R^{2}SDH: Robust Rotated Supervised Discrete Hashing

Jie Gui
Rutgers University
Piscataway, NJ, 08854, USA
guijiejie@gmail.com

Ping Li
Baidu Research
Bellevue, WA, 98004, USA
pingli98@gmail.com

ABSTRACT

Learning-based hashing has recently received considerable attentions due to its capability of supporting efficient storage and retrieval of high-dimensional data such as images, videos, and documents. In this paper, we propose a learning-based hashing algorithm called "Robust Rotated Supervised Discrete Hashing" (R^{2}SDH), by extending the previous work on "Supervised Discrete Hashing" (SDH). In R^{2}SDH, **correntropy** is adopted to replace the least square regression (LSR) model in SDH for achieving better robustness. Furthermore, considering the commonly used distance metrics such as cosine and Euclidean distance are invariant to rotational transformation, **rotation** is integrated into the original zero-one label matrix used in SDH, as additional freedom to promote flexibility without sacrificing accuracy. The rotation matrix is learned through an optimization procedure. Experimental results on three image datasets (MNIST, CIFAR-10, and NUS-WIDE) confirm that R^{2}SDH generally outperforms SDH.

KEYWORDS

supervised discrete hashing, robust M-estimator, rotation

ACM Reference Format:
Jie Gui and Ping Li. 2018. R^{2}SDH: Robust Rotated Supervised Discrete Hashing. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219955

1 INTRODUCTION

In large-scale applications in search, data mining, and machine learning, it is highly desirable that the data should be organized and indexed efficiently and accurately. As a well-established and powerful large-scale technique, **hashing** has shown promising performance and has received great attentions from researchers in data mining, machine learning, computer vision, information retrieval, and related areas due to its practical utility. Hashing generally involves generating a range of hash functions to map each instance such as an image, a video, a document or other types of data into a vector of binary code. The produced hash codes should preserve the original instance structure (e.g., similarities between the original instances). In general, distance computations can be conducted very efficiently in the Hamming space. Thus, after the data are hashed, subsequent algorithms based on pairwise comparisons can be calculated more efficiently in large-scale datasets. Due to the flexibility of binary representations, hashing methods can be applied in many situations, for instance, searching efficiently by investigating only instances falling into buckets close to the query based on the Hamming distance or using the binary codes for other tasks such as face recognition, image classification, digit recognition, and face indexing.

Generally speaking, hashing algorithms can be classified into two main categories: (A) learning-free hashing methods; and (B) learning-based (also known as data-dependent) hashing methods.

1.1 Learning-Free Hashing Methods

Learning-free hashing methods do not need training data and randomly establish a series of hash functions without any learning. There are numerous learning-free methods. If we restrict the discussion to methods which produce binary (or integer) outputs, then popular learning-free algorithms include at least the following:

- *Sign (Gaussian) random projections*: the collision probability (or empirically the hamming distance) of the hash code is proportional to the cosine similarity of the data [6, 8, 20].
- *Sign Cauchy random projections*: the collision probability is proportional to the χ^2 similarity of the data [22].
- *(b-bit) minwise hashing* (for binary data): the collision probability is proportional to the resemblance [2–4, 19, 21].
- *Consistent weighted sampling (CWS)*: the collision probability is proportional to the min-max similarity [13, 17, 18, 31].

Apparently, the advantage of learning-free methods is that they do not require any training. However, one will then have to choose appropriate hashing methods based on data, otherwise the performance might be disappointing. For example, while random projection methods (and variants such as random fourier features) are very popular, they are often not as accurate [18] (i.e., requiring long hashing code). Minwise hashing (and variants) and consistent weighted sampling generate integer hash code which are equivalent to high-dimensional sparse binary data and typically outperform random projection-based methods (in terms of accuracy versus number of hashes) in real data.

1.2 Learning-Based Hashing Methods

Recently, data-dependent (learning-based) hashing algorithms have become popular, since (potentially) learned compact hash codes would be able to efficiently and effectively organize and index large-scale data. Unlike learning-free methods which randomly construct hash functions, learning-based hashing algorithms aim to generate short hash codes by utilizing training data. An enormous volume

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. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '18, August 19–23, 2018, London, United Kingdom
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219955

of literature has been devoted to develop a wide variety of hashing algorithms which can be further divided in into the following sub-categories.

Unsupervised hashing: The training instance labels are not required in learning. For example, Gong et al. [10] presented an iterative quantization (ITQ) method that minimized the binarization loss between the original instances and hash codes. Weiss et al. [44] proposed a spectral hashing (SH) method whose objective function was similar to that of Laplacian eigenmaps [1]. Other unsupervised learning-based hashing algorithms such as anchor graph hashing (AGH) [28], inductive manifold hashing (IMH) [38] with t-distributed stochastic neighbor embedding (t-SNE) [30], and scalable graph hashing with feature transformation (SGH) [14] have also been proposed. Because unsupervised hashing does not take into consideration the training example labels, useful information key to classification may be lost. Therefore, many semi-supervised hashing and supervised hashing methods have been proposed.

Semi-supervised learning-based hashing: Hash function is learned from labeled examples and considerably more unlabeled examples. For example, Kulis and Darrell [15] presented a binary reconstructive embedding (BRE) algorithm that minimized the reconstruction error between the original Euclidean distance and the learned Hamming distance. Wang et al. [43] presented a semi-supervised hashing (SSH) method that simultaneously maximized the variance of all (unlabeled and labeled) training instances and minimized the empirical loss for pairwise labeled training instances.

