Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Probabilistic hypergraph based hash codes for social image search*

Yi XIE†, Hui-min YU†, Roland HU

(*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*)

†E-mail: yixie@zju.edu.cn; yhm2005@zju.edu.cn

Received Sept. 26, 2013; Revision accepted Mar. 30, 2014; Crosschecked June 16, 2014

Abstract: With the rapid development of the Internet, recent years have seen the explosive growth of social media. This brings great challenges in performing efficient and accurate image retrieval on a large scale. Recent work shows that using hashing methods to embed high-dimensional image features and tag information into Hamming space provides a powerful way to index large collections of social images. By learning hash codes through a spectral graph partitioning algorithm, spectral hashing (SH) has shown promising performance among various hashing approaches. However, it is incomplete to model the relations among images only by pairwise simple graphs which ignore the relationship in a higher order. In this paper, we utilize a probabilistic hypergraph model to learn hash codes for social image retrieval. A probabilistic hypergraph model offers a higher order representation among social images by connecting more than two images in one hyperedge. Unlike a normal hypergraph model, a probabilistic hypergraph model considers not only the grouping information, but also the similarities between vertices in hyperedges. Experiments on Flickr image datasets verify the performance of our proposed approach.

Key words: Hypergraph Laplacian, Probabilistic hypergraph, Hash codes, Image search
doi:10.1631/jzus.C1300268 **Document code:** A **CLC number:** TP391

1 Introduction

In the last decade, images and video have been widely stored and shared on the Internet. Many photo sharing websites, such as Flickr and Picasa, have numerous images uploaded every day. For example, Flickr has already stored over 6 billion images. Billions of images pose a challenge for conventional image search techniques. Since linear complexity is not scalable in practice, it is clearly infeasible to exhaustively compare the query with every image in the huge database. Furthermore, the visual features of a social image usually have hundreds or thousands of dimensions, which makes most large-scale image search methods suffer from the curse of dimensionality. Apart from the image search problem, storage of such massive original data is also a big problem. Consequently, exploring new representations and approaches to search large-scale datasets has become an urgent research issue.

Efficient algorithms to address this problem have drawn much attention over the last few years. Although many tree-based methods have been proposed (Arya *et al.*, 1998; Silpa-Anan and Hartley, 2008), hashing-based methods have attracted more attention as they greatly reduce memory for memory constrained applications, such as large-scale image retrieval. Hashing-based methods aim at projecting high-dimensional image features into compact binary codes so that close hash codes will be generated for similar images. With the hash codes, the Hamming distance can be efficiently computed by using logical XOR operation in the Hamming space. A similarity search can then be accomplished rapidly by finding the images that have codes within a small Hamming distance of the hash code with the query image.

* Project supported by the National Basic Research Program (973) of China (No. 2012CB316400)

In recent years, many hashing technologies have been proposed. These algorithms can be grouped into two categories: (1) supervised algorithms, e.g., boosting similarity sensitive coding (boosting SSC) (Shakhnarovich *et al.*, 2003) and the restricted Boltzmann machine (RBM) (Hinton and Salakhutdinov, 2006); (2) unsupervised algorithms, e.g., locality sensitive hashing (LSH) (Andoni and Indyk, 2006), spectral hashing (SH) (Weiss *et al.*, 2008), and hypergraph spectral hashing (HSH) (Zhuang *et al.*, 2011). These hashing-based algorithms embed high-dimensional image feature vectors into a low-dimensional Hamming space, while preserving as much semantic similarity as possible. Unsupervised algorithms use unlabeled image data to generate binary hash codes, while supervised algorithms generate hash codes considering the label information of some data points in the dataset.

However, when introducing these hashing-based methods to large-scale social image search applications, we should solve two problems: (1) Most current hashing approaches consider only pairwise information when representing the relationships between different social images, while the connection between real social media appears to be more complex. Therefore, it will be helpful to take account of the relationship not only between two images, but also among three or more images that contain local grouping information. (2) Social images such as photos on Flickr usually have contextual text information, such as titles, surrounding text, and user-generated tags that describe the images. This information of different social images is widely divergent. Some images contain tags while some do not; the numbers of tags are also different in most cases. Therefore, it is hard to measure the similarity of these social images using the same metric space.

In this paper, we aim to overcome the above difficulties by proposing a novel hypergraph-based hashing framework. Different from a conventional hypergraph framework (Zhuang *et al.*, 2011), we further exploit the correlation information among images by using a probabilistic hypergraph (Huang *et al.*, 2010) in our framework, which presents not only whether a vertex v_i belongs to a hyperedge e_j, but also the probability that $v_i \in e_j$. The scheme of our approach is illustrated in Fig. 1. In the proposed method, we first represent each social image by visual content features and semantic textual features, which are generated from the visual content and the tags of the image, respectively. Then a probabilistic hypergraph is constructed, in which the vertices denote the social images in the dataset and the hyperedges are generated by common visual features or tags between social images. Inspired by spectral hashing, we convert our hash code learning problem into a probabilistic hypergraph partition problem, which can be easily solved. Furthermore, we use a supervised learning method to learn hash codes for novel inputs, which addresses the out-of-sample extension problem. We conduct experiments on Flickr image datasets, and the experimental results demonstrate the effectiveness of our algorithm.

