SOME RESEARCH ISSUES IN HASH FUNCTION LEARNING

by

YI ZHEN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

July 2012, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

YI ZHEN

SOME RESEARCH ISSUES IN HASH FUNCTION LEARNING

by

YI ZHEN

This is to certify that I have examined the above Ph.D. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

PROF. DIT-YAN YEUNG, THESIS SUPERVISOR

PROF. MOUNIR HAMDI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering
10 July 2012

ACKNOWLEDGMENTS

This thesis would never have been possible without the guidance of my supervisor, support of my family and help from my friends.

First and foremost, I am heartily thankful to my supervisor, Prof. Dit-Yan Yeung, for his excellent guidance, caring help, and meticulous comments on my research writing. As a real model, Prof. Yeung have shown me how to be a rigorous researcher, a passionate teacher, and a responsible person.

I would like to show my appreciation for the thesis committee members, Prof. Irwin King (from Department of Computer Science and Engineering, The Chinese University of Hong Kong), Prof. Weichuan Yu (from Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology), Prof. Nevin L. Zhang, Prof. James T. Kwok, and Prof. Dit-Yan Yeung, for their insightful comments and encouraging words. I am also grateful to Prof. Raymond Wong and Prof. Qiang Yang for their invaluable suggestions on my research.

I would like to thank the members of Prof. Yeung's research group, including Wu-Jun Li, Yang Ruan, Yu Zhang, Jingni Chen, Craig Yu, Tony Ho, Emprise Chan, Zhi-Yang Wang, Nai-Yan Wang, and Jie Zhou. Many thanks to the members of the Artificial Intelligence Lab, including Bin Cao, Nathan Liu, Evan Xiang, Vincent Zheng, Sinno Pan, Chung Lee, Tengfei Liu, Weike Pan, Ning Ding, and Haodi Zhang. Thanks to the visiting members of our lab, including Yan-Ming Zhang, Guoqiang Zhong, Deming Zhai, and Li Pu. In addition, I am grateful to the mentors during my visiting to Microsoft Research Asia, including Jian-Tao Sun and Gang Wang. The friendship, collaboration, and discussion with these friends have made my research and life not only meaningful but also colorful.

Last but not least, I would like to thank my family for supporting and encouraging me throughout my study in Hong Kong. Especially, I am grateful to my wife, Ning Zhu, who has always been right behind me whenever and wherever. I would also like to thank my daughter, Joy Zhen, for her sweet smile which brings happiness and delight to the whole family. My love for all of you is beyond words.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SOME RESEARCH ISSUES IN HASH FUNCTION LEARNING

by

YI ZHEN

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Over the past decade, hashing-based methods for large-scale similarity search have sparked considerable research interest in the database, data mining and information retrieval communities. These methods achieve very fast search speed by indexing data with binary codes. Although lots of hash functions for various similarity metrics have been proposed, they often generate very long codes due to their data independence nature. In recent years, machine learning techniques have been applied to *learn* hash functions from data, forming a new research topic called *hash function learning*.

In this thesis, we study two important issues in hash function learning. On one hand, existing supervised or semi-supervised hash function learning methods, which learn hash functions from labeled data, can be regarded to be passive because they assume that the labeled data are provided in advance. Given that the data labeling process can be very costly in practice and the contribution of labeled data to hash function learning can be quite different, it may be more cost effective for the hash function learning methods to select labeled data from which to learn. To this end, we propose a novel framework, termed *active hashing*, to actively select the most informative data to label for hash function learning. Under the framework, we develop one simple method which queries data labels that the current hash functions are most uncertain about. Experiments conducted on two real data sets show obvious improvement of our active hashing algorithm over previous passive hashing methods. On the other hand, most existing hash function learning methods only work on unimodal data, which are obviously not the case in many applications, e.g., multimedia retrieval and cross-lingual document analysis. To apply

hash function learning to multimodal data, we develop three methods under the framework of *multimodal hashing* which hashes data points of multiple modalities into one common Hamming space. For aligned data, the first method is based on spectral analysis of the correlation of the multimodal data. For graph data, the second method falls into the category of latent feature models and the hash codes can be obtained through Bayesian inference. For general data, we propose a boosted co-regularization model which can be efficiently solved by stochastic gradient-based algorithms. The effectiveness of our models is validated through extensive comparative study on crossmodal multimedia retrieval.

CHAPTER 1

INTRODUCTION

Similarity search, *a.k.a.* proximity search or nearest neighbor search, plays a fundamental role in many data mining or pattern recognition problems, e.g., document retrieval, text categorization, near-duplicate detection and collaborative filtering (Yianilos, 1993; Shakhnarovich et al., 2006). Briefly speaking, similarity search aims to find the nearest neighbors, measured by some similarity measures, of a given query document.[1] Obviously, computing the similarity exhaustively between the query and every candidate document does not work for large data sets due to the prohibitive computation and storage costs.

To address the scalability issue, a number of *approximate* search techniques for similarity search have been proposed (Arya et al., 1998; Friedman et al., 1977; Andoni & Indyk, 2006b; Shakhnarovich, 2005). These methods can achieve sublinear search time at the cost of sacrificing accuracy in an acceptable sense. There are generally two approaches, namely, tree-based methods (Arya et al., 1998; Friedman et al., 1977; Weber et al., 1998) and hashing-based methods (Andoni & Indyk, 2006b; Shakhnarovich, 2005). Although tree-based methods work well on low-dimensional data, they often give no speedup on even slightly higher-dimensional data and also have very high storage demand. These two limitations make tree-based methods unappealing for applications in which high dimensionality is commonly encountered. On the contrary, hashing-based methods can perform very fast search and use substantially reduced storage by indexing data with compact binary codes. As such, considerable research effort has been spent on hashing-based approaches over the past several years. For example, lots of hash functions have been proposed for many similarity measures and successfully applied to many tasks. Most of these methods are based on random projections or permutations, and can achieve accuracy comparable with exact search asymptotically. However, they often generate very long codes mainly due to their data independence nature.

Recently, some researchers (Salakhutdinov & Hinton, 2009b; Weiss et al., 2008; Kulis & Darrell, 2009; Wang et al., 2010b; Mu et al., 2010) have made various attempts to use machine learning techniques to learn hash functions from data. The working hypothesis underlying these methods is that data-dependent hash functions learned from data can reflect the data characteristics more accurately than data-independent hash functions such as those in previous works based on random projections or permutations. In this

[1]The term 'document' is used here in a broad sense to refer to data represented in any form, including text, image, video or their combinations.

thesis, we regard learning hash functions from data as a new research topic and call it *hash function learning* (HFL). Since hashing-based methods have been applied in a vast range of areas, HFL may have great potential impact, both on the theory side and the application side, in the near future. Our focus in this work will be HFL.

According to whether or not machine learning techniques are used, we divide existing hashing-based methods into two classes: locality sensitive hashing (LSH) and hash function learning. As stated earlier, hash functions of LSH are always based on random projections or permutations, but those of HFL are learned from data. There are several perspectives from which to look into existing HFL approaches. Depending on how labeled data or side information (*a.k.a.* supervision) is used in the learning procedure, we can classify HFL methods into three categories: unsupervised methods, semi-supervised methods and supervised methods. In unsupervised HFL, only unlabeled data are used, while in supervised HFL, only labeled data are used. In semi-supervised HFL, both unlabeled and labeled data are used together to learn the hash functions. According to whether or not the learner can choose the data from which it learns, HFL methods can be classified as passive hashing methods or active hashing methods. Passive hashing learns hash functions from data provided in advance, but active hashing chooses the data from which to learn. According to the characteristics of the data being processed, HFL methods can be grouped into unimodal methods or multimodal methods. While unimodal hashing deals with data of only one modality, multimodal hashing can process data of multiple modalities. To make it clearer, we summarize the categorization of hashing-based approaches in Figure 1.1.

Though HFL methods have obtained great successes in many applications recently, research on this topic is still young and far from complete. In this thesis, we focus on two research issues, that is, *active hashing* and *multimodal hashing*, which have not been well studied to date. In the following content, we first give the motivations of these two research topics and then summarize the main contributions. After that, we outline the organization of the whole thesis and conclude this chapter with a summary of notations which are used throughout the thesis.

1.1 Motivations

1.1.1 Motivation of Active Hashing

Existing supervised or semi-supervised HFL methods can be regarded as *passive hashing*, since they assume that side-information or data labels are provided before learning. Given that the data labeling procedure can be very time costly and the quality of labeled data can be very diverse, it is better to actively choose data to label before learning from them.

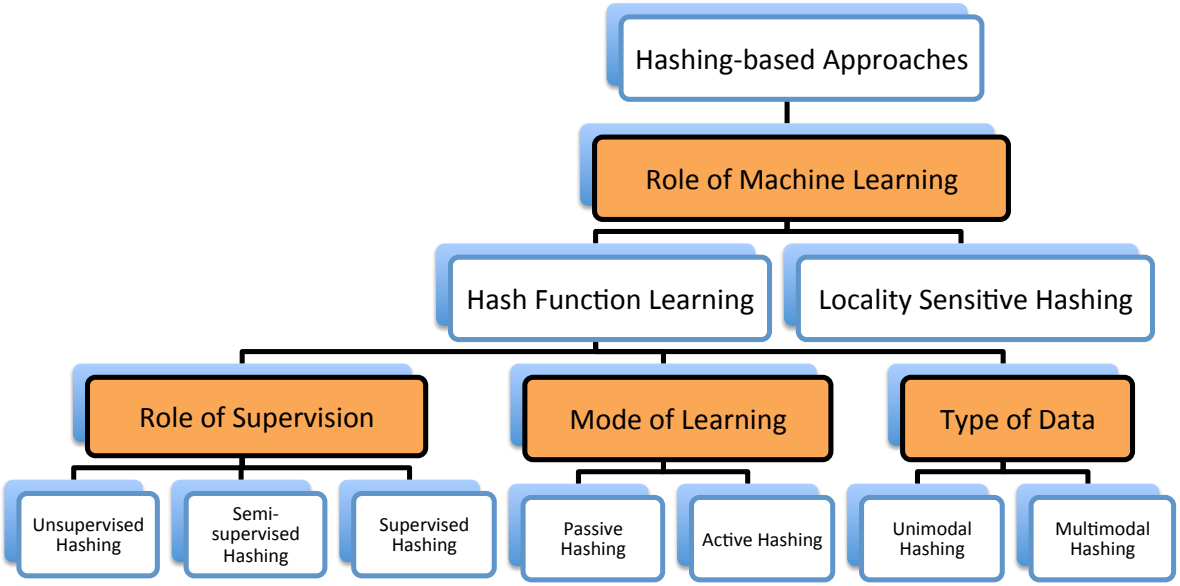


Figure 1.1: The categorization of hashing-based methods for similarity search. The orange ovals indicate the categorization criteria and the blue rectangles indicate the category names.

To this end, we propose a novel framework, called *active hashing*, to actively select the most informative unlabeled data to label for hash function learning.

1.1.2 Motivation of Multimodal Hashing

As mentioned earlier, HFL methods can be classified into unimodal methods or multimodal methods, depending on the characteristics of the data. Most previous research effort has been spent on unimodal HFL approaches, in which data points are assumed to be of a single modality. This assumption is obviously not true in many applications such as multimedia retrieval, where data are often generated from multiple modalities, for example, text, image, audio, video, and so on. To perform hashing on such multimodal data, we propose three models under the framework of *multimodal hashing* with the goal of hashing data points of different modalities into a single Hamming space, and hence multimodal similarity search can be easily performed.

1.2 Contributions

Our major contributions are:

- For active hashing, we propose a novel framework to make existing HFL methods cost effective. A simple method based on data uncertainty is developed. Experi-

mental results show that our method outperforms previous passive hashing methods by a large margin.

- For multimodal hashing, three models are proposed. For aligned data, in which different modalities are aligned, we present spectral multimodal hashing. For graph data, in which pairwise similarity values are available, we present a multimodal latent binary embedding model which comes under the umbrella of latent feature models. To handle general data, we develop a model called co-regularized hashing based on boosted co-regularization. Extensive experiments demonstrate that these methods compare favorably with the baselines.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 introduces some background knowledge, including a brief literature review of hashing-based methods for similarity search. Additionally, two related machine learning areas, namely, metric learning and active learning, are reviewed and their relationships with hash function learning are discussed.

Chapter 3 presents the active hashing framework and a batch mode active hashing algorithm.

Chapter 4 presents one multimodal hashing method for aligned data, called spectral multimodal hashing.

Chapter 5 presents a multimodal hashing model for graph data, termed multimodal latent binary embedding.

Chapter 6 presents the multimodal hashing methods for general data, called co-regularized hashing.

Chapter 7 concludes the whole thesis and proposes several research issues for future pursuit.

1.4 Notations

Some common notations used in the thesis are summarized in Table 1.1.

Table 1.1: Table of Notations

Notation	Physical Meaning
$\mathbf{a} \in \mathbb{R}^d$	a d-dimensional (column) vector denoted by a boldface lowercase letter
$a(i)$	the ith element of a vector $\mathbf{a}$
$\mathbf{1}$	a vector of all 1's with appropriate size
$\mathbf{0}$	a vector of all 0's with appropriate size
$\mathbf{A} \in \mathbb{R}^{m \times n}$	a matrix of size $m \times n$ denoted by a boldface uppercase letter
$A(i,j)$	the (i,j)th element of a matrix $\mathbf{A}$
A_{ij}	the (i,j)th element of a matrix $\mathbf{A}$
$\mathbf{I}$	an identity matrix with appropriate size
$\mathcal{A}$	a set denoted by a calligraphic uppercase letter
$\mathrm{tr}(\mathbf{A})$	trace of a matrix $\mathbf{A}$
$\mathbf{A}^T$	transpose of a matrix $\mathbf{A}$
$\lvert\mathcal{A}\rvert$	size of a set $\mathcal{A}$
$\mathrm{sgn}(a)$	the sign of a real number a
$\lVert\mathbf{x} - \mathbf{y}\rVert_H$	Hamming distance between $\mathbf{x}$ and $\mathbf{y}$

CHAPTER 2

BACKGROUND

The history of hashing may be as long as the history of computer science. Hashing methods were originally used to implement dynamic sets that support only the dictionary operations such as insert, search, and delete (Cormen et al., 2001). Conventionally, people design hash functions which map objects to bins and then construct hash tables. With the hash tables, fast dictionary operations can be performed. For example, the expected time complexity of searching an element in a hash table is only $O(1)$, although it can be $\Theta(N)$ in the worst case. A good hash table should have a low collision rate and use as little storage as possible.

In recent years, hashing has been introduced in several new problems such as similarity search (Gionis et al., 1999; Salakhutdinov & Hinton, 2009b) and data compression (Shi et al., 2009; Weinberger et al., 2009; Li et al., 2011). In this thesis, our focus is hashing for similarity search. Unlike conventional hashing methods, hashing-based similarity search methods use collision to incorporate similarity. These methods are common in indexing objects using binary codes, making search extremely fast (without much loss of accuracy) even on very large data sets.

Depending on whether or not machine learning techniques are applied to design the hash functions, existing hashing-based methods can be grouped into two categories: locality sensitive hashing (*a.k.a.* non-learning based hashing) and hash function learning (*a.k.a.* learning based hashing). While locality sensitive hashing methods have enjoyed great success over past decades, most of them are data independent and often generate very long hash codes which are practically inefficient for large-scale applications. On the contrary, hash function learning methods learn from data hash functions which reflect data characteristics more accurately and generate very compact codes. As a result, HFL methods are more desirable in real applications.

To provide a solid background for this thesis, in this chapter, we review the literature of hashing-based methods for similarity search, and two machine learning areas, namely, active learning and metric learning, which are closely related to the methods we introduce later. Specifically, in Section 2.1, we give a brief introduction of locality sensitive hashing, which is then followed by a survey of hash function learning in Section 2.2. Section 2.3 and Section 2.4 present a summarization of metric learning and active learning, separately. Finally, we summarize the chapter in Section 2.5.

2.1 Locality Sensitive Hashing

The family of *locality sensitive hashing* (LSH) algorithms have been a hot research topic since the 1990s. In this section, we briefly introduce some well-known methods. For a detailed review, the readers are referred to (Andoni & Indyk, 2006b).

The concept of LSH is very simple, that is, it aims to hash objects[1] using a number of hash functions to ensure that, for each function, the collision probability for objects that are close to each other is much higher than that for far apart objects (Indyk & Motwani, 1998). In other words, given a family of hash functions $\mathcal{H}$ and two objects $\mathbf{p}$ and $\mathbf{q}$, LSH (Charikar, 2002) requires that

$$P_{\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q})] = sim(\mathbf{p}, \mathbf{q}),$$

where h is a hash function randomly selected from $\mathcal{H}$, $P(\cdot)$ indicates the probability and $sim(\cdot, \cdot)$ returns a similarity value in the range of $[0, 1]$ based on some metric. Usually, h has only two hash values $\{0, 1\}$ and LSH uses several hash functions to generate binary codes. Therefore, LSH actually maps data objects from the original feature space to a Hamming space where the Hamming distance incorporates the similarity. It is very efficient to perform the search in the Hamming space via bit operations or data structures such as dictionaries.

LSH functions are highly dependent on the similarity measures used in the task. Up to now, many LSH functions have been proposed. In the following subsections, we introduce them separately according to the adopted similarity measures.

2.1.1 ℓ_s Distance

The most commonly used distance might be the Euclidean distance (*a.k.a.* ℓ_2 distance). In (Datar et al., 2004), LSH functions for the Euclidean distance can be constructed as follows: choose a random number w and a random projection vector $\mathbf{r} \in \mathbb{R}^D$ (each of whose coordinates is picked from a Gaussian distribution), then $h_{\mathbf{r},b}(\mathbf{p}) = \lfloor (\mathbf{r} \cdot \mathbf{p} + b)/w \rfloor$ where $b \in [0, w)$ is a random threshold and $\mathbf{p} \in \mathbb{R}^D$ is an object to be hashed. There is a publicly available software package which has implemented this algorithm.[2] Recently, Dasgupta *et al.* (Dasgupta et al., 2011) further improved the speed of the hash function construction of LSH for Euclidean distance using the Hadamard matrix.

Another common distance is ℓ_1 distance, for which the LSH hash functions are defined as (Andoni & Indyk, 2006a): for a fixed real number $w \gg R$,[3] pick random real numbers

[1] In this paper, we use objects, points and documents interchangeably.

[2] `http://www.mit.edu/~andoni/LSH/`

[3] R is a user specified number controlling the distance between points which are considered as neighbors.

$s_1, \ldots, s_D \in [0, w)$ and define $h_{s_1,\ldots,s_D}(\mathbf{p}) = (\lfloor (p_1 - s_1)/w \rfloor, \ldots, \lfloor (p_D - s_D)/w \rfloor), \mathbf{p} \in \mathbb{R}^D$.

Constructions of LSH functions for ℓ_s distance for any $s \in (0, 2]$ are also possible (Datar et al., 2004).

2.1.2 Cosine Similarity

In the data mining and information retrieval communities, especially for documents, the most commonly used metric is cosine similarity. Its value between two vectors $\mathbf{p}$ and $\mathbf{q}$ is defined as:

$$\Theta(\mathbf{p}, \mathbf{q}) = \arccos\left(\frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\| \cdot \|\mathbf{q}\|}\right), \mathbf{p}, \mathbf{q} \in \mathbb{R}^D,$$

where $\|\cdot\|$ denotes the vector ℓ_2 norm. For this similarity measure, Charikar *et al.* (Charikar, 2002) defines the following LSH family: pick a random unit vector $\mathbf{u}$ and define $h_{\mathbf{u}}(\mathbf{p}) = \text{sgn}(\mathbf{u} \cdot \mathbf{p})$. The hash function can also be viewed as partitioning the space into two half-spaces by a random hyperplane. The collision probability in this case is $P[h(\mathbf{p}) = h(\mathbf{q})] = 1 - \Theta(\mathbf{p}, \mathbf{q})/\pi$.

Manku *et al.* (Manku et al., 2007) applied the above mentioned hash functions with some modifications to document de-duplication and named the algorithm SimHash. Since then, SimHash has become one of the most well-known hashing-based methods. SimHash has also been implemented under the framework of MapReduce for cross-lingual pairwise similarity (Ture et al., 2011). Eshghi *et al.* (Eshghi & Rajaram, 2008) developed a new LSH family for cosine similarity based on concomitant rank order statistics.

2.1.3 Jaccard Similarity

Jaccard coefficient is widely used to measure similarity between sets. The definition of Jaccard Similarity is: $s(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$, which can be approximated by a simple approach based on random permutations (Broder, 1997; Broder et al., 1997). Based on this property, Broder *et al.* proposed the first LSH family for Jaccard similarity which is called minwise Hashing or MinHash (Broder et al., 1998). The hash functions of MinHash can be described simply as follows: $h_\pi(\mathcal{A}) = \min\{\pi(a) \mid a \in \mathcal{A}\}$, where π is a random permutation on the ground universe.

Recently, Li *et al.* have extended MinHash to generate more compact codes (Li & König, 2010; Li et al., 2010). Chum *et al.* (Chum et al., 2009) have proposed a geometric MinHash algorithm and applied it to the image retrieval task.

2.1.4 Kernel Similarity

In many applications, the similarity metric of interest is defined by means of kernels. As a result, developing hashing-based methods for kernel similarity is also very important. For the Pyrimid match kernel, a widely used kernel in vision problems, Pyrimid match hashing (Grauman & Darrell, 2007) was developed. The algorithm can be simply summarized as first generating an embedding of the Pyrimid match kernel, and then using the SimHash to obtain the binary codes for the embedding. Jain *et al.* (Jain et al., 2008) proposed a similar hashing algorithm for a learned Mahalanobis distance, which is equivalent to a special kernel. More generally, for shift invariant kernels, hash functions can be constructed as follows (Raginsky & Lazebnik, 2009). First get random features using Fourier transforms (Rahimi & Recht, 2007) and then threshold them randomly to give the binary codes.

Kulis *et al.* (Kulis & Grauman, 2009) implemented the idea of random projections in the kernel space and formulated KLSH. In general, KLSH uses a random set of points to form random projections and can accommodate any kernels. Recently, KLSH was extended to accept multiple kernels in (Zhang et al., 2011).

2.1.5 Summary

Besides the aforementioned algorithms, LSH functions have also been extended for some special settings, such as asymmetric Hamming distance (Dong et al., 2008; Gordo & Perronnin, 2011) and non-metric similarity (Athitsos et al., 2008; Mu & Yan, 2010).

Although theoretically effective and efficient, LSH methods have some apparent limitations. Most of all, they are data independent and thus may not reflect the data characteristics accurately. As a result, these methods always generate very long codes which are very inefficient for large-scale data sets and hence of limited practical use. It is this limitation that motivates hash function learning.

2.2 Hash Function Learning

The past five years have witnessed increasingly rapid progress in the topic of hash function learning (HFL), or learning-based hashing. The goal of HFL is to learn, rather than design, the hash functions from data so that the hash functions can generate very compact (short) hash codes. Despite being a young area, a number of HFL methods have been proposed up to now. Depending on how the label information or side-information is used in the learning procedures, we roughly classify these methods into three categories: unsupervised, semi-supervised and supervised methods.

2.2.1 Unsupervised Hash Function Learning

Unsupervised HFL methods learn hash functions from unlabeled data only and do not use labels or side-information. Although the model formulations are different from each other, the common idea is to make similar points close to each other and dissimilar points far apart in the Hamming space in which the hash codes live.[4]

Spectral hashing

The most well-known unsupervised HFL algorithm should be *spectral hashing* (SH). The objective of SH is similar to one popular dimensionality reduction method called *Laplacian Eigenmap*, meaning that points that are similar in the original feature space should have small distance in the embedded space. However, different from *Laplacian* Eigenmap which embeds data into the Euclidean space, SH maps data into the Hamming space. For a code to be efficient, the authors require that each bit has a 50% chance of being one or zero, and that different bits are independent of each other.

Putting all things together, SH is formulated as the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{y}_i} \sum_{ij} W(i,j)\|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ &\text{s.t. } \mathbf{y}_i \in \{+1,-1\}^M,\ \sum\nolimits_i^N \mathbf{y}_i = 0,\ \frac{1}{n}\sum\nolimits_i^N \mathbf{y}_i\mathbf{y}_i^T = \mathbf{I}, \end{aligned}$$

where $\mathbf{y}_i$ is the hash code of the ith point, $W(i,j)$ is the similarity between the ith and jth points, M is the length of each code and N is the number of points. Note that the authors relax the independence assumption and require the bits to be uncorrelated by constraint $\frac{1}{n}\sum_i \mathbf{y}_i\mathbf{y}_i^T = \mathbf{I}$, because it is hard to enforce independence between bits. Then above problem can be rewritten using matrix notations as follows:

$$\begin{aligned} &\min_{\mathbf{Y}} \operatorname{tr}(\mathbf{Y}^T\mathcal{L}\mathbf{Y}) \\ &\text{s.t. } Y_{ij} \in \{+1,-1\}^M, \mathbf{Y}^T\mathbf{1} = \mathbf{0}, \mathbf{Y}^T\mathbf{Y} = \mathbf{I}, \end{aligned} \tag{2.1}$$

where $\mathbf{Y}$ is the $N \times M$ code matrix, $\mathcal{L}$ is the graph *Laplacian* defined on the similarity matrix $\mathbf{W}$.

Actually, the above problem for each single bit is equivalent to a balanced graph partitioning problem and NP-hard, so Problem (2.1) is NP-hard. However, if we remove the first two constraints, the problem can easily be solved by finding K eigenvectors corresponding to the smallest eigenvalues of $\mathcal{L}$ (ignoring the eigenvector $\mathbf{1}$ corresponding to eigenvalue 0).

[4]Note that "similar" and "dissimilar" are determined by features of unlabeled data.

The solution above has two limitations. First, it involves eigen-decomposition which could be very slow when N is large. Moreover, it cannot deal with out-of-sample data points. To overcome these two limitations, the authors use eigenfunctions instead of the eigenvectors. To make the computation tractable, they simply assume that the data points are uniformly distributed in a rectangle and the similarity measure is fixed as $\exp(\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2)$. The resultant model is very fast to train and experimental results show that it is superior to LSH.

Recently, He *et al.* (He et al., 2010) proposed a kernel extension of SH, which can handle nonvectorial data, incorporate nonlinearity and places no restrictions on the uniform distribution and the fixed similarity measure. Zhang *et al.* (Zhang et al., 2010b, 2010a) used SVM to enhance SH with the aim of removing the distribution and similarity assumptions.

Binary reconstructive embeddings

Kulis *et al.* (Kulis & Darrell, 2009) developed another unsupervised HFL algorithm, namely, *binary reconstructive embeddings* (BRE), based on explicitly minimizing the reconstruction error between the original distance and the Hamming distance of the corresponding binary hash codes. BRE can be easily kernelized and does not require restrictive assumptions about the underlining data distribution.

Specifically, let M be the number of hash functions (*a.k.a.* code length), N be the number of data points, and Q be the number of landmark points. Given a data set $\mathcal{X}$, BRE defines the hash function *w.r.t.* (with respect to) the mth bit for $\mathbf{x} \in \mathcal{X}$ as:

$$h_m(\mathbf{x}) = \operatorname{sgn}\left(\sum\nolimits_{q=1}^{Q} W(m,q)\kappa(\mathbf{x}_q, \mathbf{x})\right),$$

where $\mathbf{W}$ is an $M \times Q$ projection matrices, $\{\mathbf{x}_q\}_{q=1}^{Q} \subset \mathcal{X}$ are landmark points, and $\kappa(\cdot,\cdot)$ is a user specified kernel function. Note that defining hash functions this way is very common in kernel methods such as *support vector machines* (SVM) and gives us the flexibility to work on a wide variety of data types. Therefore, given one point $\mathbf{x} \in \mathcal{X}$, we denote its corresponding binary representation as $\tilde{\mathbf{x}}$ such that its mth bit can be evaluated by $\tilde{x}(m) = (1 + h_m(\mathbf{x}))/2$.

Rather than simply choosing the $\mathbf{W}$ matrix based on random hyperplanes, they construct this matrix to achieve good reconstructions. In particular, they minimize the squared error between the original distance and the reconstructed distance *w.r.t.* $\mathbf{W}$ as follows,

$$\mathcal{O}(\mathbf{W}) = \sum\nolimits_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{N}} \left(d(\mathbf{x}_i,\mathbf{x}_j) - \tilde{d}(\mathbf{x}_i,\mathbf{x}_j)\right)^2,$$

where $\mathcal{N}$ is a set of point pairs, $d(\cdot,\cdot)$ is the original distance, $\tilde{d}(\cdot,\cdot)$ is the Hamming distance. Although the problem is non-convex and hard to optimize, the authors use a heuristic coordinate-descent algorithm to find a locally optimal $\mathbf{W}$. Experiments show that BRE achieves a state-of-the-art performance, but finding a good local optimum is hard and the performance highly depends on the applications at hand.

2.2.2 Semi-Supervised Hash Function Learning

In semi-supervised HFL methods, both unlabeled and labeled data are used. Different from original unlabeled features, the labels or side-information which often carries semantic information might be very useful for hash function learning. The basic idea of semi-supervised HFL is to consider both kinds of information to learn the hash functions.

Semantic hashing

To learn the hash codes, semantic hashing uses multiple layers of *restricted Boltzmann machines* (RBM), which are closely related to one popular dimensionality reduction framework based on neural networks (Hinton & Salakhutdinov, 2006). An RBM is an ensemble of binary vectors with a network of stochastic binary units arranged in two layers, one visible and one hidden. Given a layer of visible units $\mathbf{v} = [v_1, v_2, \ldots, v_M]$, a layer of hidden units $\mathbf{h} = [h_1, h_2, \ldots, h_N]$, and a symmetric weighting matrix $\mathbf{W}$ connecting units in different layers, the energy function of the joint configuration of all visible and hidden units is defined as:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{M} b_i v_i - \sum_{j=1}^{N} b_j h_j - \sum_{i=1}^{M}\sum_{j=1}^{N} v_i h_i W(i,j), \tag{2.2}$$

where v_i and h_j are the binary states of visible and hidden units i and j, $W(i,j)$ are the weights and b_i and b_j are bias terms. Using this energy function, a probability can be assigned to a binary vector of the visible units:

$$P(\mathbf{v}) = \sum_{\mathbf{h}} \left(e^{-E(\mathbf{v},\mathbf{h})} \Big/ \left(\sum_{\mathbf{u},\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} \right) \right).$$

An RBM lacks connections between units within a layer, hence the conditional distributions $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}|\mathbf{h})$ have convenient forms, being products of Bernoulli distributions:

$$P(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i w_{ij} v_i\right), \quad P(v_i = 1|\mathbf{h}) = \sigma\left(b_i + \sum_j w_{ij} h_j\right),$$

where $\sigma(x) = 1/(1+e^{-x})$ is the logistic sigmoid function.