Supervised learning-based hashing: Hash functions are learned using training instance labels. For example, Rastegari et al. [35] proposed predictable dual-view hashing which added the idea of support vector machines (SVMs) to hash learning. Liu et al. [27] presented a kernel-based supervised hashing (KSH) algorithm that needed a limited amount of label information, i.e., dissimilar and similar instance pairs. Other supervised data-dependent hashing methods include linear discriminant analysis based hashing (LDA-Hash) [41] and fast supervised hashing using graph cuts and decision trees (FastHash) [23, 24]. In our opinion, ranking-based algorithms [33] in which the supervised information are composed of ranking labels such as triplets are also part of the supervised hashing algorithms.

Multimodal hashing: It includes cross-modal hashing and multi-source hashing. In cross-modal hashing [16], the query denotes one modality while the output denotes another modality. For instance, given a text query, images are returned which represent the text. In multi-source hashing [39, 45], it is assumed that all views are given for a query and the goal is to learn better hash codes than unimodal hashing by fully utilizing all these views. Thus, both cross-modal hashing and multi-source hashing utilize multimodal information [5]. However, they are utilized in different situations and cross-modal hashing may have wider applications than multi-source hashing in real applications.

Deep learning-based hashing: Many algorithms have been proposed over the past few years, some of which have been successfully applied to many real applications such as image retrieval, action recognition, and image classification. As far as we know, semantic hashing [36] is the first to use deep learning for hashing. This seminal work used the stacked-restricted Boltzmann machine (RBM) to learn compact binary codes for visual search. However, the model was complicated and needed pre-training, which is inefficient in real applications. Some other related algorithms can be found in [25, 40].

In general, hash codes consist of -1 and 1 or 0 and 1. The discrete constraints imposed on the binary codes lead to complex binary optimization problems, which are generally NP-hard. To facilitate the optimization in hash code learning, most hashing algorithms first do not take into consideration the discrete constraints, then solve a relaxed and continuous problem, and finally turn real values into the approximate binary codes by thresholding (or quantization). This relaxation skill greatly simplifies the original complicated mixed integer optimization problem. However, the approximate solution is apparently suboptimal, often of low quality, and reduces the effectiveness of the final binary code. It is possibly due to the accumulated quantization error, especially when learning long binary codes. Most existing hashing methods do not take into consideration the significance of discrete optimization. Shen et al. [37] proposed a novel supervised discrete hashing (SDH) method that directly learned the binary codes without relaxation. To fully utilize label information, SDH was formulated as a least squares regression that regressed the corresponding label on each hash code. Luo et al. [29] proposed robust discrete code modeling for SDH. Gui et al. [11] proposed SDH with relaxation (SDHR).

1.3 Summary of Contribution

Least squares regression (LSR) is a widely-utilized statistical analysis method, which has been successfully used in many data mining and machine learning tasks, such as classification, clustering, dimensionality reduction, and multi-view learning. However, the ordinary LSR is sensitive to outliers [32], which motivates us to consider strategies for enhancing the robustness in SDH. To this end, we propose a robust SDH model which uses robust M-estimator [26] and can be defined such as Welsch M-estimator and Cauchy M-estimator. Furthermore, the original LSR utilize the label matrix as the regression target, which is fixed. To increase the flexibility, we add a rotation matrix to the label matrix. The rotation matrix is solved through optimization which is more flexible and maybe more accurate. We name our algorithm "Robust Rotated Supervised Discrete Hashing" (R^2SDH).

The rest of the paper is organized as follows: Section 2 briefly introduces supervised discrete hashing. Section 3 presents the R^2SDH. Section 4 reports on our experimental results. Finally, Section 5 concludes the paper.

2 REVIEW OF SUPERVISED DISCRETE HASHING (SDH)

We briefly review the related work SDH [37] in this section. Given n (See Table 1 for the notations used in this paper) examples $X = \{x_i\}_{i=1}^n$, our aim is to learn the corresponding hash codes $B = \{b_i\}_{i=1}^n \in \{-1, 1\}^{n \times l}$ to retain their similarities in the original space, where the i-th row vector b_i is the l-bits hash codes for x_i. The label matrix is denoted as $Y = \{y_i\}_{i=1}^n \in \mathbb{R}^{n \times c}$, where c denotes the number of classes. The term y_{ik} is the kth element of y_i and $y_{ik} = 1$ if x_i is from class k and 0 otherwise.

Table 1: Notation

Symbol	Description
$X \in \mathbb{R}^{n\times d}$	the data matrix
$x_i \in \mathbb{R}^{1\times d}$	the i-th example
$B \in \mathbb{R}^{n\times l}$	the hash codes
$b_i \in \mathbb{R}^{1\times l}$	the hash code for x_i (the i-th row vector of B)
$Y \in \mathbb{R}^{n\times c}$	the class labels of all training examples
$y_i \in \mathbb{R}^{1\times c}$	the i-th row vector of the label matrix Y
$P \in \mathbb{R}^{m\times l}$	the projection matrix for the nonlinear embedding
$W \in \mathbb{R}^{l\times c}$	the projection matrix for the hash code
$R \in \mathbb{R}^{c\times c}$	the rotation matrix
n	the training example size: the number of all training instances
l	the hash code length
c	the number of classes
$F(\cdot)$	the nonlinear embedding to approximate the hash code
m	the number of anchor points
$\{a_j\}_{i=1}^m$	randomly selected m anchor points from the training instances
$\phi(\cdot)$	the m-dimensional vector obtained by the kernel function
δ	the kernel parameter of last term of SDH's objective function
$g()$	the robust M-estimator
σ^2	the kernel width of the robust M-estimator
r	the Hamming radius