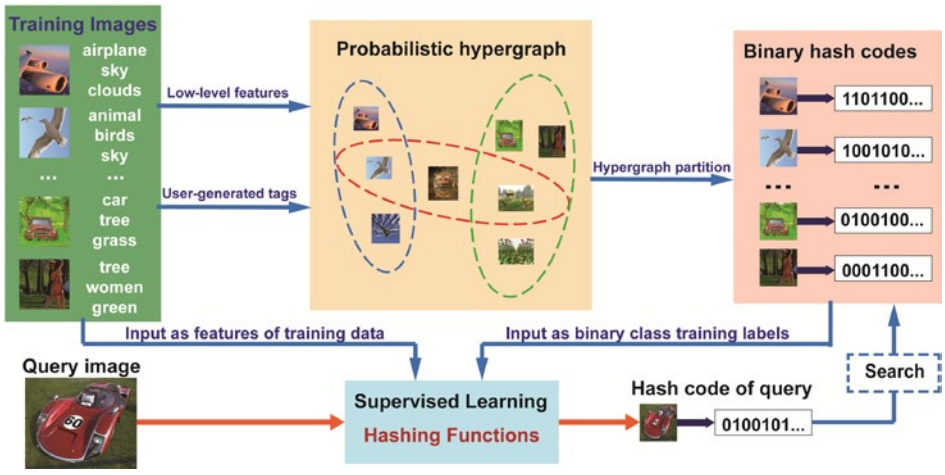


Fig. 1 Schematic of the probabilistic hypergraph based hashing algorithm

2 Related work

2.1 Hashing-based image search

As mentioned before, many hashing-based algorithms have been proposed in recent years. LSH (Andoni and Indyk, 2006) is one of the most popular methods. The basis of LSH is to design binary code so that items which have small Euclidean distances will be mapped to similar codewords. LSH generates every bit in the code by a random linear projection followed by a random threshold. The Euclidean distance can be asymptotically approached by the Hamming distance in this way. However, since random projection is data-independent, this method may lead to quite inefficient codes in practice. Several recently proposed methods pursue machine learning approaches rather than random projections to calculate data-aware hashing codes. Hinton and Salakhutdinov (2006) and Salakhutdinov and Hinton (2007) showed that multiple stacked RBMs can learn much more compact binary codes. In the greedy pre-training stage, they used contrastive divergence to achieve the convergence of the parameters layer by layer, and the derived result of each layer was treated as input to train the next layer. In the fine-tuning stage, back propagation using an objective function such as neighborhood component analysis (NCA) to refine the weights of the trained network was used. RBM-hashing showed much better performance than LSH in Torralba *et al.* (2008). However, it means that we need to estimate a large number of weights, which would make the training procedure time-consuming and demand a big size of training data. A much simpler algorithm called boosting SSC was proposed in Shakhnarovich *et al.* (2003). They first used adaBoost to classify pairwise input data as similar or non-similar, and then took each weak learner as a decision stump. The output of all weak learners on a data point was its binary hashing code. In Weiss *et al.* (2008), a promising method called SH was proposed based on graph partitioning. SH formalizes the requirements for good hash code and equates them to the graph partitioning problem. The analogy to graph partitioning suggests an efficient solution by thresholding eigenvectors of the Laplacian matrix of the similarity graph to binary codes. SH demonstrates significant improvements by comparison with LSH. In Li *et al.* (2013), an approach of spectral hashing with a semantically consistent graph (SHS) was proposed to improve the performance of SH. As an extension of SH, HSH was first proposed by Zhuang *et al.* (2011) which used a unified hypergraph to replace the simple graph in SH. HSH makes a novel improvement to SH. However, a unified hypergraph considers only the grouping information between data points while ignoring the fact that the relationship between vertices and hyperedges may be different.

2.2 Hypergraph learning

Since the hypergraph is effective for higher-order semantic relationship modeling, it has already been adopted in many information retrieval and data mining tasks (Zhou *et al.*, 2004; 2006; Xia and Hancock, 2008; Huang *et al.*, 2010; Liu *et al.*, 2011; Yu *et al.*, 2012). In Gao *et al.* (2013), a general hypergraph framework was proposed and the authors generalized the methodology of spectral clustering which originally operated on undirected graphs to hypergraphs. They further developed algorithms for hypergraph embedding and transductive classification based on a spectral hypergraph clustering algorithm. In Gao *et al.* (2013), a joint method which integrates a visual feature and tags of images in a unified hypergraph was proposed for image retrieval. It performs better than separated methods which use only visual content or tags. In Zass and Shashua (2008), the probabilistic hypergraph was adopted in an image match by solving the convex optimization problem. Huang *et al.* (2010) extended the probabilistic hypergraph model for image retrieval via hypergraph ranking; its effectiveness was demonstrated by comparison with other methods.

The above studies have verified the effectiveness of the hypergraph model in building a higher-order relationship among data. In our proposed method, we use the probabilistic hypergraph model to build the relationship among social images so that both visual content features like bag-of-visual-words (BoW) (Jiang and Ngo, 2008) and contextual information like user-generated tags can be taken into consideration. This provides more useful higher-order information for the mapping step.

3 Introduction to hypergraph

For better understanding of our method, both conventional and probabilistic hypergraphs are introduced in this section. In the simple graph model, samples like social images are represented by vertices, and an edge can connect only two vertices. Using the affinity relationship of the simple graph, many learning methods can be conducted on the graph (Zhou *et al.*, 2004; He *et al.*, 2004; 2006). However, simple graphs cannot represent high-order information. Different from the simple graph, a hyperedge in a hypergraph can connect more than two vertices, which is helpful in analyzing high-order relationship. For clarity, Table 1 lists some important notations used in this paper.

Table 1 Notations and descriptions

Notation	Description
$G=(V, E, w)$	G represent a hypergraph, where V, E, and w indicate the set of vertices, the set of hyperedges, and the weights of edges, respectively
$\boldsymbol{D}_\mathrm{v}$	Diagonal matrix of the vertex degrees
$\boldsymbol{D}_\mathrm{e}$	Diagonal matrix of the edge degrees
$\boldsymbol{H}_\mathrm{p}$	Incidence matrix for the probabilistic hypergraph
$\boldsymbol{W}$	Diagonal weight matrix with its (i, i)-th element being the edge-weight of the ith hyperedge
$\boldsymbol{\Delta}$	Laplacian matrix of the constructed hypergraph
$\delta(e)$	Degree of hyperedge e
$d(v)$	Degree of vertex v
$w(e)$	Weight of hyperedge e
n	Number of images in hypergraph learning
k	Number of bits in hashing code
$\boldsymbol{F}$	An $n\times k$ matrix representing the k-bit binary codes of all n images

3.1 Conventional hypergraph

From a vertex set V, a hyperedge set E, and the weights of the edges w, we can construct a hypergraph $G=(V, E, w)$. Here each edge e is assigned a weight $w(e)$. The hypergraph can be denoted by a $|V|\times|E|$ incidence matrix $\boldsymbol{H}$ with entries:

$$h(v_i, e_j)=\begin{cases}1, & \text{if } v_i\in e_j,\\ 0, & \text{otherwise.}\end{cases} \tag{1}$$

The conventional hypergraph model defined in Eq. (1) assigns a vertex v_i belonging to a hyperedge e_j with a binary judgment ($h(v_i, e_j)$ equals 1 or 0). This hypergraph model equally treats all the vertices in a same hyperedge and ignores the relative affinity between vertices. This may lead to the loss of some useful information, which may impact the accuracy of the hypergraph based applications.

3.2 Probabilistic hypergraph

Similar to Huang *et al.* (2010), we use a probabilistic hypergraph model to break the constraint of a conventional hypergraph. Assume that each vertex has an affinity coefficient with each hyperedge and the affinity coefficient $A(i, j)\in[0, 1]$. In our application, this assumption means that each social image may have a different affinity for a visual feature or a tag. A vertex having a greater affinity coefficient with a hyperedge means that the vertex is more likely to belong to the hyperedge, and vice versa. A probabilistic hypergraph can be represented by an incidence matrix $\boldsymbol{H}_\mathrm{p}$ defined as

$$h_\mathrm{p}(v_i, e_j)=\begin{cases}A(i, j), & \text{if } v_i\in e_j,\\ 0, & \text{otherwise.}\end{cases} \tag{2}$$

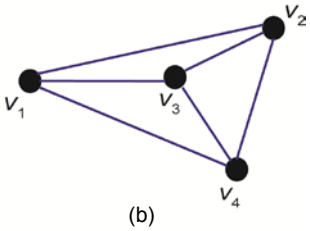
As we can see, both the local grouping information and the probability that a hyperedge contains a vertex are taken into consideration in the probabilistic hypergraph model. In this model, the correlation information between vertices and edges is represented more accurately. The definition in Eq. (1) can be taken as the special case of Eq. (2). Fig. 2 provides an example to explain the difference between a simple graph and a probabilistic hypergraph.