Recently, Salakhutdinov *et al.* (Salakhutdinov & Hinton, 2009a) demonstrated methods for stacking RBM into multiple layers, creating "deep networks" which can capture

high order correlations between the visible units at the bottom layer of the network. By choosing an architecture that progressively reduces the number of units in each layer, a high dimensional binary input vector can be mapped to a far smaller binary vector at the output. Thus, at the output each bit maps through multiple layers of nonlinearities to model the complicated subspace of the input data. If the feature values are not binary but real numbers, the first layer of visible units are modified to have a Gaussian distribution. This type of trained networks are capable of capturing higher order correlations between different layers of the network. Since the network structure gradually reduces the number of units in each layer, the high-dimensional input can be projected to a much more compact binary vector space.

A practical implementation of RBM has two major stages, an unsupervised pre-training stage and a supervised fine-tuning stage. The greedy pre-training stage is progressively executed layer by layer from input to output. After achieving convergence of the parameters of a layer via contrastive divergence, the derived activation probabilities are fixed and treated as input to drive the training of the next layer. During the fine-tuning stage, the labeled data is used to help refine the trained network through back-propagation. Specifically, a cost function is first defined to estimate the number of correctly classified points in the training set. Then, the network weights are refined to maximize this objective function through gradient descent.

Semantic hashing outperforms LSH methods in applications such as document retrieval, but it involves estimating a large number of weights. As such, it not only involves an extremely costly training procedure, but also demands sufficient labeled training data for fine-tuning.

Semi-supervised hashing

Rather than use unlabeled and labeled data in two separate stages like semantic hashing, Wang *et al.* (Wang et al., 2010a) have developed a simple semi-supervised hashing (SSH) method which uses both kinds of information in a unified framework.

Given a set of N data points, $\mathbf{X} \in \mathbb{R}^{N \times D}$, and the ith row of $\mathbf{X}$ corresponds to a D-dimensional point that will be denoted by $\mathbf{x}_i$ in the sequel, the authors want to learn M hash functions leading to a M-bit binary codes $\mathbf{Y} \in \{0,1\}^{N \times M}$ of $\mathbf{X}$. In the paper, the mth hash function is defined as,

$$h_m(\mathbf{x}) = \text{sgn}\left(\mathbf{w}_m^T \mathbf{x} + b_m\right),$$

where $\mathbf{w}$ is a D-dimensional projection vector and b_m is a scaler.[5] Without loss of gen-

[5]Note that one bit of binary code can be easily obtained by setting $y_m(\mathbf{x}) = (1 + h_m(\mathbf{x}))/2$.

erality, we assume that $\mathbf{X}$ has zero-mean ($\sum_{i=1}^{N} \mathbf{x}_i = \mathbf{0}$),[6] then the hash function can be simplified as,

$$h_m(\mathbf{x}) = \text{sgn}(\mathbf{w}_m^T \mathbf{x}).$$

Based on the labels or side-information, the authors construct a set of similar point pairs $\mathcal{S}$ and a set of dissimilar pairs $\mathcal{D}$. We use $\mathcal{SD}$ to denote the set of points involved in $\mathcal{S}$ and $\mathcal{D}$, and $\mathbf{X}_{\mathcal{SD}} \in \mathbb{R}^{l \times D}$ to denote its matrix form, where $l = |\mathcal{SD}| < N$.

In SSH, the hashing functions $\mathcal{H} = \{h_m\}_{m=1}^{M}$ can be learned by optimizing two objectives. The first objective is to maximize the empirical accuracy on $\mathcal{S}$ and $\mathcal{D}$, which is defined as the sum of difference of the total number of correctly classified pairs and that of wrongly classified pairs by each bit as follows:

$$J(\mathcal{H}) = \sum\nolimits_{m=1}^{M} \left(\sum\nolimits_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} h_m(\mathbf{x}_i) h_m(\mathbf{x}_j) - \sum\nolimits_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} h_m(\mathbf{x}_i) h_m(\mathbf{x}_j) \right).$$

Optimizing the above objective function is difficult, since the hash functions $\{h_m\}_{m=1}^{M}$ are discrete and not differentiable. As such, the $\text{sgn}(\cdot)$ operator is dropped and the following approximate objective is used instead,

$$\begin{aligned} J(\mathbf{W}) &= \sum\nolimits_{m=1}^{M} \left(\sum\nolimits_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} \mathbf{w}_m^T \mathbf{x}_i \mathbf{x}_j^T \mathbf{w}_m - \sum\nolimits_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \mathbf{w}_m^T \mathbf{x}_i \mathbf{x}_j^T \mathbf{w}_m \right) \\ &= \frac{1}{2} \text{tr} \left(\mathbf{W}^T \mathbf{X}_{\mathcal{SD}}^T \mathbf{S} \mathbf{X}_{\mathcal{SD}} \mathbf{W} \right), \end{aligned}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M] \in \mathbb{R}^{D \times M}$ and $\mathbf{S} \in \mathbb{R}^{l \times l}$ is a relational matrix incorporating pairwise relations,

$$S(i,j) = \begin{cases} +1 & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ -1 & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases}.$$

The other objective is to maximize the information conveyed by each bit of hash codes. From an information-theoretic point of view, a binary bit, which gives a balanced partition (*maximal entropy partition*) of $\mathbf{X}$ ($\sum_{i=1}^{n} h(\mathbf{x}_i) = 0$), provides the maximum information. Although finding mean-thresholded hash functions that meet the balancing requirement is NP-hard (Weiss et al., 2008), it is proved that this objective is equivalent to maximizing the variance of a bit (Wang et al., 2010a), which is lower-bounded by the scaled variance of the projected data. Thus the scaled variance of the projected data is used as the second objective,

$$R(\mathbf{W}) = \rho \sum\nolimits_{k=1}^{K} \mathbf{w}_m^T \mathbf{X}^T \mathbf{X} \mathbf{w}_m,$$

[6] If this is not the case, preprocessing can be applied on the data.

where ρ is a scaling parameter.

Combining the two objectives into one, the optimization problem of SSH is formulated as follows:

$$\begin{aligned} \max_{\mathbf{W}} \ & \frac{1}{2}\mathrm{tr}\left(\mathbf{W}^T\left(\mathbf{X}_{\mathcal{SD}}^T\mathbf{S}\mathbf{X}_{\mathcal{SD}} + \rho\mathbf{X}^T\mathbf{X}\right)\mathbf{W}\right) \\ \text{s.t. } & \mathbf{W}^T\mathbf{W} = \mathbf{I}, \end{aligned} \tag{2.3}$$

where the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ is commonly used to incorporate the uncorrelatedness of the projection directions (Weiss et al., 2008). Obviously, Problem (2.3) can be solved by spectral decomposition of the matrix $\mathbf{X}_{\mathcal{SD}}^T\mathbf{S}\mathbf{X}_{\mathcal{SD}} + \rho\mathbf{X}^T\mathbf{X}$ and $\mathbf{W}$ is actually the eigenvectors corresponding to the largest eigenvalues.

Wang *et al.* (Wang et al., 2010b) argue that it is better to consider the dependence between contiguous bits rather than treat them independently. To this end, they propose a sequential learning algorithm which obtains one bit of codes at a time and updates the relational matrix $\mathbf{S}$ before learning the next bit. Experimental studies show that the sequential algorithm achieves much better performance than semantic hashing.

Label-regularized max-margin partition

Mu *et al.* have taken the concept of margin into consideration to propose a semi-supervised HFL method termed *label-regularized max-margin partition* (LAMP) (Mu et al., 2010). Considering each hash function a binary classifier, the authors argue that larger margin between hash-induced partitions usually indicates better generalization ability to out-of-sample data.

For each hash function, the optimization problem of LAMP is formulated as follows:

$$\begin{aligned} \min_{\omega,b,\xi,\zeta,y} \ & \frac{1}{2}\|\omega\|^2 + \frac{\lambda_1}{N}\sum_i \xi_i + \frac{\lambda_2}{N}\sum_{(i,j)\in\Theta}\zeta_{ij} \\ \text{s.t. } & y_i(\omega^T\mathbf{x}_i + b) + \xi_i \geq 1, \xi_i \geq 0, \forall\ i, \\ & y_y y_j + \zeta_{ij} \geq 0, \zeta_{ij}a \geq 0, \forall\ (ij) \in \Theta, \end{aligned}$$

where ω and b are the parameters of the hash function, y is the label vector induced by the hash function, Θ denotes the set of constraints which is generated from label or side information. Since y_i is always set to be $\mathrm{sgn}(\omega^T\mathbf{x}_i + b)$ in hashing-based methods and ω can be represented by a linear combination of random landmark vectors using the

kernel-trick, the modified formulation of the problem is used:

$$\begin{aligned} \min_{\nu,b,\xi,\zeta} \ & \frac{1}{2}\nu^T \mathbf{G}\nu + \frac{\lambda_1}{N}\sum_i \xi_i + \frac{\lambda_2}{N}\sum_{(i,j)\in\Theta} \zeta_{ij} \\ \text{s.t. } & |\nu^T \mathbf{k}_i + b) + \xi_i \geq 1, \xi_i \geq 0, \forall\ i, \\ & (\nu^T \kappa_i + b)(\nu^T \kappa_j + b) + \zeta_{ij} \geq 0, \zeta_{ij} a \geq 0, \forall\ (ij) \in \Theta, \end{aligned} \tag{2.4}$$

where $\mathbf{G}$ is a matrix computed from the random landmark vectors and κ_i denotes the kernel similarity between the ith point and all the landmark vectors.

To optimize Problem (2.4), however, is difficult since it is non-convex and nonlinear. The authors decompose the problem using a *constrained-concave-convex-procedure* (CCCP) (Yuille & Rangarajan, 2001), and then solve the relaxed convex sub-problems in each iteration through an efficient cutting-plane based QP solver.

On several real data sets, LAMP outperforms kernelized LSH (Kulis & Grauman, 2009) by a large margin. However, LAMP suffers from the local optimality issue and its performance is highly dependent on the quantity and quality of labeled pairs.

2.2.3 Supervised Hash Function Learning

In supervised HFL methods, only labeled data are used. There are several forms of label information, such as data labels and pairwise constraints, for example, similar pairs or dissimilar pairs.

Boosted similarity sensitive coding

Boosted similarity sensitive coding (BoostSSC) (Shakhnarovich et al., 2003; Shakhnarovich, 2005) might be, to the best of our knowledge, the first HFL algorithm for similarity search. Taking an embedding viewpoint, the authors propose to learn the embedding of unlabeled data into the Hamming space in which weighted Hamming distance preserves similarity information in the original input space.

In their approach, they first construct two kinds of point pairs based on labels or side information. In a positive point pair (x_i, x_j), x_j is one of N nearest neighbors of x_i or vice versa. In a negative pair, two points are not neighbors. After embedding or mapping, each point is represented by an M-bit binary vector $\mathbf{y}_i = [h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), ..., h_M(\mathbf{x}_i)]$, and the weighted Hamming distance between two points is given by

$$d(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \alpha_m |h_m(\mathbf{x}_i) - h_m(\mathbf{x}_j)|,$$

where the weights α_m's and the functions h_m's that map the input vector $\mathbf{x}_i$ into binary features are learned using Boosting (Schapire, 1999; Schapire & Singer, 1999).

At each iteration of the learning stage, BoostSSC selects the parameters to minimize the following squared loss:

$$\sum_{k=1}^{K} w(k)(z(k) - d(k))^2,$$

where K is the number of training pairs, $z(k)$ is the neighborhood label ($z(k) = 1$ if the points are neighbors and $z(k) = 1$ otherwise) of the kth pair, $w(k)$ is the weight of the kth pair and $d(k)$ is the weighted Hamming distance of the kth pair computed based on current model parameters. After each iteration, BoostSSC increases the weights of wrongly classified pairs such that hash functions learned in the next iteration can be improved.

To summarize, BoostSSC is simple to code, relatively fast to train and achieves competitive performance in (Torralba et al., 2008). However, BoostSSC might be slow to converge and become trapped in the local optimums.

SPEC hashing

Inspired by the idea of distribution matching in metric learning, Lin *et al.* (Lin et al., 2010) have developed another supervised HFL method called *similarity preserving entropy-based coding* (SPEC), for which two linear time learning algorithms exist. Given a sparse semantic similarity matrix $\mathbf{S}$, the goal of SPEC hashing is to construct a new matrix $\mathbf{W}$, which is evaluated on the Hamming distance of the learned hash codes, to minimize the KL divergence between $\mathbf{S}$ and $\mathbf{W}$. Each hash function is a decision stump and the hash functions are learned in an incremental manner. Since computing the objective function value is quadratic to the number of objects, the authors proposed two approximated linear time algorithms to learn the hash functions.

Experimental results show that this method is better than spectral hashing. However, it is apparently unclear whether or not SPEC is better than its supervised or semi-supervised counterparts and whether or not its underlying assumption works well on general data sets. As such, more comparative studies are needed to validate its effectiveness.

Minimal loss hashing

Norouzi *et al.* (Norouzi & Fleet, 2011) proposed a method called *minimal loss hashing* (MLH) to learn hash functions from pairwise constraints. Intuitively speaking, MLH

wants similar training points to be mapped onto binary codes that differ by no more than ρ bits, and dissimilar points to be mapped onto codes that differ by no less than ρ bits. The objective is represented as follows,

$$\mathcal{O}(\mathbf{w}) = \sum\nolimits_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \ell_\rho(\|\mathbf{y}_i - \mathbf{y}_j\|_H, S(i,j)),$$

where $\mathcal{S}$ is the set of point pairs, $\mathbf{y}_i$ and $\mathbf{y}_j$ are codes of $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively, $S(i,j)$ is the label of the point pair $(\mathbf{x}_i, \mathbf{x}_j)$, and

$$\ell_\rho(m,s) = \begin{cases} \max(m-\rho+1, 0) & \text{if } s=1 \\ \max(\lambda \max(\rho - m + 1, 0)) & \text{if } s=0, \end{cases}$$

where ρ is a user provided hyperparameter that differentiates neighbors from non-neighbors in the Hamming space, and λ is a loss-hyperparameter controlling the ratio of the slopes of the penalties.

Though the objective is simple to understand, its formulation is discontinuous, non-convex and hence hard to optimize. To solve the optimization problem, the authors borrow the idea of structural SVM (Tsochantaridis et al., 2004) and formulate a piecewise linear upper bound on the objective function. An alternating algorithm is developed to minimize the bound. Extensive experiments show that MLH achieves a state-of-the-art performance. However, MLH may still suffer from the local optimality problem.

2.2.4 Summary

Besides existing settings, some more complex HFL settings have also been explored recently, such as data with multiple similarities (Zhang et al., 2011), data of multiple modalities (Bronstein et al., 2010) and optimizing time and accuracy in a unified framework (He et al., 2011).

The major limitation of HFL methods is that most of them involve a training procedure with complexity $O(N^2)$ or $O(D^2)$, meaning that the training data should not be very large. Although some subsampling approaches can be utilized to reduce the high computational cost, developing some cheaper learning methods is a future research issue very worthwhile studying.

2.3 Metric Learning

Metric learning is an important problem in the machine learning and pattern recognition communities. The objective is to learn an optimal metric, either linear or nonlinear, in the original feature space or the reproducing kernel Hilbert space, from the training data.

According to whether or not the label information or side-information is used to learn the metric, existing methods can be classified into unsupervised metric learning or supervised metric learning. In the following, we briefly review some typical works in each category. For a detailed review, we refer the interested readers to (Yang & Jin, 2006).

2.3.1 Unsupervised Metric Learning

One typical case of unsupervised metric learning is linear dimensionality reduction. The most classic methods are *principal component analysis* (PCA) and *multidimensional scaling* (MDS). While PCA finds the subspace that best preserves the variance of the data, MDS finds the projection that best preserves the pairwise distance. The two methods are equivalent when MDS uses Euclidean distance. Despite being simple, efficient, and guaranteed to optimize their criteria, these linear methods could be very limited because they cannot find the nonlinear structure in the data.

To reveal the nonlinear structures of data, lots of nonlinear dimensionality reduction algorithms have been proposed. ISOMAP (Tenenbaum et al., 2000) assumes that isometric properties should be preserved in both the observation space and the intrinsic embedding space. According to this assumption, ISOMAP finds the subspace that best preserves the geodesic inter-point distance. Unlike ISOMAP that tries to preserve the geodesic distance for any pair of data points, *locally linear embedding* (LLE) (Roweis & Saul, 2000) and *Laplacian Eigenmap* (Belkin & Niyogi, 2003) focus on the preservation of the local neighbor structure. As an extension of (Belkin & Niyogi, 2003), *locality preserving projection* (LPP) (He & Niyogi, 2003) finds linear projective mappings that optimally preserve the neighborhood structure of the data. Mutual information which measures the difference between probability distributions has also been introduced to dimensionality reduction methods. Related work includes *stochastic neighbor embedding* (SNE) (Hinton & Roweis, 2002) and *manifold charting* (Brand, 2002).

2.3.2 Supervised Metric Learning

Supervised metric learning algorithms are designed to learn either from the class labels or the side information which is often cast in the form of pairwise constraints (i.e., must-link constraints and cannot-link constraints). In (Xing et al., 2002), Xing *et al.* propose to learn a distance metric from the pairwise constraints. The optimal kernel is found to minimize the distance between data points in must-link constraints and simultaneously maximize the distance between data points in cannot-link constraints. *Relevance component analysis* (Shental et al., 2002) is another popular approach, in which data points in the same classes are grouped in *chunklets*, and the distance metric is computed based on the co-

variance matrix estimated from each *chunklet*. Goldberger *et al.* (Goldberger et al., 2004) develop an algorithm, abbreviated as *neighborhood component analysis*, which combines metric learning with k-nearest neighbor (KNN) classification. Globerson *et al.* (Globerson & Roweis, 2005) present an algorithm to collapse data samples in the same class into a single point and make samples belonging to different classes far apart. Recently, an information-theoretic based approach (Davis et al., 2007) developed by Davis *et al.* has been reported to achieve a state-of-the-art performance.

Empirical studies show that supervised metric learning algorithms usually outperform the unsupervised ones. However, most of the supervised metric learning algorithms need to solve non-trivial optimization problems, and thus are computationally expensive.

2.3.3 Relationship with HFL

Both metric learning and hash function learning try to learn a proper metric from data, but they are different in several aspects. First of all, the learned metric in hash function learning is Hamming distance while that of metric learning is usually the Euclidean distance. Secondly, hash function learning maps data into binary codes whereas metric learning often maps data into real vectors. Last but not least, hash function learning and metric learning have quite different applications. Hash function learning aims to speed up approximate similarity search, but metric learning is usually applied to classification and recognition applications. Therefore, hash function learning puts more focus on local similarity than conventional metric learning.

2.4 Active Learning

As a research topic originally developed by the machine learning community, active learning has been widely applied in many areas such as computer vision, data mining and information retrieval. The task of active learning is to select the most informative data for experts to label with the goal of reducing the labeling cost, which might be expensive in many tasks. Over the past few decades, a lot of algorithms have been proposed and gained great successes.

The major challenge of active learning is how to find the most informative data for specific applications effectively and efficiently. In the following, we briefly review some general criteria of data informativeness and corresponding well-known algorithms. The readers are encouraged to read (Settles, 2009; Tong, 2001) for detailed reviews. Please also note that we use instance, example and data interchangeably in the sequel.

2.4.1 Uncertainty-based Active Learning

Perhaps the simplest and the most commonly used approach is *uncertainty-based active learning* (UAL) (Lewis & Catlett, 1994; Lewis & Gale, 1994). In this approach, the learner selects the instances whose labels it is the most uncertain about for the experts to label. This method is very straightforward for classifiers with probabilistic outputs. Take binary classification problems for example, when the basic learner could predict the labels in a probabilistic way, such as $P(y = +1 \mid \mathbf{x}) = 0.8$, UAL will select the instance whose posterior probability of being positive (or negative) is nearest to 0.5. However, this simple method will not be suitable in settings where there are more than two classes. A more general UAL method selects the data points maximizing the *entropy* defined as follows:

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}} H(\mathbf{y} \mid \mathbf{x}) = \text{argmax}_{\mathbf{x}} \left(-\sum\nolimits_i P(y_i \mid \mathbf{x}, \theta) \log P(y_i \mid \mathbf{x}, \theta) \right),$$

where $H(\mathbf{y} \mid \mathbf{x}) = -\sum_i P(y_i \mid \mathbf{x}, \theta) \log P(y_i \mid \mathbf{x}, \theta)$ is called entropy which measures the uncertainty of label $\mathbf{y}$ given instance $\mathbf{x}$, y_i ranges over all possible labels and θ is the model parameter. The criterion of entropy can be easily generalized to probabilistic models for more complex structured instances, such as sequences (Settles & Craven, 2008) and trees (Hwa, 2004).

An alternative to entropy-based UAL is selecting the instance whose *most probable* label is the *least confident*, which can be formulated as follows:

$$\mathbf{x}^* = \text{argmin}_{\mathbf{x}} P(y^* \mid \mathbf{x}, \theta),$$

where $y^* = \text{argmax}_y P(y \mid \mathbf{x}, \theta)$ is the most probable class label of instance $\mathbf{x}$. This method has been shown to work especially well for information extraction tasks (Culotta & McCallum, 2005; Settles & Craven, 2008).

For classifiers without probabilistic outputs, UAL can also be applied if the outputs could be mapped to probabilities (Lindenbaum et al., 2004; Fujii et al., 1998). Take margin-based classifiers such as SVM for example, the certainty can be defined as the distance to the decision boundary (Tong & Koller, 2002).

Another general UAL method is *query-by-committee* (QBC) (Seung et al., 1992). This approach maintains a group of classifiers, called a *committee*, which are trained on the current labeled data. Each committee member represents one classifier, and is allowed to vote on any unlabeled instances. The most uncertain data are those whose labels the committee members have the largest disagreement on. Intuitively, the QBC strategy is to minimize the version space represented by the committee of classifiers.

The UAL approaches are not immune to selecting outliers, which have high uncertainty but are not helpful to the learner when labeled and incorporated into the training set. Examples are provided in (McCallum & Nigam, 1998).

2.4.2 Representativeness-based Active Learning

Although effective and easy to implement in many applications, UAL methods are always prone to querying outliers, which are useless, sometimes even harmful, for classifier training. To overcome this limitation, *representativeness-based active learning* (RAL) has been proposed. The intuition of RAL methods is that the most informative data should be the most representative of the unlabeled data.

Xu *et al.* (Xu et al., 2003) might be the first to implement the above intuition for SVM classifiers, using some simple heuristics. Instead of selecting the instances closest to the current SVM hyperplane, they first cluster the points in the margin of the current model and then query the labels of the cluster centroid. (Nguyen & Smeulders, 2004) first clusters unlabeled instances and tries to avoid querying outliers by propagating label information from cluster centroid to instances in the same cluster.

Density-based active learning algorithms, which tend to select the instances from dense regions, could also be considered a special case of RAL, because the denser the region is, the more representative the instances (located in the region) are. These RAL approaches are always used in combination with UAL methods (Xu et al., 2007; Settles & Craven, 2008). In (Xu et al., 2007), data informativeness is measured by relevance, density and diversity in the relevance feedback tasks. Similarly, (Settles & Craven, 2008) develops an information density framework for the sequence labeling task to measure the uncertainty and representativeness of instances.

In recent years, experimental design originated in statistics has been introduced as a new family of RAL methods. The seminal work is *transductive experimental design* (Yu et al., 2006), which extends traditional experimental design methods to the transductive setting for active learning. Some subsequent work includes convex relaxation of the original problem (Yu et al., 2008), and incorporation of *Laplacian* regularization (He et al., 2007) or label information (Zhen & Yeung, 2010).

2.4.3 Minimal Loss Active Learning

In many real-world applications, the learned classifiers are eventually evaluated on a test set, so a better classifier should make less error on the test set. However, none of previous approaches directly optimize this objective, and this may explain why they do not work

well in some circumstances. Intuitively, *minimal loss active learning* (MLAL) aims to select the instances, when labeled and incorporated into the training set, leading to the largest error reduction on the test set. To evaluate the test error, we have to know the true labels of test instances. However, the true labels are unknown during the model training phase, as a result, the estimated (or expected) test error is used.

Cohn *et al.* proposed a statistically optimal solution, which selects the training examples that result in the lowest error on future test examples (Cohn et al., 1996). In their analysis, this goal could be achieved by minimizing the variance of training data. The authors developed two simple algorithms with closed form solutions for regression problems. For classification problems, Roy and McCallum used a sampling approach to estimate the expected error reduction (Roy & McCallum, 2001). Later, this framework was combined with a supervised learning approach to give a dramatic improvement over conventional UAL methods (Zhu et al., 2003).

MLAL has the advantages of being near-optimal and independent of the types of classifiers. However, it may be the most prohibitively expensive strategy, because it requires not only estimating the expected future error over the unlabeled data at each learning iteration, but also retraining the classifier for each possible label of the instance. To reduce the computational cost, some researchers have resorted to subsampling the unlabeled data (Roy & McCallum, 2001) or approximate training techniques (Guo & Greiner, 2007).

2.4.4 Relationship with HFL

Active learning and active hashing have in common that both of them aim to find the most informative data for experts to label. As such, the criteria of informativeness might be similar in active learning and active hashing. However, active learning is usually applied to classification, regression and ranking problems, which are very different from the approximate similarity search to which active hashing applies. This may lead to big differences in the definitions of informativeness, the formulations of optimization problems as well as the algorithms.

2.5 Summary

In this chapter, we have reviewed hashing-based methods for similarity search, and two machine learning areas related to hash function learning. The central idea of hashing-based methods is to index data using binary codes which have the advantages of highly reduced storage cost and very fast computation speed. Different from locality sensitive

hashing which is based on random projections or permutations, hash function learning aims to learn hash functions from data automatically, and thus, as we see later in more detail, has a close relationship with metric learning and active learning.

In the next chapter, we introduce the framework of active hashing which combines the concepts behind semi-supervised hash function learning and active learning to make HFL more cost effective.

CHAPTER 3

ACTIVE HASHING

3.1 Introduction

Existing supervised and semi-supervised HFL methods can be considered *passive hashing* because they assume that the labeled data are provided beforehand. However, given the labeled data may be expensive to acquire,[1] it is more cost effective if we are able to identify and label the most informative data. In this case, hash functions can be learned efficiently with only a small number of labeled data and the labeling cost, which might be very high in practice, can be greatly reduced. Besides, as we will see in next subsection, the effectiveness of the labeled data may be quite different. In some situations, adding more labels (if not carefully chosen) into the training set may even impair the quality of the learned hash functions. As a result, it is a very worthwhile endeavor to explore methods of selecting and labeling data in an active manner for hash function learning.

To eliminate the aforementioned limitations of passive hashing, in this chapter, we propose a novel framework, called *active hashing* (AH), with the goal of selecting the most informative data to label for hash function learning.[2] Generally speaking, each active hashing iteration consists of three phases as depicted in Figure 3.1. Specifically, given the labeled data set $\mathcal{L}$, the unlabeled data set $\mathcal{U}$ and the candidate data set $\mathcal{C}$,

- Active selection phase: select the most informative data points from $\mathcal{C}$ to form set $\mathcal{A}$.
- Labeling phase: ask an oracle to label the points in $\mathcal{A}$ and update $\mathcal{L}$, $\mathcal{U}$ and $\mathcal{C}$.
- Training phase: Train hash functions based on both $\mathcal{L}$[3] and $\mathcal{U}$.

In practice, there can be several iterations before some criteria are met. For example, we have run out of labeling resources or the quality of the learned hash functions is satisfactory.

[1] Although crowdsourcing has recently been developed to acquire labels in a very cheap manner, for some domains involving privacy or confidentiality, it is inappropriate to involve the "crowd" in the labeling work.

[2] Active hashing and active learning (Angluin, 1988; Cohn et al., 1994) have quite different goals, although they share some common ideas such as selecting the most informative data for expert to label.

[3] The usage of $\mathcal{L}$ may be different in different methods. For example, SSH can construct labeled pairs $\mathcal{S}$ and $\mathcal{D}$ easily from $\mathcal{L}$.

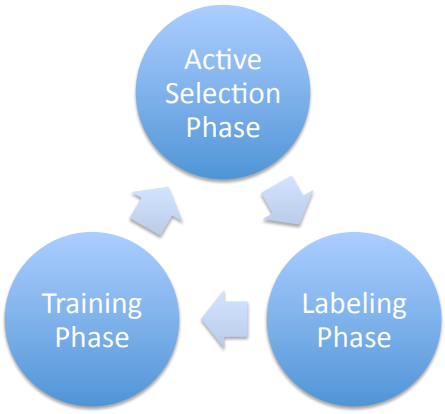


Figure 3.1: Three phases of one active hashing iteration. Each iteration starts from the active selection phase and ends by the training phase.

The remainder of this chapter is organized as follows. Section 3.2 illustrates the limitations of passive hashing to motivate our active hashing framework. In Section 3.3, we present a simple active hashing method based on the uncertainty criterion. Empirical studies conducted on several real-world data sets are presented in Section 3.4, followed by some conclusions in Section 3.5.

3.2 Motivation

In passive hashing methods such as SSH, there are two types of point pairs (similar and dissimilar) and both types are considered equally important. However, as illustrated in Fig. 3.2 below, treating both types of point pairs as equally important is not satisfactory when the data points belong to more than two classes.