SDH's objective function is defined as:

$$\min_{B,F,W} \sum_{i=1}^{n} \|y_i - b_i W\|_2^2 + \lambda \|W\|_F^2 + \upsilon \sum_{i=1}^{n} \|b_i - F(x_i)\|_2^2 \quad (1)$$
$$s.t.\ \forall i \quad b_i \in \{-1,1\}^l,$$

which can be written, equivalently, as

$$\min_{B,F,W} \|Y - BW\|_F^2 + \lambda \|W\|_F^2 + \upsilon \|B - F(X)\|_F^2 \quad (2)$$
$$s.t.\ B \in \{-1,1\}^{n\times l},$$

where $\|\cdot\|_F$ is the matrix Frobenius norm and W is the projection matrix. The first term of (2) is the least squares regression, which is used to regress the corresponding class label on each hash code. The second term of (2) is used for regularization. $F(\cdot)$ in the last term of (2) is an easy yet useful nonlinear embedding to approximate the hash code

$$F(x) = \phi(x) P, \quad (3)$$

where $\phi(x)$ is an m-dimensional vector acquired by the RBF kernel:

$$\phi(x) = [\exp\left(-\frac{\|x - a_1\|^2}{\delta}\right), \cdots, \exp\left(-\frac{\|x - a_m\|^2}{\delta}\right)]. \quad (4)$$

The terms $\{a_j\}_{j=1}^m$ are randomly selected m anchor points from the training instances, and δ is the Gaussian kernel width parameter. The matrix $P \in \mathbb{R}^{m\times l}$ projects $\phi(x)$ onto the low-dimensional space. Similar formulations as equation (3) are widely adopted in such as binary reconstructive embedding (BRE) [15] and kernel-based supervised hashing (KSH) [27].

The reason why a Gaussian kernel function is used is shown as follows: when the underlying feature embedding for the kernel is unknown, existing methods such as LSH do not apply for kernelized data in high-dimensional space. Furthermore, the usage of kernel generalizes such methods as LSH to adjust to arbitrary kernels, making it possible to keep the algorithm's sublinear time similarity search guarantees for a large number of useful similarity functions.

SDH's optimization has three steps: the F-step to solve P, the W-step to solve W, and the B-step to solve B:

F-step: If B is fixed, it is easy to solve the projection matrix P:

$$P = \left(\phi(X)^T \phi(X)\right)^{-1} \phi(X)^T B. \quad (5)$$

W-step: W is easily computed and has a closed-form solution by fixing B:

$$W = \left(B^T B + \lambda I\right)^{-1} B^T Y. \quad (6)$$

B-step: By fixing all other variables, [37] proposed solving B iteratively through a "discrete cyclic coordinate descent procedure".

3 R^{2}SDH: ROBUST ROTATED SUPERVISED DISCRETE HASHING

In this section, we introduce our proposed method in detail.

3.1 Correntropy

As using the ordinary least square regression might be sensitive to outliers, in this study, we adopt the idea of "correntropy" [26] in order to increase robustness of the hashing procedure. Specifically, we modify the original SDH formulation as follows:

$$\min_{B,F,W} \sum_{i=1}^{n} g\left((Y - BW)^i\right) + \lambda \|W\|_F^2 + \upsilon \|B - F(X)\|_F^2 \quad (7)$$
$$s.t.\ B \in \{-1,1\}^{n\times l}$$

where we recall $(Y - BW)^i$ is the i-th row of $(Y - BW)$. The term $g()$ in (7) is the robust M-estimator and can be defined such as Welsch M-estimator and Cauchy M-estimator. In this study, we focus on the Welsch M-estimator:

$$g(x) = 1 - \exp\left(-\frac{\|x\|_2^2}{\sigma^2}\right), \quad (8)$$

where σ^2 is the kernel width. We will describe a procedure for determining σ^2.

3.2 R^{2}SDH: Robust Rotated Supervised Discrete Hashing

If we rotate the label matrix, the distance between the code words of different classes remains the same. For example, for binary classification, the code words for the first class and second class are $y^1 = [1 \quad 0]$ (Fig. 1) and $y^2 = [0 \quad 1]$, respectively. If we randomly rotate y^1 and y^2 clockwise or counterclockwise, the distance between y^1 and y^2 remains the same. Therefore, we add rotation to (7) to obtain more flexibilities. R^{2}SDH's objective function is

$$\min_{B,F,W,R} \sum_{i=1}^{n} g\left((YR - BW)^i\right) + \lambda \|W\|_F^2 + \upsilon \|B - F(X)\|_F^2 \quad (9)$$
$$s.t.\ B \in \{-1,1\}^{n\times l}, RR^T = I,$$

where R is a rotation matrix.

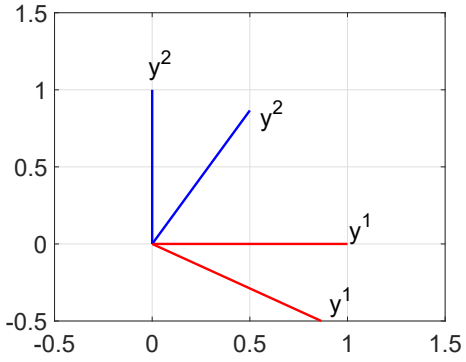

Figure 1: Random rotation of y^1 and y^2.

3.3 Optimization

The problem formulated in (9) is a mixed binary integer program with five unknown variables. Alternating optimization is adopted to solve the problem iteratively.