3.3 Introduction to hypergraph learning

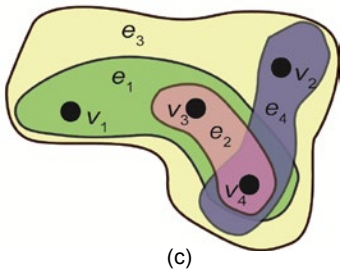
We can perform different machine learning tasks in the hypergraph model, e.g., embedding, clustering, classification, and ranking. Here we introduce the basic hypergraph partition algorithm which is used in our proposed method to help us better understand the hypergraph model. This algorithm can be conducted on both a conventional hypergraph and a probabilistic

	e_1	e_2	e_3	e_4
v_1	0.5	0	0.8	0
v_2	0	0	0.5	0.5
v_3	0.8	0.7	0.7	0
v_4	1	0.6	0.9	0.8

(a)



(b)



(c)

Fig. 2 The difference between a simple graph and a probabilistic hypergraph

(a) Incidence matrix $\boldsymbol{H}_{\mathrm{p}}$; (b) Simple graph built from $\boldsymbol{H}_{\mathrm{p}}$; (c) Probabilistic hypergraph built from $\boldsymbol{H}_{\mathrm{p}}$

hypergraph. For each vertex in the hypergraph $v\in V$, its degree can be defined by

$$d(v)=\sum_{e\in E} w(e)h(v,e), \tag{3}$$

and the edge degree of each hyperedge $e\in E$ is defined by

$$\delta(e)=\sum_{v\in V} h(v,e). \tag{4}$$

Let $\boldsymbol{D}_{\mathrm{v}}$ and $\boldsymbol{D}_{\mathrm{e}}$ denote the diagonal matrices of vertex degrees and edge degrees, respectively, and $\boldsymbol{W}$ the diagonal matrix containing the hyperedge weights. As in Zhou *et al.* (2006), a normalized cost function $\Omega(\boldsymbol{f})$ is defined for a binary partition problem in the hypergraph:

$$\Omega(\boldsymbol{f})=\frac{1}{2}\sum_{e\in E}\sum_{u,v\in e}\frac{w(e)h(u,e)h(v,e)}{\delta(e)}\left(\frac{\boldsymbol{f}(u)}{\sqrt{d(u)}}-\frac{\boldsymbol{f}(v)}{\sqrt{d(v)}}\right)^2, \tag{5}$$

where $\boldsymbol{f}$ is the label vector to be learned. Vertices having more hyperedges in common are more likely to be in the same class; minimizing the cost function $\Omega(\boldsymbol{f})$ can guarantee these vertices obtain similar labels.

Let $\boldsymbol{\Theta}=\boldsymbol{D}_{\mathrm{v}}^{-1/2}\boldsymbol{H}\boldsymbol{W}\boldsymbol{D}_{\mathrm{e}}^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{D}_{\mathrm{v}}^{-1/2}$ and define $\boldsymbol{\Delta}=\boldsymbol{I}-\boldsymbol{\Theta}$, where $\boldsymbol{I}$ denotes the identity matrix. The normalized cost function can be rewritten as

$$\begin{aligned}\Omega(\boldsymbol{f})&=\sum_{e\in E}\sum_{u,v\in e}\frac{w(e)h(u,e)h(v,e)}{\delta(e)}\left(\frac{\boldsymbol{f}^2(u)}{d(u)}-\frac{\boldsymbol{f}(u)\boldsymbol{f}(v)}{\sqrt{d(u)d(v)}}\right)\\&=\sum_{u\in V}\boldsymbol{f}^2(u)\sum_{e\in E}\frac{w(e)h(u,e)}{d(u)}\sum_{v\in V}\frac{h(v,e)}{\delta(e)}\\&\quad-\sum_{e\in E}\sum_{u,v\in e}\frac{\boldsymbol{f}(u)h(v,e)w(e)h(v,e)\boldsymbol{f}(v)}{\sqrt{d(u)d(v)}\delta(e)}\\&=\boldsymbol{f}^{\mathrm{T}}\boldsymbol{\Delta}\boldsymbol{f}.\end{aligned} \tag{6}$$

Here the second step in the derivation comes from $\sum_{e\in E}\frac{w(e)h(u,e)}{d(u)}=1$ and $\sum_{v\in V}\frac{h(v,e)}{\delta(e)}=1$. The matrix $\boldsymbol{\Delta}$ is called the hypergraph Laplacian, and we can note that $\boldsymbol{\Delta}$ is a positive semi-definite matrix. From standard results in linear algebra, the solution $\boldsymbol{f}$ to this optimization problem is the eigenvector corresponding to the smallest nonzero eigenvalue of $\boldsymbol{\Delta}$. According to Zhou *et al.* (2006), it is straightforward to extend this approach to k-way partitioning.

4 The proposed algorithm

In this section we present the proposed probabilistic hypergraph based hashing approach. We introduce the process of probabilistic hypergraph construction from social images, and then present how to obtain binary hash codes from the constructed probabilistic hypergraph. Finally, we take the out-of-sample problem into consideration.

4.1 Probabilistic hypergraph construction

In probabilistic hypergraph construction, each social image is regarded as a vertex in the probabilis-

tic hypergraph $G=(V, E, w)$. For a dataset of n social images, the constructed hypergraph contains n vertices. We use two types of features, i.e., the low-level visual content feature and high-level semantic textual information, to create the hyperedges of the generated hypergraph.

For the visual information of social images, we adopt BoW representation (Yang *et al.*, 2007; Jiang and Ngo, 2008) for image description. To generate BoW representation, local invariant features are extracted from each social image using a difference of Gaussian (DoG) detector and a scale invariant feature transform (SIFT) descriptor (Lowe, 2004). We then compute a codebook by clustering all the SIFT descriptors in their feature space into s clusters, and each detected SIFT feature is encoded into the BoW codebook by the K-means approach. After these processes, the visual content of every social image can be represented by a vector $\boldsymbol{y}_i^{\text{bow}}$, where $y_i^{\text{bow}}(l)=m$ denotes that this image has m SIFT descriptors belonging to the lth visual code of the codebook.

For textual information we consider only the top 1000 most frequently user-generated tags in our social image dataset. From all the unique tags we first calculate their frequency of occurrence, and then select the top 1000 frequent tags to build a codebook for tag information. Each social image is encoded into the codebook by the index of tags that it contains. We use $\boldsymbol{y}_i^{\text{tag}}$ to represent the textual content of each image where $y_i^{\text{tag}}(l)=1$ means that the social image v_i contains the lth tag. Fig. 3 shows the procedures of generating the visual codebook and tag codebook.

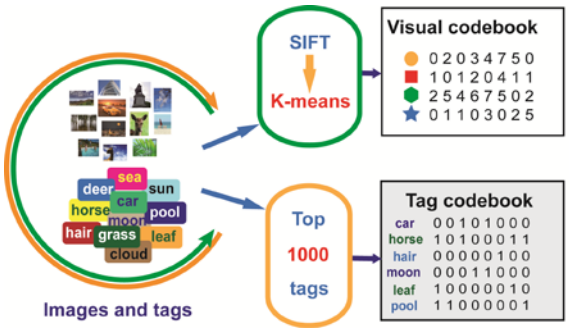


Fig. 3 Illustration of the visual-codebook and tag-codebook generating procedures

With the two vectors $\boldsymbol{y}^{\text{bow}}$ and $\boldsymbol{y}^{\text{tag}}$ for each social image, we can easily construct the hyperedges of our probabilistic hypergraph. In our algorithm each visual codeword in $\boldsymbol{y}^{\text{bow}}$ generates a hyperedge by linking the images containing the same word in the BoW feature, and we assign an affinity coefficient $A(i, j)$ to each image belonging to a content-based hyperedge e_j. The $A(i, j)$ is computed as follows:

$$A(i,j)=\begin{cases}\exp\left(-\dfrac{1}{y_i^{\text{bow}}(j)}\right), & \text{if } v_i\in e_j,\\ 0, & \text{otherwise.}\end{cases} \tag{7}$$

We use each codeword of the tag to construct a hyperedge analogously. For each image connected by hyperedge e_j, the $A(i, j)$ is assigned 0.8 if the social image v_i contains the same tag which generates the hyperedge, and if the title or the label of this social image includes this frequent tag, the $A(i, j)$ is assigned 1 to show a high affinity.