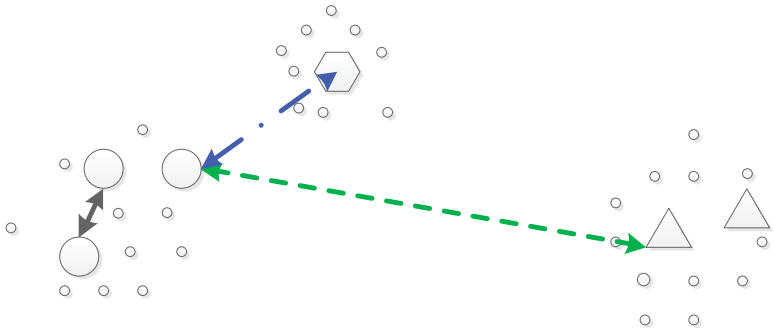
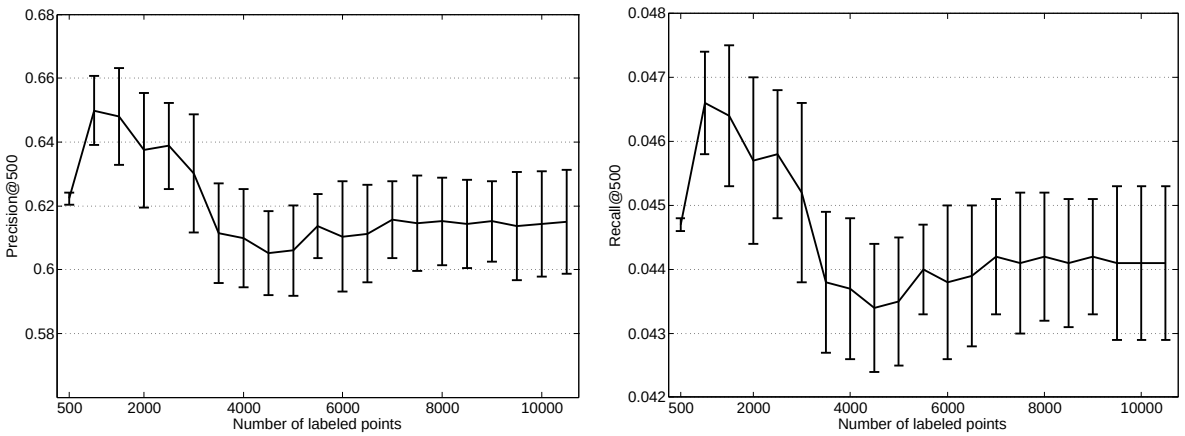


Figure 3.2: Limitations of passive hashing

In Fig. 3.2, we use circles, triangles, hexagons and smaller circles to denote labeled data of three different classes and unlabeled data respectively. There are two dissimilar pairs represented by a blue dashed line and a green dashed line. Obviously, the circle

class is more similar (closer) to the hexagon class than the triangle class. However, the two pairs induce the same weight (-1) in $\mathcal{D}$, making the learned hash functions biased and perform worse even when more labeled pairs are provided. As a result, it is very important to treat the pairs with different weights or informativeness, as we will discuss later, but passive hashing methods are unable to do so.

To see some real examples of these limitations, we conduct a group of experiments on the MNIST data set.[4] In the experiments, we trained several semi-supervised hashing models (Wang et al., 2010a) by varying the number of labeled points[5]. The performance of these models, measured by average precision and recall over ten random repeats, is plotted in Fig. 3.3 below.



(a) Precision (b) Recall

Figure 3.3: Semi-supervised hashing with a varying number of labeled data points

We can see that the model performance in terms of both precision and recall has big variance, showing that the effectiveness of labeled points is quite different. Moreover, the model performance improves as the number of labeled data points increases until about one thousand. After that, however, the performance degrades as more labeled points are added, indicating that choosing labeled data is very critical to supervised and semi-supervised HFL algorithms.

The motivation of active hashing is simple: if we select labeled data carefully, we can not only reduce the model variance but also keep improving the model quality without impairing it when more labeled data are added into the training set.

[4]Details of the data set and the evaluation methods will be introduced later.

[5]All the pairwise relations among these points will be used for training.

3.3 Uncertainty-based Active Hashing

3.3.1 Uncertainty-based Active Hashing

The challenge of active hashing is how to identify the most informative points for hash function learning. Based on a previous semi-supervised HFL model, i.e., SSH (Wang et al., 2010a), we propose an *uncertainty-based active hashing* (UAH) method.

We have already known that, in (Wang et al., 2010a), one bit code is obtained by thresholding a linear projection of a point $\mathbf{x}$, e.g., $h_m = \text{sgn}(\mathbf{w}_m^T\mathbf{x})$ for the mth bit. Intuitively speaking, the magnitude of $\mathbf{w}_m^T\mathbf{x}$ measures the certainty of current hash function h_m on the mth bit code of $\mathbf{x}$. In other words, the larger $|\mathbf{w}_m^T\mathbf{x}|$ is, the more certain h_m is on $\mathbf{x}$, and vice versa. Thus it is very straightforward to use h_m's certainty $\bar{h}_m(\mathbf{x}) = |\mathbf{w}_m^T\mathbf{x}|$ to measure the informativeness of point $\mathbf{x}$; that is, the smaller $\bar{h_m}(\mathbf{x})$ is, the less certain h_m is on $\mathbf{x}$ and hence the more informative $\mathbf{x}$ is to h_m. Therefore, given a group of M hash functions $\mathcal{H} = \{h_m\}_{m=1}^M$, we define the certainty of $\mathbf{x}$ *w.r.t.* $\mathcal{H}$ as follows,

Definition 3.3.1. *Data Certainty*

$$f(\mathcal{H}, \mathbf{x}) = \|\mathbf{W}^T\mathbf{x}\|_2,$$

where $\|\cdot\|_2$ *denotes the vector* ℓ_2 *norm.*[6]

Based on this simple definition of informativeness, one simple active hashing algorithm is to select the point with the smallest f value at each active iteration. However, as pointed out by (Hoi et al., 2006a; Guo & Schuurmans, 2007), selecting data point in such a greedy manner, i.e., one at a time, could be very inefficient in practice. Moreover, greedy selection may be suboptimal because the selected points might be very similar to each other and hence provide redundant information to the learner. As such, we propose a *batch mode active hashing* (BUAH) algorithm which can select a batch of points at a time.

Similar to the batch mode active learning algorithm for image classification studied in (Hoi et al., 2006b), we define the objective of BUAH as follows,

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T\tilde{\mathbf{f}} + \tfrac{\lambda}{M}\boldsymbol{\mu}^T\mathbf{K}\boldsymbol{\mu} \\ \text{s.t.} \quad & \boldsymbol{\mu} \in \{0,1\}^{|\mathcal{C}|}, \boldsymbol{\mu}^T\mathbf{1} = M, \end{aligned} \tag{3.1}$$

where $\tilde{\mathbf{f}}$ is the normalized certainty value of the points in a candidate set $\mathcal{C}$ and can be computed as $\tilde{f}_i = f(\mathcal{H}, \mathbf{x}_i)/f_{\max}$ and $f_{\max} = \max_{\mathbf{x}_i \in \mathcal{C}} f(\mathcal{H}, \mathbf{x}_i)$, $\boldsymbol{\mu}$ is an indicator vector whose entries control whether or not corresponding points are selected, i.e., $\mu_i = 1$ when $\mathbf{x}_i$ is selected and $\mu_i = 0$ when $\mathbf{x}_i$ is not selected. We call λ the balancing parameter, since

[6]Please note that other norms such as ℓ_1 norm can also be used to define certainty.

Algorithm 3.1 Algorithm of BUAH

Input:
$\mathcal{L}^0$ – initial set of labeled data,
$\mathcal{U}^0$ – initial set of unlabeled data,
$\mathcal{C}^0$ – initial set of candidate data,
T – number of iterations
Procedure:
for $t = 1$ **to** T **do**
Train hash functions $\mathcal{H}$ based on $\mathcal{L}^{t-1}$ and $\mathcal{U}^{t-1}$
Compute certainty value $\mathbf{f}$ of $\mathcal{C}^{t-1}$ using Definition 3.3.1
Solve Problem (3.2) and obtain $\boldsymbol{\mu}$
Choose M examples with the largest μ_i into $\mathcal{A}^t$
Request the labels of points in $\mathcal{A}^t$
Update $\mathcal{L}^t \leftarrow \mathcal{L}^{t-1} \cup \mathcal{A}^t$, $\mathcal{U}^t \leftarrow \mathcal{U}^{t-1} \setminus \mathcal{A}^t$ and $\mathcal{C}^t \leftarrow \mathcal{C}^{t-1} \setminus \mathcal{A}^t$
end for

it controls the balance between the two terms. Moreover, K is the number of points we want to select and $\mathbf{K}$ is a positive semi-definite similarity matrix defined on $\mathcal{C}$. We use cosine similarity in our experiments, however, using other similarities is also possible and we leave it as future work.

Intuitively, we can consider the first term of Eqn. (3.1), $\boldsymbol{\mu}^T\tilde{\mathbf{f}}$, to be the sum of certainty of selected examples, and the second term, $\boldsymbol{\mu}^T\mathbf{K}\boldsymbol{\mu}$, to be the sum of similarity between these selected examples. Thus minimizing the objective of Problem (3.1) is to select a set of points that are the most uncertain and dissimilar with each other. However, Problem (3.1) is an integer programming problem and NP-hard. We relax the problem by replacing the integer constraint $\boldsymbol{\mu} \in \{0,1\}^{|\mathcal{C}|}$ with continuous constraint $\mathbf{0} \leq \boldsymbol{\mu} \leq \mathbf{1}$, and arrive at the following optimization problem,

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T\tilde{\mathbf{f}} + \tfrac{\lambda}{M}\boldsymbol{\mu}^T\mathbf{K}\boldsymbol{\mu} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\mu} \leq \mathbf{1}, \boldsymbol{\mu}^T\mathbf{1} = M. \end{aligned} \tag{3.2}$$

The optimization problem (3.2) is a standard QP problem that can be solved efficiently using existing solvers (Boyd & Vandenberghe, 2004). Finally, given the estimated $\boldsymbol{\mu}$, we select M unlabeled points with the largest μ_i's.

The algorithm of BUAH is summarized in Algorithm 3.1, where the superscripts of $\mathcal{L}, \mathcal{U}, \mathcal{C}$ indicate the indices of active selection iterations and T is the total number of iterations. Since solving Problem (3.2) requires time complexity $O(|\mathcal{C}|^3)$, in practice, we can select a subset of $\mathcal{C}$ corresponding to the smallest f_i values to reduce the computational cost without much loss of accuracy.

3.4 Experiments

3.4.1 Experimental Settings

We conduct several experiments on two tasks, namely, image retrieval and text retrieval. For both tasks, we mean-center the data and normalize the feature vectors to have unit norm. Both data sets are partitioned into separate training and test sets. We first use the training set to learn hash functions. Then we retrieve from the training set 500 points with smallest Hamming distances to the query point in the test set. Two evaluation measures, precision and recall, of the retrieved points are computed for each query and then averaged over all the queries. In all the experiments, we set the size of the candidate set $|\mathcal{C}| = 5000$ and the code length $K = 24$.

Three algorithms are compared in the experiments:

- **Random Sampling** (Random), which randomly selects examples to label. This is essentially a passive hashing method.[7]
- **Greedy Active Hashing** (GAH), which selects one point with the smallest $\tilde{f}_i$ in each iteration.
- **Batch Mode Active Hashing** (BMAH), which selects a batch of M points by solving Problem (3.2).

3.4.2 Image Retrieval

Data set

The MNIST data set[8] consists of $70,000$ images of handwritten digits. Each image has 28×28 pixels and is associated with a label from 0 to 9. We use the gray-scale intensity values of the images as features resulting in a 784-dimensional vector space. The data set is split into a training set of $69,000$ images and a test set of $1,000$ images.

Comparison of BMAH and GAH

We first compare two AH algorithms, BMAH and GAH. Initially, we randomly choose 100 points and their labels from the training set to form $\mathcal{L}^0$ and use the remaining data

[7] But unlike other passive hashing methods, the labeled set does grow in size. This is similar to some active learning methods with random data selection. So it is still "active" in some sense.

[8] http://yann.lecun.com/exdb/mnist/

as $\mathcal{U}^0$. We then use BMAH and GAH individually to select M points[9] and report the retrieval results of the hash functions learned from the updated training set. We use $\lambda = 0.4$ for BMAH.[10] The procedure is repeated 10 times and the average results, as well as their standard deviations, and total time cost (in seconds) for different values of M are reported in Table 3.1.

Table 3.1: Comparison of BMAH and GAH for image retrieval

M	Method	Precision	Recall	Time Cost (seconds)
50	BMAH	0.6096±0.0009	0.0438±0.0001	195.03
	GAH	0.6096±0.0007	0.0438±0.0001	482.23
100	BMAH	0.6097±0.0007	0.0438±0.0001	214.23
	GAH	0.6100±0.0007	0.0438±0.0001	887.60
150	BMAH	0.6107±0.0009	0.0438±0.0001	352.49
	GAH	0.6120±0.0009	0.0439±0.0001	1563.03
200	BMAH	0.6113±0.0012	0.0439±0.0001	433.81
	GAH	0.6154±0.0013	0.0442±0.0001	1744.36

From Table 3.1, we observe that BMAH and GAH are comparable in performance in terms of both precision and recall, showing that there is not much redundancy among the most informative points. However, GAH has much higher computational time cost than BMAH. Therefore, we exclude GAH in the subsequent experiments.

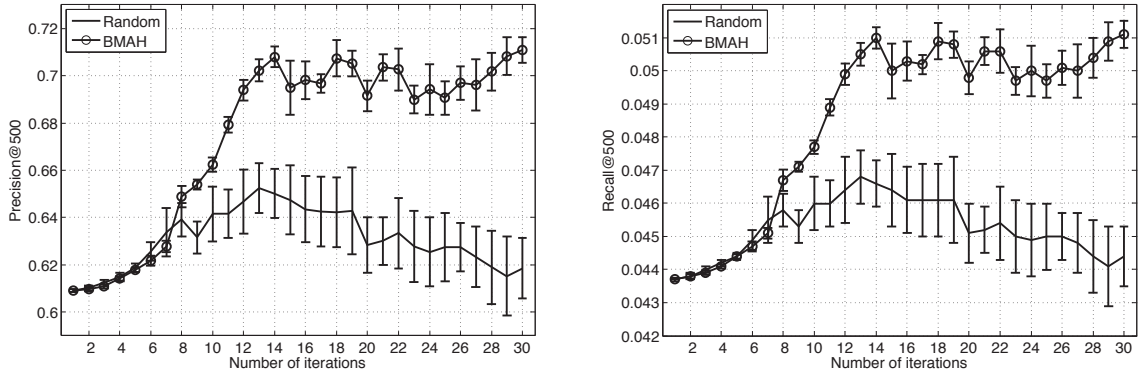
Comparison of BMAH and Random

In this section, we compare BMAH with Random. We again choose 100 random points from the training set to form $\mathcal{L}^0$ and use the remaining data as $\mathcal{U}^0$. For BMAH, we set the parameters to be $M = 100$ and $\lambda = 0.4$. The whole process is repeated 10 times and we plot the average results and their standard deviations in Fig. 3.4.

We can see from the figures that BMAH performs much better than Random, especially at the late stage. We observe that the performance of Random degrades as more labels are obtained. This can be attributed to the limitation of passive hashing as introduced in Section 3.2. On the other hand, BMAH can overcome this problem by selecting the most uncertain points because these points are likely to locate near the class boundary and hence the difference between points in a dissimilar pair may be small.

[9]We note that GAH selects only one example in each iteration, so it needs M iterations to select a total of M points. However, BMAH can select all M points in a single iteration.

[10]The reason why we set $\lambda = 0.4$ will be made clear later.



(a) Learning curves *w.r.t.* precision

(b) Learning curves *w.r.t.* recall

Figure 3.4: Learning curves of BMAH and Random for image retrieval

Varying initial label size $|\mathcal{L}^0|$

To evaluate the effect of the initial label size on BMAH and Random, we conduct some experiments by varying the value of $|\mathcal{L}^0|$. The parameters of BMAH are $M = 10$ and $\lambda = 0.4$. For each $|\mathcal{L}^0|$ value, the whole procedure is repeated 10 times and the average results, as well as standard deviations, after selecting $1,000$ points are reported in Table 3.2, where boldface numbers indicate better results.

Table 3.2: Comparison of BMAH and Random with varying initial label size $|\mathcal{L}^0|$

$\lvert\mathcal{L}^0\rvert$	Method	Precision	Recall
100	Random	0.6416±0.0103	0.0460±0.0007
	BMAH	**0.6793**±0.0033	**0.0489**±0.0002
200	Random	0.6430±0.0087	0.0461±0.0006
	BMAH	**0.6941**±0.0025	**0.0500**±0.0002
300	Random	0.6518±0.0119	0.0467±0.0009
	BMAH	**0.6985**±0.0052	**0.0502**±0.0004
400	Random	0.6474±0.0150	0.0464±0.0011
	BMAH	**0.7053**±0.0060	**0.0507**±0.0004
500	Random	0.6417±0.0135	0.0460±0.0010
	BMAH	**0.7073**±0.0090	**0.0509**±0.0007

From Table 3.2, we can easily see that BMAH is consistently better than Random. It is interesting to observe that the precision and recall of BMAH keep increasing as $|\mathcal{L}^0|$ increases, but those of Random first increase and then decrease. This is reasonable since BMAH can find informative points more accurately with more initially labeled points whereas the active selection procedure of Random is independent of $\mathcal{L}^0$.

Varying batch size M

To study the effect of the batch size on BMAH and Random, we conduct several experiments by varying M. We set $|\mathcal{L}^0| = 100$ for both methods and $\lambda = 0.4$ for BMAH. The average results over 10 repeats, and the corresponding standard deviations, after selecting $10M$ points are reported in Table 3.3.

Table 3.3: Comparison of BMAH and Random with varying batch size M

M	Method	Precision	Recall
50	Random	**0.6292**±0.0026	**0.0452**±0.0002
	BMAH	0.6262±0.0017	0.0450±0.0001
100	Random	0.6416±0.0103	0.0460±0.0007
	BMAH	**0.6793**±0.0033	**0.0489**±0.0002
150	Random	0.6392±0.0082	0.0458±0.0006
	BMAH	**0.6988**±0.0084	**0.0504**±0.0006
200	Random	0.6302±0.0130	0.0452±0.0009
	BMAH	**0.7068**±0.0046	**0.0509**±0.0003

From Table 3.3, it is easy to see that a larger value of M also induces a larger performance gap between BMAH and Random. This is quite reasonable because larger M implies that 1) BMAH will select more informative points than Random; 2) the limitation of passive hashing will be reduced more by BMAH.

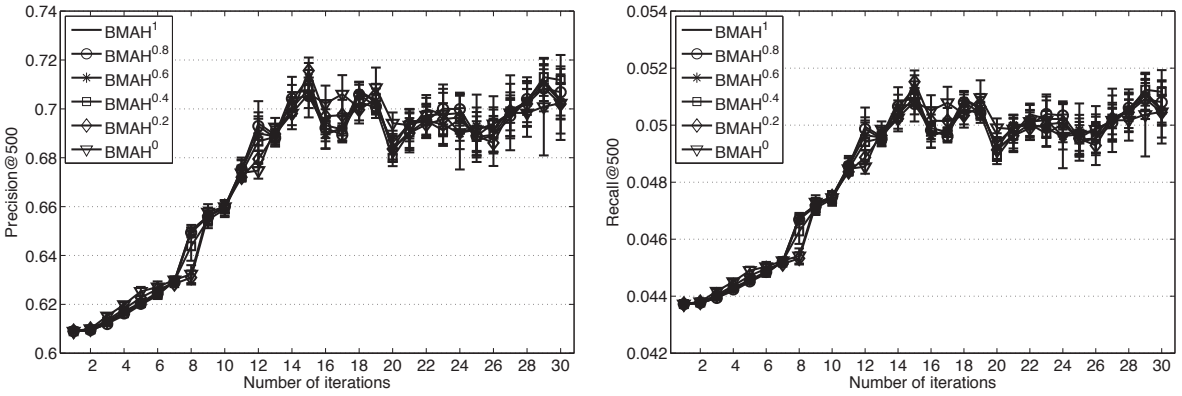
Varying balancing parameter λ

In Problem (3.2), the balancing parameter λ controls the balance between selecting the most uncertain points and reducing the redundancy among the selected data points. In this last group of experiments, we study the effect of λ by varying its value from 0 to 1. For each value of λ, the result is averaged over 10 random repeats. We plot learning curves *w.r.t.* precision and recall in Fig. 3.5. Note that the value of λ is indicated by the superscript of BMAH.

From Fig. 3.5, we observe that BMAH is not very sensitive to λ and $\lambda = 0.4$ achieves the best performance after 30 iterations. We believe that there is little redundancy among the most uncertain data points, so varying λ in such a small range does not have much effect on the model performance.

Varying code length K

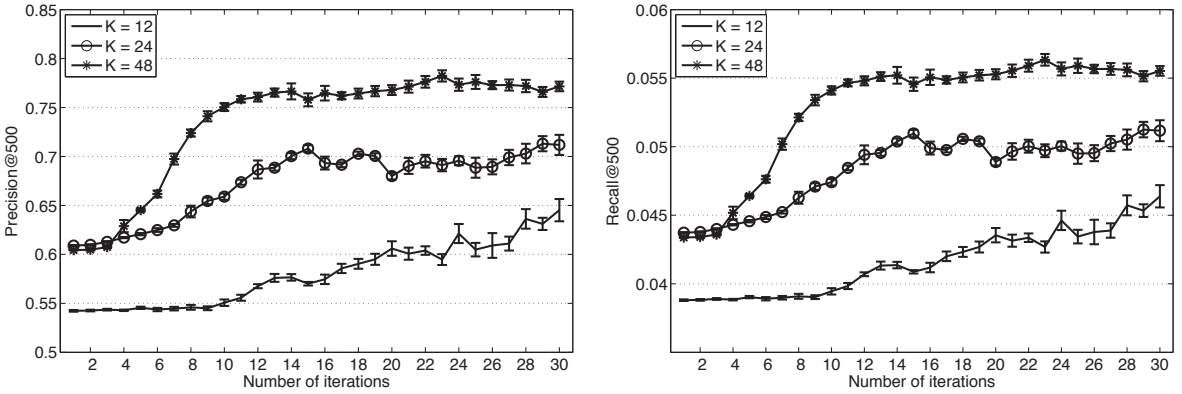
We have observed that BMAH is very effective with a fixed code length, i.e., $K = 24$. But is it also effective when we set the code length K to other values? To answer this question,



(a) Learning curves *w.r.t.* precision (b) Learning curves *w.r.t.* recall

Figure 3.5: Learning curves of BMAH with varying balancing parameter λ for image retrieval. The parameters of BMAH are set as $|\mathcal{L}^0| = 100$ and $M = 100$.

we conducted a group of experiments by running BMAH with different K values. The learning curves of BMAH averaged over 10 random repeats are plotted in Fig. 3.6.



(a) Learning curves *w.r.t.* precision (b) Learning curves *w.r.t.* recall

Figure 3.6: Learning curves of BMAH with varying code length K for text retrieval. The parameters of BMAH are set as $\lambda = 0.4, |\mathcal{L}^0| = 100$ and $M = 100$.

From Fig. 3.6, we observe that larger code length gives better performance and, for all three different code lengths, BMAH consistently improves as more labeled data are added. The results validate the effectiveness of BMAH.

3.4.3 Text Retrieval

Data set

The 20 Newsgroups (NEWS) data set[11] was originally collected for document classification. We use the popular 'bydate' version which contains 18,846 documents evenly

[11]http://people.csail.mit.edu/jrennie/20Newsgroups/

distributed across 20 categories. Each document is labeled by exactly one of the 20 labels. In our experiments, we randomly select 1,000 documents as the test set and use the remaining documents as the training set. The original data set contains 26,214 features generated by the *tf·idf* scheme (Salton & Buckley, 1988). In our experiments, we extract 1,000 features by applying principal component analysis (PCA).

Comparison of BMAH and GAH

The first group of experiments is to compare BMAH and GAH. We randomly choose 50 points and their labels to form $\mathcal{L}^0$ and use the remaining data as $\mathcal{U}^0$. Moreover, we set $\lambda = 0.4$ for BMAH. The procedure is repeated 10 times and the average results after selecting M points are reported in Table 3.4. The results in Table 3.4 are very similar to those in Table 3.1, and for the same reason, we omit GAH in the following experiments.

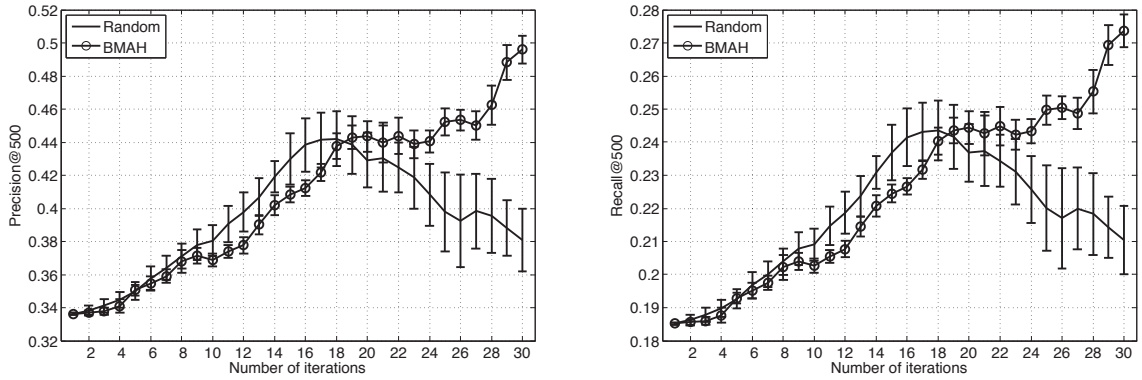
Table 3.4: Comparison of BMAH and GAH for text retrieval

M	Method	Precision	Recall	Time Cost (seconds)
50	BMAH	0.3373±0.0012	0.1857±0.0006	67.44
	GAH	0.3375±0.0011	0.1858±0.0006	473.28
100	BMAH	0.3377±0.0022	0.1859±0.0012	79.88
	GAH	0.3374±0.0017	0.1857±0.0009	958.04
150	BMAH	0.3441±0.0043	0.1894±0.0023	219.34
	GAH	0.3420±0.0033	0.1882±0.0018	2058.31
200	BMAH	0.3520±0.0043	0.1935±0.0023	469.08
	GAH	0.3492±0.0033	0.1921±0.0018	2612.79

Comparison of BMAH and Random

We compare BMAH with Random in this section. Similarly, we set $|\mathcal{L}^0| = 50$ for both methods and $M = 50$ and $\lambda = 0.4$ for BMAH. We plot the results averaged over 10 random repeats and their standard deviations in Fig. 3.7.

From the figures, we can easily see that the performance of Random is better than that of BMAH at the early stage but degrades very fast afterwards. This is reasonable because there are 20 categories in the data set and initially the selected points can only cover a small portion of them. In the beginning, Random uniformly selects points from all the categories but BMAH focuses on those points that are uncertain with respect to the current hash functions and hence might be biased. However, this kind of bias will be corrected when more labeled data points are added. Thus, the precision of BMAH continues to increase at a later stage but that of Random drops due to the limitation of passive hashing.



(a) Learning curves *w.r.t.* precision

(b) Learning curves *w.r.t.* recall

Figure 3.7: Learning curves of BMAH and Random for text retrieval

Varying initial label size $|\mathcal{L}^0|$

In this group of experiments, we vary the initial label size $|\mathcal{L}^0|$ to study its effect on BMAH and Random. The parameters are set to $M = 50$ and $\lambda = 0.4$. The whole procedure is repeated 10 times and the average results, together with standard deviations, after selecting $30M$ points are reported in Table 3.5.

Table 3.5: Comparison of BMAH and Random with varying initial label size $|\mathcal{L}^0|$

$\lvert\mathcal{L}^0\rvert$	Method	Precision	Recall
50	Random	0.3765±0.0188	0.2081±0.0103
	BMAH	**0.4993**±0.0112	**0.2752**±0.0062
100	Random	0.3731±0.0250	0.2063±0.0137
	BMAH	**0.4961**±0.0140	**0.2737**±0.0076
150	Random	0.3726±0.0155	0.2060±0.0083
	BMAH	**0.4945**±0.0124	**0.2730**±0.0065
200	Random	0.3620±0.0147	0.2002±0.0078
	BMAH	**0.5068**±0.0140	**0.2796**±0.0076
250	Random	0.3666±0.0144	0.2026±0.0079
	BMAH	**0.5064**±0.0148	**0.2792**±0.0080

In Table 3.5, BMAH consistently outperforms Random. It is interesting to observe that the precision and recall of BMAH keep increasing as $|\mathcal{L}^0|$ increases, but those of Random first increase and then drop. This can be explained by the same reason in Section 3.4.2.

Varying batch size M

In this section, we evaluate the effect of the batch size M on BMAH and Random. We set $|\mathcal{L}^0| = 50$ for both methods and $\lambda = 0.4$ for BMAH. The whole process is repeated 10 times, and the average results and standard deviations, after selecting $20M$ points, are reported in Table 3.6.

Table 3.6: Comparison of BMAH and Random with varying batch size M

M	Method	Precision	Recall
50	Random	0.4302±0.0201	0.2374±0.0106
	BMAH	**0.4399**±0.0121	**0.2426**±0.0065
100	Random	0.3432±0.0173	0.1900±0.0093
	BMAH	**0.5001**±0.0220	**0.2766**±0.0119
150	Random	0.2983±0.0170	0.1655±0.0091
	BMAH	**0.4076**±0.0239	**0.2248**±0.0131
200	Random	0.2718±0.0132	0.1511±0.0071
	BMAH	**0.3266**±0.0267	**0.1806**±0.0148

From Table 3.6, we see that increasing M will degrade the performance of both BMAH and Random. This observation is different from what we have observed in Section 3.4.2. In this data set, the limitation of passive hashing is much more severe since there are 20 categories. Hence BMAH cannot completely eliminate the limitation as in the previous task. However, the performance drop of BMAH is much slower than that of Random. This still demonstrates that BMAH is capable of overcoming the limitation of passive hashing slightly.

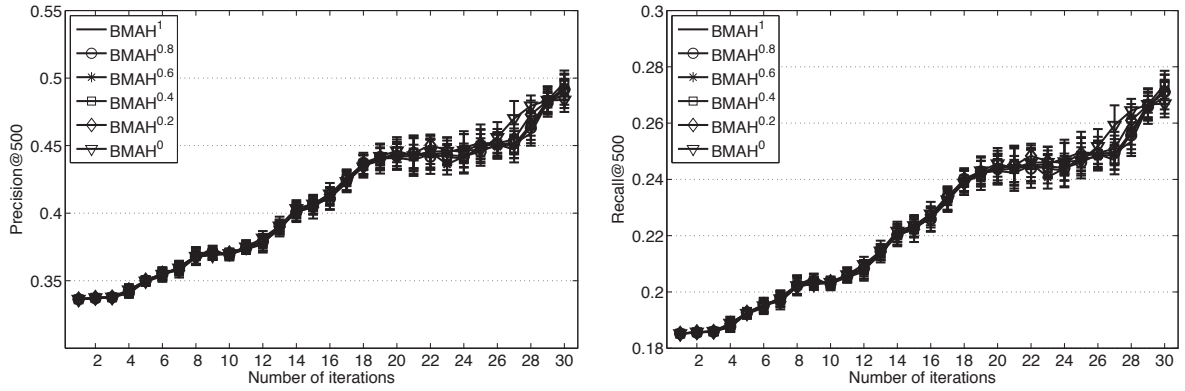
Varying balancing parameter λ

To study the sensitivity of BMAH to the balancing parameter λ, we conduct some experiments by varying λ from 0 to 1. The learning curves averaged over 10 random repeats are plotted in Fig. 3.8. Similar to before, we use superscripts to indicate the values of λ.

