journal of Machine Learning Research*, 2002.
- [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.
- [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.
- [CY07] Hong Chang and Dit-Yan Yeung. Kernel-based distance metric learning for content-based image retrieval. *Image Vision Comput.*, 25(5):695–703, 2007.
- [CZPC02] Guang-Ho Cha, Xiaoming Zhu, P. Petkovic, and Chin-Wan Chung. An efficient indexing method for nearest neighbor searches in high-dimensional image databases. *IEEE Transactions on Multimedia*, 4(1):76–87, 2002.
- [DBST02] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

REFERENCES

- [DCL08] Wei Dong, Moses Charikar, and Kai Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *SIGIR*, pages 123–130, 2008.
- [DDKD08] Réda Dehak, Najim Dehak, Patrick Kenny, and Pierre Dumouchel. Kernel combination for svm speaker verification. In *Proceedings of the Speaker and Language Recognition Workshop*, 2008.
- [dDMM04] Isaac Martín de Diego, Javier M. Moguerza, and Alberto Muñoz. Combining kernel information for support vector classification. In *Multiple Classifier Systems*, pages 102–111, 2004.
- [DGK04] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD*, pages 551–556, 2004.
- [DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th annual symposium on Computational geometry (SCG’04)*, pages 253–262. New York, NY, 2004.
- [DIO⁺09] I. Daoudi, K. Idrissi, S. E. Ouatik, A. Baskurt, and D. Aboutajdine. An efficient high-dimensional indexing method for content-based retrieval in large image databases. *Image Commun.*, 24(10):775–790, 2009.
- [DKJ⁺07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007.
- [DMSO10] Santanu Das, Bryan L. Matthews, Ashok N. Srivastava, and Nikunj C. Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 47–56, Washington, DC, USA, 2010.
- [DTX12] Lixin Duan, Ivor W. Tsang, and Dong Xu. Domain transfer multiple kernel learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 465–479, 2012.

REFERENCES

- [eFhCjL05] Rong en Fan, Pai hsuen Chen, and Chih jen Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.
- [FKM05] Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *SODA*, pages 385–394, 2005.
- [FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.
- [FSSM07] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, pages 1–8, 2007.
- [FYT09] Josef Kittler Fei Yan, Krystian Mikolajczyk and Muhammad Tahir. A comparison of l_1 norm and l_2 norm multiple kernel svms in image and video classification. In *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing*, 2009.
- [GA08] Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 352–359, 2008.
- [GA11] Mehmet Gönen and Ethem Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.
- [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.
- [GD07] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *CVPR*, 2007.

REFERENCES

- [GHP07] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.
- [GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.
- [GKK11] Mehmet Gönen, Melih Kandemir, and Samuel Kaski. Multitask learning using regularized multiple kernel learning. In *Neural Information Processing*, pages 500–509. Springer, 2011.
- [GKN12] Viktor Gl, Etienne E. Kerre, and Mike Nachtegaele. Multiple kernel learning based modality classification for medical images. In *CVPR Workshops*, pages 76–83, 2012.
- [GN08] Peter V. Gehler and Sebastian Nowozin. Infinite kernel learning. In *NIPS 2008 Workshop on "Kernel Learning: Automatic Selection of Optimal Kernels" (LK ASOK 08)*, 2008.
- [GR05a] Mark Girolami and Simon Rogers. Hierarchic bayesian models for kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 241–248, 2005.
- [GR05b] Amir Globerson and Sam Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, 2005.
- [Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In Beatrice Yormark, editor, *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pages 47–57. ACM Press, 1984.
- [GZ06] Mark Girolami and Mingjun Zhong. Data integration for classification problems employing gaussian process priors. In *NIPS*, pages 465–472, 2006.
- [HB07] Zaïd Harchaoui and Francis Bach. Image classification with segmentation graph kernels. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