D-step: Based on half-quadratic optimization [12], the problem (9) can be solved in an alternate minimization way as follows :

$$D_i = \exp\left(-\left\|(YR - BW)^i\right\|_2^2 \Big/ \sigma^2\right), \tag{10}$$

$$\sigma^2 = \theta(YR - BW)^i\left((YR - BW)^i\right)^T \Big/ c \tag{11}$$

where θ is a tuning parameter and we recall c is the number of classes and $(YR - BW)^i$ is the i-th row of $(YR - BW)$.

In the next step, we need to solve the following program:

$$\begin{aligned} \min_{B,F,W,R}\ & tr\left((YR - BW)^T D\,(YR - BW)\right) \\ & +\lambda\left\|W\right\|_F^2 + \upsilon\left\|B - F\,(X)\right\|_F^2 \\ s.t.\ & B \in \{-1,1\}^{n\times l}, RR^T = I, \end{aligned} \tag{12}$$

where $tr\,()$ is the trace norm and D is a diagonal matrix and the i-th diagonal element is D_i. The problem (12) can be solved with regard to B, F, W, and R, respectively, by fixing the other variables.

W-step: For fixed B, F, and R, W can be solved by setting the derivative of (12) to zero. Therefore, W has a closed-form solution as following:

$$W = \left(B^T DB + \lambda I\right)^{-1} B^T DYR. \tag{13}$$

R-step: For fixed B, F, and W, we rewrite (12) as

$$\begin{aligned} \min_R\ & \left\|D^{1/2}\,(YR - BW)\right\|_F^2 \\ s.t.\ & RR^T = I. \end{aligned} \tag{14}$$

Theorem 1. Suppose the singular value decomposition of $Y^T DBW$ is $U\Sigma V^T$, the closed form solution of R is $R = UV^T$.

Proof: The Lagrange function is

$$L(R,A) = \left\|D^{1/2}\,(YR - BW)\right\|_F^2 + tr\left(A\left(RR^T - I\right)\right), \tag{15}$$

which equals

$$\begin{aligned} L(R,A) &= tr\left(\left(R^T Y^T - W^T B^T\right) D\,(YR - BW)\right) \\ &\quad + tr\left(A\left(RR^T - I\right)\right) \\ &= tr\left(Y^T DYRR^T\right) - 2tr\left(W^T B^T DYR\right) \\ &\quad + tr\left(R^T AR\right) + const, \end{aligned} \tag{16}$$

where $A \in \mathbb{R}^{c\times c}$ is the Lagrange multiplier and is a symmetric matrix. Since $RR^T = R^T R = I$, (16) is equal to

$$L(R,A) = -2tr\left(W^T B^T DYR\right) + tr\left(R^T AR\right) + const. \tag{17}$$

By setting the derivative of $L(R,A)$ with respect to R to zero, we have

$$\frac{\partial L(R,A)}{R} = -Y^T DBW + AR = 0. \tag{18}$$

Therefore, we get

$$R = A^{-1} Y^T DBW. \tag{19}$$

Since $RR^T = I$, we obtain

$$A^{-1} Y^T DBWW^T B^T DYA^{-1} = I. \tag{20}$$

Thus, we have

$$A = \left(Y^T DBW\left(Y^T DBW\right)^T\right)^{1/2}. \tag{21}$$

By substituting (21) in (19), we obtain

$$\begin{aligned} R &= \left(Y^T DBW\left(Y^T DBW\right)^T\right)^{-1/2} Y^T DBW \\ &= \left(U\Sigma V^T V\Sigma U^T\right)^{-1/2} U\Sigma V^T \\ &= UV^T. \end{aligned} \tag{22}$$

This completes the proof. □

B-step: In order to solve B, (12) can be rewritten as:

$$\begin{aligned} \min_B\ & tr\left((YR - BW)^T D\,(YR - BW)\right) + \upsilon\left\|B - F\,(X)\right\|_F^2 \\ & s.t.\ B \in \{-1,1\}^{n\times l} \end{aligned} \tag{23}$$

which can be equivalently written as

$$\begin{aligned} \min_B\ & tr\left(W^T B^T DBW\right) - 2tr(B^T M) \\ & s.t.\ B \in \{-1,1\}^{n\times l}, \end{aligned} \tag{24}$$

where $M = DYRW^T + \upsilon F\,(X)$. Like in SDH [37], we also use the discrete cyclic coordinate descent method is to solve B. Due to the D matrix in the formulation, we can not simply use the result in [37] via a straightforward matrix transformation.

Let b^T be the mth column of B, $m = 1, \cdots, l$ and B' the matrix of B excluding b^T, w be the mth row of W and W' the matrix of W excluding w. Similarly, let q the m column of M and M' the matrix M excluding q. Then we have

$$\begin{aligned} & tr\left(W^T B^T DBW\right) \\ &= tr\left(\left((W')^T (B')^T + w^T b\right) D\left(B'W' + b^T w\right)\right) \\ &= const + tr\left(w^T bDb^T w\right) + 2tr\left(bDB'W'w^T\right). \end{aligned} \tag{25}$$

Since $bDb^T = \sum_i D_i b_i^2 = \sum_i D_i$ where b_i is the i-th element of b, bDb^T is a constant.

