Semi-Supervised SimHash for Efficient Document Similarity Search

Qixia Jiang and Maosong Sun
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Sci. and Tech., Tsinghua University, Beijing 100084, China
qixia.jiang@gmail.com, sms@tsinghua.edu.cn

Abstract

Searching documents that are similar to a query document is an important component in modern information retrieval. Some existing hashing methods can be used for efficient document similarity search. However, unsupervised hashing methods cannot incorporate prior knowledge for better hashing. Although some supervised hashing methods can derive effective hash functions from prior knowledge, they are either computationally expensive or poorly discriminative. This paper proposes a novel (semi-)supervised hashing method named Semi-Supervised SimHash (S^3H) for high-dimensional data similarity search. The basic idea of S^3H is to learn the optimal feature weights from prior knowledge to relocate the data such that similar data have similar hash codes. We evaluate our method with several state-of-the-art methods on two large datasets. All the results show that our method gets the best performance.

1 Introduction

Document Similarity Search (DSS) is to find similar documents to a query doc in a text corpus or on the web. It is an important component in modern information retrieval since DSS can improve the traditional search engines and user experience (Wan *et al.*, 2008; Dean *et al.*, 1999). Traditional search engines accept several terms submitted by a user as a query and return a set of docs that are relevant to the query. However, for those users who are not search experts, it is always difficult to accurately specify some query terms to express their search purposes. Unlike short-query based search, DSS queries by a full (long) document, which allows users to directly submit a page or a document to the search engines as the description of their information needs. Meanwhile, the explosion of information has brought great challenges to traditional methods. For example, Inverted List (IL) which is a primary key-term access method would return a very large set of docs for a query document, which leads to the time-consuming post-processing. Therefore, a new effective algorithm is required.

Hashing methods can perform highly efficient but approximate similarity search, and have gained great success in many applications such as Content-Based Image Retrieval (CBIR) (Ke *et al.*, 2004; Kulis *et al.*, 2009b), near-duplicate data detection (Ke *et al.*, 2004; Manku *et al.*, 2007; Costa *et al.*, 2010), etc. Hashing methods project high-dimensional objects to compact binary codes called *fingerprints* and make similar fingerprints for similar objects. The similarity search in the Hamming space[1] is much more efficient than in the original attribute space (Manku *et al.*, 2007).

Recently, several hashing methods have been proposed. Specifically, SimHash (SH) (Charikar M.S., 2002) uses random projections to hash data. Although it works well with long fingerprints, SH has poor discrimination power for short fingerprints. A kernelized variant of SH, called Kernelized Locality Sensitive Hashing (KLSH) (Kulis *et al.*, 2009a), is proposed to handle non-linearly separable data. These methods are unsupervised thus cannot incorporate prior knowledge for better hashing. Moti-

[1]Hamming space is a set of binary strings of length L.

vated by this, some supervised methods are proposed to derive effective hash functions from prior knowledge, i.e., Spectral Hashing (Weiss *et al.*, 2009) and Semi-Supervised Hashing (SSH) (Wang *et al.*, 2010a). Regardless of different objectives, both methods derive hash functions via Principle Component Analysis (PCA) (Jolliffe, 1986). However, PCA is computationally expensive, which limits their usage for high-dimensional data.

This paper proposes a novel (semi-)supervised hashing method, Semi-Supervised SimHash (S^3H), for high-dimensional data similarity search. Unlike SSH that tries to find a sequence of hash functions, S^3H fixes the random projection directions and seeks the optimal feature weights from prior knowledge to relocate the objects such that similar objects have similar fingerprints. This is implemented by maximizing the empirical accuracy on the prior knowledge (labeled data) and the entropy of hash functions (estimated over labeled and unlabeled data). The proposed method avoids using PCA which is computationally expensive especially for high-dimensional data, and leads to an efficient Quasi-Newton based solution. To evaluate our method, we compare with several state-of-the-art hashing methods on two large datasets, i.e., 20 Newsgroups (20K points) and Open Directory Project (ODP) (2.4 million points). All experiments show that S^3H gets the best search performance.

This paper is organized as follows: Section 2 briefly introduces the background and some related works. In Section 3, we describe our proposed Semi-Supervised SimHash (S^3H). Section 4 provides experimental validation on two datasets. The conclusions are given in Section 5.

2 Background and Related Works

Suppose we are given a set of N documents, $\mathcal{X} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^M\}_{i=1}^N$. For a given query doc $\mathbf{q}$, DSS tries to find its nearest neighbors in $\mathcal{X}$ or a subset $\mathcal{X}' \subset \mathcal{X}$ in which distance from the documents to the query doc $\mathbf{q}$ is less than a give threshold. However, such two tasks are computationally infeasible for large-scale data. Thus, it turns to the approximate similarity search problem (Indyk *et al.*, 1998). In this section, we briefly review some related approximate similarity search methods.