From Fig. 3.8, we see that BMAH is not very sensitive to λ, and $\lambda = 0.4$ achieves the best performance after 30 iterations. This again can be explained by the same reason in Section 3.4.2.

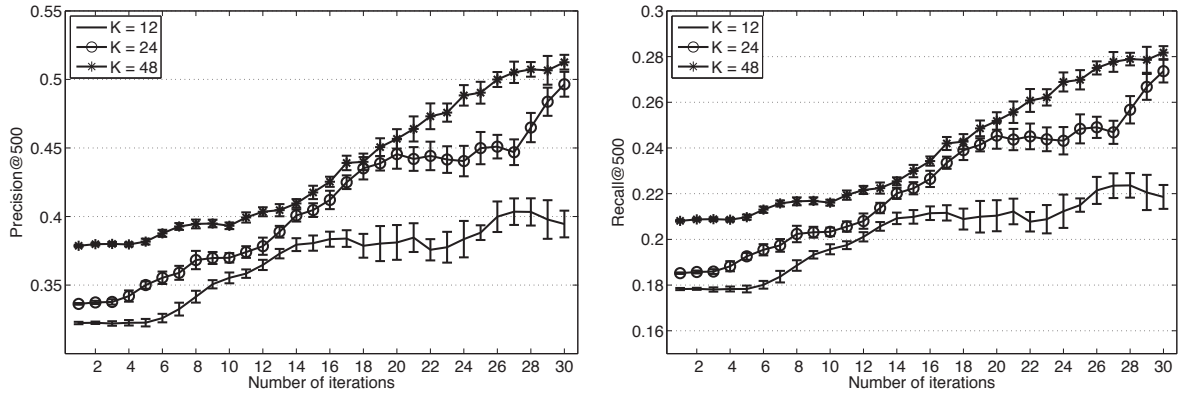
Varying code length K

In this last subsection, we study the performance of BMAH by varying the code length K. The learning curves of BMAH, averaged over 10 random repeats, with different K are plotted in Fig. 3.9.



(a) Learning curves *w.r.t.* precision (b) Learning curves *w.r.t.* recall

Figure 3.8: Learning curves of BMAH with varying balancing parameter λ for text retrieval. The parameters of BMAH are set as $|\mathcal{L}^0| = 50$ and $M = 50$.



(a) Learning curves *w.r.t.* precision (b) Learning curves *w.r.t.* recall

Figure 3.9: Learning curves of BMAH with varying code length K for text retrieval. The parameters of BMAH are set as $\lambda = 0.4, |\mathcal{L}^0| = 50$ and $M = 50$.

From Fig. 3.9, we observe that the performance of BMAH improves as more labeled data are incorporated no matter how long the hash codes are. This again validates the effectiveness of BMAH.

3.5 Conclusion

To summarize, we have proposed a new hashing framework to actively learn hash functions from both unlabeled and labeled data. We have utilized the uncertainty criterion, which is commonly used in many active learning methods, to give a simple and efficient active hashing algorithm. Experimental results show that our framework can effectively identify the most informative points for the expert to label and can overcome the limitations of existing supervised HFL methods.

Though being effective and easy to implement, the uncertainty-based active hashing method does not directly optimize the quality of the learned hash functions. As such, to take this work further, we plan to explore other criteria of informativeness for active hashing. For example, we may select the points which maximize some information gain or minimize some expected loss functions which are directly related to the quality of hash functions. Moreover, although we have demonstrated that it is very effective to select points to label first and then construct point pairs, it might be better to select point pairs directly to make it computationally more efficient. Last but not least, another possible research direction is to explore other batch mode algorithms for active hashing.

CHAPTER 4

MULTIMODAL HASHING FOR ALIGNED DATA

4.1 Introduction

As of now, almost all existing HFL methods assume that the data are unimodal, meaning that both the queries and the candidates are of the same modality. They cannot be adapted easily for multimodal search which is often encountered in many multimedia retrieval, image analysis and data mining applications. Take crossmodal multimedia retrieval for example, using an image about a historic event as query, one may want to retrieve relevant text articles that can provide more detailed information about the event. Obviously, existing unimodal methods cannot be applied directly to multimodal similarity search because they assume that all the data are of the same modality. Designing HFL methods for multimodal data is thus a very worthwhile direction to pursue.

Recently, Bronstein *et al.* (Bronstein et al., 2010) proposed a general framework which is referred to as *multimodal hashing* (MH). As illustrated in Figure 4.1 for the bimodal case, MH functions hash documents of different modalities into a common Hamming space so that fast similarity search can be performed. The key challenge of MH is to learn effective hash functions for multiple modalities efficiently from the provided information.

In this chapter, we study a simple case of multimodal hashing, in which the data from different modalities are aligned. For example, suppose there are two modalities, i.e., image and text, if each image has been aligned with one and only one text article and vice versa, we consider the data to be aligned. The alignment can be determined by applications at hand, e.g, an image and a text can be paired if they are referring to the same object. To learn MH functions for these data, we first give a basic model which learns hash functions through spectral analysis of the correlation between modalities. Besides the basic method which can only handle vectorial data and is linear, we provide a kernel extension to handle nonvectorial data and incorporate nonlinearity. We further incorporate *Laplacian* regularization for situations in which side information is also available in the data.

The rest of this chapter is organized as follows. In Section 4.2, we introduce some related work. We then present our model, *spectral multimodal hashing*, in Section 4.3. Empirical studies conducted on real-world data sets are presented in Section 4.4, before we conclude this chapter in Section 4.5.

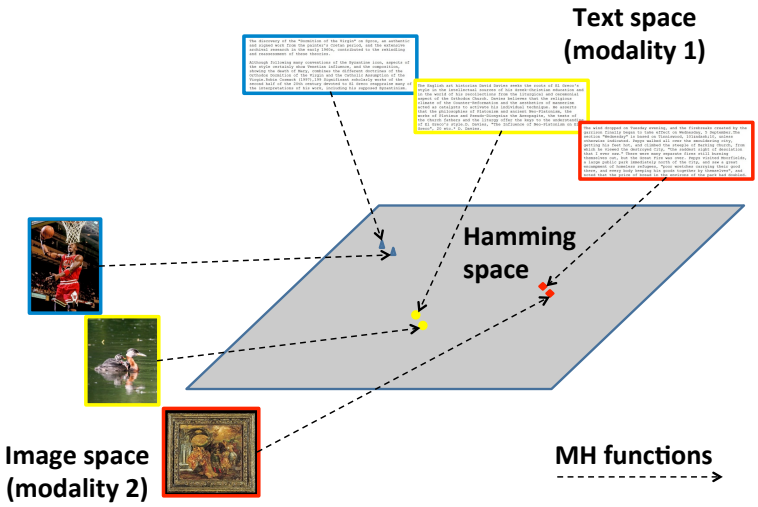


Figure 4.1: Illustration of the multimodal hashing framework. Under this framework, similar documents (with bounding boxes of the same color) of different modalities are hashed to nearby points in the Hamming space whereas dissimilar documents (with bounding boxes of different colors) are hashed to points far apart.

4.2 Related Work

Under the framework of multimodal hashing, we first introduce a recent work called cross modal similarity sensitive hashing (CMSSH) (Bronstein et al., 2010), which is, to the best of our knowledge, the first work on multimodal hashing.

Suppose we have two sets of N data points each from two modalities (*a.k.a.* feature spaces), $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^{D_x}\}_{i=1}^N$ and $\mathcal{Y} = \{\mathbf{y}_i \in \mathbb{R}^{D_y}\}_{i=1}^N$, and the corresponding points $(\mathbf{x}_i, \mathbf{y}_i)$ are paired. For applications studied in this paper, a pair $(\mathbf{x}_i, \mathbf{y}_i)$ may represent a multimedia document where $\mathbf{x}_i$ is an image and $\mathbf{y}_i$ is the corresponding text article. For notational convenience, we denote the data sets as matrices $\mathbf{X} \in \mathbb{R}^{D_x \times N}$ and $\mathbf{Y} \in \mathbb{R}^{D_y \times N}$ where each column corresponds to a data point. Without loss of generality, we assume that $\mathbf{X}, \mathbf{Y}$ have been normalized to have zero mean.

CMSSH works as follows: Given a set of similar pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ and a set of dissimilar pairs $\{(\mathbf{x}_j, \mathbf{y}_j)\}$, where $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ belong to two different modalities, CMSSH constructs two mapping functions $\xi : \mathcal{X} \to \mathbb{H}^M$ and $\eta : \mathcal{Y} \to \mathbb{H}^M$ such that, with high probability, the Hamming distance is small for similar points and large for dissimilar points. Specifically, the mth bit of Hamming representation $\mathbb{H}^M$ for $\mathcal{X}$ and $\mathcal{Y}$ can be defined by two functions, ξ_m and η_m, which are parameterized by projections p_m and q_m, respectively. In their paper, ξ_m and η_m are assumed to have the form $\xi_m(\mathbf{x}) = \text{sgn}(\mathbf{p}_m^T\mathbf{x}+a_m)$ and $\eta_m(\mathbf{y}) = \text{sgn}(\mathbf{q}_m^T\mathbf{y}+b_m)$, where $\mathbf{p}_m$ and $\mathbf{q}_m$ are D_x- and D_y-dimensional

unit vectors, a_m and b_m are scalars.

A method based on boosting is used to learn the mapping functions. The algorithm is briefly described here. First, it initializes the weight of each point pair to $w_m(k) = 1/K$ where K is the total number of point pairs. Then, for the mth bit, it selects ξ_m and η_m that maximize the following objective function:

$$\sum\nolimits_{k=1}^{K} \left(w_m(k) s_k \operatorname{sgn}(\mathbf{p}_m^T \mathbf{x}_k + a_m) \operatorname{sgn}(\mathbf{q}_m^T \mathbf{y}_k + b_m) \right),$$

where $s_k = 1$ if the kth pair is a similar pair and $s_k = -1$ otherwise. Since maximizing the objective function above is difficult, the $\operatorname{sgn}(\cdot)$ operator and bias terms a_m, b_m are dropped to give the following approximate objective function for maximization:

$$\sum\nolimits_{k=1}^{K} w_m(k) s_k (\mathbf{p}_m^T \mathbf{x}_k)(\mathbf{q}_m^T \mathbf{y}_k) = \mathbf{p}_m^T \left(\sum\nolimits_{k=1}^{K} w_m(k) s_k \mathbf{x}_k \mathbf{y}_k^T \right) \mathbf{q}_m.$$

It is easy to see that the $\mathbf{p}_m$ and $\mathbf{q}_m$ that maximize the above objective are the largest left and right singular vectors of $\mathbf{C} = \sum_{k=1}^{K} w_m(k) s_k \mathbf{x}_k \mathbf{y}_k^T$. After obtaining $\mathbf{p}_m$ and $\mathbf{q}_m$, the algorithm updates the weights with the update rule $w_{m+1}(k) = w_m(k) \exp(-s_k \xi_m(\mathbf{x}) \eta_m(\mathbf{y}))$ and then proceeds to learn $\mathbf{p}_{m+1}, \mathbf{q}_{m+1}$ for the $(m+1)$st bit.

Roughly speaking, CMSSH tries to map similar points to similar codes and dissimilar points to different codes by exploiting pairwise relations across different modalities. However, it ignores relational information within the same modality which could be very useful for hash function learning (Weiss et al., 2008; He et al., 2010). Furthermore, CMSSH can only handle vectorial data which might not be available in many applications.

Recently, Kumar *et al.* extended spectral hashing (Weiss et al., 2008) to the multi-view case, leading to a method called cross-view hashing (CVH) (Kumar & Udupa, 2011). The objective of CVH is to minimize the inter-view and intra-view Hamming distances for similar points and maximize those for dissimilar points. The optimization problem is relaxed to several generalized eigenvalue problems which can be solved by off-the-shelf methods.

4.3 Spectral Multimodal Hashing

In this section, we first formulate the multimodal hashing problem as a discrete embedding problem and show that it can be approximately solved by spectral decomposition followed by thresholding, which is similar to spectral hashing (Weiss et al., 2008) for unimodal data. But unlike spectral hashing, our focus is multimodal data, which are often encountered in a vast range of multimedia applications. Therefore, we call the proposed method *spectral multimodal hashing* (SMH). In the following, we first give a basic SMH model in Section 4.3.1 and then present the other two models as extensions in Section 4.3.2.

4.3.1 Formulation

Let there be two data matrices $\mathbf{X}^{D_x \times N}$ and $\mathbf{Y}^{D_y \times N}$ from different modalities and the corresponding points $(\mathbf{x}_i, \mathbf{y}_i)$ be paired. For applications studied in this paper, a pair $(\mathbf{x}_i, \mathbf{y}_i)$ may represent a multimedia document where $\mathbf{x}_i$ is an image and $\mathbf{y}_i$ is the corresponding text article. Without loss of generality, we assume that $\mathbf{X}, \mathbf{Y}$ have been normalized to have zero mean. We want to learn two sets of hash functions $\{h_m\}_{m=1}^M$ and $\{g_m\}_{m=1}^M$ to give M-bit binary codes of $\mathbf{X}$ and $\mathbf{Y}$, respectively.

In this paper, we use thresholded linear projection to define the hash functions. More specifically, the mth hash functions for both modalities are defined as follows:

$$h_m(\mathbf{x}) = \text{sgn}(\mathbf{x}^T \mathbf{w}_x^{(m)} + t_x), \quad g_m(\mathbf{y}) = \text{sgn}(\mathbf{y}^T \mathbf{w}_y^{(m)} + t_y),$$

where $\mathbf{w}_x^{(m)} \in \mathbb{R}^{D_x}, \mathbf{w}_y^{(m)} \in \mathbb{R}^{D_y}$ correspond to two projection directions. The corresponding Hamming bits can be obtained as

$$b_m(\mathbf{x}) = \frac{1 + h_m(\mathbf{x})}{2}, \quad b_m(\mathbf{y}) = \frac{1 + g_m(\mathbf{y})}{2}. \tag{4.1}$$

Let the binary vectors $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_M(\mathbf{x}))^T$ and $\mathbf{g}(\mathbf{y}) = (g_1(\mathbf{y}) \ldots, g_M(\mathbf{y}))^T$ denote the projections of points $\mathbf{x}$ and $\mathbf{y}$. The goal of our basic SMH model is to seek the projections that maximize the correlation between variables in the projected space (Hamming space). Intuitively, two hash codes are more correlated in the Hamming space if the corresponding points in the original space are similar and less correlated otherwise. Moreover, the hash codes should be balanced in the sense that each bit has equal chance of being 1 and -1 and the hash bits should be independent of each other (Weiss et al., 2008). As a result, SMH can be formulated as the following constrained optimization problem:

$$\begin{aligned} \max_{\{\mathbf{w}_x^{(m)}, \mathbf{w}_y^{(m)}\}_{m=1}^M} \quad & \frac{\mathbb{E}(\mathbf{h}^T \mathbf{g})}{\sqrt{\mathbb{E}(\mathbf{h}^T \mathbf{h}) \mathbb{E}(\mathbf{g}^T \mathbf{g})}} \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^N h_m(\mathbf{x}_i) = 0, \ m = 1, \ldots, M \\ & \textstyle\sum_{i=1}^N g_m(\mathbf{y}_i) = 0, \ m = 1, \ldots, M \\ & \textstyle\sum_{i=1}^N h_m(\mathbf{x}_i) h_n(\mathbf{x}_i) = 0, \ \forall m \neq n \\ & \textstyle\sum_{i=1}^N g_m(\mathbf{y}_i) g_n(\mathbf{y}_i) = 0, \ \forall m \neq n, \end{aligned} \tag{4.2}$$

where the expectation is taken with respect to the data distribution in the corresponding feature space. This problem is difficult to solve even without the constraints since the objective function is non-differentiable. Moreover, the balancing constraints make the problem NP-hard (Weiss et al., 2008).

Similar to (Wang et al., 2010a), we relax the problem by dropping the sgn(·) operator, the thresholds t_x and t_y and the balancing constraints. Instead, we implicitly enforce the constraints by preprocessing the data through mean-centering. Hence we arrive at the following optimization problem for one bit:[1]

$$\begin{aligned} \max_{\mathbf{w}_x, \mathbf{w}_y} \quad & \mathbb{E}(\mathbf{w}_x^T \mathbf{x}\, \mathbf{w}_y^T \mathbf{y}) \\ \text{s.t.} \quad & \mathbb{E}((\mathbf{w}_x^T \mathbf{x})^2) = 1,\, \mathbb{E}((\mathbf{w}_y^T \mathbf{y})^2) = 1, \end{aligned} \tag{4.3}$$

which in fact is the standard form of *canonical correlation analysis* (CCA) (Hotelling, 1936). Approximating the expectation by empirical expectation, we rewrite Problem (4.3) as follows:

$$\begin{aligned} \max_{\mathbf{w}_x, \mathbf{w}_y} \quad & \mathbf{w}_x \mathbf{C}_{xy} \mathbf{w}_y \\ \text{s.t.} \quad & \mathbf{w}_x \mathbf{C}_{xx} \mathbf{w}_x = 1,\, \mathbf{w}_y \mathbf{C}_{yy} \mathbf{w}_y = 1, \end{aligned} \tag{4.4}$$

where $\mathbf{C}_{xy} = \frac{1}{N}\mathbf{X}\mathbf{Y}^T$, $\mathbf{C}_{xx} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$, and $\mathbf{C}_{yy} = \frac{1}{N}\mathbf{Y}\mathbf{Y}^T$.

This problem is equivalent to the following generalized eigenvalue problem:

$$\mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{xy}^T \mathbf{w}_x = \lambda^2 \mathbf{C}_{xx} \mathbf{w}_x. \tag{4.5}$$

The solution $\mathbf{w}_x$ is the eigenvector that corresponds to the largest eigenvalue. With the $\mathbf{w}_x$ thus computed, we can compute $\mathbf{w}_y$ as

$$\mathbf{w}_y = \frac{1}{\lambda}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\mathbf{w}_x. \tag{4.6}$$

With projection vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ computed, one common approach of getting the binary codes is simply using the sgn(·) operator. However, this may separate the points located near the boundary, impairing the model especially when the data distribution is dense in that area. To overcome this shortcoming, we use two thresholds, a fixed threshold of zero and a learned threshold, to get the binary codes.

The learning-based threshold can be obtained as follows. For each projection, we first divide the range of projected values into N_b bins and then calculate the relative data density of each bin as $P_t = N_t/N, t = 1, \ldots, N_b$, with N_t stands for the number of points in the tth bin. The cost of cutting the tth bin is defined as follows,

$$C_t = \left(\sum\nolimits_{\hat{t}=1}^{t-1} P_{\hat{t}}\right)^2 + \left(\sum\nolimits_{\hat{t}=t+1}^{N_b} P_{\hat{t}}\right)^2 + P_t,$$

which measures the relative density of the tth bin and the relative density of its both sides. Intuitively, if C_t is small, a boundary cutting through the tth bin will separate

[1]For notational simplicity, we omit the indices of the hash functions.

a sparse area and make the points located evenly at its both sides. Actually, C_t is an adapted surrogate of the average size of a proper hash bucket which should be as small as possible for nearest neighbor search (Cayton & Dasgupta, 2007). We then use the center of the bin with the smallest C_t as the threshold and denote it as t_x or t_y.

Now we are ready to generate binary bits with 0 and t_x. For example, given $\mathbf{w}_x$ and $\mathbf{x}^*$, we have

$$h_1(\mathbf{x}^*) = \text{sgn}(\mathbf{w}_x^T\mathbf{x}^*), \quad h_2(\mathbf{x}^*) = \text{sgn}(\mathbf{w}_x^T\mathbf{x}^* - t_x), \tag{4.7}$$

and given $\mathbf{w}_y$ and $\mathbf{y}^*$, we have

$$g_1(\mathbf{y}^*) = \text{sgn}(\mathbf{w}_y^T\mathbf{y}^*), \quad g_2(\mathbf{y}^*) = \text{sgn}(\mathbf{w}_y^T\mathbf{y}^* - t_y). \tag{4.8}$$

The basic SMH algorithm is summarized in Algorithm 4.1.

Algorithm 4.1 Algorithm of SMH

Input:
$\mathbf{X}$, $\mathbf{Y}$ – data matrices
M – number of hash functions
Procedure:
Compute $\mathbf{C}_{xx}, \mathbf{C}_{xy}, \mathbf{C}_{yy}$.
Obtain M eigenvectors corresponding to the M largest eigenvalues of the generalized eigenvalue problem (4.5) as $\mathbf{w}_x$'s.
Obtain the corresponding $\mathbf{w}_y$'s using Equation (4.6).
Learn thresholds t_x and t_y.
Obtain the hash codes of points $\mathbf{x}^*$ and $\mathbf{y}^*$ using Equations (4.7), (4.8) & (4.1).

4.3.2 Extensions

Kernel SMH

The SMH model presented in the previous subsection has two limitations. First, it can only handle vectorial data. Second, the projection before thresholding is linear. In this subsection, we propose a kernel extension of SMH, abbreviated as KSMH thereafter, to overcome these limitations.

Let $\mathcal{K}(\cdot,\cdot)$ be a valid kernel function and $\phi(\cdot)$ be the corresponding function that maps data points in the original input space to the kernel-induced feature space. In the sequel, we use $\mathbf{\Phi}(\mathbf{X}) = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)]$ and $\mathbf{\Phi}(\mathbf{Y}) = [\phi(\mathbf{y}_1), \ldots, \phi(\mathbf{y}_N)]$ to denote the data matrices in the kernel-induced feature space.

Taking the kernel approach (Schölkopf et al., 2001)(Kulis & Darrell, 2009), we represent $\mathbf{w}_x$ and $\mathbf{w}_y$ as linear combinations of two groups of landmark points in the kernel-induced feature space, i.e.,

$$\mathbf{w}_x = \mathbf{\Phi}(\hat{\mathbf{X}})^T\boldsymbol{\alpha}, \quad \mathbf{w}_y = \mathbf{\Phi}(\hat{\mathbf{Y}})^T\boldsymbol{\beta},$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{D_x \times P}$ and $\hat{\mathbf{X}} \in \mathbb{R}^{D_x \times P}$ are two landmark sets, in which the points are randomly chosen from $\mathbf{X}$ and $\mathbf{Y}$, respectively. We note that although the landmark points should be sampled from the corresponding data distribution and be sufficiently representative, it is enough in practice to select the landmarks randomly from the training set. $\boldsymbol{\alpha} \in \mathbb{R}^{P \times 1}, \boldsymbol{\beta} \in \mathbb{R}^{P \times 1}$ are combination coefficients. To reduce the computational cost, P is usually a small number compared to N.

The objective function of Problem (4.3) can now be rewritten as

$$\frac{\boldsymbol{\alpha}^T \mathbf{K}_{\hat{x}x} \mathbf{K}_{y\hat{y}} \boldsymbol{\beta}}{\sqrt{\boldsymbol{\alpha}^T \mathbf{K}_{\hat{x}x} \mathbf{K}_{x\hat{x}} \boldsymbol{\alpha} \boldsymbol{\beta}^T \mathbf{K}_{\hat{y}y} \mathbf{K}_{y\hat{y}} \boldsymbol{\beta}}}, \tag{4.9}$$

where $\mathbf{K}_{\hat{x}x} = \mathbf{K}_{x\hat{x}}^T = \boldsymbol{\Phi}(\hat{\mathbf{X}})^T \boldsymbol{\Phi}(\mathbf{X})$ and $\mathbf{K}_{\hat{y}y} = \mathbf{K}_{y\hat{y}}^T = \boldsymbol{\Phi}(\hat{\mathbf{Y}})^T \boldsymbol{\Phi}(\mathbf{Y})$.

Since the objective function above can lead to degenerate solutions as discussed in (Hardoon et al., 2004), we penalize the norms of $\mathbf{w}_x$ and $\mathbf{w}_y$ in the denominator of (4.9) and arrive at the following alternative form:

$$\frac{\boldsymbol{\alpha}^T \mathbf{K}_{\hat{x}x} \mathbf{K}_{y\hat{y}} \boldsymbol{\beta}}{\sqrt{\boldsymbol{\alpha}^T (\mathbf{K}_{\hat{x}x} \mathbf{K}_{x\hat{x}} + \kappa \mathbf{K}_{\hat{x}\hat{x}}) \boldsymbol{\alpha} \boldsymbol{\beta}^T (\mathbf{K}_{\hat{y}y} \mathbf{K}_{y\hat{y}} + \kappa \mathbf{K}_{\hat{y}\hat{y}}) \boldsymbol{\beta}}}, \tag{4.10}$$

where $\mathbf{K}_{\hat{x}\hat{x}} = \boldsymbol{\Phi}(\hat{\mathbf{X}})^T \boldsymbol{\Phi}(\hat{\mathbf{X}}), \mathbf{K}_{\hat{y}y} = \boldsymbol{\Phi}(\hat{\mathbf{Y}})^T \boldsymbol{\Phi}(\hat{\mathbf{Y}})$ and $\kappa > 0$ is a regularization parameter.

After some simple relaxations and manipulations similar to SMH, $\boldsymbol{\alpha}$ can be obtained by solving the following generalized eigenvalue problem:

$$\mathbf{K}_{\hat{x}x} \mathbf{K}_{y\hat{y}} (\mathbf{K}_{\hat{y}y} \mathbf{K}_{y\hat{y}} + \kappa \mathbf{K}_{\hat{y}\hat{y}})^{-1} \mathbf{K}_{\hat{y}y} \mathbf{K}_{x\hat{x}} \boldsymbol{\alpha} = \lambda^2 (\mathbf{K}_{\hat{x}x} \mathbf{K}_{x\hat{x}} + \kappa \mathbf{K}_{\hat{x}\hat{x}}) \boldsymbol{\alpha}. \tag{4.11}$$

After obtaining $\boldsymbol{\alpha}$, we compute

$$\boldsymbol{\beta} = \frac{1}{\lambda} (\mathbf{K}_{\hat{y}y} \mathbf{K}_{y\hat{y}} + \kappa \mathbf{K}_{\hat{y}\hat{y}})^{-1} \mathbf{K}_{\hat{y}y} \mathbf{K}_{x\hat{x}} \boldsymbol{\alpha}. \tag{4.12}$$

We can also learn the thresholds t_x and t_y using the same approach presented in the last section. For any new point $\mathbf{x}^*$, two bits of binary code can be obtained as

$$h_1(\mathbf{x}^*) = \text{sgn}(\mathbf{k}_{x^*}^T \boldsymbol{\alpha}), \quad h_2(\mathbf{x}^*) = \text{sgn}(\mathbf{k}_{x^*}^T \boldsymbol{\alpha} - t_x) \tag{4.13}$$

and for $\mathbf{y}^*$, we have

$$g_1(\mathbf{y}^*) = \text{sgn}(\mathbf{k}_{y^*}^T \boldsymbol{\beta}), \quad g_2(\mathbf{y}^*) = \text{sgn}(\mathbf{k}_{y^*}^T \boldsymbol{\beta} - t_y), \tag{4.14}$$

where $\mathbf{k}_{y^*} = \boldsymbol{\Phi}(\hat{\mathbf{X}})^T \phi(\mathbf{x}^*)$ and $\mathbf{k}_{y^*} = \boldsymbol{\Phi}(\hat{\mathbf{Y}})^T \phi(\mathbf{y}^*)$.

The algorithm of KSMH is summarized in Algorithm 4.2.

Algorithm 4.2 Algorithm of KSMH

Input:
$\mathbf{X}$, $\mathbf{Y}$ – data matrices
$\mathcal{K}(\cdot,\cdot)$ – kernel function
M – number of hash functions
κ – regularization parameter
Procedure:
Compute $\mathbf{K}_{\hat{x}x}, \mathbf{K}_{\hat{y}y}, \mathbf{K}_{\hat{x}\hat{x}}, \mathbf{K}_{\hat{y}\hat{y}}$.
Obtain M eigenvectors corresponding to the M largest eigenvalues of the generalized eigenvalue problem (4.11) as $\boldsymbol{\alpha}$'s.
Obtain the corresponding $\boldsymbol{\beta}$'s using Equation (4.12).
Learn thresholds t_x and t_y.
Obtain the hash codes of points $\mathbf{x}^*$ and $\mathbf{y}^*$ using Equations (4.13), (4.14) & (4.1).

Regularized kernel SMH

Both SMH and KSMH proposed above aim at maximizing the correlation between variables in different modalities while ignoring the relational information within each modularity. Moreover, it is unclear how to make use of side information such as labels in the two models in case such information is available in the data.

Inspired by (Blaschko et al., 2008), we further extend KSMH by adding two *Laplacian* regularization terms to the objective (4.10). We name this new model RKSMH, whose objective is:

$$\frac{\boldsymbol{\alpha}^T \mathbf{K}_{\hat{x}x} \mathbf{K}_{y\hat{y}} \boldsymbol{\beta}}{\sqrt{\boldsymbol{\alpha}^T (\mathbf{K}_{\hat{x}x} \mathbf{R}_x \mathbf{K}_{x\hat{x}} + \kappa \mathbf{K}_{\hat{x}\hat{x}}) \boldsymbol{\alpha} \boldsymbol{\beta}^T (\mathbf{K}_{\hat{y}y} \mathbf{R}_y \mathbf{K}_{y\hat{y}} + \kappa \mathbf{K}_{\hat{y}\hat{y}}) \boldsymbol{\beta}}},$$

where $\mathbf{R}_x = (\mathbf{I} + \gamma \mathcal{L}_x), \mathbf{R}_y = (\mathbf{I} + \gamma \mathcal{L}_y)$, $\gamma > 0$ is a parameter controlling the impact of regularization, and $\mathcal{L}_x, \mathcal{L}_y$ are graph *Laplacians* (Chung, 1996) that incorporate some information about $\mathbf{X}$ and $\mathbf{Y}$, respectively. We note that the *Laplacian* matrix is defined as $\mathcal{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with $D(i,i) = \sum_{j=1}^{N} W(i,j)$.[2] We note that the *Laplacian* matrix can be computed efficiently by using an anchor graph (Liu et al., 2010).

The regularizers $\mathbf{R}_x$ and $\mathbf{R}_y$ not only can exploit relational information of a single modality but can also incorporate into the model side information when it is available. For example, $\mathcal{L}_x$ can incorporate structural or geometric information in the input space $\mathbf{X}$ by defining $\mathbf{W}_x$ as

$$W_x(i,j) = \begin{cases} \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma^2}\right) & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

where $d(\cdot,\cdot)$ is the Euclidean distance between two points and σ is a user-specified width parameter. In our experiments, we regard two points as neighbors if either one is among

[2]To avoid being cluttered, we omit the subscripts here.

the K nearest neighbors of the other one in the feature space. We call this type of *Laplacian* the feature-based *Laplacian*.