Furthermore, we assign a weight parameter $w(e_j)$ to each hyperedge $e_j\in E$. The weight of each hyperedge is calculated as follows:

$$w(e_j)=\frac{\sum_{v_i\in e_j}A(i,j)}{\sum_{v_i\in V}A(i,j)}. \tag{8}$$

However, a tag often provides more information than one dimension of BoW representation, but the tag codebook appears to be sparser in most cases. This means that using Eq. (8) to calculate all the hyperedge weights is unfair for a tag-based hyperedge. We set a parameter $\lambda\in[0, 1]$ to balance this effect. The weight parameters can be updated by

$$w'(e_j)=\begin{cases}\lambda\cdot w(e_j), & \text{if } e_j \text{ is content-based,}\\ (1-\lambda)\cdot w(e_j), & \text{if } e_j \text{ is tag-based.}\end{cases} \tag{9}$$

4.2 Spectral hashing on the probabilistic hypergraph

Semantic hashing seeks a mapping function which can map data points into binary codes. In the constructed hypergraph model $G(V, E, w)$, each vertex $v\in V$ corresponds to a data point. We want to find a

good mapping function Φ between vertex v and k-dimensional binary codewords f, which can be defined as

$$\Phi: v \to f \in \{-1,1\}^k. \tag{10}$$

As mentioned in Weiss *et al.* (2008), a good mapping function should have the following qualities:

1. Compactness: requiring only a small number of bits to code the full dataset.

2. Consistency: preserving the similarity by mapping similar data points to similar binary codewords.

3. Extendibility: easy handling of the out-of-sample inputs.

To preserve the vertex similarity in Hamming space, we expect to map vertices sharing more common hyperedges with more similar codewords. To meet this property, we seek a hashing function that minimizes the average Hamming distance between similar vertices in the same edge. For coding efficiency, we demand each bit of the codewords has a 50% chance of being 1 or −1, and different bits are independent of each other.

According to Weiss *et al.* (2008), the k-bit spectral hashing problem can be regarded as finding k independent balanced partitions through the lowest cut cost in a simple graph. Similarly, our problem can be solved by finding the lowest normalized cost function $\Omega(\boldsymbol{f})$ in the k-ary balanced probabilistic hypergraph partition problem described in Zhou *et al.* (2006). This also quite adequately meets the requirements above.

Let $\boldsymbol{F}=[\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_k]$ denote the binary codes of all n vertices. $\boldsymbol{F}$ is an $n\times k$ matrix, each row corresponding to k-bit hashing code of a vertex $v\in V$, and the ith column $\boldsymbol{f}_i$ is an n-dimensional vector containing the ith bit of all n vertices. As in Zhou *et al*. (2006), from the normalized cost functions (5) and (6), we can obtain the optimal objective function for our hashing algorithm:

$$\begin{aligned} &\min \sum_{i=1}^{k} \boldsymbol{f}_i^{\mathrm{T}} \boldsymbol{\Delta} \boldsymbol{f}_i = \mathrm{tr}(\boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Delta} \boldsymbol{F}) \\ &\text{s.t. } F(i, j) \in \{-1, 1\},\ \boldsymbol{F}^{\mathrm{T}} \boldsymbol{1} = \boldsymbol{0},\ \boldsymbol{F}^{\mathrm{T}} \boldsymbol{F} = \boldsymbol{I}. \end{aligned} \tag{11}$$

Here $\boldsymbol{\Delta}$ is the hypergraph Laplacian of the constructed probabilistic hypergraph. According to Weiss *et al.* (2008), this is a non-deterministic polynomial (NP) hard problem, but by removing the constraint $F(i, j)\in \{-1, 1\}$, we can formulate the problem to the hypergraph partition problem in Zhou *et al.* (2006), whose solution is the k eigenvectors of $\boldsymbol{\Delta}$ with minimal eigenvalues (after excluding the eigenvector with trivial eigenvalue 0). Then we can obtain binary codes by thresholding these eigenvectors to 1 and −1.

4.3 Out-of-sample extension

The above step generates only the binary codes for the social images in the hypergraph. However, for the novel query images that do not belong to our training dataset, it is hardly realistic to generate the binary codes for them by updating the hypergraph. This problem, called out-of-sample extension, is often resolved by the Nyström method (Bengio *et al.*, 2004) and SH method (Weiss *et al.*, 2008). Nevertheless, these methods are not suitable for our social image search problem since they are either computationally expensive or rely on a very restrictive assumption of data distribution. In this work we adopt the strategy used in Zhang *et al.* (2010) and Zhuang *et al.* (2011), which considers the out-of-sample problem as a supervised learning process.

We use the generated binary codes in $\boldsymbol{F}$ to train k binary classifiers. The ith column $\boldsymbol{f}_i$ is regarded as a label vector of the ith binary classifier $\Phi^i(\boldsymbol{x})$. Here $\boldsymbol{x}$ is the input vector combined by $\boldsymbol{y}^{\mathrm{bow}}$ and $\boldsymbol{y}^{\mathrm{tag}}$. After the supervised learning by inputting features of the training social images, we obtain k binary classifiers $\Phi^1(\boldsymbol{x})$, $\Phi^2(\boldsymbol{x})$, …, $\Phi^k(\boldsymbol{x})$. We concatenate them to build the final hashing function $\Phi(\boldsymbol{x})$ for the k-bit hashing problem. In our algorithm, the support vector machine (SVM) with non-linear kernels is used as the classifier. The whole procedure of our algorithm is listed in Algorithm 1.

Algorithm 1 Probabilistic hypergraph based hashing

Input: Social images $v_1, v_2, \ldots, v_n$.

Output: Binary codes $\boldsymbol{F}$ and hashing function $\Phi(\boldsymbol{x})$.