2.1 SimHash

SimHash (SH) is first proposed by Charikar (Charikar M.S., 2002). SH uses random projections as hash functions, i.e.,

$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x}) = \begin{cases} +1, & if\ \mathbf{w}^T\mathbf{x} \geq 0 \\ -1, & otherwise \end{cases} \quad (1)$$

where $\mathbf{w} \in \mathbb{R}^M$ is a vector randomly generated. SH specifies the distribution on a family of hash functions $\mathcal{H} = \{h\}$ such that for two objects $\mathbf{x}_i$ and $\mathbf{x}_j$,

$$\Pr_{h\in\mathcal{H}}\{h(\mathbf{x}_i) = h(\mathbf{x}_j)\} = 1 - \frac{\theta(\mathbf{x}_i, \mathbf{x}_j)}{\pi} \quad (2)$$

where $\theta(\mathbf{x}_i, \mathbf{x}_j)$ is the angle between $\mathbf{x}_i$ and $\mathbf{x}_j$. Obviously, SH is an unsupervised hashing method.

2.2 Kernelized Locality Sensitive Hashing

A kernelized variant of SH, named Kernelized Locality Sensitive Hashing (KLSH) (Kulis *et al.*, 2009a), is proposed for non-linearly separable data. KLSH approximates the underling Gaussian distribution in the implicit embedding space of data based on central limit theory. To calculate the value of hashing fuction $h(\cdot)$, KLSH projects points onto the eigenvectors of the kernel matrix. In short, the complete procedure of KLSH can be summarized as follows: 1) randomly select P (a small value) points from $\mathcal{X}$ and form the kernel matrix, 2) for each hash function $h(\phi(\mathbf{x}))$, calculate its weight $\boldsymbol{\omega} \in \mathbb{R}^P$ just as Kernel PCA (Schölkopf *et al.*, 1997), and 3) the hash function is defined as:

$$h(\phi(\mathbf{x})) = \mathrm{sign}(\sum_{i=1}^{P} \omega_i \cdot \kappa(\mathbf{x}, \mathbf{x}_i)) \quad (3)$$

where $\kappa(\cdot, \cdot)$ can be any kernel function.

KLSH can improve hashing results via the kernel trick. However, KLSH is unsupervised, thus designing a data-specific kernel remains a big challenge.

2.3 Semi-Supervised Hashing

Semi-Supervised Hashing (SSH) (Wang *et al.*, 2010a) is recently proposed to incorporate prior knowledge for better hashing. Besides $\mathcal{X}$, prior knowledge in the form of similar and dissimilar object-pairs is also required in SSH. SSH tries to find L optimal hash functions which have maximum empirical accuracy on prior knowledge and maximum entropy by finding the top L eigenvectors of an extended covariance matrix[2] via PCA or SVD.

However, despite of the potential problems of numerical stability, SVD requires massive computational space and $O(M^3)$ computational time where M is feature dimension, which limits its usage for high-dimensional data (Trefethen *et al.*, 1997). Furthermore, the variance of directions obtained by PCA decreases with the decrease of the rank (Jolliffe, 1986). Thus, lower hash functions tend to have smaller entropy and larger empirical errors.

2.4 Others

Some other related works should be mentioned. A notable method is Locality Sensitive Hashing (LSH) (Indyk *et al.*, 1998). LSH performs a random linear projection to map similar objects to similar hash codes. However, LSH suffers from the efficiency problem that it tends to generate long codes (Salakhutdinov *et al.*, 2007). LAMP (Mu *et al.*, 2009) considers each hash function as a binary partition problem as in SVMs (Burges, 1998). Spectral Hashing (Weiss *et al.*, 2009) maintains similarity between objects in the reduced Hamming space by minimizing the averaged Hamming distance[3] between similar neighbors in the original Euclidean space. However, spectral hashing takes the assumption that data should be distributed uniformly, which is always violated in real-world applications.

3 Semi-Supervised SimHash

In this section, we present our hashing method, named Semi-Supervised SimHash (S^{3}H). Let $\mathcal{X}_L = \{(\mathbf{x}_1, c_1) \ldots (\mathbf{x}_u, c_u)\}$ be the labeled data, $c \in \{1 \ldots C\}$, $\mathbf{x} \in \mathbb{R}^M$, and $\mathcal{X}_U = \{\mathbf{x}_{u+1} \ldots \mathbf{x}_N\}$ the unlabeled data. Let $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_U$. Given the labeled data $\mathcal{X}_L$, we construct two sets, attraction set Θ_a and repulsion set Θ_r. Specifically, any pair $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_a$, $i, j \leq u$, denotes that $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same class, i.e., $c_i = c_j$, while any pair $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_r$, $i, j \leq u$, denotes that $c_i \neq c_j$. Unlike previews works that attempt to find L optimal hyperplanes, the basic idea of S^{3}H is to fix L random hyperplanes and to find an optimal feature-weight vector to relocate the objects such that similar objects have similar codes.

[2]The extended covariance matrix is composed of two components, one is an unsupervised covariance term and another is a constraint matrix involving labeled information.

[3]Hamming distance is defined as the number of bits that are different between two binary strings.