$\mathcal{L}_x$ can also be used to incorporate side information such as labels by defining $\mathbf{W}_x$ as

$$W_x(i,j) = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ have the same label} \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

We call this *Laplacian* the label-based *Laplacian* thereafter. Note that $\mathcal{L}_y$ can be defined similarly.

In RKSMH, $\boldsymbol{\alpha}$ can be obtained by solving the following generalized eigenvalue problem:

$$\mathbf{K}_{\hat{x}x}\mathbf{K}_{y\hat{y}}(\mathbf{K}_{\hat{y}y}\mathbf{R}_y\mathbf{K}_{y\hat{y}} + \kappa\mathbf{K}_{\hat{y}\hat{y}})^{-1}\mathbf{K}_{\hat{y}y}\mathbf{K}_{x\hat{x}}\boldsymbol{\alpha} = \lambda^2(\mathbf{K}_{\hat{x}x}\mathbf{R}_x\mathbf{K}_{x\hat{x}} + \kappa\mathbf{K}_{\hat{x}\hat{x}})\boldsymbol{\alpha}, \tag{4.17}$$

and $\boldsymbol{\beta}$ can be computed as

$$\boldsymbol{\beta} = \frac{1}{\lambda}(\mathbf{K}_{\hat{y}y}\mathbf{R}_y\mathbf{K}_{y\hat{y}} + \kappa\mathbf{K}_{\hat{y}\hat{y}})^{-1}\mathbf{K}_{\hat{y}y}\mathbf{K}_{x\hat{x}}\boldsymbol{\alpha}. \tag{4.18}$$

The thresholding procedure of RKSMH is the same as those of KSMH and SMH. The algorithm is summarized in Algorithm 4.3.

Algorithm 4.3 Algorithm of RKSMH

Input:
$\mathbf{X}$, $\mathbf{Y}$ – data matrices
$\mathcal{K}(\cdot,\cdot)$ – kernel function
M – number of hash functions
κ, γ – regularization parameters
Procedure:
Compute $\mathbf{K}_{\hat{x}x}, \mathbf{K}_{\hat{y}y}, \mathbf{K}_{\hat{x}\hat{x}}, \mathbf{K}_{\hat{y}\hat{y}}, \mathbf{R}_x, \mathbf{R}_y$.
Obtain M eigenvectors corresponding to the M largest eigenvalues of the generalized eigenvalue problem (4.17) as $\boldsymbol{\alpha}$'s.
Obtain the corresponding $\boldsymbol{\beta}$'s using Equation (4.18).
Learn thresholds t_x and t_y.
Obtain the hash codes of points $\mathbf{x}^*$ and $\mathbf{y}^*$ using Equations (4.13), (4.14) & (4.1).

Beyond two modalities

Our SMH models can easily accommodate more than two modalities, using the corresponding extensions of CCA and KCCA (Hardoon et al., 2004)(Blaschko et al., 2008).

Taking SMH for example, suppose we have K modalities and want to learn K projection vectors $\{\mathbf{w}_1, \cdots, \mathbf{w}_K\}$, we solve the following generalized eigenvalue problem:

$$\begin{pmatrix} \mathbf{C}_{11} & \cdots & \mathbf{C}_{1K} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{K1} & \cdots & \mathbf{C}_{KK} \end{pmatrix} \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_K \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{C}_{11} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}_{KK} \end{pmatrix} \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_K \end{pmatrix},$$

where $\mathbf{C}_{ij}$ is the covariance matrix between modalities i and j, and $\mathbf{C}_{ij} = \mathbf{C}_{ji}^T$.

For KSMH, we select landmark points from each modality and index them with $\{\hat{1}, \cdots, \hat{K}\}$. The corresponding eigenvalue problem, for projection vectors $\{\alpha_1, \cdots, \alpha_K\}$, is:

$$\begin{pmatrix} \mathbf{K}_{\hat{1}1}\mathbf{K}_{1\hat{1}} & \cdots & \mathbf{K}_{\hat{1}1}\mathbf{K}_{K\hat{K}} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_{\hat{K}K}\mathbf{K}_{1\hat{1}} & \cdots & \mathbf{K}_{\hat{K}K}\mathbf{K}_{K\hat{K}} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_K \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{K}_{\hat{1}1}\mathbf{K}_{1\hat{1}} + \kappa\mathbf{K}_{\hat{1}\hat{1}} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{K}_{\hat{K}K}\mathbf{K}_{K\hat{K}} + \kappa\mathbf{K}_{\hat{K}\hat{K}} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_K \end{pmatrix},$$

where $\mathbf{K}_{\hat{i}i}$ is the kernel matrix between the landmark points and the training data points and $\mathbf{K}_{\hat{i}\hat{i}}$ is the kernel matrix for the landmark points, for the ith modality. The generalized eigenvalue problems for RKSMH are similar, and we omit them here due to space limitations.

With the projection vectors learned, we can apply the same thresholding process to each modality and get the binary codes easily.

4.4 Experiments

We conduct several experiments to compare SMH and its extensions with some other related methods. Through the experiments, we want to answer the following questions for each method:

1. How does SMH perform when compared with other state-of-the-art hashing models on crossmodal retrieval task?
2. How does SMH perform when compared with other state-of-the-art hashing models on unimodal retrieval task?

4.4.1 Data Sets

In our experiments, we use two publicly available data sets that are, to the best of our knowledge, the only two up-to-date public data sets involving multiple modalities at large scale.

The first data set, named *Wiki*, is based on a set of Wikipedia featured articles provided by (Rasiwasia et al., 2010).[3] It contains a total of 2,866 documents (image-text pairs), each of which consists of an image and a text article. Each document is annotated with a label chosen from ten semantic classes. The data set has been split into a training set of 2,173 documents and a test set of 693 documents. The image representation scheme is based on

[3] `http://www.svcl.ucsd.edu/projects/crossmodal/`

Table 4.1: Characteristics of Data Sets

Data set	D_x	D_y	# of points	# of classes
Wiki	128	10	2,866	10
Flickr	500	1000	186,577	10

the popular *scale invariant feature transformation* (SIFT) (Lowe, 2004) with a codebook of 128 words. Representation of a text article is based on its probability distribution over topics derived from a *latent Dirichlet allocation* (LDA) model (Blei et al., 2003) with ten topics.

The second data set, named *Flickr* thereafter, is a subset of the NUS-WIDE database[4] which is based on images from Flickr.com (Chua et al., 2009). We prune the original data set and keep only the points belonging to at least one of the ten largest classes. The data set contains a total of 186,577 image-text pairs, each of which belongs to at least one of ten possible labels (*a.k.a.* concepts). The data set has been split into a training set of 185,577 pairs and a test set of 1,000 pairs. The images are represented by a 500-dimensional SIFT representation. The text is simply represented by the number of occurrences of the 1,000 most frequently used tags according to the image.

Some characteristics of the two data sets are summarized in Table 4.1.

4.4.2 Experimental Settings

To mimic real multimedia retrieval systems, we consider two tasks in our experiments: crossmodel and uni-modal retrieval. In cross-modal retrieval, the query and the database belong to different modalities, for example, an image is used as a query and a set of text is used as a database. In uni-modal retrieval, the query and the database belong to the same modality. For each task, we first train the models on the training set, and then use documents in the test set as queries and the training set as database.

We use two evaluation measures in both cases, namely, *mean average precision* (MAP) and precision at a fixed Hamming radius. MAP is a measure widely used by the information retrieval community (Baeza-Yates & Ribeiro-Neto, 1999; Rasiwasia et al., 2010). Specifically, the MAP for a set of queries is the mean of the *average precision* (AP) scores for each query, with

$$\mathrm{AP} = \frac{1}{L} \sum_{r=1}^{N} P(r) \times \delta(r),$$

where r is the rank position, N is the number of retrieved documents, $\delta(r)$ is a binary

[4] `http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm`

function that returns 1 if the document at rank position r is relevant[5] to the query and 0 otherwise, $P(r)$ is the precision of relevance at position r, and L is the total number of relevant documents in the retrieved set. MAP is sometimes referred to geometrically as the area under the precision-recall curve for a set of queries (Turpin & Scholer, 2006). Thus a larger value of MAP indicates a better performance. To get the precision at Hamming radius d, we first retrieve all the documents which have Hamming distance at most d to the query and then compute the precision of the retrieved documents. Similar to MAP, larger values of precision indicate better performance. In all experiments, we set the rank position $r = 100$ and the Hamming radius $d = 2$.

4.4.3 Results

In the following experiments, we randomly select $P = 500$ data points from the training set as landmarks for KSMH and RKSMH and repeat the process ten times. Hence for these two methods, we report the average results with the corresponding standard deviations. Moreover, linear kernel is used, Laplacians are defined based on labels and the parameters are set to $\kappa = 10^{-4}$, $\gamma = 0.1$ for the Wiki data set and $\gamma = 100$ for the Flickr data set. Besides, to reduce computational cost on the Flickr data set, we use a subset of 5,000 instances from the training set to train the models but the retrieval tasks are still conducted on the whole training set.

Comparison for crossmodel retrieval

We first compare the four multimodal hashing methods, i.e., CMSSH, SMH, KSMH and RKSMH, for crossmodel retrieval. The results for different code lengths M on the Wiki data set are reported in Table 4.2 & 4.3, and those on the Flickr data set are reported in Table 4.4 & 4.5.

Table 4.2: Performance comparison for Image-Text retrieval on Wiki

Method	Measure	Image query – Text database		
		$M = 4$	$M = 8$	$M = 16$
CMSSH	MAP	0.1660	0.1640	0.1751
	Precision	0.1150	0.1501	**0.3487**
SMH	MAP	0.1937	0.2290	0.2140
	Precision	0.1252	**0.1640**	0.2200
KSMH	MAP	0.1909 ± 0.0032	$\mathbf{0.2194 \pm 0.0052}$	0.2177 ± 0.0058
	Precision	$\mathbf{0.1253 \pm 0.0008}$	0.1635 ± 0.0045	0.2186 ± 0.0130
RKSMH	MAP	$\mathbf{0.1918 \pm 0.0021}$	0.2189 ± 0.0036	$\mathbf{0.2200 \pm 0.0046}$
	Precision	0.1219 ± 0.0008	0.1633 ± 0.0032	0.2172 ± 0.0084

[5]In the experiments, an image is relevant to a text if they share the same class label and vice versa.

Table 4.3: Performance comparison for Text-Image retrieval on Wiki

Method	Measure	Text query – Image database		
		$M = 4$	$M = 8$	$M = 16$
CMSSH	MAP	0.1928	0.1746	0.1950
	Precision	0.1142	0.1389	0.1320
SMH	MAP	**0.2208**	**0.2784**	**0.3494**
	Precision	0.1258	**0.1800**	**0.3047**
KSMH	MAP	0.2207 ± 0.0039	0.2765 ± 0.0052	0.3108 ± 0.0059
	Precision	$\mathbf{0.1264 \pm 0.0013}$	0.1782 ± 0.0071	0.2780 ± 0.0118
RKSMH	MAP	0.2088 ± 0.0038	0.2559 ± 0.0065	0.3171 ± 0.0075
	Precision	0.1228 ± 0.0008	0.1740 ± 0.0045	0.2774 ± 0.0106

Table 4.4: Performance comparison for Image-Text retrieval on Flickr

Method	Measure	Image query – Text database		
		$M = 4$	$M = 8$	$M = 16$
CMSSH	MAP	0.3723	0.3822	0.4100
	Precision	0.3458	0.3503	0.4104
SMH	MAP	0.3463	0.3872	0.4159
	Precision	0.3451	0.4130	0.4100
KSMH	MAP	$\mathbf{0.4604 \pm 0.0116}$	0.4747 ± 0.0136	0.4718 ± 0.0066
	Precision	$\mathbf{0.3778 \pm 0.0052}$	0.4148 ± 0.0066	0.4390 ± 0.0111
RKSMH	MAP	0.4482 ± 0.0125	$\mathbf{0.4782 \pm 0.0052}$	$\mathbf{0.4865 \pm 0.0037}$
	Precision	0.3709 ± 0.0038	$\mathbf{0.4185 \pm 0.0065}$	$\mathbf{0.4690 \pm 0.0094}$

From the tables, we can see that all three SMH models outperform CMSSH by a large margin on both data sets. Among our three models, RKSMH performs the best on both data sets, indicating the effectiveness of *Laplacian* regularization. We also note that KSMH achieves performance similar to that of SMH on the Wiki data set and better performance than SMH on the Flickr data set, showing that the kernel extension is quite useful.

Comparison for unimodel retrieval

In this section, we compare the four multimodal and two well-known unimodel hashing-based methods for unimodel retrieval. It should be noted that multimodel hashing algorithms learn hash functions from both modalities whereas the unimodel hashing algorithms learn hash functions from only one modality. The results are summarized in Table 4.6 & 4.7 for the Wiki data set, and Table 4.8 & 4.9 for the Flickr data set.

Similar to the results of crossmodal retrieval, our models outperform CMSSH on both data sets and the performance gap is larger on the Flickr data set. As expected, RKSMH achieves the best performance among our methods and KSMH is better than SMH. Note

Table 4.5: Performance comparison for Text-Image retrieval on Flickr

Method	Measure	Text query – Image database		
		$M = 4$	$M = 8$	$M = 16$
CMSSH	MAP	**0.4824**	0.4829	0.4712
	Precision	0.3467	0.3616	**0.5286**
SMH	MAP	0.3590	0.4056	0.4520
	Precision	0.3447	**0.4346**	0.4460
KSMH	MAP	0.4683 ± 0.0146	0.4849 ± 0.0119	0.4860 ± 0.0102
	Precision	$\mathbf{0.3839 \pm 0.0054}$	0.4260 ± 0.0073	0.4537 ± 0.0114
RKSMH	MAP	0.4610 ± 0.0066	$\mathbf{0.5013 \pm 0.0081}$	$\mathbf{0.5098 \pm 0.0050}$
	Precision	0.3750 ± 0.0057	0.4241 ± 0.0082	0.4751 ± 0.0141

Table 4.6: Performance comparison for image retrieval on Wiki

Method	Measure	Image query – Image database		
		$M = 4$	$M = 8$	$M = 16$
SH	MAP	0.1559	0.1545	0.1552
	Precision	0.1084	0.1084	0.1083
CMSSH	MAP	0.1640	0.1683	0.1743
	Precision	0.1139	0.1171	0.1294
SMH	MAP	**0.1773**	**0.1930**	**0.1903**
	Precision	**0.1178**	**0.1352**	**0.1536**
KSMH	MAP	0.1759 ± 0.0014	0.1912 ± 0.0031	0.1892 ± 0.0019
	Precision	0.1173 ± 0.0004	0.1343 ± 0.0011	0.1533 ± 0.0029
RKSMH	MAP	0.1747 ± 0.0014	0.1848 ± 0.0020	0.1884 ± 0.0019
	Precision	0.1155 ± 0.0006	0.1324 ± 0.0007	0.1512 ± 0.0022

that our methods perform better than one state-of-the-art unimodal hashing methods, namely, *spectral hashing*, indicating that information from other modalities can help to learn good hash codes for unimodal retrieval. As a result, SMH, especially RKSMH, is also very useful for unimodal retrieval systems.

4.5 Conclusion

In this chapter, we have proposed spectral multimodal hashing (SMH) under the framework of multimodal hashing, the goal of which is to perform similarity search on data of multiple modalities. SMH learns the hash codes through spectral analysis of the modality correlation. Experimental results show that our SMH model outperforms the state-of-the-art methods.

In the future, we wish to relax the data alignment assumption of SMH and develop more general multimodal hashing methods. In addition, we would like to apply SMH to other applications such as multimodal medical image registration.

Table 4.7: Performance comparison for text retrieval on Wiki

Method	Measure	Text query – Text database		
		$M = 4$	$M = 8$	$M = 16$
SH	MAP	**0.3068**	0.3986	0.5590
	Precision	0.1084	0.1086	0.1721
CMSSH	MAP	0.3024	0.4737	0.5364
	Precision	0.1739	0.2752	0.4179
SMH	MAP	0.2850	0.4461	0.5563
	Precision	0.1358	0.2648	0.5704
KSMH	MAP	0.3066 ± 0.0220	0.4627 ± 0.0173	0.5590 ± 0.0068
	Precision	$\mathbf{0.1397 \pm 0.0041}$	0.2742 ± 0.0226	0.5741 ± 0.0217
RKSMH	MAP	0.2891 ± 0.0046	$\mathbf{0.5078 \pm 0.0046}$	$\mathbf{0.5697 \pm 0.0041}$
	Precision	0.1328 ± 0.0013	$\mathbf{0.2786 \pm 0.0144}$	$\mathbf{0.5927 \pm 0.0143}$

Table 4.8: Performance comparison for image retrieval on Flickr

Method	Measure	Image query – Image database		
		$M = 4$	$M = 8$	$M = 16$
SH	MAP	0.3743	0.3780	0.3793
	Precision	0.3449	0.3449	0.3449
CMSSH	MAP	0.4064	0.4262	0.4304
	Precision	0.3396	0.3376	0.3424
SMH	MAP	0.3753	0.4326	0.4388
	Precision	0.3450	0.3786	0.4075
KSMH	MAP	$\mathbf{0.4498 \pm 0.0061}$	0.4642 ± 0.0051	0.4606 ± 0.0034
	Precision	$\mathbf{0.3746 \pm 0.0059}$	$\mathbf{0.4095 \pm 0.0103}$	0.4342 ± 0.0126
RKSMH	MAP	0.4390 ± 0.0062	$\mathbf{0.4726 \pm 0.0055}$	$\mathbf{0.4783 \pm 0.0029}$
	Precision	0.3668 ± 0.0050	0.4020 ± 0.0070	$\mathbf{0.4403 \pm 0.0082}$

Table 4.9: Performance comparison for text retrieval on Flickr

Method	Measure	Text query – Text database		
		$M = 4$	$M = 8$	$M = 16$
SH	MAP	0.3753	0.3756	0.3752
	Precision	0.3449	0.3449	0.3449
CMSSH	MAP	0.4762	0.5197	**0.5832**
	Precision	0.3824	0.3962	0.4112
SMH	MAP	0.3769	0.4650	0.5031
	Precision	0.3449	0.3838	0.4356
KSMH	MAP	$\mathbf{0.4866 \pm 0.0135}$	0.5132 ± 0.0098	0.5177 ± 0.0095
	Precision	$\mathbf{0.3839 \pm 0.0062}$	0.4342 ± 0.0121	0.4760 ± 0.0112
RKSMH	MAP	0.4723 ± 0.0153	$\mathbf{0.5245 \pm 0.0103}$	0.5441 ± 0.0068
	Precision	0.3777 ± 0.0036	$\mathbf{0.4394 \pm 0.0073}$	$\mathbf{0.5117 \pm 0.0127}$

CHAPTER 5

MULTIMODAL HASHING FOR GRAPH DATA

5.1 Introduction

So far as we know, existing multimodal hashing methods including CMSSH and CVH have achieved successes in several applications but they also have some apparent limitations. First, both models can only deal with vectorial data which may not be available in some applications. Besides, they both involve eigendecomposition operations which may be very costly especially when the data dimensionality is high. Furthermore, CMSSH has been developed for shape retrieval and medical image alignment applications and CVH for people search applications in the natural language processing area. These applications are very different from those studied in this thesis.

Although spectral multimodal hashing has solved some of the issues mentioned above, it is also based on eigendecomposition and requires the data to be aligned. In this chapter, we study hashing-based similarity search in the context of multimodal graph data, in which pairwise similarity of data points is provided and different modalities are no longer aligned.

We propose a probabilistic latent factor model, called *multimodal latent binary embedding* (MLBE), to learn hash functions for multiple modalities. As a generative model, the hash codes are binary latent factors in a common Hamming space which determine the generation of both intra-modality and inter-modality similarities as observed either directly or indirectly. Although getting a full Bayesian solution is intractable, we devise an efficient algorithm for learning the binary latent factors based on *maximum a posteriori* (MAP) estimation. Compared to its counterparts (Bronstein et al., 2010; Kumar & Udupa, 2011), MLBE can:

(a) be interpreted easily in a principled manner;

(b) be extended easily;

(c) avoid overfitting via parameter learning;

(d) support efficient learning algorithms.

The remainder of this chapter is organized as follows. Section 5.2 presents the model formulation, the learning algorithm, some model extensions as well as discussion. Experimental validation of MLBE conducted using both synthetic data and two realistic data sets is presented in Section 5.3. Finally, Section 5.4 concludes the chapter.

5.2 Multimodal Latent Binary Embedding

5.2.1 Model Formulation

For simplicity of our presentation, we focus exclusively on the bimodal case in this chapter, but it is very easy to extend MLBE for more than two modalities. As a running example, we assume that the data come from two modalities $\mathcal{X}$ and $\mathcal{Y}$ corresponding to the image modality and text modality, respectively.

The observations in MLBE are intra-modality and inter-modality similarities. Specifically, there are two symmetric intra-modality similarity matrices $\mathbf{S}^x \in \mathbb{R}^{I \times I}$ and $\mathbf{S}^y \in \mathbb{R}^{J \times J}$, where I and J denote the number of data points in modality $\mathcal{X}$ and that in modality $\mathcal{Y}$, respectively. In case the observed data are only available in the form of feature vectors, different ways can be used to convert them into similarity matrices. For the image data $\mathcal{X}$, the similarities in $\mathbf{S}^x$ could be computed from the corresponding Euclidean distances between feature vectors. For the text data $\mathcal{Y}$, the similarities in $\mathbf{S}^y$ could be the cosine similarities between bag-of-words representations.

In addition, there is an inter-modality similarity matrix $\mathbf{S}^{xy} \in \{1, 0\}^{I \times J}$, where 1 and 0 denote similar and dissimilar relationships, respectively, between the corresponding entities. Note that it is common to specify cross-modality similarities this way, because it is very difficult if not impossible to specify real-valued cross-modality similarities objectively. The binary similarity values in $\mathbf{S}^{xy}$ can often be determined based on their semantics. Take multimedia retrieval for example, if an image and a text article are both for the same historic event, their similarity will be set to 1. Otherwise, their similarity will be 0.

Our probabilistic generative model has latent variables represented by several matrices. First, there are two sets of binary latent factors, $\mathbf{U} \in \{+1, -1\}^{I \times K}$ for $\mathcal{X}$ and $\mathbf{V} \in \{+1, -1\}^{J \times K}$ for $\mathcal{Y}$, where each row in $\mathbf{U}$ or $\mathbf{V}$ corresponds to one data point and can be interpreted as the hash code of that point. In addition, there are two intra-modality weighting matrices, $\mathbf{W}^x \in \mathbb{R}^{K \times K}$ for $\mathcal{X}$ and $\mathbf{W}^y \in \mathbb{R}^{K \times K}$ for $\mathcal{Y}$, and an inter-modality weighting variable $w > 0$. The basic assumption of MLBE is that the observed intra-modality and inter-modality similarities are generated from the binary latent factors, intra-modality weighting matrices and inter-modality weighting variable. Note that the

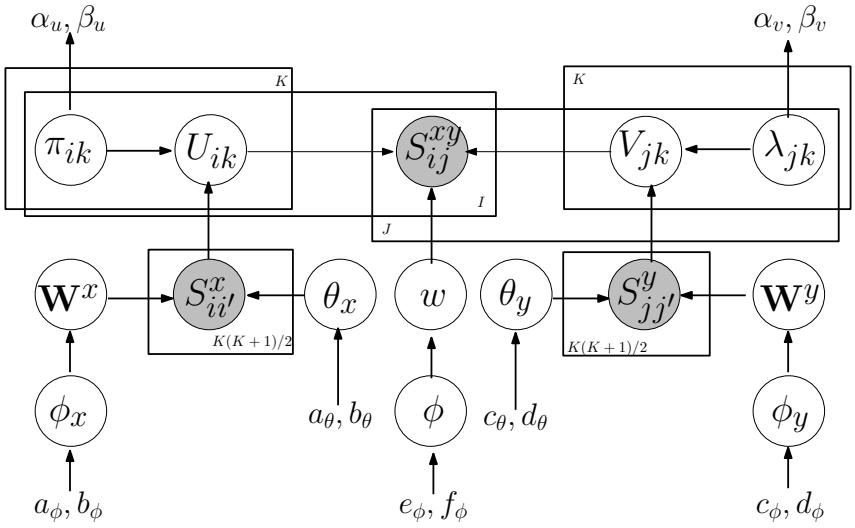


Figure 5.1: Graphical model representation of MLBE

real-valued weighting matrices and weighting variable are needed for generating the similarities because the values in the latent factors $\mathbf{U}$ and $\mathbf{V}$ are discrete.

The graphical model representation of MLBE is depicted in Figure 5.1, in which shaded nodes are used for observed variables and empty ones for latent variables as well as parameters which are also defined as random variables. The others are hyperparameters, which will be denoted collectively by Ω in the sequel for notational convenience.

We first consider the likelihood functions of MLBE. Given $\mathbf{U}, \mathbf{V}, \mathbf{W}^x, \mathbf{W}^y, \theta_x$ and θ_y, the conditional probability density functions of the intra-modality similarity matrices $\mathbf{S}^x$ and $\mathbf{S}^y$ are defined as

$$p(\mathbf{S}^x \mid \mathbf{U}, \mathbf{W}^x, \theta_x) = \prod_{i=1}^{I} \prod_{i'=1}^{I} \mathcal{N}(S_{ii'}^x \mid \mathbf{u}_i^T \mathbf{W}^x \mathbf{u}_{i'}, \frac{1}{\theta_x}),$$

$$p(\mathbf{S}^y \mid \mathbf{V}, \mathbf{W}^y, \theta_y) = \prod_{j=1}^{J} \prod_{j'=1}^{J} \mathcal{N}(S_{jj'}^y \mid \mathbf{v}_j^T \mathbf{W}^y \mathbf{v}_{j'}, \frac{1}{\theta_y}),$$

where $\mathbf{u}_i$ and $\mathbf{u}_{i'}$ denote the ith and i'th rows of $\mathbf{U}$, $\mathbf{v}_i$ and $\mathbf{v}_{j'}$ denote the jth and j'th rows of $\mathbf{V}$, and $\mathcal{N}(x \mid \mu, \sigma^2)$ is the probability density function of the univariate Gaussian distribution with mean μ and variance σ^2.

Given $\mathbf{U}, \mathbf{V}$ and w, the conditional probability mass function of the inter-modality similarity matrix $\mathbf{S}^{xy}$ is given by

$$p(\mathbf{S}^{xy} \mid \mathbf{U}, \mathbf{V}, w) = \prod_{i=1}^{I} \prod_{j=1}^{J} \left[\mathrm{Bern}(S_{ij}^{xy} \mid \gamma(w \mathbf{u}_i^T \mathbf{v}_j))\right]^{O_{ij}},$$

where $\mathrm{Bern}(x \mid \mu)$ is the probability mass function of the Bernoulli distribution with parameter μ, O_{ij} is an indicator variable which is equal to 1 if S_{ij}^{xy} is observed and 0

otherwise, and $\gamma(x) = 1/(1+\exp(-x))$ is the logistic sigmoid function to ensure that the parameter μ of the Bernoulli distribution is in the range $(0, 1)$.

To complete the model formulation, we also need to define prior distributions on the latent variables and hyperprior distributions on the parameters. For the matrix $\mathbf{U}$, we impose a prior independently and identically on each element of $\mathbf{U}$ as follows:[1]

$$p(U_{ik} \mid \pi_{ik}) = \text{Bern}(U_{ik} \mid \pi_{ik}),$$
$$p(\pi_{ik} \mid \alpha_u, \beta_u) = \text{Beta}(\pi_{ik} \mid \alpha_u, \beta_u),$$

where $\text{Beta}(\mu \mid a, b)$ is the probability density function of the beta distribution with hyperparameters a and b. This particular choice is mainly due to the computational advantage of using conjugate distributions so that we can integrate out π_{ik}, as a form of Bayesian averaging, to give the following prior distribution on $\mathbf{U}$:

$$p(\mathbf{U} \mid \alpha_u, \beta_u) = \prod_{i=1}^{I} \prod_{k=1}^{K} \text{Bern}(U_{ik} \mid \frac{\alpha_u}{\alpha_u + \beta_u}).$$

Similarly, we define the prior distribution on $\mathbf{V}$ as:

$$p(\mathbf{V} \mid \alpha_v, \beta_v) = \prod_{j=1}^{J} \prod_{k=1}^{K} \text{Bern}(V_{jk} \mid \frac{\alpha_v}{\alpha_v + \beta_v}).$$

For the weighting matrices $\mathbf{W}^x$ and $\mathbf{W}^y$, we impose Gaussian prior distributions on them:

$$p(\mathbf{W}^x \mid \phi_x) = \prod_{k=1}^{K} \prod_{d=k}^{K} \mathcal{N}(W^x_{kd} \mid 0, \frac{1}{\phi_x}),$$
$$p(\mathbf{W}^y \mid \phi_y) = \prod_{k=1}^{K} \prod_{d=k}^{K} \mathcal{N}(W^y_{kd} \mid 0, \frac{1}{\phi_y}).$$

The weighting variable w has to be strictly positive to enforce a positive relationship between the inner product of two hash codes and the inter-modality similarity. So we impose the half-normal prior distribution on w:

$$p(w \mid \phi) = \mathcal{HN}(w \mid \phi) = e^{-\frac{\phi}{2}w^2} \sqrt{\frac{2\phi}{\pi}}.$$

Because the parameters $\theta_x, \theta_y, \phi_x, \phi_y$ and ϕ are all random variables, we also impose hyperprior distributions on them. The gamma distribution is used for all these distribu-

[1] Conventionally, the Bernoulli distribution is defined for discrete random variables which take the value 1 for success and 0 for failure. Here, the discrete random variables take values from $\{-1, +1\}$ instead assuming an implicit linear mapping from $\{0, 1\}$.

tions:

$$
\begin{aligned}
p(\theta_x \mid a_\theta, b_\theta) &= \mathrm{Gam}(\theta_x \mid a_\theta, b_\theta), \\
p(\theta_y \mid c_\theta, d_\theta) &= \mathrm{Gam}(\theta_y \mid c_\theta, d_\theta), \\
p(\phi_x \mid a_\phi, b_\phi) &= \mathrm{Gam}(\phi_x \mid a_\phi, b_\phi), \\
p(\phi_y \mid c_\phi, d_\phi) &= \mathrm{Gam}(\phi_y \mid c_\phi, d_\phi), \\
p(\phi \mid e_\phi, f_\phi) &= \mathrm{Gam}(\phi \mid e_\phi, f_\phi),
\end{aligned}
$$

where $\mathrm{Gam}(\tau \mid a, b) = \frac{1}{\Gamma(a)} b^a \tau^{a-1} e^{-b\tau}$ denotes the probability density function of the gamma distribution with hyperparameters a and b, and $\Gamma(\cdot)$ is the gamma function.