1 Generate bag-of-visual-words $\boldsymbol{y}_i^{\mathrm{bow}}$ and tag information vector $\boldsymbol{y}_i^{\mathrm{tag}}$ for each social image v_i in the dataset;

2 Construct probabilistic hypergh $G=(V, E, w)$;

3 Compute the affinity coefficients for all the vertices in each content-based hyperedge using Eq. (7);

4 Assign affinity coefficients for vertices in tag-based hyperedges to 0.8 or 1;

5 Compute the hyperedge weight by Eqs. (8) and (9);

6 Compute the incidence matrix $\boldsymbol{H}_{\mathrm{p}}$ using Eq. (2) and diagonal matrices $\boldsymbol{D}_{\mathrm{v}}$, $\boldsymbol{D}_{\mathrm{e}}$ using Eqs. (3) and (4);
7 Compute $\boldsymbol{\Delta}=\boldsymbol{I}-\boldsymbol{D}_{\mathrm{v}}^{-1/2}\boldsymbol{H}\boldsymbol{W}\boldsymbol{D}_{\mathrm{e}}^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{D}_{\mathrm{v}}^{-1/2}$;
8 Generate $\boldsymbol{F}$ by computing the k eigenvectors of $\boldsymbol{\Delta}$ with minimal eigenvalues;
9 Threshold $\boldsymbol{F}$ to 1 and −1;
10 **for** $i\leftarrow 1$ to k **do**
11 Use the ith column of $\boldsymbol{F}$ as binary labels to train the SVM classifier $\Phi^{i}(\boldsymbol{x})$;
12 **end for**
13 Concatenate $\Phi^{1}(\boldsymbol{x})$, $\Phi^{2}(\boldsymbol{x})$, …, $\Phi^{k}(\boldsymbol{x})$ to build the hashing function $\Phi(\boldsymbol{x})$.

5 Experiments evaluation

In this section we carry out experiments on a social image dataset NUS-WIDE to evaluate our proposed method by comparing it with several state-of-the-art hash algorithms, namely LSH, SH, SHS, and HSH. We also extend our approach to specific applications, such as object retrieval and scene matching, using object image dataset NUS-WIDE-OBJECT and scene image dataset NUS-WIDE-SCENE.

5.1 Dataset and experimental settings

We use a widely adopted NUS-WIDE dataset (Chua *et al.*, 2009) to conduct our experiments. The NUS-WIDE dataset contains about 270 000 Flickr social images and these images are labeled with 81 semantic concept classes (e.g., animal, cars, and sky). Fig. 4 illustrates several example images from this dataset. NUS-WIDE is a multi-label dataset which means each social image in NUS-WIDE may belong to more than one class. In our experiment, the image labels are used as the ground-truth to evaluate the image search performance. If an image shares at least one common label with the query image, we consider it as a correct search result. NUS-WIDE-OBJECT and NUS-WIDE-SCENE are subsets of the NUS-WIDE dataset, which contains 30 000 object images of 31 object categories and 34 926 scene images of 33 scene categories.

NUS-WIDE provides different image features and tag information for all the images in the dataset. We use the 500-D bag-of-visual-words feature and 1000-D tag vector which corresponds to the most frequently used 1000 tags in the experiment. Thus, the length s of the visual codebook is 500. The parameter λ is assigned to a different value in different cases. In our work, precision is adopted to evaluate the retrieval result, which is the fraction of retrieved images that are relevant to the query.

Fig. 4 Several example images and their associated tags in the NUS-WIDE dataset

5.2 Experimental results and analysis

5.2.1 Experiments on NUS-WIDE

In our work, the proposed probabilistic hypergraph based hashing algorithm is compared with competitive hash methods including LSH (Andoni and Indyk, 2006), SH (Weiss *et al.*, 2008), and SHS (Li *et al.*, 2013), and we also assess the performance of probabilistic hypergraph hashing against HSH (Zhuang *et al.*, 2011). To verify the effectiveness of weight reassignment, both probabilistic hypergraph hashing with equal hyperedge weights (PHH) and probabilistic hypergraph hashing with hyperedge weight updating using Eqs. (8) and (9) (PHH-W) are compared with the above methods. First, we randomly select 30 000 images in the dataset to construct the probabilistic hypergraph. All these images are then mapped to binary hash codes using different hashing algorithms. For PHH-W, the parameter λ is empirically set to 0.1 here. Five thousand images are randomly chosen from all 81 classes in the 269 648 images of NUS-WIDE as query images. Then we rank the other images according to the Hamming distance to the query images.

Fig. 5 shows the nearest neighbor retrieval results of different methods. The code length ranges from 4 to 32 bits and the precisions of the first 50, 100, and 150 searched neighbors are illustrated. Since there are several frequently occurring labels (e.g., person, animal) shared by a large proportion of images, even random selection can achieve a precision over 0.2. So, it can be observed that LSH has relatively low precision results than other algorithms, and

as the number of bits increases, its convergence rate is small. SH works better than LSH since it is a data-aware method. HSH and SHS improve the performance of SH by employing a hypergraph structure and semantically consistent graph. As expected, PHH outperforms these methods in almost all cases, which verifies the effectiveness of the probabilistic hypergraph model. PHH-W further enhances the performance of PHH. It is mainly because we reassign higher hyperedge weights for tag-based hyperedges using λ=0.1, and these hyperedges are often more sensitive to semantic labels. Fig. 6 illustrates the precision curves with different nearest neighbors retrieved using the above methods. The results also show that PHH-W and PHH achieve the best performance consistently. Among all the algorithms, HSH and SHS have relatively good performance, but our method outperforms both of them. The main possible reason is that we take the affinity between vertices and edges into consideration in hypergraph construction. This provides more important information for hash code learning.

From the outstanding performance of PHH-W, we can infer that weight-updating of hyperedges plays an important role in our method. So, we carry out experiments to further test the effect of λ. We set different values to λ and the results are demonstrated in Fig. 7, from which we can see that the performance of our method gradually deteriorates as λ increases from 0.1 to 0.5 (λ=0.5 means equal hyperedge weights). This proves that since the tag-based hyperedges contain more semantic information but have a sparser codebook, they should be reassigned much higher weights to approach the better performance. However, when λ is set to 0.05, the performance deteriorates compared to λ=0.1. This may be caused by ignoring too much visual information.

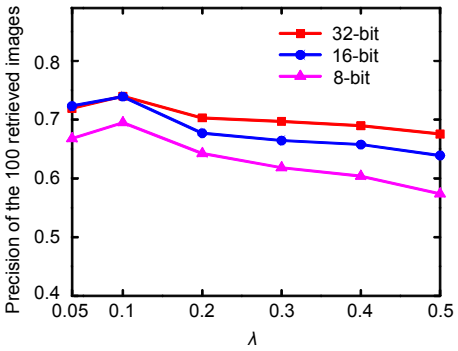


Fig. 7 Performance of our method with respect to different values of λ

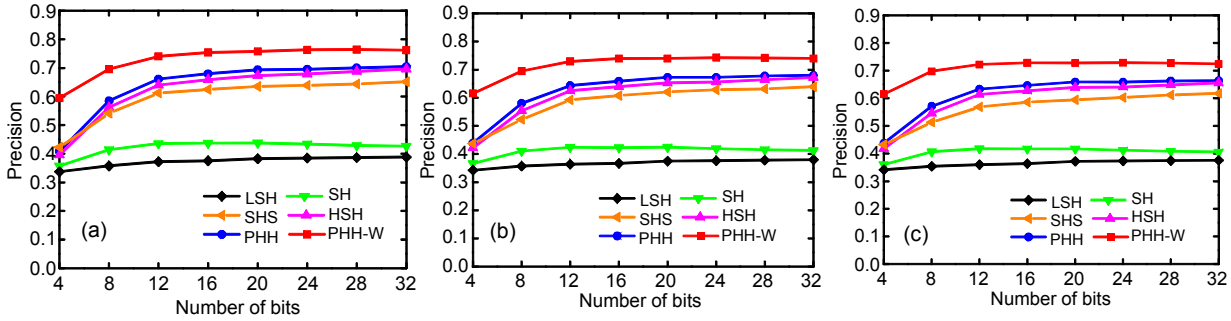


Fig. 5 Precision of the top 50 (a), 100 (b), and 150 (c) neighbors at different numbers of bits