3.1 Data Representation

Since L random hyperplanes are fixed, we can represent a object $\mathbf{x} \in \mathcal{X}$ as its relative position to these random hyperplanes, i.e.,

$$\mathbf{D} = \mathbf{\Lambda} \cdot \mathbf{V} \tag{4}$$

where the element $\mathrm{V}_{ml} \in \{+1, -1, 0\}$ of $\mathbf{V}$ indicates that the object $\mathbf{x}$ is above, below or just in the l-th hyperplane with respect to the m-th feature, and $\mathbf{\Lambda} = \mathrm{diag}(|x_1|, |x_2|, \ldots, |x_M|)$ is a diagonal matrix which, to some extent, reflects the distance from $\mathbf{x}$ to these hyperplanes.

3.2 Formulation

Hashing maps the data set $\mathcal{X}$ to an L-dimensional Hamming space for compact representations. If we represent each object as Equation (4), the l-th hash function is then defined as:

$$h_l(\mathbf{x}) = \hbar_l(\mathbf{D}) = \mathrm{sign}(\mathbf{w}^T \mathbf{d}_l) \tag{5}$$

where $\mathbf{w} \in \mathbb{R}^M$ is the feature weight to be determined and $\mathbf{d}_l$ is the l-th column of the matrix $\mathbf{D}$.

Intuitively, the "contribution" of a specific feature to different classes is different. Therefore, we hope to incorporate this side information in S^{3}H for better hashing. Inspired by (Madani *et al.*, 2009), we can measure this contribution over $\mathcal{X}_L$ as in Algorithm 1. Clearly, if objects are represented as the occurrence numbers of features, the output of Algorithm 1 is just the conditional probability $\mathrm{Pr}(\mathrm{class}|\mathrm{feature})$. Finally, each object $(\mathbf{x}, c) \in \mathcal{X}_L$ can be represented as an $M \times L$ matrix $\mathbf{G}$:

$$\mathbf{G} = \mathrm{diag}(\nu_{1,c}, \nu_{2,c}, \ldots, \nu_{M,c}) \cdot \mathbf{D} \tag{6}$$

Note that, one pair $(\mathbf{x}_i, \mathbf{x}_j)$ in Θ_a or Θ_r corresponds to $(\mathbf{G}_i, \mathbf{G}_j)$ while $(\mathbf{D}_i, \mathbf{D}_j)$ if we ignore features' contribution to different classes.

Furthermore, we also hope to maximize the empirical accuracy on the labeled data Θ_a and Θ_r and

Algorithm 1: Feature Contribution Calculation

```
for each $(\mathbf{x}, c) \in \mathcal{X}_L$ do
    for each $f \in \mathbf{x}$ do
        $\nu_f \leftarrow \nu_f + x_f$;
        $\nu_{f,c} \leftarrow \nu_{f,c} + x_f$;
    end
end
for each feature $f$ and class $c$ do
    $\nu_{f,c} \leftarrow \frac{\nu_{f,c}}{\nu_f}$;
end
```

maximize the entropy of hash functions. So, we define the following objective for $\hbar(\cdot)$s:

$$J(\mathbf{w}) = \frac{1}{N_p}\sum_{l=1}^{L}\Big\{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\Theta_a}\hbar_l(\mathbf{x}_i)\hbar_l(\mathbf{x}_j) - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\Theta_r}\hbar_l(\mathbf{x}_i)\hbar_l(\mathbf{x}_j)\Big\} + \lambda_1\sum_{l=1}^{L}\mathrm{H}(\hbar_l) \quad (7)$$

where $N_p = |\Theta_a| + |\Theta_r|$ is the number of attraction and repulsion pairs and λ_1 is a tradeoff between two terms. Wang *et al.* have proven that hash functions with maximum entropy must maximize the variance of the hash values, and vice-versa (Wang *et al.*, 2010b). Thus, $\mathrm{H}(\hbar(\cdot))$ can be estimated over the labeled and unlabeled data, $\mathcal{X}_L$ and $\mathcal{X}_U$.

Unfortunately, direct solution for above problem is non-trivial since Equation (7) is not differentiable. Thus, we relax the objective and add an additional regularization term which could effectively avoid overfitting. Finally, we obtain the total objective:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{N_p}\sum_{l=1}^{L}\{\sum_{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_a}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\psi(\mathbf{w}^T\mathbf{g}_{j,l}) \\ &- \sum_{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_r}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\psi(\mathbf{w}^T\mathbf{g}_{j,l})\} \\ &+ \frac{\lambda_1}{2N}\sum_{l=1}^{L}\{\sum_{i=1}^{u}\psi^2(\mathbf{w}^T\mathbf{g}_{i,l}) + \sum_{i=u+1}^{N}\psi^2(\mathbf{w}^T\mathbf{d}_{i,l})\} \\ &- \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2\end{aligned} \quad (8)$$

where $\mathbf{g}_{i,l}$ and $\mathbf{d}_{i,l}$ denote the l-th column of $\mathbf{G}_i$ and $\mathbf{D}_i$ respectively, and $\psi(t)$ is a piece-wise linear function defined as:

$$\psi(t) = \begin{cases} T_g & t > T_g \\ t & -T_g \le t \le T_g \\ -T_g & t < -T_g \end{cases} \quad (9)$$

This relaxation has a good intuitive explanation. That is, similar objects are desired to not only have the similar fingerprints but also have sufficient large projection magnitudes, while dissimilar objects are desired to not only differ in their fingerprints but also have large projection margin. However, we do not hope that a small fraction of object-pairs with very large projection magnitude or margin dominate the complete model. Thus, a piece-wise linear function $\psi(\cdot)$ is applied in S^{3}H.