Similarly, since $B^T M = \begin{bmatrix} (B')^T M' & (B')^T q \\ bM' & bq \end{bmatrix}$, we have

$$tr(B^T M) = const + tr(bq). \tag{26}$$

Therefore, (24) is equal to

$$\begin{aligned} &\min tr(b\left(DB'W'w^T - q\right)) \\ &s.t. \quad b \in \{-1, 1\}^{1\times l}. \end{aligned} \tag{27}$$

The solution to (27) is

$$b = sign\left(q - DB'W'w^T\right)^T. \tag{28}$$

F-step: This step is the same as that of SDH. The pseudocode of R^{2}SDH is shown in Algorithm 1.

Algorithm 1 Robust Rotated Supervised Discrete Hashing (R^{2}SDH)

Inputs: training data matrix $X = \{x_i\}_{i=1}^n$; label matrix $Y = \{y_i\}_{i=1}^n$; hash code length l; parameter λ; maximum iteration number t.
Output: hash codes $\{b_i\}_{i=1}^n \in \{-1, 1\}^{n\times l}$

Randomly select m instances $\{a_j\}_{i=1}^m$ from the training data and obtain the $\phi(x)$ through the Gaussian kernel function;
Initialize B as a $\{-1, 1\}^{n\times l}$ matrix randomly;
Initialize D and R as identity matrixes;
Use (13) to initialize W;
Use (5) to initialize P;
repeat
  D-step Use (10) to solve D;
  B-step Use discrete cyclic coordinate descent to solve B based on (28) ;
  W-step Use (13) to solve W;
  F-step Use (5) to solve P;
  R-step Use (22) to solve R;
until convergence

4 EXPERIMENTS

In this section, we investigate the performance of our proposed R^{2}SDH algorithm. All of our experiments have been conducted on a server with an Intel Xeon processor (2.80 GHz), 128 GB RAM, and configured with Microsoft Windows Server 2008 and MATLAB 2014b.

We conducted experiments using three large-scale image datasets, MNIST[1], CIFAR-10[2], and NUS-WIDE[3]. The proposed algorithm R^{2}SDH is compared with representative hashing algorithms such as BRE [15], SDHR[11], KSH [27], SSH [43], FastHash [23, 24], AGH [28], and IMH [38] with t-SNE [30]. For iterative quantization (ITQ) [9, 10], we utilize both its unsupervised version PCA-ITQ and supervised version CCA-ITQ. Canonical correlation analysis (CCA) is utilized as the preprocessing for CCA-ITQ. We utilize the public MATLAB codes and the parameters suggested by the authors. More specifically, for SDH and R^{2}SDH, λ and υ are empirically set to 1 and 1e-5, respectively; the maximum iteration number t is set to 5. The parameter θ in R^{2}SDH is estimated by 10-fold cross validation. For AGH, IMH, SDH, and R^{2}SDH, we use 1,000 randomly sampled anchor points.

The experimental results are reported based on Hamming ranking (the mean of average precision, MAP), accuracy, training time, and hash lookup including precision, recall, F-measure of Hamming radius two (@r =2). We set the Hamming radius r to be 2 as in [7, 37]. The F-measure's definition is 2×precision×recall/(precision + recall). The following evaluation metric is also used to measure the performance of the various methods: precision at N samples (precision@sample = N), which is defined as the percentage of true neighbors among the top N retrieved samples. We set N to 500 as in [7]. Note that a query is considered to be a false example if no instance is returned when calculating precisions. Ground truths are the example labels.

4.1 Results on the MNIST Dataset

MNIST has 70,000 784-dimensional handwritten digit images from '0' to '9'. Every image is cropped and normalized to 28×28 pixels. This dataset is split into a test set with 1,000 instances and a training set with all remaining instances. The experimental results on MNIST are listed in Table 2. R^{2}SDH outperforms the other methods in terms of precision@r=2, recall@r=2, F-measure@r=2, MAP, and accuracy, while PCA-ITQ performs best in terms of training time. However, R^{2}SDH's overall performance is much better than that of PCA-ITQ. For example, R^{2}SDH's F-measure@r=2 and MAP are more than four times and two times better than that of PCA-ITQ on this situation. Moreover, the precision@sample=500, precision of Hamming radius two, recall of Hamming radius two, F-measure of Hamming radius two, MAP, and accuracy curves are plotted in Figs. 2-7, respectively. Only some methods are presented due to space limitations. R^{2}SDH performs better than the other methods in most cases. Moreover, R^{2}SDH particularly performs better than the other methods in terms of precision@sample = 500.

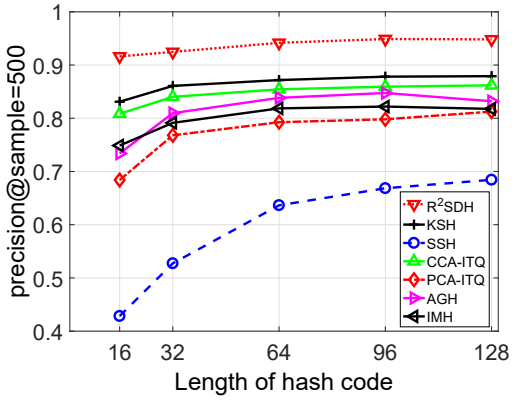

Figure 2: Precision@sample=500 on the MNIST data set with different number of hashing bits.

[1] http://yann.lecun.com/exdb/mnist/
[2] https://www.cs.toronto.edu/~kriz/cifar.html
[3] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

Table 2: Experimental results on the MNIST data set when the number of hashing bits is 16.

