A Probabilistic Model for Multimodal Hash Function Learning

Yi Zhen and Dit-Yan Yeung
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{yzhen,dyyeung}@cse.ust.hk

ABSTRACT

In recent years, both hashing-based similarity search and multimodal similarity search have aroused much research interest in the data mining and other communities. While hashing-based similarity search seeks to address the scalability issue, multimodal similarity search deals with applications in which data of multiple modalities are available. In this paper, our goal is to address both issues simultaneously. We propose a probabilistic model, called *multimodal latent binary embedding* (MLBE), to learn hash functions from multimodal data automatically. MLBE regards the binary latent factors as hash codes in a common Hamming space. Given data from multiple modalities, we devise an efficient algorithm for the learning of binary latent factors which corresponds to hash function learning. Experimental validation of MLBE has been conducted using both synthetic data and two realistic data sets. Experimental results show that MLBE compares favorably with two state-of-the-art models.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.4 [**Information Systems Applications**]: Miscellaneous; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms*

Keywords

Hash Function Learning, Binary Latent Factor Models, Multimodal Similarity Search, Metric Learning

1. INTRODUCTION

Similarity search, *a.k.a.* nearest neighbor search, is a fundamental problem in many data mining, database, and information retrieval applications [1, 34]. Given a query document[1], the similarity search problem can be regarded as finding one or more nearest neighbors of the query from a database according to some similarity measure chosen.

There are two challenging issues to address in similarity search, namely, the scalability and task-specificity issues [1, 33]. The scalability issue addresses the challenge when the database searched is of very large scale, possibly containing millions or even billions of documents. As for the task-specificity issue, the concern is that the similarity measure used should not be a generic one, but it should be specific to the application at hand to ensure that the nearest neighbors found are relevant.

Two major approaches have been proposed to address the scalability issue. One approach employs tree-based methods to organize data using data structures based on trees [3, 11]. The other approach uses hashing-based methods to map documents into bins such that collisions in the hash table reflect nearest neighbor relationships [12, 10, 9]. Although tree-based methods work well on low-dimensional data, they can easily degenerate into brute-force search as the data dimensionality increases. This limitation makes tree-based methods unappealing for real applications in which high dimensionality is commonly encountered. On the contrary, hashing-based methods can index data with compact binary codes and hence can perform very fast search without suffering from the curse of dimensionality.

The most well-known hashing-based methods belong to the locality-sensitive hashing (LSH) family [5, 17, 7, 2], in which hash functions are constructed (but not learned) based on random projections or permutations. Due to their simplicity and effectiveness, LSH algorithms have been successfully applied to many applications [19, 37, 42]. However, they often generate very long codes mainly due to their data independence nature. In other words, the hash functions are designed for some standard similarity measures which may not suit the application at hand. This is an example of the task-specificity issue in similarity search.

To address the two issues mentioned above, some research attempts in the past few years pursue a data-dependent approach by applying machine learning techniques to learn the hash functions from data automatically. We refer to this new direction as hash function learning (HFL).

To the best of our knowledge, Shakhnarovich *et al.* [35] made the first attempt to learn hash functions using a well-known machine learning algorithm, namely, a boosting algorithm [32]. Later, a method called semantic hashing [31] was proposed based on stacked restricted Boltzmann ma-

[1]In this paper, we use the term 'document' in a generic sense to refer to data from any modality, such as text, image, audio or even their combinations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'12, August 12–16, 2012, Beijing, China.
Copyright 2012 ACM 978-1-4503-1462-6/12/08... $15.00.

chines (RBMs) [16]. Using a large image database consisting of millions of images [36], it has been demonstrated that these two methods are much more effective than LSH. In a subsequent method called spectral hashing [40], a metric learning approach is used and the hash codes are found by spectral decomposition. Although spectral hashing is faster and more effective than the previous two methods, it takes a restrictive and unrealistic assumption that the data are uniformly distributed in a hyper-rectangle. Several new methods have since been proposed to relax this restrictive assumption, such as self-taught hashing [44], binary reconstructive embeddings [18], distribution matching [22], and shift-invariant kernel hashing [29]. Compared to spectral hashing, they have shown superior performance. Some more recent HFL methods focus on hashing in several special and important settings, including kernelized hashing [26, 14, 27], semi-supervised hashing [38, 39, 23], composite hashing [43], and active hashing [45].