As a result, Equation (8) is a simply unconstrained optimization problem, which can be efficiently solved by a notable Quasi-Newton algorithm, i.e., L-BFGS (Liu *et al.*, 1989). For description simplicity, only attraction set Θ_a is considered and the extension to repulsion set Θ_r is straightforward. Thus, the gradient of $\mathcal{L}(\mathbf{w})$ is as follows:

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathbf{w})}{\partial\mathbf{w}} &= \frac{1}{N_p}\sum_{l=1}^{L}\Big\{\sum_{\substack{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_a,\\ |\mathbf{w}^T\mathbf{g}_{i,l}|\le T_g}}\psi(\mathbf{w}^T\mathbf{g}_{j,l})\cdot\mathbf{g}_{i,l} \\ &+ \sum_{\substack{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_a,\\ |\mathbf{w}^T\mathbf{g}_{j,l}|\le T_g}}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\cdot\mathbf{g}_{j,l}\Big\} \\ &+ \frac{\lambda_1}{N}\sum_{l=1}^{L}\Big\{\sum_{\substack{i=1,\\ |\mathbf{w}^T\mathbf{g}_{i,l}|\le T_g}}^{u}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\cdot\mathbf{g}_{i,l} \\ &+ \sum_{\substack{i=u+1,\\ |\mathbf{w}^T\mathbf{d}_{i,l}|\le T_g}}^{N}\psi(\mathbf{w}^T\mathbf{d}_{i,l})\cdot\mathbf{d}_{i,l}\Big\} - \lambda_2\mathbf{w}\end{aligned} \quad (10)$$

Note that $\partial\psi(t)/\partial t = 0$ when $|t| > T_g$.

3.3 Fingerprint Generation

When we get the optimal weight $\mathbf{w}^*$, we generate fingerprints for given objects through Equation (5). Then, it tunes to the problem how to efficiently obtain the representation as in Figure 4 for a object. After analysis, we find: 1) hyperplanes are randomly generated and we only need to determine which sides of these hyperplanes the given object lies on, and 2) in real-world applications, objects such as docs are always very sparse. Thus, we can avoid heavy computational demands and efficiently generate fingerprints for objects.

In practice, given an object $\mathbf{x}$, the procedure of generating an L-bit fingerprint is as follows: it maintains an L-dimensional vector initialized to zero. Each feature $f \in \mathbf{x}$ is firstly mapped to an L-bit hash value by *Jenkins Hashing Function*[4]. Then,

[4]http://www.burtleburtle.net/bob/hash/doobs.html

Algorithm 2: Fast Fingerprint Generation

```
INPUT: $\mathbf{x}$ and $\mathbf{w}^*$;
initialize $\boldsymbol{\alpha} \leftarrow 0, \boldsymbol{\beta} \leftarrow 0, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^L$;
for each $f \in \mathbf{x}$ do
    randomly project $f$ to $\mathbf{h}_f \in \{-1, +1\}^L$;
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + x_f \cdot w_f^* \cdot \mathbf{h}_f$;
end
for $l = 1$ to $L$ do
    if $\alpha_l > 0$ then
        $\beta_l \leftarrow 1$;
    end
end
RETURN $\boldsymbol{\beta}$;
```

these L bits increment or decrement the L components of the vector by the value $x_f \times w_f^*$. After all features processed, the signs of components determine the corresponding bits of the final fingerprint. The complete algorithm is presented in Algorithm 2.

3.4 Algorithmic Analysis

This section briefly analyzes the relation between S^{3}H and some existing methods. For analysis simplicity, we assume $\psi(t) = t$ and ignore the regularization terms. So, Equation (8) can be rewritten as follows:

$$J(\mathbf{w})_{S^3H} = \frac{1}{2}\mathbf{w}^T[\sum_{l=1}^{L} \boldsymbol{\Gamma}_l(\boldsymbol{\Phi}^+ - \boldsymbol{\Phi}^-)\boldsymbol{\Gamma}_l^T]\mathbf{w} \quad (11)$$

where Φ_{ij}^+ equals to 1 when $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_a$ otherwise 0, Φ_{ij}^- equals to 1 when $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_r$ otherwise 0, and $\boldsymbol{\Gamma}_l = [\mathbf{g}_{1,l} \ldots \mathbf{g}_{u,l}, \mathbf{d}_{u+1,l} \ldots \mathbf{d}_{N,l}]$. We denote $\sum_l \boldsymbol{\Gamma}_l \boldsymbol{\Phi}^+ \boldsymbol{\Gamma}_l^T$ and $\sum_l \boldsymbol{\Gamma}_l \boldsymbol{\Phi}^- \boldsymbol{\Gamma}_l^T$ as $\mathbf{S}^+$ and $\mathbf{S}^-$ respectively. Therefore, maximizing above function is equivalent to maximizing the following:

$$\widetilde{J}(\mathbf{w})_{S^3H} = \frac{|\mathbf{w}^T\mathbf{S}^+\mathbf{w}|}{|\mathbf{w}^T\mathbf{S}^-\mathbf{w}|} \quad (12)$$