Method	precision@r=2	recall@r=2	F-measure@r=2	MAP	accuracy	training time (sec)
R^2SDH	**0.9373**	**0.8801**	**0.9078**	**0.9364**	**0.967**	52.3
SDH	0.8908	0.8569	0.8735	0.9276	0.925	39.1
SDHR	0.9301	0.8693	0.8987	0.9206	0.955	78.4
BRE	0.5741	0.1263	0.2071	0.3646	0.701	2884.7
KSH	0.8869	0.6287	0.7358	0.8670	0.906	318.8
SSH	0.3070	0.4235	0.3559	0.3569	0.525	68.6
CCA-ITQ	0.8136	0.4875	0.6097	0.7515	0.863	6.6
FastHash	0.5560	0.2310	0.3264	0.5253	0.588	1017.4
PCA-ITQ	0.6965	0.1107	0.1910	0.4260	0.732	**4.5**
AGH	0.6132	0.4712	0.5329	0.5995	0.785	6.8
IMH	0.6694	0.4238	0.5190	0.5996	0.813	32.1

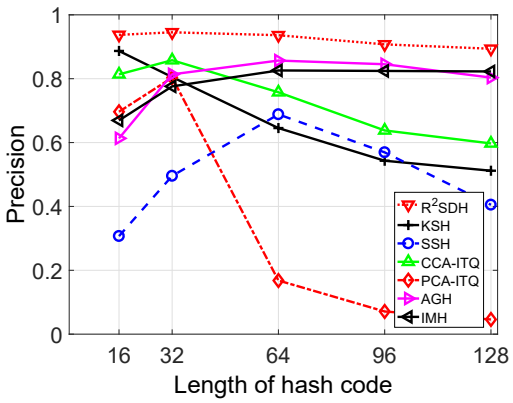

Figure 3: Precision of Hamming radius two on the MNIST data set with different number of hashing bits.

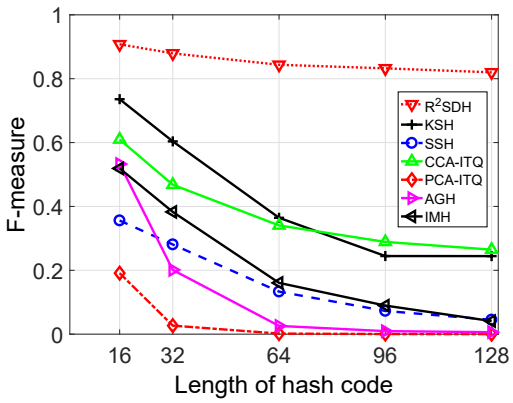

Figure 5: F-measure of Hamming radius two on the MNIST data set with different number of hashing bits.

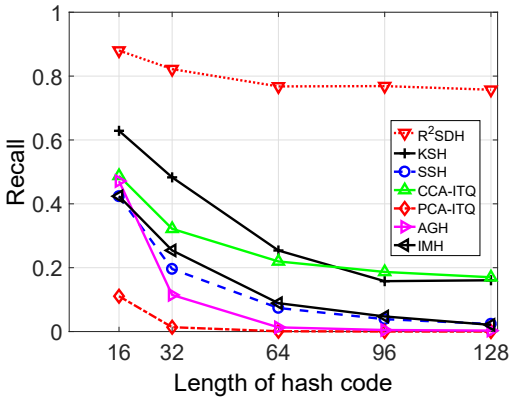

Figure 4: Recall of Hamming radius two on the MNIST data set with different number of hashing bits.

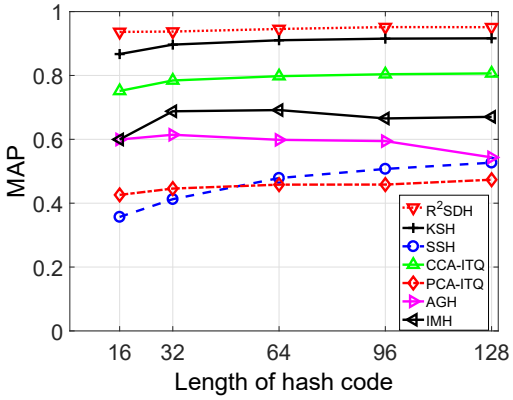

Figure 6: MAP on the MNIST data set with different number of hashing bits.

4.2 Results on the CIFAR-10 Dataset

As a subset of the celebrated 80M tiny image collection [42], there are 60,000 images on CIFAR-10 from 10 classes with 6,000 instances for every class. Every image in this dataset is represented as a 512-dimensional GIST feature vector [34]. The dataset is split into a training set with 59,000 instances and a test set with all remaining instances. The experimental results on CIFAR-10 are listed in

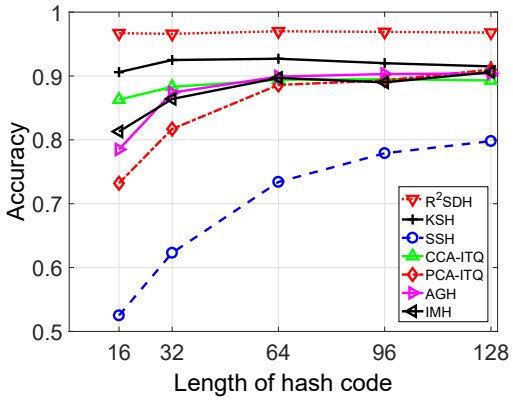

Figure 7: Accuracy on the MNIST data set with different number of hashing bits.