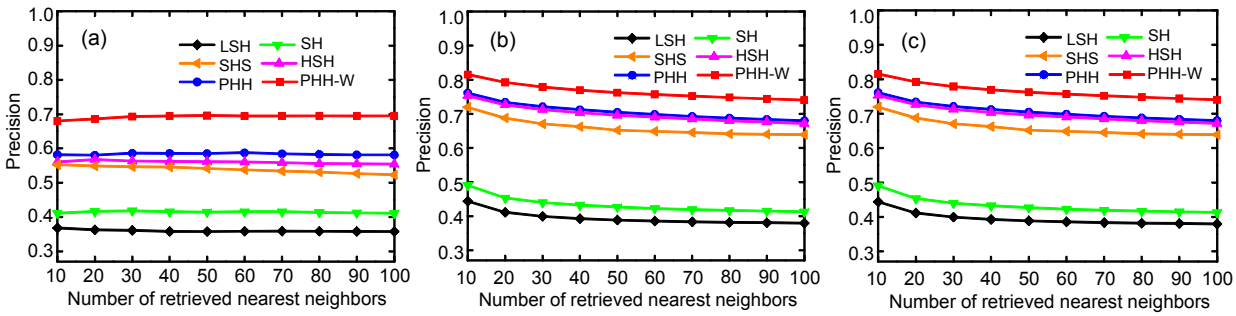


Fig. 6 Precision under different numbers of retrieved nearest neighbors at 8-bit (a), 16-bit (b), and 32-bit (c)

5.2.2 Experiments on specific applications

To further evaluate the performance of our proposed algorithm, we extend it to object image retrieval and scene image match applications using two subsets of NUS-WIDE. For the NUS-WIDE-OBJECT dataset, 17927 images are selected for probabilistic hypergraph learning and 2000 images are randomly chosen from all the 31 object classes as queries. For the NUS-WIDE-SCENE dataset, we perform a similar operation using 17463 images for training hypergraph and 2000 images from 33 scene classes for query. Then we convert all these images into hash codes using different hashing algorithms mentioned before. Tables 2 and 3 demonstrate the average precisions of queries coming from all 31 object classes and 33 scene classes, respectively. The number of searched neighbors in Hamming distance is set to 50 and the parameter λ of PHH-W is assigned 0.1 here.

We can see that PHH-W and PHH have the best and second best performance in most cases for both

Table 2 The average precision of each label class on the NUS-WIDE-OBJECT dataset using different methods

Class	Average precision (%)											
	16-bit						32-bit					
	LSH	SH	SHS	HSH	PHH	PHH-W	LSH	SH	SHS	HSH	PHH	PHH-W
bear	8.16	12.04	22.45	23.98	**28.52**	**57.22**	8.26	10.58	28.74	49.19	**50.34**	**64.58**
birds	11.73	15.04	23.17	22.26	**26.96**	**53.64**	12.12	13.81	32.16	34.62	**36.89**	**58.40**
boats	18.29	21.89	36.23	37.69	**41.46**	**45.26**	19.74	22.38	47.78	48.47	**50.60**	**51.00**
book	3.44	5.68	8.52	**9.22**	**10.40**	6.26	4.08	5.60	10.42	**11.90**	**12.80**	7.68
cars	17.78	25.74	50.98	54.99	**57.17**	**62.11**	19.04	25.73	51.46	54.84	**55.89**	**59.07**
cat	9.03	10.99	27.33	29.35	**35.46**	**78.96**	9.31	10.66	45.61	51.38	**54.50**	**72.37**
computer	4.99	7.38	13.26	10.06	**14.04**	**14.84**	4.78	7.70	26.52	21.96	**34.37**	**29.40**
coral	14.76	19.73	55.87	57.60	**57.75**	**66.13**	16.59	17.13	56.28	**57.54**	56.68	**63.63**
cow	6.28	7.75	**14.21**	9.82	13.11	**24.07**	6.69	7.82	**21.54**	18.88	20.01	**33.88**
dog	9.53	11.62	35.62	37.91	**43.08**	**77.46**	9.69	12.17	48.69	53.39	**54.50**	**72.37**
elk	6.61	7.55	13.87	11.73	**15.01**	**26.62**	6.87	6.95	16.29	15.67	**16.65**	**30.49**
fish	14.40	18.01	48.95	53.63	**55.55**	**68.34**	15.42	15.36	52.85	54.39	**54.85**	**67.34**
flags	4.61	5.12	7.28	7.16	**8.74**	**9.64**	5.33	5.07	9.41	9.54	**9.90**	**12.30**
flowers	22.76	32.58	56.83	57.66	**59.15**	**70.49**	24.40	28.93	57.35	58.79	**59.46**	**68.45**
fox	6.69	8.76	**13.26**	12.92	13.11	**17.39**	7.85	8.45	15.81	15.93	**16.05**	**19.61**
horses	6.57	8.44	10.25	9.09	**12.05**	**63.55**	7.33	8.61	25.28	**27.05**	26.34	**55.50**
leaf	16.90	21.95	30.76	33.33	**35.62**	**43.09**	18.96	20.36	36.29	37.85	**38.41**	**43.42**
plane	18.14	25.40	48.82	47.73	**49.38**	**63.18**	19.16	24.53	**50.80**	49.19	47.64	**60.39**
rocks	17.62	28.27	32.64	**33.30**	**34.09**	31.89	18.29	26.39	40.34	**44.67**	**44.91**	37.29
sand	13.25	16.41	20.54	**21.11**	20.87	**23.79**	14.52	14.98	26.50	**28.83**	28.31	**28.92**
sign	6.55	11.29	14.83	**15.28**	**15.55**	14.14	7.73	10.90	19.89	20.12	**21.00**	**40.21**
statue	5.75	7.07	18.65	19.89	**23.64**	**25.87**	6.59	6.66	27.73	27.14	**30.83**	**52.05**
sun	20.21	25.11	30.01	31.26	**31.41**	**43.14**	23.95	23.90	34.18	**35.03**	34.52	**44.77**
tiger	6.54	11.48	22.38	21.30	**24.21**	**34.92**	7.45	9.70	35.53	**38.70**	**37.90**	32.97
tower	9.44	12.93	27.42	30.50	**32.21**	**41.58**	9.80	12.05	29.44	30.52	**32.78**	**49.11**
toy	8.76	11.56	36.28	37.91	**41.72**	**44.76**	9.06	11.60	43.28	**45.05**	**44.39**	42.09
train	11.60	15.57	50.14	52.06	**59.49**	**71.20**	11.73	15.72	59.03	58.44	**61.57**	**60.90**
tree	17.96	24.80	**32.56**	30.48	31.93	**48.44**	18.94	24.38	38.57	37.11	**38.61**	**48.94**
vehicle	22.50	32.07	43.34	44.16	**46.79**	**47.30**	24.13	32.08	45.62	47.00	**47.90**	**48.23**
whales	5.45	8.86	12.57	13.18	**14.79**	**14.08**	6.67	8.02	15.33	**16.96**	**17.24**	14.26
zebra	4.56	7.44	12.21	**13.00**	**13.19**	7.54	5.43	8.76	17.93	**22.43**	**31.21**	12.50