5.2.2 Learning

With the probabilistic graphical model formulated in the previous subsection, we can now devise an algorithm to learn the binary latent factors $\mathbf{U}$ and $\mathbf{V}$ which give the hash codes we need. A fully Bayesian approach would infer the posterior distributions of $\mathbf{U}$ and $\mathbf{V}$, possibly using some sampling techniques. However, such methods are often computationally demanding. For computational efficiency, we devise an efficient alternating learning algorithm in this paper based on MAP estimation.

We first update U_{ik} while fixing all other variables. To find the MAP estimate of U_{ik}, we define a loss function with respect to U_{ik} in Definition 5.2.1:

Definition 5.2.1.

$$
\begin{aligned}
\mathcal{L}_{ik} = &- \frac{\theta_x}{2} \sum_{l \neq i}^{I} \left[\left(S_{il}^x - \mathbf{u}_l^T \mathbf{W}^x \mathbf{u}_i^+ \right)^2 - \left(S_{il}^x - \mathbf{u}_l^T \mathbf{W}^x \mathbf{u}_i^- \right)^2 \right] \\
&- \frac{\theta_x}{2} \left[\left(S_{ii}^x - \mathbf{u}_i^{+T} \mathbf{W}^x \mathbf{u}_i^+ \right)^2 - \left(S_{ii}^x - \mathbf{u}_i^{-T} \mathbf{W}^x \mathbf{u}_i^- \right)^2 \right] \\
&+ \sum_{j=1}^{J} O_{ij} \left[S_{ij}^{xy} \log \frac{\rho_{ij}^+}{\rho_{ij}^-} + (1 - S_{ij}^{xy}) \log \frac{1 - \rho_{ij}^+}{1 - \rho_{ij}^-} \right] + \log \frac{\alpha_u}{\beta_u},
\end{aligned}
$$

where $\mathbf{u}_i^+$ *is the* i*th row of* $\mathbf{U}$ *with* $U_{ik} = 1$, $\mathbf{u}_i^-$ *is the* i*th row of* $\mathbf{U}$ *with* $U_{ik} = -1$, $\rho_{ij}^+ = \gamma(w \mathbf{v}_j^T \mathbf{u}_i^+)$, *and* $\rho_{ij}^- = \gamma(w \mathbf{v}_j^T \mathbf{u}_i^-)$.

With $\mathcal{L}_{ik}$, we can state the following theorem:

Theorem 5.2.1. *The MAP solution of* U_{ik} *is equal to 1 if* $\mathcal{L}_{ik} \geq 0$ *and* -1 *otherwise.*

Proof 5.2.1. *To obtain the MAP solution of* U_{ik}*, it suffices to compare the following two posterior probabilities:*

$$
\begin{aligned}
p_+ &= \Pr(U_{ik} = 1 \mid U_{-ik}, \mathbf{V}, \mathbf{W}^x, w, \mathbf{S}^x, \mathbf{S}^{xy}, \theta_x), \\
p_- &= \Pr(U_{ik} = -1 \mid U_{-ik}, \mathbf{V}, \mathbf{W}^x, w, \mathbf{S}^x, \mathbf{S}^{xy}, \theta_x).
\end{aligned}
$$

Specifically, we compute the log ratio of the two probabilities, which is larger than or equal to zero if $p_+ \geq p_-$ and smaller than zero otherwise. The log ratio can be evaluated as follows:

$$\begin{aligned}
&\log \frac{\Pr(U_{ik}=1 \mid U_{-ik}, \mathbf{V}, \mathbf{W}^x, w, \mathbf{S}^x, \mathbf{S}^{xy}, \theta_x)}{\Pr(U_{ik}=-1 \mid U_{-ik}, \mathbf{V}, \mathbf{W}^x, w\mathbf{S}^x, \mathbf{S}^{xy}, \theta_x)} \\
&\quad = \log \frac{\Pr(\mathbf{S}^x \mid U_{ik}=1, U_{-ik}, \mathbf{W}^x, \theta_x)}{\Pr(\mathbf{S}^x \mid U_{ik}=-1, U_{-ik}, \mathbf{W}^x, \theta_x)} \\
&\quad + \log \frac{\Pr(\mathbf{S}^{xy} \mid U_{ik}=1, U_{-ik}, w, \mathbf{V})}{\Pr(\mathbf{S}^{xy} \mid U_{ik}=-1, U_{-ik}, w, \mathbf{V})} \\
&\quad + \log \frac{\Pr(U_{ik}=1 \mid \alpha_u, \beta_u)}{\Pr(U_{ik}=-1 \mid \alpha_u, \beta_u)} \\
&\quad = -\frac{\theta_x}{2} \sum_{l \neq i}^{I} \left[\left(S_{il} - \mathbf{u}_l^T \mathbf{W}^x \mathbf{u}_i^+\right)^2 - \left(S_{il} - \mathbf{u}_l^T \mathbf{W}^x \mathbf{u}_i^-\right)^2 \right] \\
&\quad - \frac{\theta_x}{2} \left[\left(S_{ii} - \mathbf{u}_i^{+T} \mathbf{W}^x \mathbf{u}_i^+\right)^2 - \left(S_{ii} - \mathbf{u}_i^{-T} \mathbf{W}^x \mathbf{u}_i^-\right)^2 \right] \\
&\quad + \sum_{j=1}^{J} O_{ij} \left[S_{ij}^{xy} \log \frac{\rho_{ij}^+}{\rho_{ij}^-} + (1 - S_{ij}^{xy}) \log \frac{1-\rho_{ij}^+}{1-\rho_{ij}^-} \right] + \log \frac{\alpha_u}{\beta_u},
\end{aligned}$$

where U_{-ik} denotes all the elements in $\mathbf{U}$ except U_{ik}. The log ratio thus computed gives exactly $\mathcal{L}_{ik}$. This completes the proof.

Similarly, we have Definition 5.2.2 and Theorem 5.2.2 for the MAP estimation of $\mathbf{V}$. The proof of Theorem 5.2.2 is similar and so it is omitted in the paper due to page limitations.

Definition 5.2.2.

$$\begin{aligned}
\mathcal{Q}_{jl} = &- \frac{\theta_y}{2} \sum_{l \neq j}^{J} \left[\left(S_{jl}^y - \mathbf{v}_l^T \mathbf{W}^y \mathbf{v}_j^+\right)^2 - \left(S_{jl}^y - \mathbf{v}_l^T \mathbf{W}^y \mathbf{v}_j^-\right)^2 \right] \\
&- \frac{\theta_y}{2} \left[\left(S_{jj}^y - \mathbf{v}_j^{+T} \mathbf{W}^y \mathbf{v}_j^+\right)^2 - \left(S_{jj}^y - \mathbf{v}_j^{-T} \mathbf{W}^y \mathbf{v}_j^-\right)^2 \right] \\
&+ \sum_{i=1}^{I} O_{ij} \left[S_{ij}^{xy} \log \frac{\lambda_{ij}^+}{\lambda_{ij}^-} + (1 - S_{ij}^{xy}) \log \frac{1-\lambda_{ij}^+}{1-\lambda_{ij}^-} \right] + \log \frac{\alpha_v}{\beta_v},
\end{aligned}$$

where $\mathbf{v}_j^+$ is the jth row of $\mathbf{V}$ with $V_{jk} = 1$, $\mathbf{v}_j^-$ is the jth row of $\mathbf{V}$ with $V_{jk} = -1$, $\lambda_{ij}^+ = \gamma(w\mathbf{u}_i^T \mathbf{v}_j^+)$, and $\lambda_{ij}^- = \gamma(w\mathbf{u}_i^T \mathbf{v}_i^-)$.

Theorem 5.2.2. *The MAP solution of V_{ik} is equal to 1 if $\mathcal{Q}_{jl} \geq 0$ and -1 otherwise.*

With $\mathbf{U}, \phi_x$ and θ_x fixed, we can compute the MAP estimate of $\mathbf{W}^x$ using Theorem 5.2.3 below.

Theorem 5.2.3. *The MAP solution of* $\mathbf{W}^x$ *is*

$$\mathbf{w}^x = \left(\mathbf{A}^T\mathbf{M}_2\mathbf{A} + \frac{\phi_x}{\theta_x}\mathbf{M}_1\right)^{-1}\mathbf{A}^T\mathbf{M}_2\mathbf{s}^x,$$

where $\mathbf{w}^x$ *is a* K^2*-dimensional column vector taken in a columnwise manner from* $\mathbf{W}^x$*,* $\mathbf{s}^x$ *is an* I^2*-dimensional column vector taken in a columnwise manner from* $\mathbf{S}^x$*,* $\mathbf{A} = \mathbf{U} \otimes \mathbf{U}$*,* $\mathbf{M}_1$ *is a diagonal matrix with each diagonal entry equal to 1 if it is the linear index of the upper-right portion of* $\mathbf{W}^x$ *and 0 otherwise, and* $\mathbf{M}_2$ *is similarly defined but with a different size which is determined by* $\mathbf{S}^x$*.*

Proof. The negative log of the posterior distribution of $\mathbf{W}^x$ can be written as:

$$\begin{aligned}
&-\log p(\mathbf{W}^x \mid \mathbf{S}^x, \mathbf{U}, \theta_x, \phi_x) \qquad (5.1)\\
&\quad = -\log P(\mathbf{W}^x \mid \phi_x) - \log P(\mathbf{S}^x \mid \mathbf{U}, \mathbf{W}^x, \theta_x) + \tilde{C}\\
&\quad = \frac{\phi_x}{2}\sum_{k=1}^{K}\sum_{d=k}^{K}(W_{kd}^x)^2 + \frac{\theta_x}{2}\sum_{i=1}^{I}\sum_{i'=i}^{I}\left(S_{ii'}^x - \mathbf{u}_i^T\mathbf{W}^x\mathbf{u}_{i'}\right)^2 + \tilde{C}\\
&\quad = \frac{\phi_x}{2}\mathbf{w}^{xT}\mathbf{M}_1\mathbf{w}^x + \frac{\theta_x}{2}\left(\mathbf{s}^x - \mathbf{A}\mathbf{w}^x\right)^T\mathbf{M}_2\left(\mathbf{s}^x - \mathbf{A}\mathbf{w}^x\right) + \tilde{C}\\
&\quad = \frac{1}{2}\mathbf{w}^{xT}\left(\theta_x\mathbf{A}^T\mathbf{M}_2\mathbf{A} + \phi_x\mathbf{M}_1\right)\mathbf{w}^x - \theta_x\mathbf{s}^{xT}\mathbf{M}_2\mathbf{A}\mathbf{w}^x + \tilde{C},
\end{aligned}$$

where $\tilde{C}$ is a constant term independent of $\mathbf{W}^x$.

Setting the derivative of Equation (5.1) to zero, we get

$$\mathbf{w}^x = \left(\mathbf{A}^T\mathbf{M}_2\mathbf{A} + \frac{\phi_x}{\theta_x}\mathbf{M}_1\right)^{-1}\mathbf{A}^T\mathbf{M}_2\mathbf{s}^x.$$

This completes the proof. □

Similarly, we have Theorem 5.2.4 for $\mathbf{W}^y$.

Theorem 5.2.4. *The MAP solution of* $\mathbf{W}^y$ *is*

$$\mathbf{w}^y = \left(\mathbf{B}^T\tilde{\mathbf{M}}_2\mathbf{B} + \frac{\phi_y}{\theta_y}\mathbf{M}_1\right)^{-1}\mathbf{B}^T\tilde{\mathbf{M}}_2\mathbf{s}^y,$$

where $\mathbf{w}^y, \mathbf{s}^y, \mathbf{B}$ *and* $\tilde{\mathbf{M}}_2$ *are also defined similarly.*

To obtain the MAP estimate of w, we minimize the negative log posterior $p(w \mid \mathbf{U}, \mathbf{V}, \mathbf{S}^{xy}, \phi)$, which is equivalent to the following objective function:

$$\mathcal{L}_w = \frac{\phi}{2}w^2 - \sum_{i=1}^{I}\sum_{j=1}^{J}\left\{O_{ij}\left[S_{ij}^{xy}\log\lambda_{ij} + \left(1 - S_{ij}^{xy}\right)\log\left(1 - \lambda_{ij}\right)\right]\right\},$$

where $\lambda_{ij} = \gamma\left(w\mathbf{u}_i^T\mathbf{v}_j\right)$.

Although the objective function is convex with respect to w, there is no closed-form solution. Nevertheless, due to its convexity, we can obtain the global minimum easily using a gradient descent algorithm. The gradient can be evaluated as follows:

$$\nabla w = \phi \cdot w - \sum_{i=1}^{I}\sum_{j=1}^{J}\left\{O_{ij}\left[S_{ij}^{xy}\left(1-\lambda_{ij}\right)\mathbf{u}_i^T\mathbf{v}_j - \left(1-S_{ij}^{xy}\right)\lambda_{ij}\mathbf{u}_i^T\mathbf{v}_j\right]\right\}. \tag{5.2}$$

As for the parameters, closed-form solutions exist for their MAP estimates which are summarized in the following theorem.

Theorem 5.2.5. *The MAP estimates of $\theta_x, \theta_y, \phi_x, \phi_y$ and ϕ are:*

$$\theta_x = \frac{I(I+1) + 4(a_\theta - 1)}{4b_\theta + 2\sum_{i=1}^{I}\sum_{i'=i}^{I}\left(S_{ii'}^{x} - \mathbf{u}_i^T\mathbf{W}^x\mathbf{u}_{i'}\right)^2}, \tag{5.3}$$

$$\theta_y = \frac{J(J+1) + 4(c_\theta - 1)}{4d_\theta + 2\sum_{j=1}^{J}\sum_{j'=j}^{J}\left(S_{jj'}^{y} - \mathbf{v}_j^T\mathbf{W}^y\mathbf{v}_{j'}\right)^2}, \tag{5.4}$$

$$\phi_x = \frac{K(K+1) + 4(a_\phi - 1)}{4b_\phi + 2\sum_{k=1}^{K}\sum_{d=k}^{K}\left(W_{kd}^{x}\right)^2}, \tag{5.5}$$

$$\phi_y = \frac{K(K+1) + 4(c_\phi - 1)}{4d_\phi + 2\sum_{k=1}^{K}\sum_{d=k}^{K}\left(W_{kd}^{y}\right)^2}, \tag{5.6}$$

$$\phi = \frac{2e_\phi - 1}{2f_\phi + w^2}. \tag{5.7}$$

Theorem 5.2.5 can be proved easily. Briefly speaking, we first find the posterior distribution of each parameter and then compute the optimal value by setting its derivative to zero. Details of the proof are omitted here.

To summarize, the learning algorithm of MLBE is presented in Algorithm 5.1.

5.2.3 Out-of-Sample Extension

Algorithm 5.1 tells us how to learn the hash functions for the observed bimodal data based on their intra-modality and inter-modality similarities. However, the hash codes can only be computed this way for the training data. In many applications, after learning the hash functions, it is necessary to obtain the hash codes for out-of-sample data points as well. One naive approach would be to incorporate the out-of-sample points into the original training set and then learn the hash functions from scratch. However, this approach is computationally unappealing due to its high computational cost especially when the training set is large.

Algorithm 5.1 Learning algorithm of MLBE

Input:
$\mathbf{S}^x$, $\mathbf{S}^y$, $\mathbf{S}^{xy}$ – similarity matrices
$\mathbf{O}$ – observation indicator variables for $\mathbf{S}^{xy}$
K – number of hash functions
Ω – hyperparameters
Procedure:
Initialize all the latent variables and parameters except $\mathbf{W}^x, \mathbf{W}^y$.
while not converged **do**
  Update $\mathbf{W}^x$ using Theorem 5.2.3.
  Update ϕ_x using Equation (5.5).
  Update each element of $\mathbf{U}$ using Theorem 5.2.1.
  Update θ_x using Equation (5.3).
  Update $\mathbf{W}^y$ using Theorem 5.2.4.
  Update ϕ_y using Equation (5.6).
  Update each element of $\mathbf{V}$ using Theorem 5.2.2.
  Update θ_y using Equation (5.4).
  Update w by gradient descent using Equation (5.2).
  Update ϕ using Equation (5.7).
end while

In this subsection, we propose a simple yet very effective method for finding the hash codes of out-of-sample points. The method is based on a simple and natural assumption that the latent variables and parameters for the training data can be fixed while computing the hash codes for the out-of-sample points.

Specifically, we first train the MLBE model using some training data selected from both modalities.[2] Using the latent variables and parameters learned from the training points, we can find the hash codes for the out-of-sample points by applying Theorem 5.2.1 or Theorem 5.2.2. For illustration, Algorithm 5.2 shows how to compute the hash code for an out-of-sample point $\mathbf{x}^*$ from modality $\mathcal{X}$ using the latent variables and parameters learned from two training sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{Y}}$. It is worth noting that the hash code for each out-of-sample point can be computed independently. The implication is that the algorithm is highly parallelizable, making it potentially applicable to very large data sets. The same can also be done to out-of-sample points from the other modality with some straightforward modifications.

5.2.4 Complexity Analysis

The computational cost of the above learning algorithm is mainly spent on updating $\mathbf{U}, \mathbf{V}, \mathbf{W}^x, \mathbf{W}^y$, and w.

The complexity of updating an entry U_{ik} is $O(IK^2 + JK)$, which grows linearly with the number of points in each modality. Updating $\mathbf{W}^x$ requires inverting a $K^2 \times K^2$ matrix.

[2] We do not make any assumption on how the training data are selected. They may be selected randomly for simplicity or carefully based on how representative they are. Random selection is used in our experiments.

Algorithm 5.2 Algorithm for out-of-sample extension

Input:
$\hat{\mathbf{S}}^x$ – intra-modality similarities for $\mathbf{x}^*$ and $\hat{\mathcal{X}}$
$\hat{\mathbf{S}}^{xy}$ – inter-modality similarities for $\mathbf{x}^*$ and $\hat{\mathcal{Y}}$
$\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{W}}^x, \hat{w}, \hat{\theta}_x$ – learned variables
α_u, β_u – hyperparameters
Procedure:
Initialize $\mathbf{u}^*$.
while not converged **do**
  Update each element of $\mathbf{u}^*$ using Theorem 5.2.1.
end while

Since K is usually very small, this step can be performed efficiently. The complexity of evaluating the gradient ∇w is linear in the number of observations of the inter-modality similarities. We note that the complexity can be greatly reduced if the similarity matrices are sparse, which is often the case in real applications.

5.2.5 Discussion

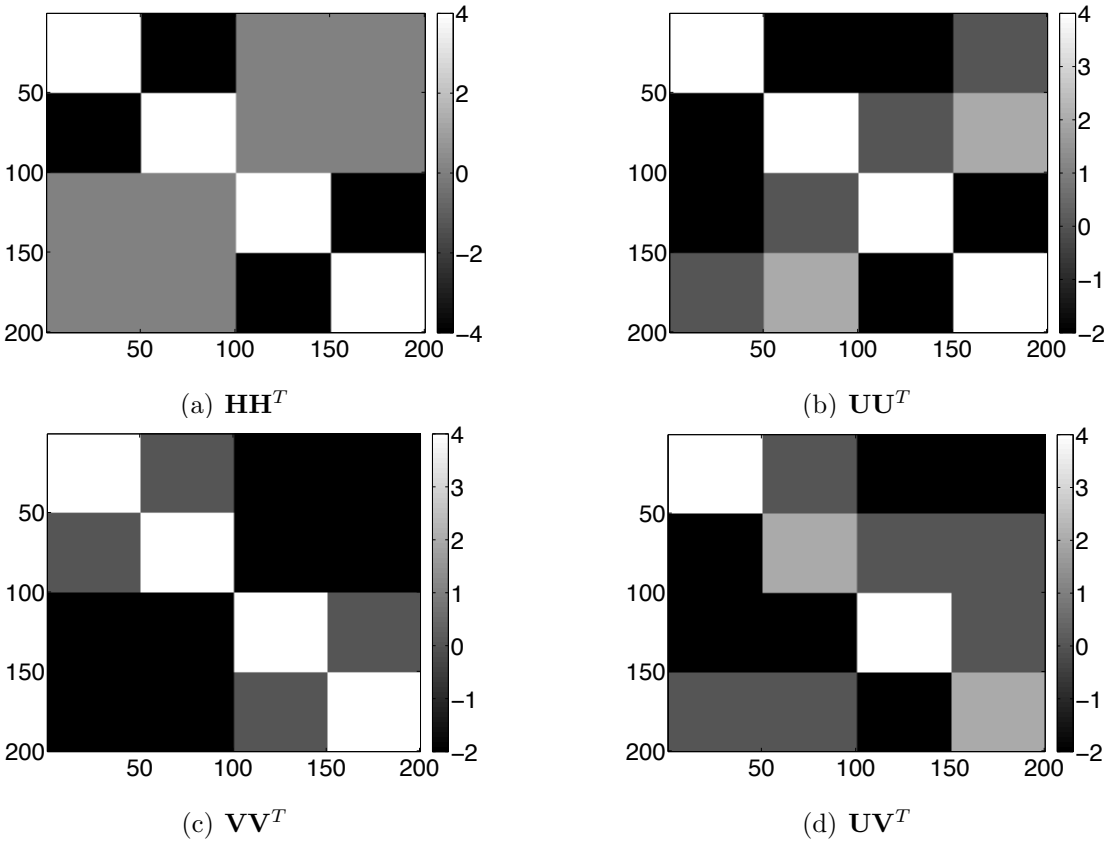
Our work is closely related to binary latent factor models (Meeds et al., 2006; Heller & Ghahramani, 2007) but there exist some significant differences. First, the binary latent factors in MLBE are used as hash codes for multimodal similarity search, while the latent factors in (Meeds et al., 2006; Heller & Ghahramani, 2007) are treated as cluster membership indicators which are used for clustering and link prediction applications. Moreover, the model formulations are very different. In MLBE, the prior distributions on the binary latent factors are simple Bernoulli distributions, but in (Meeds et al., 2006; Heller & Ghahramani, 2007), the priors on the binary latent factors are Indian buffet processes (Griffiths & Ghahramani, 2005). Furthermore, from a matrix factorization point of view, MLBE simultaneously factorizes multiple matrices but the method in (Meeds et al., 2006) factorizes only one matrix.

5.3 Experiments

We first present an illustrative example on synthetic data in Section 5.3.1. It is then followed by experiments on two publicly available real-world data sets. Section 5.3.2 presents the experimental settings and then Sections 5.3.3 and 5.3.4 present the results.

5.3.1 Illustration on Synthetic Data

There are four groups of data points with each group consisting of 50 points. We associate each group with one of four hash codes, namely, $[1, 1, -1, -1]$, $[-1, -1, 1, 1]$, $[1, -1, 1, -1]$



(a) $\mathbf{H}\mathbf{H}^T$ (b) $\mathbf{U}\mathbf{U}^T$

(c) $\mathbf{V}\mathbf{V}^T$ (d) $\mathbf{U}\mathbf{V}^T$

Figure 5.2: Illustration of MLBE

and $[-1, 1, -1, 1]$, and use a 200×4 matrix $\mathbf{H}$ to denote the hash codes of all 200 points. We generate $\mathbf{W}^x$ and $\mathbf{W}^y$ from $\mathcal{N}(\cdot \mid 1, 0.01)$ and $\mathcal{N}(\cdot \mid 5, 0.01)$, respectively. Based on the latent representation, we generate the similarity matrices $\mathbf{S}^x$ and $\mathbf{S}^y$ using $\mathcal{N}(S^x_{il} \mid \mathbf{h}_i^T\mathbf{W}^x\mathbf{h}_l, 0.01)$ and $\mathcal{N}(S^y_{jl} \mid \mathbf{h}_i^T\mathbf{W}^y\mathbf{h}_l, 0.01)$, respectively. Moreover, we set $S^{xy}_{ij} = 1$ if $\mathbf{h}_i = \mathbf{h}_j$ and $S^{xy}_{ij} = 0$ otherwise, assuming that all entries in $\mathbf{S}^{xy}$ are observed, i.e., $O_{ij} = 1, \forall i, j$.

Based on the similarities generated, we train MLBE to obtain the hash codes $\mathbf{U}$ and $\mathbf{V}$. Because the bits of the hash codes are interchangeable, it is more appropriate to use inner products of the hash codes to illustrate the similarity structures, as shown in Figure 5.2. Note that the whiter the area is, the more similar the points involved are. Figure 5.2(a) depicts the ground-truth similarity structure, Figure 5.2(b) and 5.2(c) show the learned intra-modality similarity structure for each modality, and Figure 5.2(d) shows the learned inter-modality similarity structure. As we can see, the whiter areas in the last three subfigures are in the same locations as those in Figure 5.2(a). In other words, both the intra-modality and inter-modality similarity structures revealed by the learned hash codes are very close to the ground truth, showing the effectiveness of MLBE.

5.3.2 Experimental Settings

We have conducted several comparative experiments on two real-world data sets, which are, to the best of our knowledge, the largest publicly available multimodal data sets that are fully paired and labeled. Both data sets are bimodal with the image and text modalities but the feature representations are different. Each data set is partitioned into a database set and a separate query set.

We compare MLBE with CMSSH[3] and CVH[4] on two common crossmodal retrieval tasks. Specifically, we use a text query to retrieve similar images from the image database and use an image query to retrieve similar texts from the text database. Since the data sets are fully labeled, meaning that each document (image or text) has one or more semantic labels, it is convenient to use these labels to decide the ground-truth neighbors.

We use *mean Average Precision* (mAP) as the performance measure. Given a query and a set of R retrieved documents, the *Average Precision* (AP) is defined as

$$\mathrm{AP} = \frac{1}{L} \sum\nolimits_{r=1}^{R} P(r)\, \delta(r),$$

where L is the number of true neighbors in the retrieved set, $P(r)$ denotes the precision of the top r retrieved documents, and $\delta(r) = 1$ if the rth retrieved document is a true neighbor and $\delta(r) = 0$ otherwise. We then average the AP values over all the queries in the query set to obtain the mAP measure. The larger the mAP, the better the performance. In the experiments, we set $R = 50$.

We also report two types of performance curves, namely, *precision-recall* curve and *recall* curve. Given a query set and a database, both curves can be obtained by varying the Hamming radius of the retrieved points and evaluating the precision, recall and number of retrieved points accordingly.

For MLBE, the intra-modality similarity matrices are computed based on the feature vectors. We first compute the Euclidean distance d between two feature vectors and then transform it into a similarity measure $s = e^{-d^2/2\sigma^2}$, where the parameter σ^2 is fixed to 1 for both data sets. The inter-modality similarity matrices are simply determined by the class labels. Since MLBE is not sensitive to the hyperparameters, we simply set all of them to 1. Besides, we initialize $\mathbf{U}$ and $\mathbf{V}$ using the results of CVH, set the initial values of $\theta_x, \theta_y, \phi_x, \phi_y$ to 0.01, and use a fixed learning rate 10^{-4} for updating w.

In all the experiments, the size of the training set, which is randomly selected from the database set for each modality, is set to 300 and only 0.1% of the random selected

[3]The implementation is kindly provided by the authors.

[4]Because the code is not publicly available, we implemented the method ourselves.

entries of $\mathbf{S}^{xy}$ are observed.[5] To be fair, all three models are trained on the same training set.

5.3.3 Results on Wiki Data Set

The Wiki data set is generated from a group of 2,866 wikipedia documents provided by (Rasiwasia et al., 2010). Each document is an image-text pair and is labeled with exactly one of 10 semantic classes. The images are represented by 128-dimensional SIFT (Lowe, 2004) feature vectors. The text articles are represented by the probability distributions over 10 topics, which are derived from a latent Dirichlet allocation (LDA) model (Blei et al., 2003). We use 80% of the data as the database set and the remaining 20% to form the query set.

The mAP values for MLBE and the two baselines are reported in Table 5.1. The *precision-recall* curves and *recall* curves for the three methods are plotted in Figure 5.3.

Table 5.1: mAP comparison on Wiki

Task	Method	Code Length		
		$K = 8$	$K = 16$	$K = 24$
Image Query vs. Text Database	MLBE	**0.3810**	**0.2561**	**0.1915**
	CVH	0.2592	0.2190	0.1767
	CMSSH	0.2438	0.2014	0.1757
Text Query vs. Image Database	MLBE	**0.4955**	**0.3209**	0.2143
	CVH	0.3474	0.3094	**0.2576**
	CMSSH	0.2044	0.2286	0.2256

We can see that MLBE significantly outperforms CVH and CMSSH when the code length is small. As the code length increases, the performance gap gets smaller. We conjecture that MLBE may get trapped in local minima during the learning process when the code length is too large.

Besides, we observe that as the code length increases, the performance of all three methods degrades. We note that this phenomenon has also been observed in (Wang et al., 2010a; Liu et al., 2011). A possible reason is that the learned models will be farther from the optimal solutions when the code length gets larger.

5.3.4 Results on Flickr Data Set

The Flickr data set consists of 186,577 image-tag pairs, which are pruned from the NUS dataset (Chua et al., 2009) by keeping the pairs that belong to one of the 10 largest

[5]We have tried larger training sets, e.g., of sizes 500 and 1,000, in our experiments but there is no significant performance improvement. So we omit the results due to space limitations.

classes. Each pair is annotated by at least one of 10 labels. The image features are 500-dimensional SIFT features and the text features are 1000-dimensional vectors obtained by performing PCA on the original tag occurrence features. We use 99% of the data as the database set and the remaining 1% to form the query set.