REFERENCES

- [HBHW06] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Learning a kernel function for classification with small training samples. In *ICML*, pages 401–408, 2006.
- [HCC12] Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Multiple kernel fuzzy clustering. pages 120–134, 2012.
- [HCK09] Mingqing Hu, Yiqiang Chen, and James Tin-Yau Kwok. Building sparse multiple-kernel svm classifiers. *IEEE Transactions on Neural Networks*, 20:827–839, 2009.
- [HCRB11] Junfeng He, Shih-Fu Chang, Regunathan Radhakrishnan, and Claus Bauer. Compact hashing with joint optimization of search accuracy and time. In *CVPR*, pages 753–760, 2011.
- [HCX08] Junfeng He, Shih-Fu Chang, and Lexing Xie. Fast kernel learning for spatial pyramid matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- [HHL⁺11] Han-Pang Huang, Tz-Hau Huang, Yi-Hung Liu, Zhi-Hao Kang, and Jyh-Tong Teng. A brain-controlled rehabilitation system with multiple kernel learning. In *SMC*, pages 591–596, 2011.
- [HJ85] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, New York, 1985.
- [HJL07] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of International Conference on Machine Learning*, pages 361–368, 2007.
- [HJZY13] Steven CH Hoi, Rong Jin, Peilin Zhao, and Tianbao Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, 2013.
- [HL08] Steven C. H. Hoi and Michael R. Lyu. A multimodal and multilevel ranking scheme for large-scale video retrieval. *IEEE Transactions on Multimedia*, 10(4):607–619, 2008.

REFERENCES

- [HLC06] Steven C. H. Hoi, Michael R. Lyu, and Edward Y. Chang. Learning the unified kernel machines for classification. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 187–196, 2006.
- [HLC08] Steven C.H. Hoi, Wei Liu, and Shih-Fu Chang. Semi-supervised distance metric learning for collaborative image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 2008.
- [HLC10] Junfeng He, Wei Liu, and Shih-Fu Chang. Scalable similarity search with optimized kernel hashing. In *KDD*, pages 1129–1138, 2010.
- [HLJ06] Steven C.H. Hoi, Michael R. Lyu, and Rong Jin. A unified log-based relevance feedback scheme for image retrieval. *IEEE Trans. KDE*, 18(4):509–204, 2006.
- [HLLM06] Steven C.H. Hoi, Wei Liu, Michael R. Lyu, and Wei-Ying Ma. Learning distance metrics with contextual constraints for image retrieval. In *CVPR*, June 17–22 2006.
- [HSS08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.
- [HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.
- [IM98] P. Indyk and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.
- [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.
- [JB11] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *CVPR*, pages 873–880, 2011.

REFERENCES

- [JDR⁺09] Saketha N. Jagarlapudi, Dinesh, Raman, Chiranjib Bhattacharyya, Aharon Ben-Tal, and Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. In *Advances in Neural Information Processing Systems*, pages 844–852. 2009.
- [JDS08] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV (1)*, pages 304–317, 2008.
- [JHY10] Rong Jin, Steven C. H. Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *ALT*, pages 390–404, 2010.
- [JKDD12] Prateek Jain, Brian Kulis, Jason V. Davis, and Inderjit S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 2012.
- [JSJY08] Shuiwang Ji, Liang Sun, Rong Jin, and Jieping Ye. Multi-label multiple kernel learning. In *Advances in Neural Information Processing Systems*, 2008.
- [JVV12a] A. Jain, S. V. N. Vishwanathan, and M. Varma. Spg-gmkl: Generalized multiple kernel learning with a million kernels. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2012.
- [JVV12b] Ashesh Jain, S. V. N. Vishwanathan, and Manik Varma. Spf-gmkl: generalized multiple kernel learning with a million kernels. In *KDD*, pages 750–758, 2012.
- [JZH11] Ivor W. Tsang Jinfeng Zhuang and Steven C.H. Hoi. Two-layer multiple kernel learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, volume 15, pages 909–917, 2011.
- [KBL08] Marius Kloft, Ulf Brefeld, and Pavel Laskov. Non-sparse multiple kernel learning. *NIPS Workshop on Kernel Learning*, 2008.
- [KBS⁺09] Marius Kloft, Ulf Brefeld, Soeren Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate ℓ_p -norm multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 997–1005, 2009.