The best and second best results in each case are marked in bold for clarity

Table 3 The average precision of each label class on the NUS-WIDE-SCENE dataset using different methods

Class	Average precision (%)											
	16-bit						32-bit					
	LSH	SH	SHS	HSH	PHH	PHH-W	LSH	SH	SHS	HSH	PHH	PHH-W
airport	52.55	52.41	56.15	58.57	**59.27**	**61.23**	52.91	54.98	57.29	55.66	**57.43**	**62.95**
beach	66.14	68.23	72.48	73.62	**73.76**	**75.56**	67.49	69.21	73.74	74.06	**74.10**	**75.26**
bridge	53.88	55.50	60.38	60.19	**61.25**	**66.34**	54.60	56.28	62.94	63.46	**64.83**	**68.87**
buildings	54.57	57.06	61.54	63.82	**64.38**	**68.61**	54.95	58.60	66.47	67.07	**67.31**	**69.03**
castle	57.76	59.73	65.77	64.45	**65.95**	**66.93**	57.39	62.75	66.93	66.53	**67.12**	**68.29**
cityscape	54.00	56.81	64.15	**67.26**	67.13	**69.35**	53.98	56.90	67.52	**70.78**	**70.96**	70.45
clouds	68.01	71.00	**74.30**	73.18	73.89	**77.84**	69.60	71.60	**75.51**	74.75	75.20	**77.00**
frost	37.76	41.05	48.73	50.37	**54.10**	**66.44**	36.93	42.22	55.36	58.56	**61.04**	**65.25**
garden	39.34	42.52	47.64	47.94	**48.35**	**59.69**	39.21	44.36	51.28	**54.34**	53.72	**61.12**
glacier	61.29	62.93	68.84	68.67	**69.54**	**73.37**	61.40	63.68	71.32	72.40	**73.68**	**74.27**
grass	57.03	60.22	**63.78**	62.60	63.49	**71.57**	57.76	61.99	67.34	**68.43**	67.97	**72.61**
harbor	68.36	70.91	73.87	**75.28**	**75.62**	74.24	67.41	70.32	75.28	**79.02**	**79.97**	75.40
house	55.44	57.96	62.42	61.19	**62.95**	**68.58**	56.43	59.93	66.07	66.50	**66.94**	**67.82**
lake	68.26	70.52	74.28	74.17	**74.92**	**78.12**	69.83	71.04	75.94	75.54	**76.16**	**77.56**
moon	56.40	57.76	63.35	**64.16**	63.56	**67.92**	58.55	60.02	65.07	64.86	**65.66**	**69.35**
mountain	62.50	65.05	70.44	69.67	**71.27**	**74.92**	63.08	66.06	72.83	72.64	**73.89**	**76.15**
nighttime	51.41	54.40	62.29	64.00	**64.46**	**66.70**	53.33	55.34	63.71	65.77	**67.09**	**67.85**
ocean	68.12	70.67	76.60	76.74	**77.20**	**77.67**	69.57	71.90	77.01	76.88	**77.28**	**77.15**
plants	47.73	50.89	59.48	58.71	**60.25**	**68.81**	48.40	53.16	64.52	64.70	**65.03**	**69.53**
railroad	43.89	44.22	56.76	**59.08**	**61.02**	57.51	45.10	46.02	60.34	**70.29**	**69.88**	61.43
rainbow	64.34	69.09	72.29	**72.89**	**73.59**	71.11	67.50	68.32	73.68	**74.11**	**74.25**	72.14
reflection	64.02	66.14	**71.24**	69.37	70.20	**75.97**	64.76	67.17	**73.59**	71.76	72.84	**75.39**
road	44.92	47.45	52.79	53.82	**54.49**	**62.48**	45.83	49.41	55.68	**59.41**	59.40	**65.08**
sky	67.39	69.66	71.98	72.14	**72.78**	**77.08**	68.73	70.63	73.27	73.94	**74.38**	**76.42**
snow	51.16	54.01	60.79	62.47	**64.55**	**74.08**	51.85	55.01	64.21	66.90	**68.30**	**72.14**
street	26.69	30.52	41.09	42.69	**44.42**	**52.14**	27.17	31.94	46.72	48.31	**48.50**	**52.30**
sunset	68.90	71.70	74.93	74.10	**75.04**	**79.61**	71.00	71.53	75.49	75.24	**75.73**	**78.71**
temple	46.03	48.00	54.28	53.01	**55.02**	**58.77**	46.13	49.29	58.79	**59.32**	**61.38**	58.97
town	45.50	47.34	55.56	56.33	**56.41**	**60.98**	45.33	49.05	59.43	**60.79**	60.64	**60.90**
valley	65.30	68.49	73.23	72.41	**73.58**	**76.51**	65.81	69.74	75.94	75.31	**76.33**	**77.14**
water	62.59	64.86	68.87	69.38	**70.17**	**75.85**	63.52	66.83	70.35	71.57	**72.28**	**75.04**
waterfall	33.37	39.47	47.56	46.14	**51.30**	**64.42**	32.62	45.30	62.23	60.92	**64.67**	**64.96**
window	35.18	37.54	44.83	45.70	**46.43**	**56.91**	34.85	39.95	50.75	51.82	**52.13**	**60.65**

The best and second best results in each case are marked in bold for clarity

object image retrieval and scene image match applications. It demonstrates that our probabilistic hypergraph model and weight-reassignment process are really helpful in generating more effective codes for image search and other applications.

In object image retrieval, PHH-W shows much better performance than other approaches. This is mainly because tag information can represent objects better than visual information, and the weight-updating process of PHH-W enhances the influence of tag-based edges, which in turn promotes the search performance.

In scene image match, the average precisions of different label classes are significantly higher than those of object retrieval. The main reason is that NUS-WIDE-SCENE contains more multi-label

images, which improves the search accuracy. Since scene match requires more visual information than object retrieval, the performance gap between PHH-W and PHH is not as notable as before.

Fig. 8 displays the top six search results of two object query images from NUS-WIDE-OBJECT and two scene query images from NUS-WIDE-SCENE with all six methods at 16-bit, from which we can see that PHH-W and PHH obtain better results by generating less irrelevant images. For example, when considering the second query which has the labels sun and tree, most results returned by PHH-W contain both of the two labels while most results returned by other methods contain only one label or even none of them.

5.3 Analysis of computational complexity

We also analyze the efficiency of our hashing algorithm. The computational cost of our method mainly comes from eigendecomposition and SVM training. For n training images, the time complexities for these two stages are $O(n^3)$ and $O(kdn^2)$ respectively, where k is the length of hashing code and d is the dimensionality of input features. Compared with some state-of-the-art algorithms, our approach has relatively high computational cost in the training process. However, the training stage is a one-time cost. Once the hashing functions are obtained, the generation of the binary hash code is fast. Fig. 9 shows the time cost per image for the training process of our algorithm and SH.