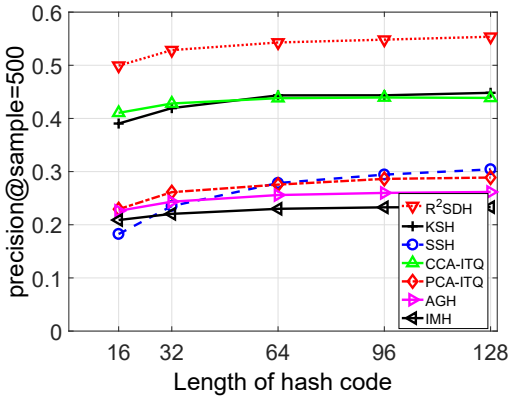

Figure 8: Precision@sample=500 on the CIFAR-10 data set with the number of hashing bits from 16 to 128.

Table 3 when the number of hashing bits is 32. Precision@r=2, recall@r=2, F-measure@r=2, MAP, accuracy, and training time, are reported. For SSH, 5,000 labeled examples are used for similarity matrix construction. R^{2}SDH performs better than SDH in terms of precision@r=2, recall@r=2, F-measure@r=2, MAP, and accuracy. For instance, the recall@r=2 of R^{2}SDH is 1.02-times higher than that of SDH. Furthermore, from the last column in Table 3, it can be seen that the training time of both R^{2}SDH and SDH are about one minute, which is quite efficient. In contrast, it takes FastHash and KSH about 20 minutes and more than 40 minutes to train, respectively. Specifically, the training of R^{2}SDH is about 16-times and 36-times more efficient than FastHash and KSH, respectively, on this situation. SDHR, SSH, CCA-ITQ, PCA-ITQ, AGH, and IMH are also very efficient; however, R^{2}SDH generally outperforms them. The precision at 500 instances (precision@sample = 500), precision@r=2, and accuracy versus the number of hashing bits are plotted in Figs. 8-10, respectively. Due to space limitations, only some algorithms are illustrated in the corresponding figure. In regard to precision@r=2, R^{2}SDH performs better than the other algorithms when the number of hashing bits is larger than 32, and KSH outperforms the other methods when the number of hashing bits is 16. R^{2}SDH performs the best according to precision at 500 instances (precision@sample = 500) and accuracy, demonstrating the effectiveness of R^{2}SDH.

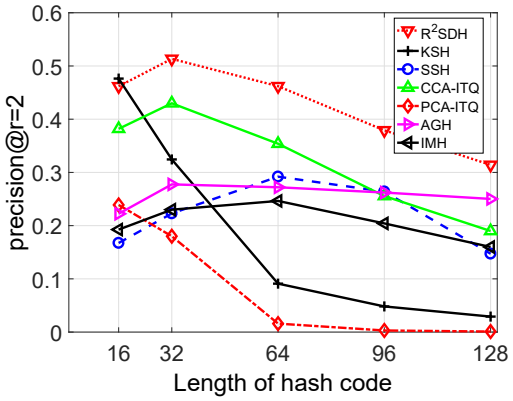

Figure 9: Precision of Hamming radius two on the CIFAR-10 data set with the number of hashing bits from 16 to 128.

4.3 Results on the NUS-WIDE Dataset

The NUS-WIDE data set has about 270,000 images collected from Flickr. NUS-WIDE contains 81 ground truth concept labels, with every image having multiple labels. The true neighbors of a query are defined as the images sharing at least one labels with the query image. We utilize the provided 500-dimensional bag-of-words features. As in [28], the 21 most frequent label are used for testing. For each class, 100 images are uniformly sampled for the query set and the remaining images are used for training. For this large database, all the training examples are utilized for the efficient R^{2}SDH, SDH, and CCA-ITQ. For BRE, MLH and KSH, 5,000 images are sampled for training. Experimental results are given in Table 4. Since SDHR can not solve multi-label learning and can only be used for

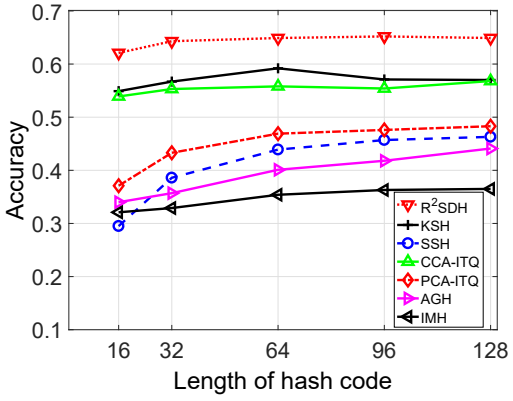

Figure 10: Accuracy on the CIFAR-10 data set with the number of hashing bits from 16 to 128.

Table 3: Experimental results on the CIFAR-10 data set when the number of hashing bits is 32.

Method	precision@r=2	recall@r=2	F-measure@r=2	MAP	accuracy	training time (sec)
R^2SDH	0.5134	**0.1622**	**0.2465**	0.4417	0.643	72.3
SDH	0.5121	0.1586	0.2422	0.4396	0.641	47.0
SDHR	**0.5169**	0.1445	0.2259	0.4408	0.627	61.6
BRE	0.2423	0.0122	0.0232	0.1505	0.353	232.4
KSH	0.3244	0.0312	0.0569	0.4421	0.567	2639.3
SSH	0.2229	0.1097	0.1470	0.1894	0.386	32.9
CCA-ITQ	0.4299	0.0426	0.0775	0.3330	0.553	7.1
FastHash	0.4358	0.0899	0.1490	**0.6009**	**0.657**	1200
PCA-ITQ	0.1801	4.8e-4	9.5e-4	0.1714	0.433	**4.8**
AGH	0.2775	0.0058	0.0113	0.1547	0.357	7.6
IMH	0.2300	0.0244	0.0441	0.1724	0.329	45.6

Table 4: Experimental results on the NUS-WIDE database when the number of hashing bits is 64.