REFERENCES

- [KG09] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.
- [KL02] Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of 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 *ICML*, pages 465–472, 2006.
- [KSTC02] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Optimizing kernel alignment over combinations of kernels. Technical Report 121, Department of Computer Science, Royal Holloway, University of London, UK, 2002.
- [KTI03] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of International Conference on Machine Learning*, pages 321–328, 2003.
- [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 International Conference on Machine Learning*, pages 323–330, 2002.
- [LCB⁺04] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
- [LCW03] Beita Li, Edward Y. Chang, and Yi-Leh Wu. Discovery of a perceptual distance function for measuring image similarity. *Multimedia Syst.*, 8(6):512–522, 2003.
- [LDC⁺04] Gert R. G. Lanckriet, Minghua Deng, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Pacific Symposium on Biocomputing*, pages 300–311, 2004.

REFERENCES

- [LG08] Chris Longworth and Mark J. F. Gales. Multiple kernel learning for speaker verification. In *ICASSP*, pages 1581–1584, 2008.
- [LG09] Chris Longworth and Mark J. F. Gales. Combining derivative and parametric kernels for speaker verification. *IEEE Transactions on Audio, Speech & Language Processing*, 17(4):748–757, 2009.
- [LJJ08] Jung-Eun Lee, Rong Jin, and Anil K. Jain. Rank-based distance metric learning: An application to image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, 2008.
- [LJW⁺07] Q. Lv, W. Josephson, Z. Wang, M. S. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*. Vienna, Austria, 2007.
- [LLF08] Yen-Yu Lin, Tyng-Luh Liu, and Chiou-Shann Fuh. Dimensionality reduction for data in multiple feature representations. In *Advances in Neural Information Processing Systems*, pages 961–968, 2008.
- [LLF11] Yen-Yu Lin, Tyng-Luh Liu, and Chiou-Shann Fuh. Multiple kernel learning for dimensionality reduction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(6):1147–1160, 2011.
- [LSDJ06a] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, 2006.
- [LSDJ06b] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, 2006.
- [LTL⁺12] Pengpeng Liang, Gregory Teodoro, Haibin Ling, Erik Blasch, Genshe Chen, and Li Bai. Multiple kernel learning for vehicle detection in wide area motion imagery. In *FUSION*, pages 1629–1636, 2012.

REFERENCES

- [LWS08] Xuchun Li, Lei Wang, and Eric Sung. Adaboost with svm-based component classifiers. *Eng. Appl. of AI*, 21(5):785–795, 2008.
- [MGL11] Brian McFee, Carolina Galleguillos, and Gert R. G. Lanckriet. Contextual object localization with multiple kernel nearest neighbor. pages 570–585, 2011.
- [MKZ11] Sören Sonnenburg Maris Kloft, Ulf Brefled and Alexander Zien. ℓ_p -norm multiple kernel learning. *Journal of Machine Learning Research*, 12:953–997, 2011.
- [Mm06] Javier M. Moguerza and Alberto munoz. Support vector machines with applications. *Statistical Science*, 21(3):322–336, 2006.
- [MMR⁺01] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.
- [MRB01] Thomas Melzer, Michael Reiter, and Horst Bischof. Nonlinear feature extraction using generalized canonical correlation analysis. In *ICANN*, pages 353–360, 2001.
- [MRW⁺99] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pages 41–48, 1999.
- [MSX⁺11] Andr Filipe Torres Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mrio A. T. Figueiredo. Online learning of structured predictors with multiple kernels. pages 507–515, 2011.
- [MSY10] Yadong Mu, Jialie Shen, and Shuicheng Yan. Weakly-supervised hashing in kernel space. In *CVPR*, pages 3344–3351, 2010.
- [NH08] Canh Hao Nguyen and Tu Bao Ho. An efficient kernel matrix evaluation measure. *Pattern Recognition*, 41:3366–3372, 2008.
- [NS96] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