Most existing HFL methods can only be applied to unimodal data. However, for a growing number of applications, it is also common to conduct similarity search on multimodal data with data belonging to multiple modalities. For example, given an image about a historic event, one may want to find some text articles that describe the event in detail. As a result, developing multimodal HFL methods is a very worthwhile research direction to explore. Up to now, however, only very few such attempts have been made [6, 20].

In this paper, we study hashing-based similarity search in the context of multimodal data. We propose a probabilistic latent factor model, called *multimodal latent binary embedding* (MLBE), to learn hash functions from multimodal data automatically. 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. Given data from multiple modalities, we devise an efficient algorithm for learning the binary latent factors based on *maximum a posteriori* (MAP) estimation. Compared to its counterparts [6, 20], 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 paper is organized as follows. Section 2 briefly introduces some recent related work. Section 3 presents our model and the learning algorithm. Experimental validation of MLBE conducted using both synthetic data and two realistic data sets is presented in Section 4. Finally, Section 5 concludes the paper.

2. RELATED WORK

Our work bears some resemblance to metric learning which aims at learning similarity or distance measures from data [41]. Such data-dependent similarity measures are generally more effective than their data-independent counterparts. Although a lot of research has been conducted on metric learning, multimodal metric learning is still largely unexplored even though multimodal data are commonly found in many applications. Some recent efforts have been made on non-hashing-based methods [21, 28]. Compared with hashing-based methods, these methods do not have the merits of low storage requirement and high search speed.

To the best of our knowledge, Bronstein *et al.* proposed the first hashing-based model, called cross-modal similarity sensitive hashing (CMSSH) thereafter, for multimodal similarity search [6]. Specifically, given a set of similar and dissimilar point pairs, CMSSH constructs two groups of linear hash functions (for the bimodal case) such that, with high probability, the Hamming distance after mapping is small for similar points and large for dissimilar points. In their formulation, each hash function (for one bit) can be obtained by solving a singular value decomposition (SVD) problem and the hash functions are learned sequentially in a standard boosting manner. However, CMSSH ignores the intra-modality relational information which could be very useful [40].

Recently, Kumar *et al.* extended spectral hashing [40] to the multi-view case, leading to a method called cross-view hashing (CVH) [20]. 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.

Both CMSSH and CVH have archieved 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 paper.

Our work is also related to binary latent factor models [25, 15] 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 [25, 15] 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 [25, 15], the priors on the binary latent factors are Indian buffet processes [13]. Furthermore, from a matrix factorization point of view, MLBE simultaneously factorizes multiple matrices but the method in [25] factorizes only one matrix.

3. MULTIMODAL LATENT BINARY EMBEDDING

We present MLBE in detail in this section. We use boldface uppercase and lowercase letters to denote matrices and vectors, respectively. For a matrix $\mathbf{A}$, its (i, j)th element is denoted by A_{ij}.

3.1 Model Formulation

For simplicity of our presentation, we focus exclusively on the bimodal case in this paper, 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

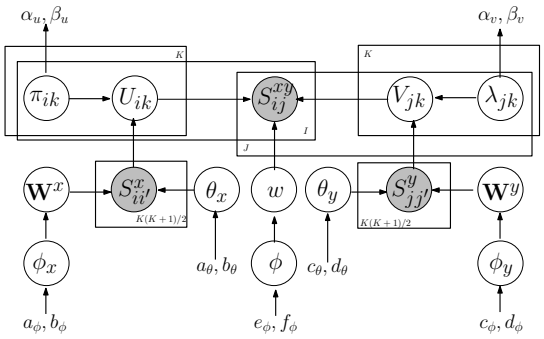


Figure 1: Graphical model representation of MLBE

$\mathcal{Y}$ 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-modal similarities this way, because it is very difficult if not impossible to specify real-valued cross-modal 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 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 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

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

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

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

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

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^{xy}_{ij} 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:[2]

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

where $\mathrm{Beta}(\mu \mid a, b)$ 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 $\mathbf{U}$:

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

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

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

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

[2]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 $\{0, 1\}$.