Fig. 8 Top six returned images of example queries with different methods at 16-bit, with irrelevant results being denoted by red boxes (References to color refer to the online version of this figure)

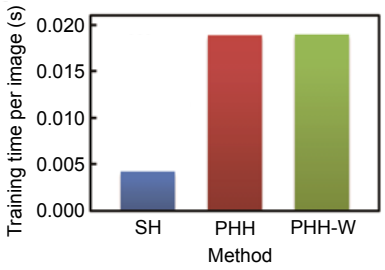


Fig. 9 Time cost comparison for the training stage

Our experiments are run on a workstation with 2.53 GHz CPU and 56 GB memory. The results show that SH is more efficient than PHH and PHH-W in the training stage, and that PHH and PHH-W have similar computational costs. Although the training of our method is slower than that of SH, the performance of our method improves a lot by employing the probabilistic hypergraph model and SVM classifiers. We will try some optimization strategies to further optimize the efficiency of our approach in future work.

6 Conclusions

In this paper, we propose a novel hashing framework for social image search, in which the probabilistic hypergraph is used to represent the relevance between social images. We first extract the visual content feature and tag information from social images and then generate a probabilistic hypergraph based on them. The binary hash codes of social images in the hypergraph are learned by a spectral graph partitioning algorithm on a probabilistic hypergraph. Furthermore, the out-of-sample problem is resolved by different hashing functions obtained in a supervised learning. We also introduce a weight reassignment process to balance the influences of visual information and tag information. The effectiveness of our approach is demonstrated by experimental results on the NUS-WIDE dataset. To further verify the performance of our approach, we extend it to object image retrieval and scene image match applications. Experimental results show that our algorithm outperforms other competitive methods.

In future work we will exploit the adaptive weight learning process to automatically update the weights of hyperedges and extend our algorithm to other social media or cross-media environments.

References

Andoni, A., Indyk, P., 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. 47th Annual IEEE Symp. on Foundations of Computer Science, p.459-468. [doi:10.1109/FOCS.2006.49]

Arya, S., Mount, D.M., Netanyahu, N.S., *et al.*, 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *JACM*, **45**(6):891-923. [doi:10.1145/293347.293348]

Bengio, Y., Delalleau, O., le Roux, N., *et al.*, 2004. Learning eigenfunctions links spectral embedding and kernel PCA. *Neur. Comput.*, **16**(10):2197-2219. [doi:10.1162/0899766041732396]

Chua, T.S., Tang, J., Hong, R., *et al.*, 2009. NUS-WIDE: a real-world web image database from National University of Singapore. Proc. ACM Int. Conf. on Image and Video Retrieval, p.48. [doi:10.1145/1646396.1646452]

Gao, Y., Wang, M., Zha, Z.J., *et al.*, 2013. Visual-textual joint relevance learning for tag-based social image search. *IEEE Trans. Image Process.*, **22**(1):363-376. [doi:10.1109/TIP.2012.2202676]

He, J., Li, M., Zhang, H.J., *et al.*, 2004. Manifold-ranking based image retrieval. Proc. 12th Annual ACM Int. Conf. on Multimedia, p.9-16. [doi:10.1145/1027527.1027531]

He, J., Li, M., Zhang, H.J., *et al.*, 2006. Generalized manifold-ranking-based image retrieval. *IEEE Trans. Image Process.*, **15**(10):3170-3177. [doi:10.1109/TIP.2006.877491]

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. *Science*, **313**(5786):504-507. [doi:10.1126/science.1127647]

Huang, Y., Liu, Q., Zhang, S., *et al.*, 2010. Image retrieval via probabilistic hypergraph ranking. IEEE Conf. on Computer Vision and Pattern Recognition, p.3376-3383. [doi:10.1109/CVPR.2010.5540012]

Jiang, Y.G., Ngo, C.W., 2008. Bag-of-visual-words expansion using visual relatedness for video indexing. Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.769-770. [doi:10.1145/1390334.1390495]

Li, P., Wang, M., Cheng, J., *et al.*, 2013. Spectral hashing with semantically consistent graph for image indexing. *IEEE Trans. Multimedia*, **15**(1):141-152. [doi:10.1109/TMM.2012.2199970]

Liu, H., le Pendu, P., Jin, R., *et al.*, 2011. A hypergraph-based method for discovering semantically associated itemsets. IEEE 11th Int. Conf. on Data Mining, p.398-406. [doi:10.1109/ICDM.2011.12]

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, **60**(2):91-110. [doi:10.1023/B:VISI.0000029664.99615.94]

Salakhutdinov, R., Hinton, G.E., 2007. Learning a nonlinear embedding by preserving class neighborhood structure. Int. Conf. on Artificial Intelligence and Statistics, p.412-419.

Shakhnarovich, G., Viola, P., Darrel, T., 2003. Fast pose

estimation with parameter-sensitive hashing. Proc. 9th IEEE Int. Conf. on Computer Vision, p.750-757. [doi:10.1109/ICCV.2003.1238424]

Silpa-Anan, C., Hartley, R., 2008. Optimised KD-trees for fast image descriptor matching. IEEE Conf. on Computer Vision and Pattern Recognition, p.1-8. [doi:10.1109/CVPR.2008.4587638]

Torralba, A., Fergus, R., Weiss, Y., 2008. Small codes and large image databases for recognition. IEEE Conf. on Computer Vision and Pattern Recognition, p.1-8. [doi:10.1109/CVPR.2008.4587633]

Weiss, Y., Torralba, A., Fergus, R., 2008. Spectral hashing. 21st Advances in NIPS, p.1753-1760.

Xia, S., Hancock, E.R., 2008. Clustering using class specific hyper graphs. Structural, Syntactic, and Statistical Pattern Recognition, p.318-328. [doi:10.1007/978-3-540-89689-0_36]

Yang, J., Jiang, Y.G., Hauptmann, A.G., *et al.*, 2007. Evaluating bag-of-visual-words representations in scene classification. Proc. Int. Workshop on Multimedia Information Retrieval, p.197-206. [doi:10.1145/1290082.1290111]

Yu, J., Tao, D., Wang, M., 2012. Adaptive hypergraph learning and its application in image classification. *IEEE Trans. Image Process.*, **21**(7):3262-3272. [doi:10.1109/TIP.2012.2190083]

Zass, R., Shashua, A., 2008. Probabilistic graph and hypergraph matching. IEEE Conf. on Computer Vision and Pattern Recognition, p.1-8. [doi:10.1109/CVPR.2008.4587500]

Zhang, D., Wang, J., Cai, D., *et al.*, 2010. Self-taught hashing for fast similarity search. Proc. 33rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.18-25. [doi:10.1145/1835449.1835455]

Zhou, D., Bousquet, O., Lal, T.N., *et al.*, 2004. Learning with local and global consistency. 17th Advances in NIPS, p.321-328.

Zhou, D., Huang, J., Schölkopf, B., 2006. Learning with hypergraphs: clustering, classification, and embedding. 19th Advances in NIPS, p.1601-1608.

Zhuang, Y., Liu, Y., Wu, F., *et al.*, 2011. Hypergraph spectral hashing for similarity search of social image. Proc. 19th ACM Int. Conf. on Multimedia, p.1457-1460. [doi:10.1145/2072298.2072039]