REFERENCES

- [OZ08] Cheng Soon Ong and Alexander Zien. An automated combination of kernels for predicting protein subcellular localization. In *Proc. 8th Intl. Workshop on Algorithms in Bioinformatics (WABI'08)*, pages 186–197, Karlsruhe, Germany, 2008.
- [PCI⁺07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [PJG08] B. Kulis P. Jain and K. Grauman. Fast image search for learned metrics. In *CVPR*, 2008.
- [PMD00] Dmitry Pavlov, Jianchang Mao, and Byron Dom. Scaling-up support vector machines using boosting algorithm. In *ICPR*, pages 2219–2222, 2000.
- [PWCG01] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Research in Computational Molecular Biology*, pages 249–255, 2001.
- [QL05] Shibin Qiu and Terran Lane. Multiple kernel learning for support vector regression. Technical report, Computer Science Department, University of New Mexico, 2005.
- [QL09] Shibin Qiu and Terran Lane. A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 190–199, 2009.
- [QMC05] Arun Qamra, Yan Meng, and Edward Y. Chang. Enhanced perceptual distance functions and indexing for image replica recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):379–391, 2005.
- [QT09] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009.

REFERENCES

- [RBCG07] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 775–782, 2007.
- [RBCG08] Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 11:2491–2521, 2008.
- [RGZ⁺08] Rouhollah Rahmani, Sally A. Goldman, Hui Zhang, Sharath R. Cholleti, and Jason E. Fritts. Localized content-based image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1902–1912, 2008.
- [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.
- [RL09] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.
- [Rob81] John T. Robinson. The k-d-b-tree: A search structure for large multi-dimensional dynamic indexes. *SIGMOD*, pages 10–18, 1981.
- [RSBP11] Vincent Rapp, Thibaud Senechal, Kevin Bailly, and Lionel Prevost. Multiple kernel learning svm and statistical validation for facial landmark detection. In *FG*, pages 265–271, 2011.
- [SGR08] Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 1040–1047, 2008.
- [SGR10] Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. *Machine Learning*, 79(1-2):73–103, 2010.
- [SH75] M.I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 151–162, 1975.
- [SHS01] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *COLT/EuroCOLT*, pages 416–426, 2001.

REFERENCES

- [SJHL06] Luo Si, Rong Jin, Steven C.H. Hoi, and Michael R. Lyu. Collaborative image retrieval via regularized metric learning. *ACM Multimedia Systems Journal*, 12(1):34–44, 2006.
- [SRB07] Frédéric Suard, Alain Rakotomamonjy, and Abdelaziz Bensrhair. Model selection in pedestrian detection using multiple kernel learning. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 270–275. IEEE, 2007.
- [SRS05] Sören Sonnenburg, Gunnar Rätsch, and Christin Schäfer. Learning interpretable svms for biological sequence classification. In *RECOMB*, pages 389–407, 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 Alex Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [SS10] Niranjan Subrahmanya and Yung C. Shin. Sparse multiple kernel learning for signal processing applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:788–798, 2010.
- [SS12] Taiji Suzuki and Masashi Sugiyama. Fast learning rate of multiple kernel learning: Trade-off between sparsity and smoothness. In *AISTATS*, pages 1152–1183, 2012.
- [SSM97] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ICANN*, pages 583–588, 1997.
- [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.
- [STC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