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 distributions:

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

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.

3.2 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 3.1:

Definition 3.1.

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

where $\mathbf{u}_i^+$ is the ith row of $\mathbf{U}$ with $U_{ik}=1$, $\mathbf{u}_i^-$ is the ith 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 3.1. *The MAP solution of U_{ik} is equal to 1 if $\mathcal{L}_{ik} \geq 0$ and -1 otherwise.*

The proof of Theorem 3.1 can be found in Appendix A.1.

Similarly, we have Definition 3.2 and Theorem 3.2 for the MAP estimation of $\mathbf{V}$. The proof of Theorem 3.2 is similar and so it is omitted in the paper due to page limitations.

Definition 3.2.

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

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 3.2. *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 3.3 below. The proof is in Appendix A.2.

Theorem 3.3. *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 $\mathbf{S}^x$.

Similarly, we have Theorem 3.4 for $\mathbf{W}^y$.

Theorem 3.4. *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:

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

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:

$$\begin{aligned}
\nabla w = &\phi\cdot w \qquad (1)\\
&-\sum_{i=1}^{I}\sum_{j=1}^{J}\left\{O_{ij}\left[S^{xy}_{ij}\left(1-\lambda_{ij}\right)\mathbf{u}_i^T\mathbf{v}_j - \left(1-S^{xy}_{ij}\right)\lambda_{ij}\mathbf{u}_i^T\mathbf{v}_j\right]\right\}.
\end{aligned}$$

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

THEOREM 3.5. *The MAP estimates of $\theta_x, \theta_y, \phi_x, \phi_y$ and ϕ are:*

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

$$\theta_y = \frac{J(J+1) + 4(c_\theta - 1)}{4d_\theta + 2\sum_{j=1}^{J}\sum_{j'=j}^{J}\left(S^y_{jj'} - \mathbf{v}_j^T\mathbf{W}^y\mathbf{v}_{j'}\right)^2}, \quad (3)$$

$$\phi_x = \frac{K(K+1) + 4(a_\phi - 1)}{4b_\phi + 2\sum_{k=1}^{K}\sum_{d=k}^{K}\left(W^x_{kd}\right)^2}, \quad (4)$$

$$\phi_y = \frac{K(K+1) + 4(c_\phi - 1)}{4d_\phi + 2\sum_{k=1}^{K}\sum_{d=k}^{K}\left(W^y_{kd}\right)^2}, \quad (5)$$

$$\phi = \frac{2e_\phi - 1}{2f_\phi + w^2}. \quad (6)$$

Theorem 3.5 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 1.

Algorithm 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

begin
  Initialize all the latent variables and parameters except $\mathbf{W}^x, \mathbf{W}^y$;
  while *not converged* **do**
    Update $\mathbf{W}^x$ using Theorem 3.3;
    Update ϕ_x using Equation (4);
    Update each element of $\mathbf{U}$ using Theorem 3.1;
    Update θ_x using Equation (2);
    Update $\mathbf{W}^y$ using Theorem 3.4;
    Update ϕ_y using Equation (5);
    Update each element of $\mathbf{V}$ using Theorem 3.2;
    Update θ_y using Equation (3);
    Update w by gradient descent using Equation (1);
    Update ϕ using Equation (6).
end

3.3 Out-of-Sample Extension

Algorithm 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.

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.[3] 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 3.1 or Theorem 3.2. For illustration, Algorithm 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.

Algorithm 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

begin
  Initialize $\mathbf{u}^*$;
  while *not converged* **do**
    Update each element of $\mathbf{u}^*$ using Theorem 3.1.
end

3.4 Complexity Analysis

The computational cost of the 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. 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.