Clearly, Equation (12) is analogous to Linear Discriminant Analysis (LDA) (Duda *et al.*, 2000) except for the difference: 1) measurement. S^{3}H uses similarity while LDA uses distance. As a result, the objective function of S^{3}H is just the reciprocal of LDA's. 2) embedding space. LDA seeks the best separative direction in the original attribute space. In contrast, S^{3}H firstly maps data from $\mathbb{R}^M$ to $\mathbb{R}^{M\times L}$ through the following projection function

$$\phi(\mathbf{x}) = \mathbf{x} \cdot [\text{diag}(\text{sign}(\mathbf{r}_1)), \ldots, \text{diag}(\text{sign}(\mathbf{r}_L))] \quad (13)$$

where $\mathbf{r}_l \in \mathbb{R}^M, l = 1, \ldots, L$, are L random hyperplanes. Then, in that space ($\mathbb{R}^{M\times L}$), S^{3}H seeks a direction[5] that can best separate the data.

From this point of view, it is obvious that the basic SH is a special case of S^{3}H when $\mathbf{w}$ is set to $\mathbf{e} = [1, 1, \ldots, 1]$. That is, SH firstly maps the data via $\phi(\cdot)$ just as S^{3}H. But then, SH directly separates the data in that feature space at the direction $\mathbf{e}$.

Analogously, we ignore the regularization terms in SSH and rewrite the objective of SSH as:

$$J(\mathbf{W})_{SSH} = \frac{1}{2}\,\text{tr}[\mathbf{W}^T\mathbf{X}(\boldsymbol{\Phi}^+ - \boldsymbol{\Phi}^-)\mathbf{X}^T\mathbf{W}] \quad (14)$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L] \in \mathbb{R}^{M\times L}$ are L hyperplanes and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$. Maximizing this objective is equivalent to maximizing the following:

$$\widetilde{J}(\mathbf{W})_{SSH} = \frac{|\,\text{tr}[\mathbf{W}^T\mathbf{S}'^+\mathbf{W}]|}{|\,\text{tr}[\mathbf{W}^T\mathbf{S}'^-\mathbf{W}]|} \quad (15)$$

where $\mathbf{S}'^+ = \mathbf{X}\boldsymbol{\Phi}^+\mathbf{X}^T$ and $\mathbf{S}'^- = \mathbf{X}\boldsymbol{\Phi}^-\mathbf{X}^T$. Equation (15) shows that SSH is analogous to Multiple Discriminant Analysis (MDA) (Duda *et al.*, 2000). In fact, SSH uses top L best-separative hyperplanes in the original attribute space found via PCA to hash the data. Furthermore, we rewrite the projection function $\phi(\cdot)$ in S^{3}H as:

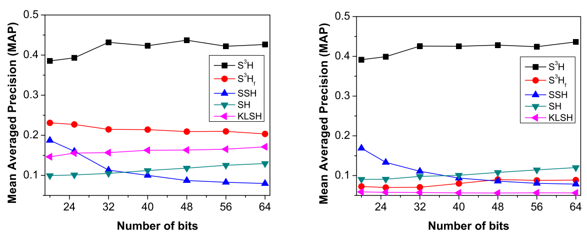
$$\phi(\mathbf{x}) = \mathbf{x} \cdot [\mathbf{R}_1, \ldots, \mathbf{R}_L] \quad (16)$$

where $\mathbf{R}_l = \text{diag}(\text{sign}(\mathbf{r}_l))$. Each $\mathbf{R}_l$ is a mapping from $\mathbb{R}^M$ to $\mathbb{R}^M$ and corresponds to one embedding space. From this perspective, unlike SSH, S^{3}H globally seeks a direction that can best separate the data in L different embedding spaces simultaneously.

4 Experiments

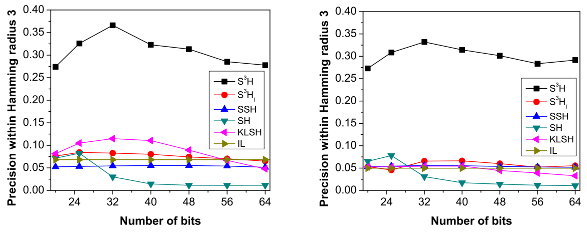
We use two datasets 20 Newsgroups and Open Directory Project (ODP) in our experiments. Each document is represented as a vector of occurrence numbers of the terms within it. The class information of docs is considered as prior knowledge that two docs within a same class should have more similar fingerprints while two docs within different classes should have dissimilar fingerprints. We will demonstrate that our S^{3}H can effectively incorporate this prior knowledge to improve the DSS performance.