REFERENCES

- [Sto74] M. Stone. Cross-Validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974.
- [Suz11] Taiji Suzuki. Unifying framework for fast learning rate of non-sparse multiple kernel learning. In *NIPS*, pages 1575–1583, 2011.
- [SWS⁺00] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. PAMI*, 22(12):1349–1380, 2000.
- [SWS05] E Sung, Lei Wang, and Eric Sung. A study of adaboost with svm based weak learners. *Proceedings 2005 IEEE International Joint Conference on Neural Networks 2005*, 1:196–201, 2005.
- [TC01] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM Press.
- [TCY09] Lei Tang, Jianhui Chen, and Jieping Ye. On multiple kernel learning with multiple labels. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1255–1260, 2009.
- [TmCK05] Ivor W. Tsang, Pak ming Cheung, and James T. Kwok. Kernel relevant component analysis for distance metric learning. In *In IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 954–959. IJCNN, 2005.
- [TS10] Ryota Tomioka and Taiji Suzuki. Sparsity-accuracy trade-off in mkl. arXiv:1001.2615, 2010.
- [TYSK09] Yufei Tao, Ke Yi, Cheng Sheng, and Panos Kalnis. Quality and efficiency in high dimensional nearest neighbor search. In *SIGMOD Conference*, pages 563–576, 2009.
- [Uhl91] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, 1991.

REFERENCES

- [Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [VB09] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 1065–1072, 2009.
- [VGVZ09] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *ICCV*, pages 606–613, 2009.
- [VR07] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1–8, 2007.
- [VsAV10] S. V. N. Vishwanathan, Zhaonan sun, Nawanol Ampornpunt, and Manik Varma. Multiple kernel learning and the smo algorithm. In *NIPS*, pages 2361–2369, 2010.
- [VSN08] Seyyed Majid Valiollahzadeh, Abolghasem Sayadiyan, and Mohammad Nazari. Face detection using adaboosted svm-based component classifier. *CoRR*, abs/0812.2575, 2008.
- [WBS06] Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 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.
- [WHJT12] Shuhui Wang, Qingming Huang, Shuqiang Jiang, and Qi Tian. Smkl: Scalable semi-supervised multiple kernel learning for real-world image applications. pages 1259–1274, 2012.
- [WJHT10a] Shuhui Wang, Shuqiang Jiang, Qingming Huang, and Qi Tian. S3mkl: scalable semi-supervised multiple kernel learning for image data mining. In *ACM Multimedia*, pages 163–172, 2010.

REFERENCES

- [WJHT10b] Shuhui Wang, Shuqiang Jiang, Qingming Huang, and Qi Tian. S3mkl: scalable semi-supervised multiple kernel learning for image data mining. In *ACM Multimedia*, pages 163–172, 2010.
- [WKC10] Jun Wang, Ondrej Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431, 2010.
- [WSB06] Mingrui Wu, Bernhard Schölkopf, and Gökhan H. Bakir. A direct method for building sparse kernel learning algorithms. *Journal of Machine Learning Research*, 7:603–624, 2006.
- [XH11] Hao Xia and Steven C. H. Hoi. Mkboost: A framework of multiple kernel boosting. In *SDM*, pages 199–210, 2011.
- [XH13] Hao Xia and Steven C. H. Hoi. Mkboost: A framework of multiple kernel boosting. *IEEE Trans. Knowl. Data Eng.*, pages 1574–1586, 2013.
- [XHJZ13] Hao Xia, Steven C. H. Hoi, Rong Jin, and Peilin Zhao. Online multiple kernel similarity learning for visual similarity search. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1–14, 2013.
- [XJKL08] Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems*, 2008.
- [XJY⁺09] Zenglin Xu, Rong Jin, Jieping Ye, Michael R. Lyu, and Irwin King. Non-monotonic feature selection. In *Proceedings of International Conference on Machine Learning*, pages 1145–1152, 2009.
- [XJY⁺10] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *International Conference on Machine Learning*, pages 1175–1182, 2010.
- [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.