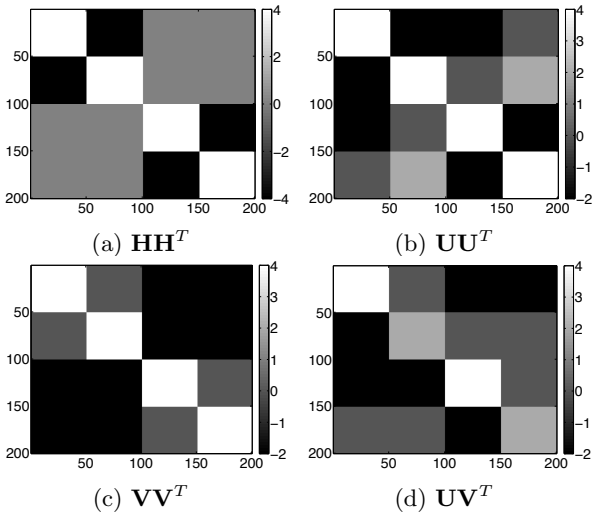
4. EXPERIMENTS

We first present an illustrative example on synthetic data in Section 4.1. It is then followed by experiments on two publicly available real-world data sets. Section 4.2 presents the experimental settings and then Sections 4.3 and 4.4 present the results. The code and data can be downloaded at `http://www.cse.ust.hk/~dyyeung/code/mlbe/`.

4.1 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, $[1, 1, -1, -1]$, $[-1, -1, 1, 1]$, $[1, -1, 1, -1]$ 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.

[3]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.



(a) $\mathbf{HH}^T$ (b) $\mathbf{UU}^T$ (c) $\mathbf{VV}^T$ (d) $\mathbf{UV}^T$

Figure 2: Illustration of MLBE

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 2. Note that the whiter the area is, the more similar the points involved are. Figure 2(a) depicts the ground-truth similarity structure, Figure 2(b) and 2(c) show the learned intra-modality similarity structure for each modality, and Figure 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 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.

4.2 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[4] and CVH[5] on two common cross-modal 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

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

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 randomly selected entries of $\mathbf{S}^{xy}$ are observed.[6] To be fair, all three models are trained on the same training set.

4.3 Results on Wiki Data Set

The Wiki data set is generated from a group of 2,866 Wikipedia documents provided by [30]. 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 [24] feature vectors. The text articles are represented by the probability distributions over 10 topics, which are derived from a latent Dirichlet allocation (LDA) model [4]. 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 1. The *precision-recall* curves and *recall* curves for the three methods are plotted in Figure 3.