[5]The direction is determined by concatenating $\mathbf{w}$ L times.



(a) (b)

Figure 1: Mean Averaged Precision (MAP) for different number of bits for hash ranking on 20 Newsgroups. (a) 10K features. (b) 30K features.



(a) (b)

Figure 2: Precision within Hamming radius 3 for hash lookup on 20 Newsgroups. (a) 10K features. (b) 30K features.

We use Inverted List (IL) (Manning *et al.*, 2002) as the baseline. In fact, given a query doc, IL returns all the docs that contain any term within it. We also compare our method with three state-of-the-art hashing methods, i.e., KLSH, SSH and SH. In KLSH, we adopt the RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\delta^2})$, where the scaling factor δ^2 takes 0.5 and the other two parameters p and t are set to be 500 and 50 respectively. The parameter λ in SSH is set to 1. For S^{3}H, we simply set the parameters λ_1 and λ_2 in Equation (8) to 4 and 0.5 respectively. To objectively reflect the performance of S^{3}H, we evaluate our S^{3}H with and without Feature Contribution Calculation algorithm (FCC) (Algorithm 1). Specifically, FCC-free S^{3}H (denoted as S^{3}H$_f$) is just a simplification when $\mathbf{G}$s in S^{3}H are simply set to $\mathbf{D}$s.

For quantitative evaluation, as in literature (Wang *et al.*, 2010b; Mu *et al.*, 2009), we calculate the precision under two scenarios: hash lookup and hash ranking. For hash lookup, the proportion of good neighbors (have the same class label as the query) among the searched objects within a given Hamming radius is calculated as precision. Similarly to (Wang *et al.*, 2010b; Weiss *et al.*, 2009), for a query document, if no neighbors within the given Hamming radius can be found, it is considered as zero precision. Note that, the precision of IL is the proportion of good neighbors among the whole searched objects. For hash ranking, all the objects in $\mathcal{X}$ are ranked in terms of their Hamming distance from the query document, and the top K nearest neighbors are returned as the result. Then, Mean Averaged Precision (MAP) (Manning *et al.*, 2002) is calculated. We also calculate the averaged intra- and inter- class Hamming distance for various hashing methods. Intuitively, a good hashing method should have small intra-class distance while large inter-class distance.

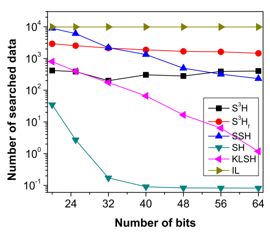
We test all the methods on a PC with a 2.66 GHz processor and 12GB RAM. All experiments repeate 10 times and the averaged results are reported.

4.1 20 Newsgroups

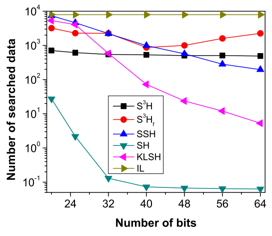
20 Newsgroups[6] contains 20K messages, about 1K messages from each of 20 different newsgroups. The entire vocabulary includes 62,061 words. To evaluate the performance for different feature dimensions, we use Chi-squared feature selection algorithm (Forman, 2003) to select 10K and 30K features. The averaged message length is 54.1 for 10K features and 116.2 for 30K features. We randomly select 4K massages as the test set and the remain 16K as the training set. To train SSH and S^{3}H, from the training set, we randomly generate 40K message-pairs as Θ_a and 80K message-pairs as Θ_r.

For hash ranking, Figure 1 shows MAP for various methods using different number of bits. It shows that performance of SSH decreases with the growing of hash bits. This is mainly because the variance of the directions obtained by PCA decreases with the decrease of their ranks. Thus, lower bits have larger empirical errors. For S^{3}H, FCC (Algorithm 1) can significantly improve the MAP just as discussed in Section 3.2. Moreover, the MAP of FCC-free S^{3}H (S^{3}H$_f$) is affected by feature dimensions while FCC-based (S^{3}H) is relatively stable. This implies FCC can also improve the satiability of S^{3}H. As we see, S^{3}H$_f$ ignores the contribution of features to different classes. However, besides the local description of data locality in the form of object-pairs, such

[6]http://www.cs.cmu.edu/afs/cs/project/theo-3/www/



(a)



(b)

Figure 3: Averaged searched sample numbers using 4K query messages for hash lookup. (a) 10K features. (b) 30K features.

(global) information also provides a proper guidance for hashing. So, for S^3H_f, the reason why its results with 30K features are worse than the results with 10K features is probably because S^3H_f learns to hash only according to the local description of data locality and many not too relevant features lead to relatively poor description. In contrast, S^3H can utilize global information to better understand the similarity among objects. In short, S^3H obtains the best MAP for all bits and feature dimensions.