Table 5.2: mAP comparison on Flickr

Task	Method	Code Length		
		$K = 8$	$K = 16$	$K = 24$
Image Query vs. Text Database	MLBE	**0.6322**	**0.6608**	0.5104
	CVH	0.5361	0.4871	0.4605
	CMSSH	0.5155	0.5333	**0.5150**
Text Query vs. Image Database	MLBE	**0.5626**	**0.5970**	0.4296
	CVH	0.5260	0.4856	**0.4553**
	CMSSH	0.5093	0.4594	0.4053

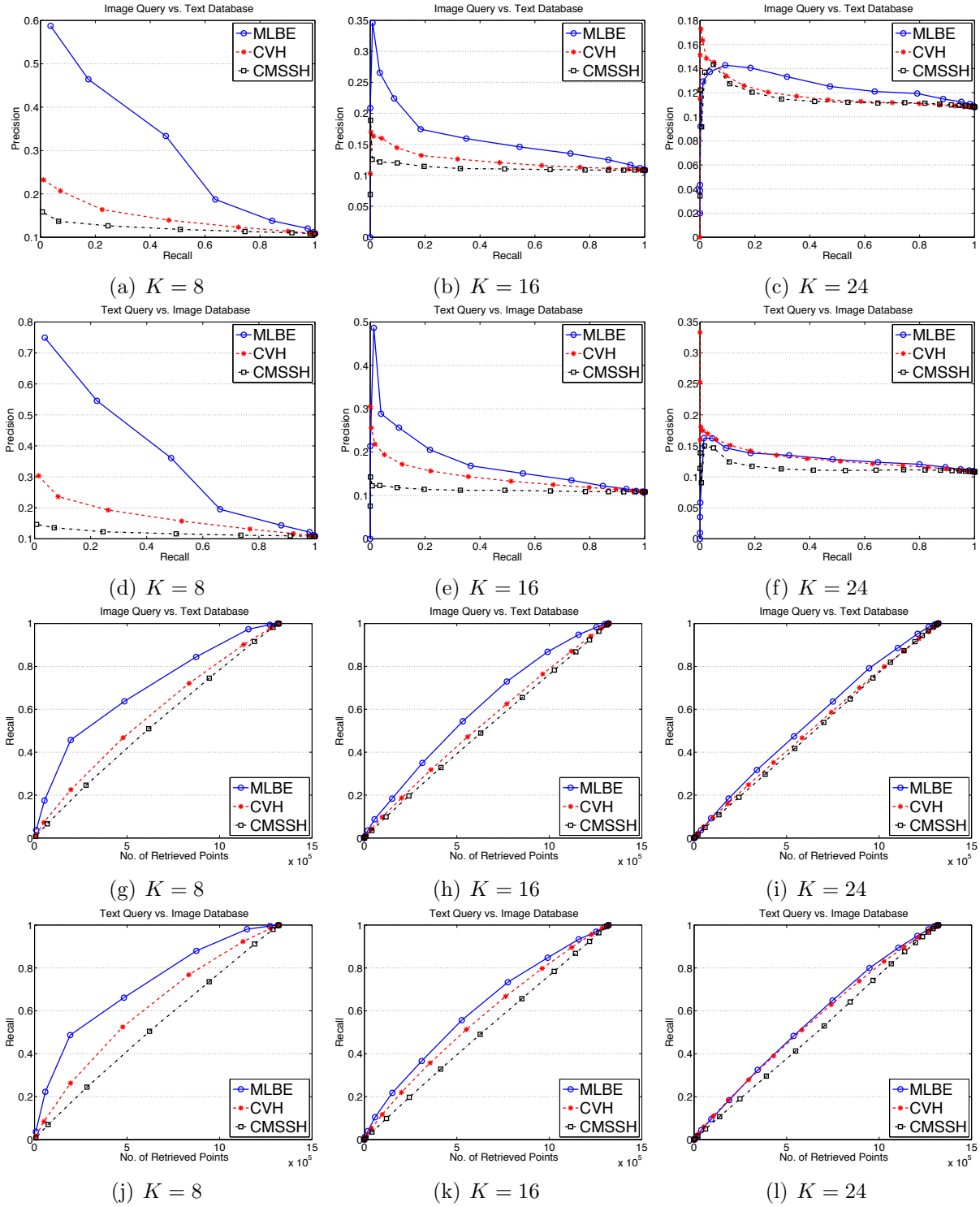
The mAP results are reported in Table 5.2. Similar to the results on Wiki, we observe that MLBE outperforms its counterparts by a large margin when the code length is small.

The corresponding *precision-recall* curves and *recall* curves are plotted in Figure 5.4. We note that MLBE has the best overall performance.

5.4 Conclusion

In this chapter, we have proposed a novel probabilistic model, called multimodal latent binary embedding (MLBE), for multimodal hash function learning for graph data. As a latent factor model, MLBE regards the binary latent factors as hash codes and hence maps data points from multiple modalities to a common Hamming space in a principled manner. Although finding exact posterior distributions of the latent factors is intractable, we have devised an efficient alternating learning algorithm based on MAP estimation. Experimental results show that MLBE compares favorably with two state-of-the-art models.

For our future research, we will go beyond the point estimation approach presented in this paper to explore a more Bayesian treatment based on variational inference for enhanced robustness and efficiency. We would also like to extend MLBE to determine the code length K automatically from data. This is an important yet largely unaddressed issue in existing methods. Moreover, we also plan to apply MLBE to other tasks such as multimodal medical image registration.



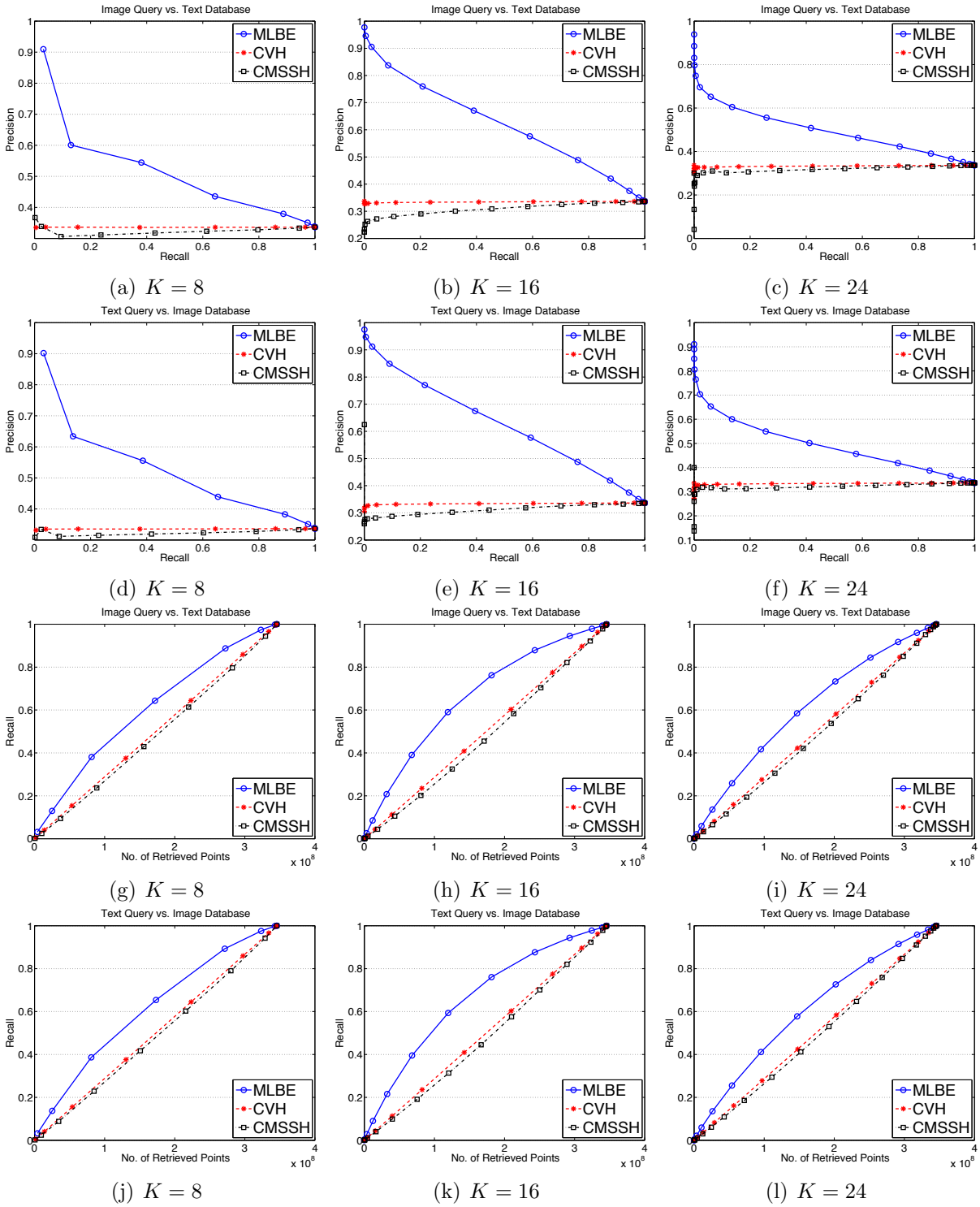
(a) $K = 8$ (b) $K = 16$ (c) $K = 24$

(d) $K = 8$ (e) $K = 16$ (f) $K = 24$

(g) $K = 8$ (h) $K = 16$ (i) $K = 24$

(j) $K = 8$ (k) $K = 16$ (l) $K = 24$

Figure 5.3: Precision-recall curves and recall curves on Wiki



(a) $K = 8$ (b) $K = 16$ (c) $K = 24$

(d) $K = 8$ (e) $K = 16$ (f) $K = 24$

(g) $K = 8$ (h) $K = 16$ (i) $K = 24$

(j) $K = 8$ (k) $K = 16$ (l) $K = 24$

Figure 5.4: Precision-recall curves and recall curves on Flickr

CHAPTER 6

MULTIMODAL HASHING FOR GENERAL DATA

6.1 Introduction

As stated earlier, the SMH model and the MLBE model require the data points in different modalities to be aligned or organized in graphs. However, these assumptions might not be the case in some applications.

In this chapter, we present a novel model for data in general form. Specifically, we develop Co-Regularized Hashing (CRH), which is based on a boosted co-regularization framework. For each bit of the hash codes, CRH learns a group of hash functions, one for each modality, by minimizing a novel loss function. Although the loss function is non-convex, it is in a special form which can be expressed as a difference of convex functions. As a consequence, the Concave-Convex Procedure (CCCP) (Yuille & Rangarajan, 2001) can be applied to solve the optimization problem iteratively. We use a stochastic gradient method in each CCCP iteration. After learning the hash functions for one bit, CRH proceeds to learn more bits via a boosting procedure such that the bias introduced by the hash functions can be sequentially minimized.

In the following, we present the CRH model in Section 6.2. Empirical study conducted on three real data sets is reported in Section 6.3 before the conclusion in Section 6.4.

6.2 Co-Regularized Hashing

6.2.1 Objective Function

Suppose that there are two sets of data points from two modalities,[1] e.g., $\{\mathbf{x}_i \in \mathcal{X}\}_{i=1}^{I}$ for a set of I images from some feature space $\mathcal{X}$ and $\{\mathbf{y}_j \in \mathcal{Y}\}_{j=1}^{J}$ for a set of J textual documents from another feature space $\mathcal{Y}$. We also have a set of N inter-modality point pairs $\Theta = \{(\mathbf{x}_{a_1}, \mathbf{y}_{b_1}), (\mathbf{x}_{a_2}, \mathbf{y}_{b_2}), \ldots, (\mathbf{x}_{a_N}, \mathbf{y}_{b_N})\}$, where, for the nth pair, a_n and b_n are indices of the points in $\mathcal{X}$ and $\mathcal{Y}$, respectively. We further assume that each pair has a label $s_n = 1$ if $\mathbf{x}_{a_n}$ and $\mathbf{y}_{b_n}$ are similar and $s_n = 0$ otherwise. The notion of inter-modality

[1]For simplicity of our presentation, we focus on the bimodal case here and leave the discussion on extension to more than two modalities to Section 6.2.4.

similarity varies from application to application. For example, if an image includes a tiger and a textual document is a research paper on tigers, they should be labeled as similar. On the other hand, it is highly unlikely to label the image as similar to a textual document on basketball.

For each bit of the hash codes, we define two linear hash functions as follows:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}_x^T \mathbf{x}) \quad \text{and} \quad g(\mathbf{y}) = \text{sgn}(\mathbf{w}_y^T \mathbf{y}),$$

where $\text{sgn}(\cdot)$ denotes the sign function, and $\mathbf{w}_x$ and $\mathbf{w}_y$ are projection vectors which, ideally, should map similar points to the same hash bin and dissimilar points to different bins. Our goal is to achieve HFL by learning $\mathbf{w}_x$ and $\mathbf{w}_y$ from the multimodal data.

To achieve this goal, we propose to minimize the following objective function *w.r.t.* (with respect to) $\mathbf{w}_x$ and $\mathbf{w}_y$:

$$\mathcal{O} = \frac{1}{I}\sum_{i=1}^{I} \ell_i^x + \frac{1}{J}\sum_{j=1}^{J} \ell_j^y + \gamma \sum_{n=1}^{N} \omega_n \ell_n^* + \frac{\lambda_x}{2}\|\mathbf{w}_x\|^2 + \frac{\lambda_y}{2}\|\mathbf{w}_y\|^2, \tag{6.1}$$

where ℓ_i^x and ℓ_j^y are intra-modality loss terms for modalities $\mathcal{X}$ and $\mathcal{Y}$, respectively. In this work, we define them as:

$$\ell_i^x = \left[1 - f(\mathbf{x}_i)(\mathbf{w}_x^T \mathbf{x}_i)\right]_+ = \left[1 - |\mathbf{w}_x^T \mathbf{x}_i|\right]_+,$$
$$\ell_j^y = \left[1 - g(\mathbf{y}_j)(\mathbf{w}_y^T \mathbf{y}_j)\right]_+ = \left[1 - |\mathbf{w}_y^T \mathbf{y}_j|\right]_+,$$

where $[a]_+$ is equal to a if $a \geq 0$ and 0 otherwise. We note that the intra-modality loss terms are similar to the hinge loss in the (linear) support vector machine but have quite different meaning. Conceptually, we want the projected values to be far away from 0 and hence expect the hash functions learned to have good generalization ability (Mu et al., 2010). For the inter-modality loss term ℓ_n^*, we associate with each point pair a weight ω_n, with $\sum_{n=1}^{N} \omega_n = 1$, to normalize the loss as well as compute the bias of the hash functions. In this paper, we define ℓ_n^* as

$$\ell_n^* = s_n d_n^2 + (1 - s_n)\tau(d_n),$$

where $d_n = \mathbf{w}_x^T \mathbf{x}_{a_n} - \mathbf{w}_y^T \mathbf{y}_{b_n}$ and $\tau(d)$ is called the smoothly clipped inverted squared deviation (SCISD) function.

The SCISD function was first proposed in (Quadrianto & Lampert, 2011). It can be defined as follows:

$$\tau(d) = \begin{cases} -\frac{1}{2}d^2 + \frac{a\lambda^2}{2} & \text{if } |d| \leq \lambda \\ \frac{d^2 - 2a\lambda|d| + a^2\lambda^2}{2(a-1)} & \text{if } \lambda < |d| \leq a\lambda \\ 0 & \text{if } a\lambda < |d|, \end{cases}$$

where a and λ are two user-specified parameters. The SCISD function penalizes projection vectors that result in small distance between dissimilar points after projection. A more important property is that it can be expressed as a difference of two convex functions. Specifically, we can express $\tau(d) = \tau_1(d) - \tau_2(d)$ where

$$\tau_1(d) = \begin{cases} 0 & \text{if } |d| \leq \lambda \\ \frac{ad^2 - 2a\lambda|d| + a\lambda^2}{2(a-1)} & \text{if } \lambda < |d| \leq a\lambda \\ \frac{1}{2}d^2 - \frac{a\lambda^2}{2} & \text{if } a\lambda < |d| \end{cases} \quad \text{and} \quad \tau_2(d) = \frac{1}{2}d^2 - \frac{a\lambda^2}{2}.$$

6.2.2 Optimization

Though the objective function (6.1) is nonconvex *w.r.t.* $\mathbf{w}_x$ and $\mathbf{w}_y$, we can optimize it *w.r.t.* $\mathbf{w}_x$ and $\mathbf{w}_y$ in an alternating manner. Take $\mathbf{w}_x$ for example, we remove the irrelevant terms and get the following objective:

$$\frac{1}{I}\sum_{i=1}^{I} \ell_i^x + \frac{\lambda_x}{2}\|\mathbf{w}_x\|^2 + \gamma \sum_{n=1}^{N} \omega_n \ell_n^*, \tag{6.2}$$

where

$$\ell_i^x = \begin{cases} 0 & \text{if } |\mathbf{w}_x^T \mathbf{x}_i| \geq 1 \\ 1 - \mathbf{w}_x^T \mathbf{x}_i & \text{if } 0 \leq \mathbf{w}_x^T \mathbf{x}_i < 1 \\ 1 + \mathbf{w}_x^T \mathbf{x}_i & \text{if } -1 < \mathbf{w}_x^T \mathbf{x}_i < 0 \end{cases}.$$

It is easy to realize that the objective function (6.2) can be expressed as a difference of two convex functions in different cases. As a consequence, we can use CCCP to solve the non-convex optimization problem iteratively with each iteration minimizing a convex upper bound of the original objective function.

Briefly speaking, given an objective function $f_0(x) - g_0(x)$ where both f_0 and g_0 are convex, CCCP works iteratively as follows. The variable x is first randomly initialized to $x^{(0)}$. At the tth iteration, CCCP minimizes the following convex upper bound of $f_0(x) - g_0(x)$ at location $x^{(t)}$:

$$f_0(x) - \big(g_0(x^{(t)}) + \partial_x g_0(x^{(t)})(x - x^{(t)})\big),$$

where $\partial_x g_0(x^{(t)})$ is the first derivative of $g(x)$ at $x^{(t)}$. This optimization problem can be solved using any convex optimization solver to obtain $x^{(t+1)}$. Given an initial value $x^{(0)}$, the solution sequence $\{x^{(t)}\}$ found by CCCP is guaranteed to reach a local minimum or a saddle point.

For our problem, the optimization problem at the tth iteration is minimizing the following upper bound of Equation (6.2) *w.r.t.* $\mathbf{w}_x$:

$$\mathcal{O}_x = \frac{\lambda_x\|\mathbf{w}_x\|^2}{2} + \gamma \sum_{n=1}^{N} \omega_n \left(s_n d_n^2 + (1 - s_n)\zeta_n^x\right) + \frac{1}{I}\sum_{i=1}^{I} \ell_i^x, \tag{6.3}$$

where $\zeta_n^x = \tau_1(d_n) - \tau_2(d_n^{(t)}) - d_n^{(t)}\mathbf{x}_{a_n}^T(\mathbf{w}_x - \mathbf{w}_x^{(t)}), d_n^{(t)} = (\mathbf{w}_x^{(t)})^T\mathbf{x}_{a_n} - \mathbf{w}_y^T\mathbf{y}_{b_n}$, and $\mathbf{w}_x^{(t)}$ is the value of $\mathbf{w}_x$ at the tth iteration.

To find a local optimal solution of problem (6.3), we can use any gradient based methods. In this work, we develop a stochastic gradient solver based on Pegasos (Shalev-Shwartz et al., 2007), which is known to be one of the fastest solvers for margin-based classifiers. Specifically, we randomly select k point from each modality and l point pairs to evaluate the gradient at each iteration.

The key step of our method is to evaluate the gradient of objective function (6.3) *w.r.t.* $\mathbf{w}_x$, which can be computed as

$$\frac{\partial \mathcal{O}_x}{\partial \mathbf{w}_x} = 2\gamma \sum_{n=1}^{N} \omega_n s_n d_n \mathbf{x}_{a_n} + \gamma \sum_{n=1}^{N} \omega_n \boldsymbol{\mu}_n^x + \lambda_x \mathbf{w}_x - \frac{1}{I}\sum_{i=1}^{I} \boldsymbol{\pi}_i^x, \tag{6.4}$$

where $\boldsymbol{\mu}_n^x = (1 - s_n)\left(\frac{\partial \tau_1}{\partial d_n} - d_n^{(t)}\right)\mathbf{x}_{a_n}$,

$$\frac{\partial \tau_1}{\partial d_n} = \begin{cases} 0 & \text{if } |d_n| \le \lambda \\ \frac{ad_n - 2a\lambda\,\mathrm{sgn}(d_n)}{(a-1)} & \text{if } \lambda < |d_n| \le a\lambda \\ d_n & \text{if } a\lambda < |d_n|. \end{cases} \quad \text{and } \boldsymbol{\pi}_i^x = \begin{cases} 0 & \text{if } |\mathbf{w}_x^T\mathbf{x}_i| \ge 1 \\ \mathrm{sgn}\left(\mathbf{w}_x^T\mathbf{x}_i\right)\mathbf{x}_i & \text{if } |\mathbf{w}_x^T\mathbf{x}_i| < 1 \end{cases}.$$

Similarly, the objective function for the optimization problem *w.r.t.* $\mathbf{w}_y$ at the tth CCCP iteration is:

$$\mathcal{O}_y = \frac{\lambda_y \|\mathbf{w}_y\|^2}{2} + \gamma \sum_{n=1}^{N} \omega_n \left(s_n d_n^2 + (1 - s_n)\zeta_n^y\right) + \frac{1}{J}\sum_{j=1}^{I} \ell_j^y, \tag{6.5}$$

where $\zeta_n^y = \tau_1(d_n) - \tau_2(d_n^{(t)}) + d_n^{(t)}\mathbf{y}_{b_n}^T(\mathbf{w}_y - \mathbf{w}_y^{(t)}), d_n^{(t)} = \mathbf{w}_x^T\mathbf{x}_{a_n} - (\mathbf{w}_y^{(t)})^T\mathbf{y}_{b_n}$, $\mathbf{w}_y^{(t)}$ is the value of $\mathbf{w}_y$ at the tth iteration and

$$\ell_j^y = \begin{cases} 0 & \text{if } |\mathbf{w}_y^T\mathbf{y}_j| \ge 1 \\ 1 - \mathbf{w}_y^T\mathbf{y}_j & \text{if } 0 \le \mathbf{w}_y^T\mathbf{y}_j < 1 \\ 1 + \mathbf{w}_y^T\mathbf{y}_j & \text{if } -1 < \mathbf{w}_y^T\mathbf{y}_j < 0 \end{cases}.$$

The corresponding gradient is given by

$$\frac{\partial \mathcal{O}_y}{\partial \mathbf{w}_y} = -2\gamma \sum_{n=1}^{N} \omega_n s_n d_n \mathbf{y}_{b_n} - \gamma \sum_{n=1}^{N} \omega_n \boldsymbol{\mu}_n^y + \lambda_y \mathbf{w}_y - \frac{1}{J}\sum_{j=1}^{I} \boldsymbol{\pi}_j^y, \tag{6.6}$$

where $\boldsymbol{\mu}_n^y = (1 - s_n)\left(\frac{\partial \tau_1}{\partial d_n} - d_n^{(t)}\right)\mathbf{y}_{b_n}$ and

$$\boldsymbol{\pi}_j^y = \begin{cases} 0 & \text{if } |\mathbf{w}_y^T\mathbf{y}_j| \ge 1 \\ \mathrm{sgn}\left(\mathbf{w}_y^T\mathbf{y}_j\right)\mathbf{y}_j & \text{if } |\mathbf{w}_y^T\mathbf{y}_j| < 1 \end{cases}.$$

6.2.3 Algorithm

So far we have only discussed how to learn the hash functions for one bit of the hash codes. To learn the hash functions for multiple bits, one could repeat the same procedure and treat the learning for each bit independently. However, as reported in previous studies (Wang et al., 2010a; Liu et al., 2011), it is very important to take into consideration the relationships between different bits in HFL. In other words, to learn compact hash codes, we should coordinate the learning of hash functions for different bits.

To this end, we take the standard AdaBoost (Freund & Schapire, 1997) approach to learn multiple bits sequentially. Intuitively, this approach allows learning of the hash functions in later stages to be aware of the bias introduced by their antecedents. The overall algorithm of CRH is summarized in Algorithm 6.1.

Algorithm 6.1 Co-Regularized Hashing

Input:
$\mathcal{X}, \mathcal{Y}$ – multimodal data
Θ – inter-modality point pairs
K – code length
$\lambda_x, \lambda_y, \gamma$ – regularization parameters
a, λ – parameters for SCISD function
Output:
$\mathbf{w}_x^{(k)}, k = 1, \ldots, K$ – projection vectors for $\mathcal{X}$
$\mathbf{w}_y^{(k)}, k = 1, \ldots, K$ – projection vectors for $\mathcal{Y}$

Procedure:
Initialize $\omega_n^{(1)} = 1/N$, $\forall n \in \{1, 2, \ldots, N\}$.
for $k = 1$ **to** K **do**
  repeat
    Optimize Equation (6.3) to get $\mathbf{w}_x^{(k)}$;
    Optimize Equation (6.5) to get $\mathbf{w}_y^{(k)}$;
  until convergence.
  Compute error of current hash functions:

$$\epsilon_k = \sum\nolimits_{n=1}^{N} \omega_n^{(k)} \mathbf{I}_{[s_n \neq h_n]},$$

  where $\mathbf{I}_{[a]} = 1$ if a is true and $\mathbf{I}_{[a]} = 0$ otherwise, and

$$h_n = \begin{cases} 1 & f(\mathbf{x}_{a_n}) = g(\mathbf{y}_{b_n}) \\ 0 & f(\mathbf{x}_{a_n}) \neq g(\mathbf{y}_{b_n}) \end{cases}.$$

  Set $\beta_k = \epsilon_k / (1 - \epsilon_k)$.
  Update the weight for each point pair:

$$\omega_n^{(k+1)} = \omega_n^{(k)} \beta_k^{1 - \mathbf{I}_{[s_n \neq h_n]}}.$$

end for

The first computationally expensive part of the algorithm is to evaluate the gradients. The time complexity is $O((k+l)d)$, where d is the data dimensionality, and k and l are the numbers of random points and random pairs, respectively, for the stochastic gradient solver. In our experiments, we set $k = 1$ and $l = 500$. We notice that further increasing the two numbers brings no significant performance improvement. We leave the theoretical study of the impact of k and l to our future work. Another major computational cost comes from updating the weights of the inter-modality point pairs. The time complexity is $O(dN)$, where N is the number of inter-modality point pairs.

To summarize, our algorithm scales linearly with the number of inter-modality point pairs and the data dimensionality. In practice, the number of inter-modality point pairs is usually small, making our algorithm very efficient.

6.2.4 Extensions

We briefly discuss two possible extensions of CRH in this subsection. First, we note that it is easy to extend CRH to learn nonlinear hash functions via the kernel trick (Shawe-Taylor & Cristianini, 2004). Specifically, according to the generalized representer theorem (Schölkopf et al., 2001), we can represent the projection vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ as

$$\mathbf{w}_x = \sum\nolimits_{i=1}^{I} \alpha_i \phi_x(\mathbf{x}_i) \quad \text{and} \quad \mathbf{w}_y = \sum\nolimits_{j=1}^{J} \beta_j \phi_y(\mathbf{y}_j),$$

where $\phi_x(\cdot)$ and $\phi_y(\cdot)$ are kernel-induced feature maps for modalities $\mathcal{X}$ and $\mathcal{Y}$, respectively. Then the objective function (6.1) can be expressed in kernel form and kernel-based hash functions can be learned by minimizing a new but very similar objective function.

Another possible extension is to make CRH support more than two modalities. Taking a new modality $\mathcal{Z}$ for example, we need to incorporate into Equation (6.1) the following terms: loss and regularization terms for $\mathcal{Z}$, and all pairwise loss terms involving $\mathcal{Z}$ and other modalities, e.g., $\mathcal{X}$ and $\mathcal{Y}$.

For both extensions, it is straightforward to adapt the algorithm presented above to solve the new optimization problems.

6.2.5 Discussion

CRH is closely related to a recent multimodal metric learning method called MultiNPP (Quadrianto & Lampert, 2011), because CRH uses a loss function for inter-modality point pairs which is similar to MultiNPP. However, CRH is a general framework and other loss functions for inter-modality point pairs can also be adopted. The two methods have at least three significant differences. First, our focus is on HFL while MultiNPP is on

metric learning through embedding. Second, in addition to the inter-modality loss term, the objective function in CRH includes two intra-modality loss terms for large margin HFL while MultiNPP only has a loss term for the inter-modality point pairs. Third, CRH uses boosting to sequentially learn the hash functions but MultiNPP does not take this aspect into consideration.

As discussed briefly in (Quadrianto & Lampert, 2011), one may first use MultiNPP to map multimodal data into a common real space and then apply any unimodal HFL method for multimodal hashing. However, this naive two-stage approach has some limitations. First, both stages can introduce information loss which impairs the quality of the hash functions learned. Second, a two-stage approach generally needs more computational resources. These two limitations can be overcome by using a one-stage method such as CRH.

6.3 Experiments

6.3.1 Experimental Settings

In our experiments, we compare CRH with two state-of-the-art multimodal hashing methods, namely, CMSSH (Bronstein et al., 2010)[2] and CVH (Kumar & Udupa, 2011),[3] for two crossmodal retrieval tasks: (1) *image query vs. text database*; (2) *text query vs. image database*. The goal of each retrieval task is to find from the text (image) database the nearest neighbors for the image (text) query.

We use two benchmark data sets which are, to the best of our knowledge, the largest fully paired and labeled multimodal data sets. We further divide each data set into a database set and a query set. To train the models, we randomly select a group of documents from the database set to form the training set. Moreover, we randomly select 0.1% of the point pairs from the training set. For fair comparison, all models are trained on the same training set and the experiments are repeated 5 times.

The mean average precision (mAP) is used as the performance measure. To compute the mAP, we first evaluate the average precision (AP) of a set of R retrieved documents as $\mathrm{AP} = \frac{1}{L}\sum_{r=1}^{R} P(r)\,\delta(r)$, where L is the number of true neighbors in the retrieved set, $P(r)$ denotes the precision of the top r retrieved documents, and $\delta(r) = 1$ if the rth retrieved document is a true neighbor and $\delta(r) = 0$ otherwise. The mAP is then computed by averaging the AP values over all the queries in the query set. The larger the mAP, the better the performance. In the experiments, we set $R = 50$. Besides, we also report the

[2]We used the implementation generously provided by the authors.

[3]We implemented the method ourselves because the code is not publicly available.

precision and recall within a fixed Hamming radius.

We use cross-validation to choose the parameters for CRH and find that the model performance is only mildly sensitive to the parameters. As a result, in all experiments, we set $\lambda_x = 0.01, \lambda_y = 0.01, \gamma = 1000, a = 3.7$, and $\lambda = 1/a$. Besides, unless specified otherwise, we fix the training set size to 2,000 and the code length K to 24.

6.3.2 Results on Wiki Data Set

The Wiki data set, generated from Wikipedia featured articles, consists of 2,866 image-text pairs.[4] In each pair, the text is an article describing some events or people and the image is closely related to the content of the article. The images are represented by 128-dimensional SIFT (Lowe, 2004) feature vectors, while the text articles are represented by the probability distributions over 10 topics learned by a latent Dirichlet allocation (LDA) model (Blei et al., 2003). Each pair is labeled with one of 10 semantic classes. We simply use these class labels to identify the neighbors. Moreover, we use 80% of the data as the database set and the remaining 20% to form the query set.

The mAP values of the three methods and a method based on binarizing MultiNPP (Bin-MultiNPP) are reported in Table 6.1. We can see that CRH outperforms CVH and CMSSH under all settings and CVH performs slightly better than CMSSH. We note that CMSSH ignores the intra-modality relational information and CVH simply treats each bit independently. Hence the performance difference is expected. Also, we can see that binarizing MultiNPP directly always achieves the worst performance.