Table 1: mAP comparison on Wiki

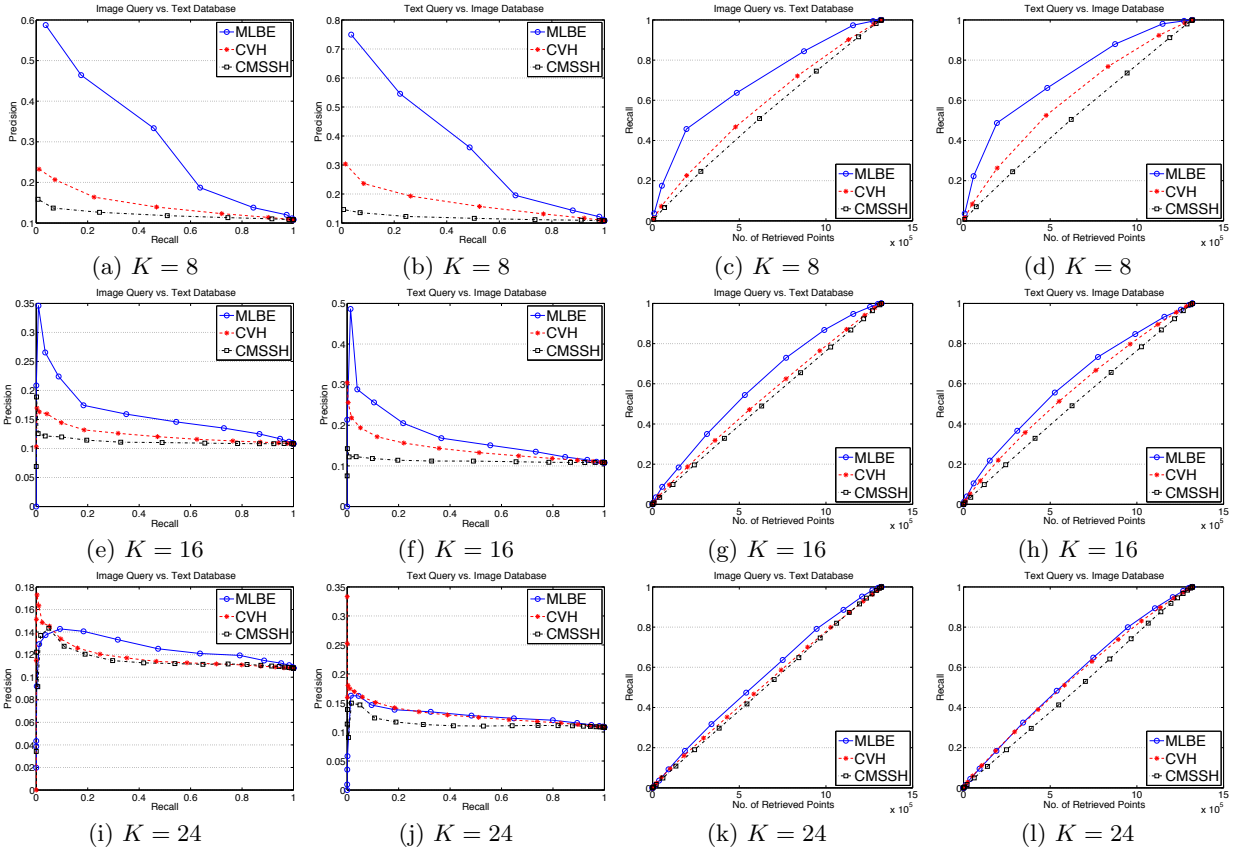
Task	Method	Code Length		
		$K = 8$	$K = 16$	$K = 24$
Image Query vs. Text Database	MLBE	**0.3810**	**0.2561**	**0.1915**
	CVH	0.2592	0.2190	0.1767
	CMSSH	0.2438	0.2014	0.1757
Text Query vs. Image Database	MLBE	**0.4955**	**0.3209**	0.2143
	CVH	0.3474	0.3094	**0.2576**
	CMSSH	0.2044	0.2286	0.2256

We can see that MLBE significantly outperforms CVH and CMSSH when the code length is small. As the code

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

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

[6]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.



(a) $K = 8$ (b) $K = 8$ (c) $K = 8$ (d) $K = 8$

(e) $K = 16$ (f) $K = 16$ (g) $K = 16$ (h) $K = 16$

(i) $K = 24$ (j) $K = 24$ (k) $K = 24$ (l) $K = 24$

Figure 3: Precision-recall curves and recall curves on Wiki

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 [38, 23]. A possible reason is that the learned models will be farther from the optimal solutions when the code length gets larger.

4.4 Results on Flickr Data Set

The Flickr data set consists of 186,577 image-tag pairs, which are pruned from the NUS dataset [8] by keeping the pairs that belong to one of the 10 largest 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 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 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 4. We note that MLBE has the best overall performance.