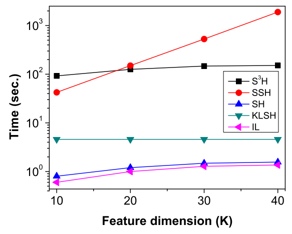
For hash lookup, Figure 2 presents the precision within Hamming radius 3 for different number of bits. It shows that IL even outperforms SH. This is because few objects can be hashed by SH into one hash bucket. Thus, for many queries, SH fails to return any neighbor even in a large Hamming radius of 3. Clearly, S^3H outperforms all the other methods for different number of hash bits and features.

The number of messages searched by different methods are reported in Figure 3. We find that the number of searched data of S^3H (with/without FCC) decreases much more slowly than KLSH, SH and SSH with the growing of the number of hash bits. As discussed in Section 3.4, this mainly benefits from the design of S^3H that S^3H (globally) seeks a direction that can best separate the data in L embedding spaces simultaneously. We also find IL returns a large number of neighbors of each query message which leads to its poor efficiency.

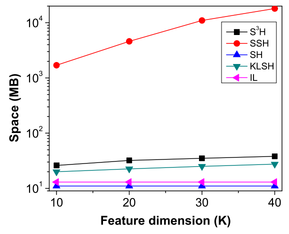
The averaged intra- and inter- class Hamming distance of different methods are reported in Table 1. As it shows, S^3H has relatively larger margin (Δ) between intra- and inter-class Hamming distance. This indicates that S^3H is more effective to make similar points have similar fingerprints while keep the dissimilar points away enough from each other.

	intra-class	inter-class	Δ
S^3H	13.1264	15.6342	**2.5078**
S^3H_f	12.5754	13.3479	0.7725
SSH	6.4134	6.5262	0.1128
SH	15.3908	15.6339	0.2431
KLSH	10.2876	10.8713	0.5841

Table 1: Averaged intra- and inter- class Hamming distance of 20 Newsgroups for 32-bit fingerprint. Δ is the difference between the averaged inter- and intra- class Hamming distance. Large Δ implies good hashing.



(a)



(b)

Figure 4: Computational complexity of training for different feature dimensions for 32-bit fingerprint. (a) Training time (sec). (b) Training space cost (MB).

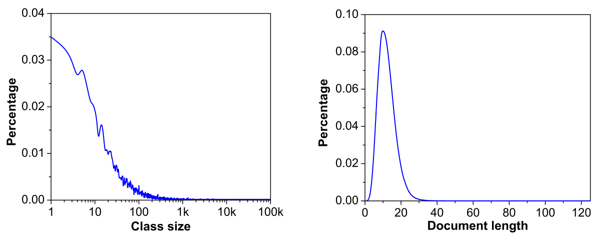
Figure 4 shows the (training) computational complexity of different methods. We find that the time and space cost of SSH grows much faster than SH, KLSH and S^3H with the growing of feature dimension. This is mainly because SSH requires SVD to find the optimal hashing functions which is computational expensive. Instead, S^3H seeks the optimal feature weights via L-BFGS, which is still efficient even for very high-dimensional data.

4.2 Open Directory Project (ODP)

Open Directory Project (ODP)[7] is a multilingual open content directory of web links (docs) organized by a hierarchical ontology scheme. In our experiment, only English docs[8] at level 3 of the category tree are utilized to evaluate the performance. In short, the dataset contains 2,483,388 docs within 6,008 classes. There are totally 862,050 distinct words and each doc contains 14.13 terms on average. Since docs are too short, we do not conduct

[7]http://rdf.dmoz.org/

[8]The title together with the corresponding short description of a page are considered as a document in our experiments.



(a) (b)

Figure 5: Overview of ODP data set. (a) Class distribution at level 3. (b) Distribution of document length.

	intra-class	**inter-class**	Δ
S^3H	14.0029	15.9508	**1.9479**
S^3H_f	14.3801	15.5260	1.1459
SH	14.7725	15.6432	0.8707
KLSH	9.3382	10.5700	1.2328

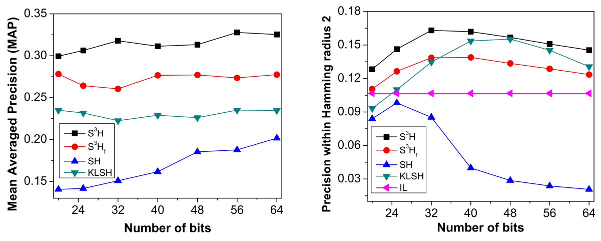
Table 2: Averaged intra- and inter- class Hamming distance of ODP for 32-bit fingerprint (860K features). Δ is the difference between averaged intra- and inter- class Hamming distance.

feature selection[9]. An overview of ODP is shown in Figure 5. We randomly sample 10% docs as the test set and the remain as the training set. Furthermore, from training set, we randomly generate 800K doc-pairs as Θ_a, and 1 million doc-pairs as Θ_r. Note that, since there are totally over 800K features, it is extremely inefficient to train SSH. Therefore, we only compare our S^3H with IL, KLSH and SH.