REFERENCES

- [XNLS04] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, 2004.
- [XrLyZs10] Zhang Xiang-rong, Hu Long-ying, and Wang Zhi-sheng. Multiple kernel support vector regression for economic forecasting. In *Management Science and Engineering (ICMSE), 2010 International Conference on*, pages 129–134. IEEE, 2010.
- [XTX13] Xinxing Xu, Ivor W. Tsang, and Dong Xu. Soft margin multiple kernel learning. *IEEE Trans. Neural Netw. Learning Syst.*, 24(5):749–761, 2013.
- [XWHJ12] Hao Xia, Pengcheng Wu, Steven C. H. Hoi, and Rong Jin. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *SIGIR*, pages 55–64, 2012.
- [YC07] Dit-Yan Yeung and Hong Chang. A kernel approach for semisupervised metric learning. *IEEE Transactions on Neural Networks*, 18(1):141–149, 2007.
- [YCJ07] Jieping Ye, Jianhui Chen, and Shuiwang Ji. Discriminant kernel and regularization parameter learning via semidefinite programming. In *International Conference on Machine Learning*, pages 1095–1102, 2007.
- [YHC09] Yiming Ying, Kaizhu Huang, and Colin Campbell. Enhanced protein fold recognition through a novel data integration approach. *BMC Bioinformatics*, 10, 2009.
- [YHL11] Chi-Yuan Yeh, Chi-Wei Huang, and Shie-Jue Lee. A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38(3):2177–2186, 2011.
- [Yia93] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA’93)*, pages 311–321. ACM/SIAM, 1993.
- [YJC08] Jieping Ye, Shuiwang Ji, and Jianhui Chen. Multi-class discriminant kernel learning via convex programming. pages 719–758, 2008.

REFERENCES

- [YJM⁺10] Liu Yang, Rong Jin, Lily B. Mummert, Rahul Sukthankar, Adam Goode, Bin Zheng, Steven C. H. Hoi, and Mahadev Satyanarayanan. A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):30–44, 2010.
- [YKMT09] Fei Yan, Josef Kittler, Krystian Mikolajczyk, and Muhammad Atif Tahir. Non-sparse multiple kernel learning for fisher discriminant analysis. In *ICDM*, pages 1064–1069, 2009.
- [YKMT12] Fei Yan, Josef Kittler, Krystian Mikolajczyk, and Muhammad Atif Tahir. Non-sparse multiple kernel fisher discriminant analysis. pages 607–642, 2012.
- [YMB⁺10] Fei Yan, Krystian Mikolajczyk, Mark Barnard, Hongping Cai, and Josef Kittler. p norm multiple kernel fisher discriminant analysis for object and image categorisation. In *CVPR*, pages 3626–3632, 2010.
- [YMJ⁺12] Tianbao Yang, Mehrdad Mahdavi, Rong Jin, Lijun Zhang, and Yang Zhou. Multiple kernel learning from noisy labels by stochastic programming. In *ICML*, 2012.
- [YXY⁺11] Haiqin Yang, Zenglin Xu, Jieping Ye, Irwin King, and Michael R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, 2011.
- [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.
- [ZKGL04] Xiaojin Zhu, Jaz S. Kandola, Zoubin Ghahramani, and John D. Lafferty. Non-parametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2004.
- [ZKZ09] Bin Zhao, James T. Kwok, and Changshui Zhang. Multiple kernel clustering. In *SDM*, pages 638–649, 2009.

REFERENCES

- [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*, pages 113–122, 2011.
- [ZO07] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 1191–1198, Corvalis, Oregon, 2007.
- [ZR08] Xiaolong Zhang and Fang Ren. Improving svm learning accuracy with adaboost. *International Conference on Natural Computation*, 3:221–225, 2008.
- [ZTH09] Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. Simplenpk1: simple non-parametric kernel learning. In *Proceedings of International Conference on Machine Learning*, 2009.
- [ZWCL10] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25, 2010.