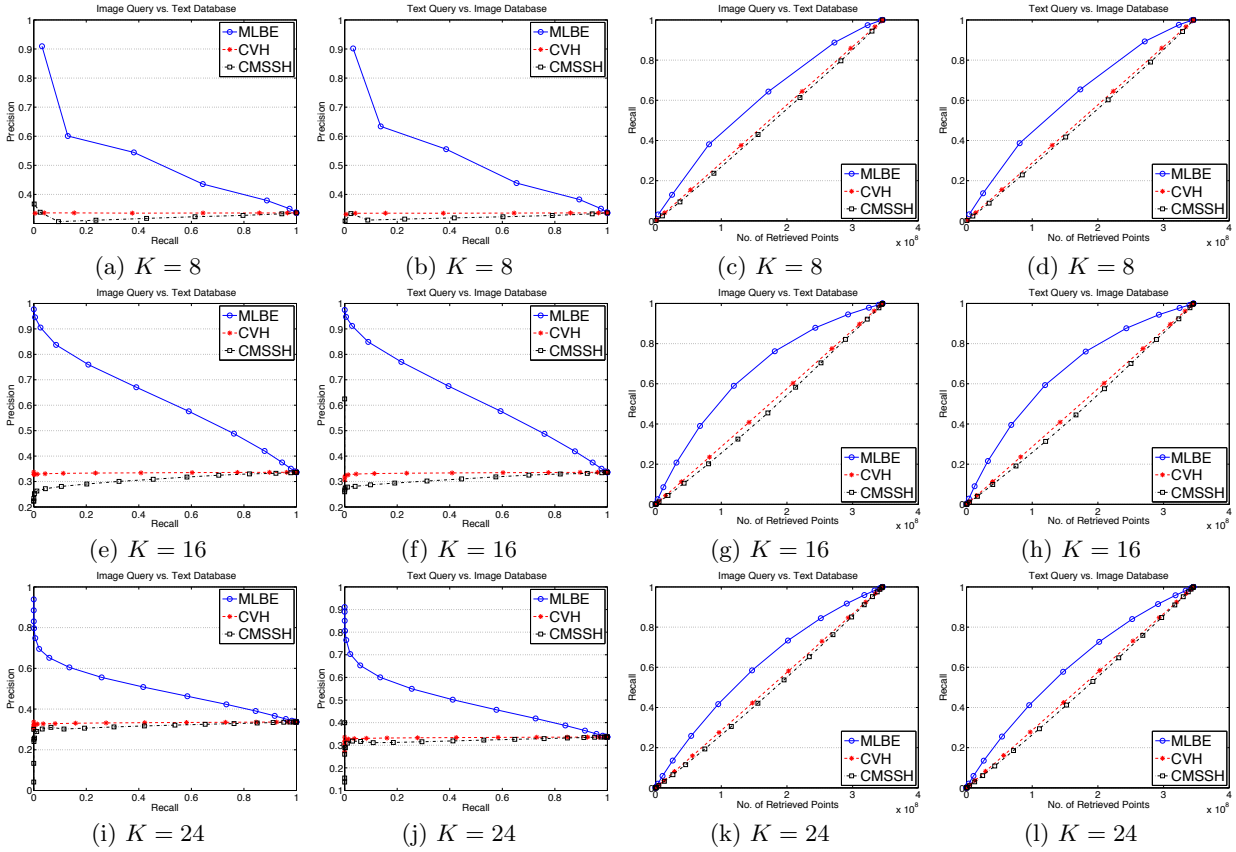
5. CONCLUSIONS

In this paper, we have presented a novel probabilistic model for multimodal hash function learning. As a latent factor model, the model 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. 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. Besides, we also plan to apply MLBE to other tasks such as multimodal medical image registration.

6. ACKNOWLEDGMENTS

This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.



(a) $K = 8$ (b) $K = 8$ (c) $K = 8$ (d) $K = 8$

(e) $K = 16$ (f) $K = 16$ (g) $K = 16$ (h) $K = 16$

(i) $K = 24$ (j) $K = 24$ (k) $K = 24$ (l) $K = 24$

Figure 4: Precision-recall curves and recall curves on Flickr

7. REFERENCES

[1] A. Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

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

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, 1998.

[6] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, 2010.

[7] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[8] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. NUS-WIDE: A real-world web image database from National University of Singapore. In *CIVR*, 2009.

[9] A. Dasgupta, R. Kumar, and T. Sarlos. Fast locality-sensitive hashing. In *KDD*, 2011.

[10] K. Eshghi and S. Rajaram. Locality sensitive hash functions based on concomitant rank order statistics. In *KDD*, 2008.