Table 6.1: mAP comparison on Wiki

Task	Method	Code Length		
		$K = 24$	$K = 48$	$K = 64$
Image Query vs. Text Database	CRH	$\mathbf{0.2537 \pm 0.0206}$	$\mathbf{0.2399 \pm 0.0185}$	$\mathbf{0.2392 \pm 0.0131}$
	CVH	0.2043 ± 0.0150	0.1788 ± 0.0149	0.1732 ± 0.0072
	CMSSH	0.1965 ± 0.0123	0.1780 ± 0.0080	0.1624 ± 0.0073
	Bin-MultiNPP	0.1790 ± 0.0202	0.1672 ± 0.0130	0.1628 ± 0.0175
Text Query vs. Image Database	CRH	$\mathbf{0.2896 \pm 0.0214}$	$\mathbf{0.2882 \pm 0.0261}$	$\mathbf{0.2989 \pm 0.0293}$
	CVH	0.2714 ± 0.0164	0.2304 ± 0.0104	0.2156 ± 0.0202
	CMSSH	0.2179 ± 0.0161	0.2094 ± 0.0072	0.2040 ± 0.0135
	Bin-MultiNPP	0.1925 ± 0.0173	0.1927 ± 0.0284	0.1847 ± 0.0195

We further compare the three methods on several aspects in Figure 6.1. We first vary the code length K and plot the precision within a Hamming radius of 2 in subfigures 6.1(a) and 6.1(b). As K increases, the performance of CRH also improves but the other two methods cannot benefit from increasing K. We then vary the size of the training set

[4] `http://www.svcl.ucsd.edu/projects/crossmodal/`

in subfigures 6.1(c) and 6.1(d). Although CVH performs the best when the training set is small, its performance is gradually surpassed by CRH as the size increases. In the remaining subfigures, we plot the precision-recall curves and recall curves for all three methods. It is obvious that CRH outperforms its two counterparts by a large margin.

6.3.3 Results on Flickr Data Set

The Flickr data set consists of 186,577 image-tag pairs pruned from the NUS data set[5] (Chua et al., 2009) by keeping the pairs that belong to one of the 10 largest classes. The images are represented by 500-dimensional SIFT vectors. To obtain more compact representations of the tags, we perform PCA on the original tag occurrence features and obtain 1000-dimensional feature vectors. Each pair is annotated by at least one of 10 semantic labels, and two points are defined as neighbors if they share at least one label. We use 99% of the data as the database set and the remaining 1% to form the query set.

The mAP values of the three methods are reported in Table 6.2. In the task of image query vs. text database, CRH performs comparably to CMSSH, which is better than CVH. However, in the other task, CRH achieves the best performance. At the same time, we observe that CRH beat Bin-MultiNPP by a large margin.

Table 6.2: mAP comparison on Flickr

Task	Method	Code Length		
		$K = 24$	$K = 48$	$K = 64$
Image Query vs. Text Database	CRH	0.5259 ± 0.0094	0.4990 ± 0.0075	$\mathbf{0.4929 \pm 0.0064}$
	CVH	0.4717 ± 0.0035	0.4515 ± 0.0041	0.4471 ± 0.0023
	CMSSH	$\mathbf{0.5287 \pm 0.0123}$	$\mathbf{0.5098 \pm 0.0141}$	0.4911 ± 0.0220
	Bin-MultiNPP	0.4775 ± 0.0211	0.4527 ± 0.0143	0.4446 ± 0.0259
Text Query vs. Image Database	CRH	$\mathbf{0.5364 \pm 0.0021}$	$\mathbf{0.5185 \pm 0.0050}$	$\mathbf{0.5064 \pm 0.0055}$
	CVH	0.4598 ± 0.0020	0.4519 ± 0.0029	0.4477 ± 0.0058
	CMSSH	0.5029 ± 0.0321	0.4815 ± 0.0101	0.4660 ± 0.0298
	Bin-MultiNPP	0.4767 ± 0.0147	0.4524 ± 0.0115	0.4450 ± 0.0125

Similar to the previous subsection, we have conducted a group of experiments to compare the three methods on several aspects and report the results in Figure 6.2. We first compare the precision under different code lengths in subfigures 6.2(a) and 6.2(a). In almost all code lengths, CRH outperforms the other two methods. The results for varying the size of the training set are plotted in subfigures 6.2(c) and 6.2(c). As more training data are used, CRH always performs better but the performance of CVH and CMSSH has high variance. Finally, the precision-recall curves and recall curves are shown in the

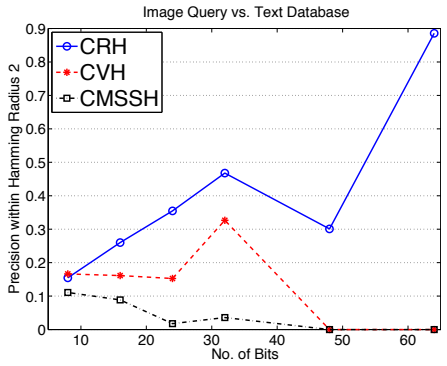
[5] `http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm`

remaining subfigures. Similar to the results on Wiki, CRH performs the best. However, the performance gap is smaller here.

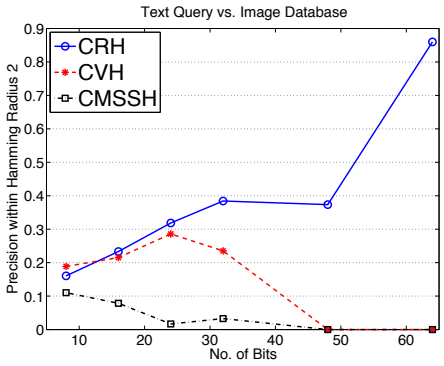
6.4 Conclusion

In this chapter, we have presented a novel method for multimodal hash function learning based on a boosted co-regularization framework which is named co-regularized hashing (CRH). In CRH, there is no data assumption such as those the data are aligned or organized in graphs. Because the objective function of the optimization problem is in the form of a difference of convex functions, we develop an efficient learning algorithm based on CCCP and a stochastic gradient method. Experimental study based on two benchmark data sets shows that CRH outperforms two state-of-the-art multimodal hashing methods.

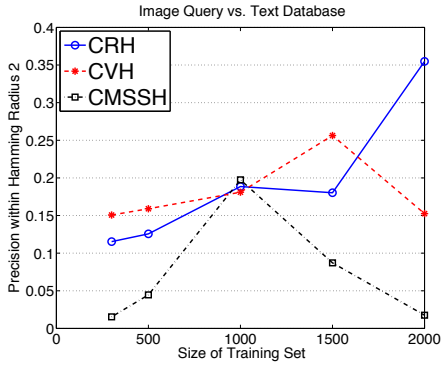
To take this work further, we would like to conduct theoretical analysis of CRH and apply it to some other tasks such as multimodal medical image alignment. Another possible research issue is to develop sublinear optimization algorithms to further improve the scalability of CRH.



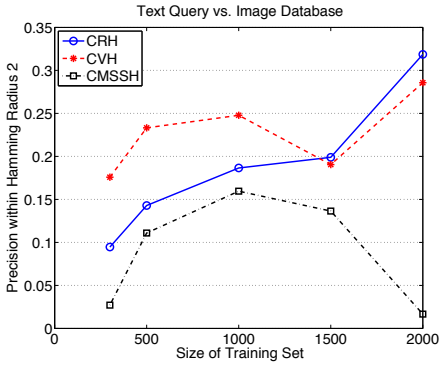
(a) Varying Code Length



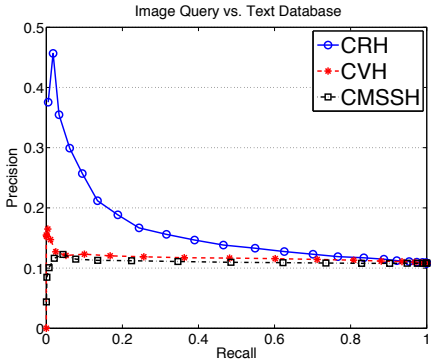
(b) Varying Code Length



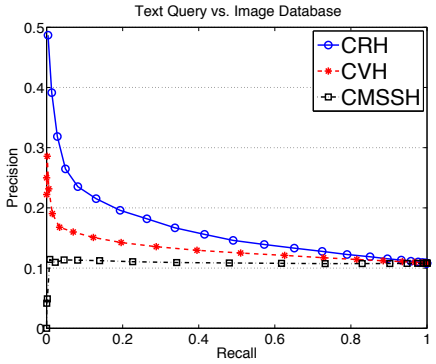
(c) Varying Training Set



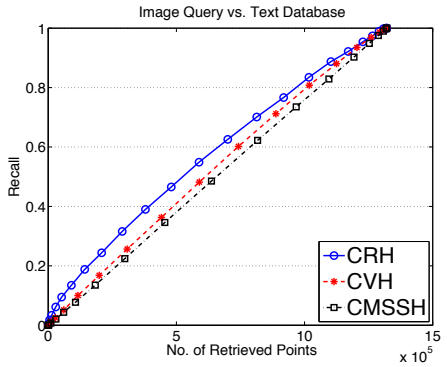
(d) Varying Training Set



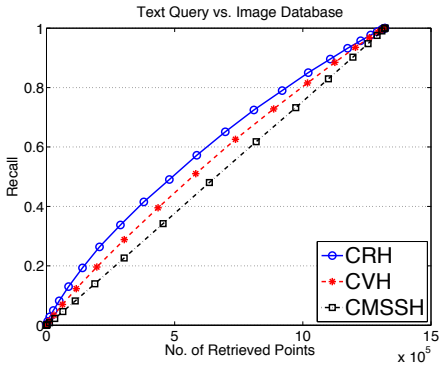
(e) Pre-Rec Curve



(f) Pre-Rec Curve

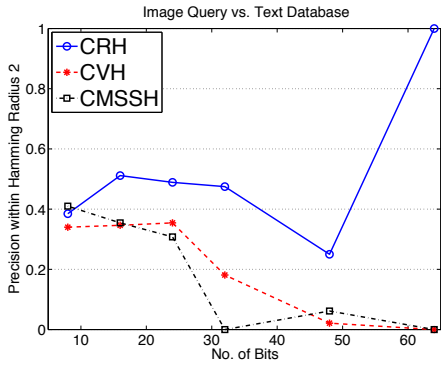


(g) Recall Curve

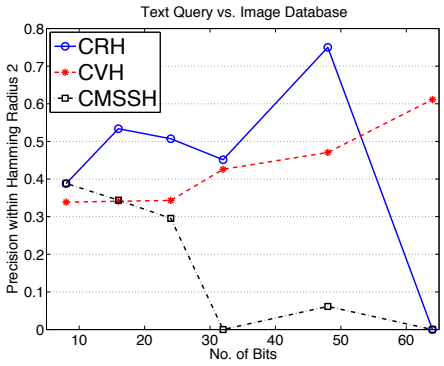


(h) Recall Curve

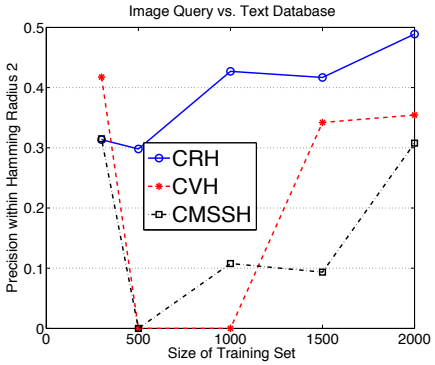
Figure 6.1: Results on Wiki



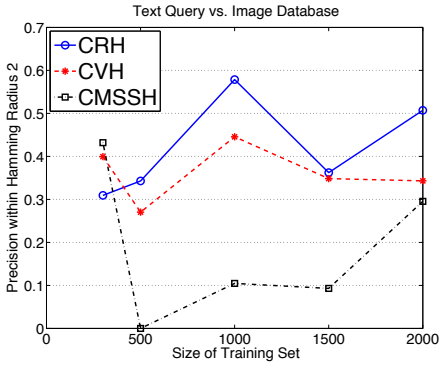
(a) Varying Code Length



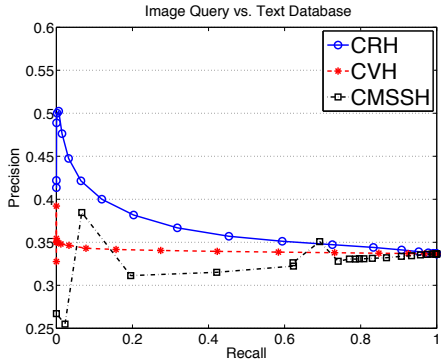
(b) Varying Code Length



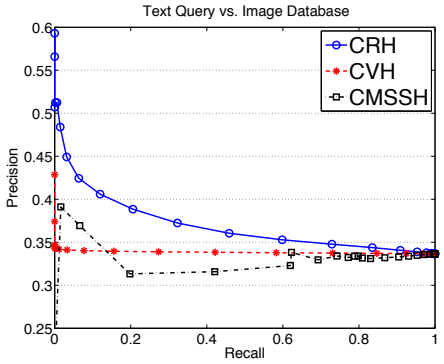
(c) Varying Training Set



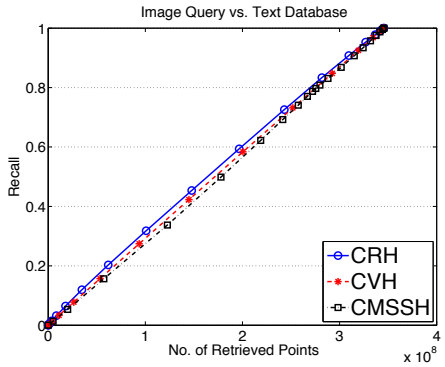
(d) Varying Training Set



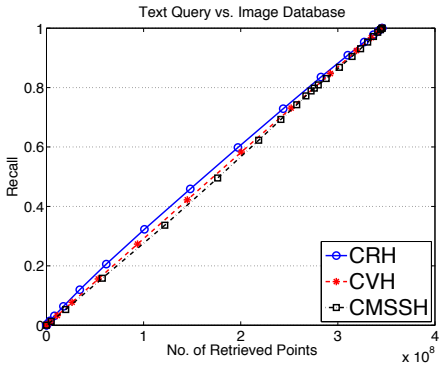
(e) Pre-Rec Curve



(f) Pre-Rec Curve



(g) Recall Curve



(h) Recall Curve

Figure 6.2: Results on Flickr

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

As a new research topic in the machine learning area, hash function learning has attracted plenty of interest in recent years. Different from locality sensitive hashing methods which design data-independent hash functions for specific similarity measures, hash function learning methods learn data-dependent hash functions directly from the data.

In the present thesis, we explore two research issues in hash function learning, namely, active hashing and multimodal hashing. Active hashing aims to actively select the data from which to learn hash functions, whereas multimodal hashing focuses on learning the hash functions for data involving multiple modalities. The contributions of this thesis are summarized as follows:

Active Hashing To eliminate the data selection bias and reduce the labeling cost of existing supervised or semi-supervised hash function learning methods, we propose the active hashing framework. Under this framework, the hash functions are learned from the data which are carefully selected before learning. To evaluate the data informativeness, a simple yet effective criterion which is based on uncertainty is proposed. We also present a batch mode algorithm for the data selection procedure. Through extensive experiments, we demonstrate that active hashing outperforms passive hashing by a large margin.

Spectral Multimodal Hashing To learn hash functions for multimodal data, we propose the spectral multimodal hashing method (SMH). Given that the data in different modalities are aligned, that is, every object has a representation in each modality, SMH learns the hash functions through spectral analysis of modality correlation. Due to its resemblance to canonical correlation analysis in statistics, SMH is easy to extend to support kernel similarity and more than two modalities. We also propose a novel method to learn the thresholds for the hash functions. Experimental results show that our SMH model outperforms the state-of-the-art methods.

Multimodal Latent Binary Embedding For the multimodal data which are organized in similarity graphs, we propose a novel probabilistic model called multimodal latent binary embedding (MLBE) to learn the hash functions. As a latent factor

model, MLBE regards the binary latent factors as hash codes and hence maps data points from multiple modalities to a common Hamming space in a principled probabilistic manner. Although finding exact posterior distributions of the latent factors (hash codes) is intractable, we devise an efficient alternating learning algorithm based on MAP estimation. Experimental results show that MLBE compares favorably with state-of-the-art multimodal hashing models.

Co-regularized Hashing For the general multimodal data, we propose a new method based on two popular supervised learning techniques: boosting and co-regularization, and name it co-regularized hashing (CRH). We no longer assume that the data are aligned or organized in graphs in CRH. To learn each bit of the hash codes, the objective function is in the form of a difference of convex functions, and we develop an efficient learning algorithm based on the convex-concave procedure (CCCP) and a stochastic gradient method. Hash functions for different bits are learned using a standard boosting method. Comparative studies based on benchmark data sets show that CRH outperforms two state-of-the-art multimodal hashing methods.

7.2 Future Work

To take our work further, we wish to study the following topics in the near future:

Challenging issues in Multimodal Hashing In multimodal hash function learning, there are several challenging issues worth further studying. Among other things, it is the most important to learn an appropriate length of hash codes automatically from data. It is well known that the hash codes generated by HFL methods are always much shorter than those generated by LSH methods, however, there is no benchmark result on learning the code length.

Hierarchical Hash Function Learning Most existing HFL methods assume that different hash bits are equally important, in other words, there is no structure in the hash codes. This assumption can be easily violated in applications where similarity search should be performed in different resolutions or levels. Although several researchers have made some preliminary attempts on generating multiple bits based on one hash function (Liu et al., 2011), to the best of our knowledge, no result has been reported on hash functions with structures.

We propose to develop hierarchical HFL methods. In addition to supporting different levels of search, the structural information of hash codes can be used to reduce the search space and hence further improve the scalability. For example, one can use coarse-level hash codes to find a candidate space and then fine-level hash codes

for nearest neighbors. Moreover, hierarchical HFL can be used to combine different hashing models. For example, given that some hash functions are sensitive to recall while others are sensitive to precision, we can organize different kinds of hash codes in such a hierarchy that the new model can take advantages of different models.

Sublinear Hash Function Learning Many HFL algorithms rely on expensive matrix operations such as eigen-decompostion and inversion. Obviously, they are not suitable for very high-dimensional data, such as text and image, unless the data are properly preprocessed. Although recent progress shows that linear time complexity can be expected (Lin et al., 2010), complicated approximation techniques are required in these methods. Furthermore, even linear time algorithms are sometimes not feasible in real large-scale applications.

Recently, a family of sublinear algorithms has emerged in the machine learning area (Hazan et al., 2011). These algorithms have been applied to learning support vector machines (SVM) for classification problems. We believe that it is also possible to develop sublinear HFL algorithm, because some HFL algorithms such as CRH are similar to SVM training algorithms in some sense.

Collaborative Filtering and Hash Function Learning Almost all HFL methods focus on similarity search applications. Nevertheless, other large-scale data mining tasks can also benefit from HFL. One outstanding example is collaborative filtering (CF), which is a very important technique for data mining and, more specifically, the recommender systems. So far as we know, several non-learning based hashing methods have been used to improve collaborative filtering (Shi et al., 2009; Weinberger et al., 2009), and we believe that HFL can make further improvement.

One possible approach may be to formulate CF as a multimodal hashing problem. To recommend movies to users, we just find nearest neighbor movies of the users in a common Hamming space. The learning problem needs to take into consideration the collaborative rating matrix as well as other side information, and the whole learning procedure should be conducted in sublinear time.

Bibliography

Andoni, A., & Indyk, P. (2006a). Efficient algorithms for substring near neighbor problem. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1203–1212.

Andoni, A., & Indyk, P. (2006b). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 459–468.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*(4), 319–342.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, *45*(6), 891–923.

Athitsos, V., Potamias, M., Papapetrou, P., & Kollios, G. (2008). Nearest neighbor retrieval using distance-based hashing. In *IEEE 24th International Conference on Data Engineering*, pp. 327–336.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. ACM Press.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Blaschko, M. B., Lampert, C. H., & Gretton, A. (2008). Semi-supervised Laplacian regularization of kernel canonical correlation analysis. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 133–145.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brand, M. (2002). Charting a manifold. In *Advances in Neural Information Processing Systems 15*, pp. 961–968.

Broder, A. Z. (1997). On the resemblance and containment of documents. In *International Conference on Compression and Complexity of Sequences*, pp. 21–29.

Broder, A. Z., Charikar, M., Frieze, A. M., & Mitzenmacher, M. (1998). Min-wise independent permutations. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 327–336.

Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997). Syntactic clustering of the web. In *Proceedings of the 6th International Conference on World Wide Web*, pp. 1157–1166.

Bronstein, M. M., Bronstein, A. M., Michel, F., & Paragios, N. (2010). Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3594–3601.

Cayton, L., & Dasgupta, S. (2007). A learning framework for nearest neighbor search. In *Advances in Neural Information Processing Systems 20*, pp. 233–240.

Charikar, M. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pp. 380–388.

Chua, T.-S., Tang, J., Hong, R., Li, H., Luo, Z., & Zheng, Y.-T. (2009). NUS-WIDE: A real-world web image database from National University of Singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pp. 48:1–48:9.

Chum, O., Perdoch, M., & Matas, J. (2009). Geometric min-hashing: Finding a (thick) needle in a haystack. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 17–24.

Chung, F. R. K. (1996). *Spectral Graph Theory*. American Mathematical Society.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms*. MIT Press.

Culotta, A., & McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence*, pp. 746–751.

Dasgupta, A., Kumar, R., & Sarlos, T. (2011). Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1073–1081.

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pp. 253–262.

Davis, J. V., Kulis, B., Jain, P., Sra, S., & Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 209–216.

Dong, W., Charikar, M., & Li, K. (2008). Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 123–130.

Eshghi, K., & Rajaram, S. (2008). Locality sensitive hash functions based on concomitant rank order statistics. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 221–229.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3*(3), 209–226.

Fujii, A., Tokunaga, T., Inui, K., & Tanaka, H. (1998). Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, *24*(4), 573–597.

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 518–529.

Globerson, A., & Roweis, S. (2005). Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems 18*, pp. 451–458.

Goldberger, J., Roweis, S., Hinton, G. E., & Salakhutdinov, R. (2004). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pp. 513–520.

Gordo, A., & Perronnin, F. (2011). Asymmetric distances for binary embeddings. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 729–736.

Grauman, K., & Darrell, T. (2007). Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Griffiths, T. L., & Ghahramani, Z. (2005). Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems 18*, pp. 475–482.

Guo, Y., & Greiner, R. (2007). Optimistic active learning using mutual information. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 823–829.

Guo, Y., & Schuurmans, D. (2007). Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems 20*, pp. 593–600.

Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, *16*(12), 2639–2664.

Hazan, E., Koren, T., & Srebro, N. (2011). Beating SGD: Learning SVMs in sublinear time. In *Advances in Neural Information Processing Systems 24*, pp. 1233–1241.

He, J., Liu, W., & Chang, S.-F. (2010). Scalable similarity search with optimized kernel hashing. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1129–1138.

He, J., Radhakrishnan, R., Chang, S.-F., & Bauer, C. (2011). Compact hashing with joint optimization of search accuracy and time. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 753–760.

He, X., Min, W., Cai, D., & Zhou, K. (2007). Laplacian optimal design for image retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 119–126.

He, X., & Niyogi, P. (2003). Locality preserving projections. In *Advances in Neural Information Processing Systems 16*.

Heller, K. A., & Ghahramani, Z. (2007). A nonparametric Bayesian approach to modeling overlapping clusters. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pp. 524–531.

Hinton, G. E., & Roweis, S. (2002). Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pp. 833–840.

Hinton, G. E., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–509.

Hoi, S. C. H., Jin, R., & Lyu, M. R. (2006a). Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web*, pp. 633–642.

Hoi, S. C., Jin, R., Zhu, J., & Lyu, M. R. (2006b). Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 417–424.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, *28*(3/4), 321–377.

Hwa, R. (2004). Sample selection for statistical parsing. *Computational Linguistics*, *30*(3), 253–276.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 604–613.

Jain, P., Kulis, B., & Grauman, K. (2008). Fast image search for learned metrics. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Kulis, B., & Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems 22*, pp. 1042–1050.

Kulis, B., & Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of IEEE 12th International Conference on Computer Vision*, pp. 2130 –2137.

Kumar, S., & Udupa, R. (2011). Learning hash functions for cross-view similarity search. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1360–1365.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*, pp. 148–156.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12.

Li, P., & König, A. C. (2010). *b*-Bit minwise hashing. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 671–680.

Li, P., König, A. C., & Gui, W. (2010). *b*-Bit minwise hashing for estimating three-way similarities. In *Advances in Neural Information Processing Systems 23*, pp. 1387–1395.

Li, P., Shrivastava, A., Moore, J., & König, A. C. (2011). Hashing algorithms for large-scale learning. In *Advances in Neural Information Processing Systems 24*, pp. 2672–2680.

Lin, R.-S., Ross, D. A., & Yagnik, J. (2010). SPEC hashing: Similarity preserving algorithm for entropy-based coding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 848–854.

Lindenbaum, M., Markovitch, S., & Rusakov, D. (2004). Selective sampling for nearest neighbor classifiers. *Machine Learning*, *54*(2), 125–152.

Liu, W., He, J., & Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 679–686.

Liu, W., Wang, J., Kumar, S., & Chang, S.-F. (2011). Hashing with graphs. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 1–8.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Manku, G. S., Jain, A., & Das Sarma, A. (2007). Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 141–150.

McCallum, A., & Nigam, K. (1998). Employing EM and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 350–358.

Meeds, E., Ghahramani, Z., Neal, R., & Roweis, S. T. (2006). Modeling dyadic data with binary latent factors. In *Advances in Neural Information Processing Systems 19*, pp. 977–984.

Mu, Y., Shen, J., & Yan, S. (2010). Weakly-supervised hashing in kernel space. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3344–3351.

Mu, Y., & Yan, S. (2010). Non-metric locality-sensitive hashing. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 539–544.

Nguyen, H. T., & Smeulders, A. (2004). Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 79–86.

Norouzi, M., & Fleet, D. J. (2011). Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 353–360.

Quadrianto, N., & Lampert, C. H. (2011). Learning multi-view neighborhood preserving projections. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 425–432.

Raginsky, M., & Lazebnik, S. (2009). Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems 22*, pp. 1509–1517.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pp. 1177–1184.

Rasiwasia, N., Costa Pereira, J., Coviello, E., Doyle, G., Lanckriet, G. R., Levy, R., & Vasconcelos, N. (2010). A new approach to cross-modal multimedia retrieval. In *Proceedings of the 18th ACM International Conference on Multimedia*, pp. 251–260.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning*, pp. 441–448.

Salakhutdinov, R., & Hinton, G. E. (2009a). Deep Boltzmann machines. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 448–455.

Salakhutdinov, R., & Hinton, G. E. (2009b). Semantic hashing. *International Journal of Approximate Reasoning*, *50*, 969–978.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523.

Schapire, R. E. (1999). A brief introduction to Boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 1401–1406.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, pp. 416–426.

Settles, B. (2009). Active learning literature survey. Tech. rep., University of Wisconsin–Madison.

Settles, B., & Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1070–1079.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. In *the 5th Annual Workshop on Computational Learning Theory*, pp. 287–294.

Shakhnarovich, G. (2005). *Learning Task-Specific Similarity*. Ph.D. thesis, Massachusetts Institute of Technology.

Shakhnarovich, G., Darrell, T., & Indyk, P. (Eds.). (2006). *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press.

Shakhnarovich, G., Viola, P., & Darrell, T. (2003). Fast pose estimation with parameter-sensitive hashing. In *Proceedings of IEEE 9th International Conference on Computer Vision*, pp. 750–757.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 807–814.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shental, N., Hertz, T., Weinshall, D., & Pavel, M. (2002). Adjustment learning and relevant component analysis. In *Proceedings of the 7th European Conference on Computer Vision*, pp. 776–792.

Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A., Strehl, A., & Vishwanathan, S. (2009). Hash kernels. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 496–503.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Tong, S. (2001). *Active Learning: Theory and Applications*. Ph.D. thesis, Stanford University.

Tong, S., & Koller, D. (2002). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, *2*, 45–66.

Torralba, A., Fergus, R., & Weiss, Y. (2008). Small codes and large image databases for recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 104–111.

Ture, F., Elsayed, T., & Lin, J. (2011). No free lunch: Brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 943–952.

Turpin, A., & Scholer, F. (2006). User performance versus precision measures for simple search tasks. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 11–18.

Wang, J., Kumar, S., & Chang, S.-F. (2010a). Semi-supervised hashing for scalable image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3424–3431.

Wang, J., Kumar, S., & Chang, S.-F. (2010b). Sequential projection learning for hashing with compact codes. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1127–1134.

Weber, R., Schek, H.-J., & Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pp. 194–205.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 1113–1120.

Weiss, Y., Torralba, A., & Fergus, R. (2008). Spectral hashing. In *Advances in Neural Information Processing Systems 21*, pp. 1753–1760.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pp. 505–512.

Xu, Z., Yu, K., Tresp, V., Xu, X., & Wang, J. (2003). Representative sampling for text classification using support vector machines. In *Proceedings of the 25th European Conference on IR Research*, pp. 393–407.

Xu, Z., Akella, R., & Zhang, Y. (2007). Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the 29th European Conference on IR Research*, pp. 246–257.

Yang, L., & Jin, R. (2006). Distance metric learning: A comprehensive survey. Tech. rep., Michigan State University.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 311–321.

Yu, K., Bi, J., & Tresp, V. (2006). Active learning via transductive experimental design. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 1081–1088.

Yu, K., Zhu, S., Xu, W., & Gong, Y. (2008). Non-greedy active learning for text categorization using convex transductive experimental design. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 635–642.

Yuille, A. L., & Rangarajan, A. (2001). The concave-convex procedure (CCCP). In *Advances in Neural Information Processing Systems 14*, pp. 1–8.

Zhang, D., Wang, F., & Si, L. (2011). Composite hashing with multiple information sources. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 225–234.

Zhang, D., Wang, J., Cai, D., & Lu, J. (2010a). Extensions to self-taught hashing: Kernelisation and supervision. In *SIGIR Workshop on Feature Generation and Selection for Information Retrieval*.

Zhang, D., Wang, J., Cai, D., & Lu, J. (2010b). Self-taught hashing for fast similarity search. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 18–25.

Zhang, W., Gao, K., Zhang, Y., & Li, J. (2011). Efficient approximate nearest neighbor search with integrated binary codes. In *Proceedings of the 19th ACM international conference on Multimedia*, pp. 1189–1192.

Zhen, Y., & Yeung, D.-Y. (2010). Supervised experimental design and its application to text retrieval. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 299–306.

Zhu, X., Lafferty, J., & Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using Gaussian fields and Harmonic functions. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.