The search performance is given in Figure 6. Figure 6(a) shows the MAP for various methods using different number of bits. It shows KLSH outperforms SH, which mainly contributes to the kernel trick. S^3H and S^3H_f have higher MAP than KLSH and SH. Clearly, FCC algorithm can improve the MAP of S^3H for all bits. Figure 6(b) presents the precision within Hamming radius 2 for hash lookup. We find that IL outperforms SH since SH fails for many queries. It also shows that S^3H (with FCC) can obtain the best precision for all bits.

Table 2 reports the averaged intra- and inter-class Hamming distance for various methods. It shows that S^3H has the largest margin (Δ). This demonstrates S^3H can measure the similarity among the data better than KLSH and SH.

[9]We have tested feature selection. However, if we select 40K features via Chi-squared feature selection method, documents are represented by 3.15 terms on average. About 44.9% documents are represented by no more than 2 terms.



(a) (b)

Figure 6: Retrieval performance of different methods on ODP. (a) Mean Averaged Precision (MAP) for different number of bits for hash ranking. (b) Precision within Hamming radius 2 for hash lookup.

We should emphasize that KLSH needs 0.3ms to return the results for a query document for hash lookup, and S^3H needs $<$0.1ms. In contrast, IL requires about 75ms to finish searching. This is mainly because IL always returns a large number of objects (dozens or hundreds times more than S^3H and KLSH) and requires much time for post-processing.

All the experiments show S^3H is more effective, efficient and stable than the baseline method and the state-of-the-art hashing methods.

5 Conclusions

We have proposed a novel supervised hashing method named Semi-Supervised Simhash (S^3H) for high-dimensional data similarity search. S^3H learns the optimal feature weights from prior knowledge to relocate the data such that similar objects have similar fingerprints. This is implemented by maximizing the empirical accuracy on labeled data together with the entropy of hash functions. The proposed method leads to a simple Quasi-Newton based solution which is efficient even for very high-dimensional data. Experiments performed on two large datasets have shown that S^3H has better search performance than several state-of-the-art methods.

6 Acknowledgements

We thank Fangtao Li for his insightful suggestions. We would also like to thank the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China under Grant No. 60873174.

References

Christopher J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121-167.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th annual ACM symposium on Theory of computing*, pages 380-388.

Gianni Costa, Giuseppe Manco and Riccardo Ortale. 2010. An incremental clustering scheme for data deduplication. *Data Mining and Knowledge Discovery*, 20(1):152-187.

Jeffrey Dean and Monika R. Henzinge. 1999. Finding Related Pages in the World Wide Web. *Computer Networks*, 31:1467-1479.

Richard O. Duda, Peter E. Hart and David G. Stork. 2000. Pattern classification, 2nd edition. Wiley-Interscience.

George Forman 2003. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289-1305.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th annual ACM symposium on Theory of computing*, pages 604-613.

Ian Jolliffe. 1986. Principal Component Analysis. *Springer-Verlag*, New York.

Yan Ke, Rahul Sukthankar and Larry Huston. 2004. Efficient near-duplicate detection and sub-image retrieval. In *Proceedings of the ACM International Conference on Multimedia*.

Brian Kulis and Kristen Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the 12th International Conference on Computer Vision*, pages 2130-2137.

Brian Kulis, Prateek Jain and Kristen Grauman. 2009. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2143-2157.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1): 503-528.

Omid Madani, Michael Connor and Wiley Greiner. 2009. Learning when concepts abound. *The Journal of Machine Learning Research*, 10:2571-2613.

Gurmeet Singh Manku, Arvind Jain and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141-150.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. 2002. An introduction to information retrieval. *Spring*.

Yadong Mu, Jialie Shen and Shuicheng Yan. 2010. Weakly-Supervised Hashing in Kernel Space. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pages 3344-3351.

Ruslan Salakhutdinov and Geoffrey Hintona. 2007. Semantic hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*.

Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. 1997. Kernel principal component analysis. *Advances in Kernel Methods - Support Vector Learning*, pages 583-588. MIT.

Lloyd N. Trefethen and David Bau. 1997. Numerical linear algebra. *Society for Industrial Mathematics*.

Xiaojun Wan, Jianwu Yang and Jianguo Xiao. 2008. Towards a unified approach to document similarity search using manifold-ranking of blocks. *Information Processing & Management*, 44(3):1032-1048.

Jun Wang, Sanjiv Kumar and Shih-Fu Chang. 2010a. Semi-Supervised Hashing for Scalable Image Retrieval. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pages 3424-3431.

Jun Wang, Sanjiv Kumar and Shih-Fu Chang. 2010b. Sequential Projection Learning for Hashing with Compact Codes. In *Proceedings of International Conference on Machine Learning*.

Yair Weiss, Antonio Torralba and Rob Fergus. 2009. Spectral hashing. In *Proceedings of Advances in Neural Information Processing Systems*.