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

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[13] T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. In *NIPS 18*, 2005.

[14] J. He, W. Liu, and S.-F. Chang. Scalable similarity search with optimized kernel hashing. In *KDD*, 2010.

[15] K. A. Heller and Z. Ghahramani. A nonparametric Bayesian approach to modeling overlapping clusters. In *AISTATS*, 2007.

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

[17] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998.

[18] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS 22*, 2009.

[19] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[20] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, 2011.

[21] D. Lee, M. Hofmann, F. Steinke, Y. Altun, N. D. Cahill, and B. Schölkopf. Learning similarity measure

for multi-modal 3D image registration. In *CVPR*, 2009.

[22] R.-S. Lin, D. A. Ross, and J. Yagnik. SPEC hashing: Similarity preserving algorithm for entropy-based coding. In *CVPR*, 2010.

[23] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.

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

[25] E. Meeds, Z. Ghahramani, R. Neal, and S. T. Roweis. Modeling dyadic data with binary latent factors. In *NIPS 19*, 2006.

[26] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *CVPR*, 2010.

[27] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[28] N. Quadrianto and C. H. Lampert. Learning multi-view neighborhood preserving projections. In *ICML*, 2011.

[29] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS 22*, 2009.

[30] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. R. Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM MM*, 2010.

[31] R. Salakhutdinov and G. E. Hinton. Semantic hashing. In *SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[32] R. E. Schapire. A brief introduction to Boosting. In *IJCAI*, 1999.

[33] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.

[34] G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, March 2006.

[35] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, 2003.

[36] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[37] F. Ture, T. Elsayed, and J. Lin. No free lunch: Brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity. In *SIGIR*, 2011.

[38] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[39] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *ICML*, 2010.

[40] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS 21*, 2008.

[41] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *NIPS 15*, 2002.

[42] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu. Complementary hashing for approximate nearest neighbor search. In *ICCV*, 2011.

[43] D. Zhang, F. Wang, and L. Si. Composite hashing with multiple information sources. In *SIGIR*, 2011.

[44] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR*, 2010.

[45] Y. Zhen and D.-Y. Yeung. Active hashing and its application to image and text retrieval. *Data Mining and Knowledge Discovery*, To appear.

APPENDIX

A. PROOF OF THEOREMS

A.1 Proof of Theorem 3.1

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

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

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:

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

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.

A.2 Proof of Theorem 3.3

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

$$\begin{aligned}
&-\log p(\mathbf{W}^x \mid \mathbf{S}^x, \mathbf{U}, \theta_x, \phi_x) \qquad (7) \\
=&-\log P(\mathbf{W}^x \mid \phi_x) - \log P(\mathbf{S}^x \mid \mathbf{U}, \mathbf{W}^x, \theta_x) + \tilde{C} \\
=&\frac{\phi_x}{2} \sum_{k=1}^{K} \sum_{d=k}^{K} (W_{kd}^x)^2 + \frac{\theta_x}{2} \sum_{i=1}^{I} \sum_{i'=i}^{I} \left(S_{ii'}^x - \mathbf{u}_i^T \mathbf{W}^x \mathbf{u}_{i'} \right)^2 + \tilde{C} \\
=&\frac{\phi_x}{2} \mathbf{w}^{xT} \mathbf{M}_1 \mathbf{w}^x + \frac{\theta_x}{2} \left(\mathbf{s}^x - \mathbf{A}\mathbf{w}^x \right)^T \mathbf{M}_2 \left(\mathbf{s}^x - \mathbf{A}\mathbf{w}^x \right) + \tilde{C} \\
=&\frac{1}{2} \mathbf{w}^{xT} \left(\theta_x \mathbf{A}^T \mathbf{M}_2 \mathbf{A} + \phi_x \mathbf{M}_1 \right) \mathbf{w}^x - \theta_x \mathbf{s}^{xT} \mathbf{M}_2 \mathbf{A} \mathbf{w}^x + \tilde{C},
\end{aligned}$$

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

Setting the derivative of Equation (7) 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.