Method	precision@r=2	recall@r=2	F-measure@r=2	MAP	training time (sec)
R^2SDH	0.3605	**0.9950**	**0.5293**	0.4918	890.7
SDH	**0.4957**	0.3025	0.3757	0.5609	593.8
BRE	0.0129	6.8375e-7	1.3674e-6	0.5022	41660.2
KSH	0.2528	0.0329	0.0582	**0.7334**	4142.7
SSH	0.4405	0.3713	0.4030	0.5786	114.4
CCA-ITQ	0.2839	8.0079e-4	0.0016	0.6127	46.1
FastHash	0.2043	0.0032	0.0062	0.5253	12564.7
PCA-ITQ	0.0944	0.0026	0.0051	0.5556	31.6
AGH	0.4500	0.0040	0.0080	0.5256	**21.6**
IMH	0.4434	0.0117	0.0228	0.5478	61.1

single-label learning, there is no result for SDHR in Table 4. R^2SDH performs the best in two evaluation metrics, demonstrating the effectiveness of our method on the retrieval task of multi-label data.

5 CONCLUSION

In this paper, we propose a data-dependent (learning-based) hashing algorithm named "robust rotated supervised discrete hashing" (R^2SDH) by extending "supervised discrete hashing" (SDH). R^2SDH used correntropy rather than least squares regression as in SDH. Thus, R^2SDH is more robust than SDH. Furthermore, R^2SDH used rotation for the label matrix, which offer additional flexibility. As expected, the R^2SDH's performance is better than that of SDH in most cases. Real-world image classification and digit recognition experiments demonstrate the effectiveness and efficiency of the proposed method.

Hashing algorithms can reduce computational costs, but the costs are still high for large-scale high-dimensional data. How to speed hashing algorithms such as R^2SDH is an issue worth studying. There are some possible ways, for instance, selecting representative instances to represent each class rather than using all instances.

ACKNOWLEDGEMENT

Jie Gui was supported by NSFC-61572463, NSF-Bigdata-1419210 and NSF-III-1360971.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.
[2] Andrei Z. Broder. 1997. On the Resemblance and Containment of Documents. In *the Compression and Complexity of Sequences*. Positano, Italy, 21–29.
[3] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 1998. Min-Wise Independent Permutations. In *STOC*. Dallas, TX, 327–336.
[4] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the Web. In *WWW*. Santa Clara, CA, 1157 – 1166.
[5] Guoqing Chao and Shiliang Sun. 2016. Alternative Multiview Maximum Entropy Discrimination. *IEEE Transactions on Neural Networks and Learning Systems* 27, 7 (2016), 1445–1456.
[6] Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*. Montreal, Canada, 380–388.
[7] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *Conference on Computer Vision and Pattern Recognition*. 2475–2483.
[8] Michel X. Goemans and David P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite Programming. *Journal of ACM* 42, 6 (1995), 1115–1145.
[9] Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Conference on Computer Vision and Pattern Recognition*. 817–824.
[10] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929.
[11] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. 2018. Supervised Discrete Hashing with Relaxation. *IEEE Transactions on Neural Networks and Learning Systems* 29, 3 (2018), 608–617.
[12] Ran He, Tieniu Tan, Liang Wang, and Wei-Shi Zheng. 2012. $l_{2,1}$ Regularized correntropy for robust feature selection. In *Conference on Computer Vision and Pattern Recognition*. 2504–2511.
[13] Sergey Ioffe. 2010. Improved Consistent Sampling, Weighted Minhash and L1 Sketching. In *ICDM*. Sydney, AU, 246–255.

[14] Qing-Yuan Jiang and Wu-Jun Li. 2015. Scalable graph hashing with feature transformation. In *International Joint Conference on Artificial Intelligence*. 2248–2254.

[15] Brian Kulis and Trevor Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *Neural Information Processing Systems*. 1042–1050.

[16] Shaishav Kumar and Raghavendra Udupa. 2011. Learning hash functions for cross-view similarity search. In *International Joint Conference on Artificial Intelligence*. 1360–1365.

[17] Ping Li. 2015. 0-Bit Consistent Weighted Sampling. In *KDD*. Sydney, Australia, 665–674.

[18] Ping Li. 2017. Linearized GMM Kernels and Normalized Random Fourier Features. In *KDD*. 315–324.

[19] Ping Li and Arnd Christian König. 2010. b-Bit Minwise Hashing. In *WWW*. Raleigh, NC, 671–680.

[20] Ping Li, Michael Mitzenmacher, and Anshumali Shrivastava. 2014. Coding for Random Projections. In *ICML*.

[21] Ping Li, Art B Owen, and Cun-Hui Zhang. 2012. One Permutation Hashing. In *NIPS*. Lake Tahoe, NV.

[22] Ping Li, Gennady Samorodnitsky, and John Hopcroft. 2013. Sign Cauchy Projections and Chi-Square Kernel. In *NIPS*. Lake Tahoe, NV.

[23] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *Conference on Computer Vision and Pattern Recognition*. 1963–1970.

[24] Guosheng Lin, Chunhua Shen, and Anton van den Hengel. 2015. Supervised hashing using graph cuts and boosted decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 11 (2015), 2317–2331.

[25] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep Supervised Hashing for Fast Image Retrieval. In *Conference on Computer Vision and Pattern Recognition*. 2064–2072.

[26] Weifeng Liu, Puskal P Pokharel, and José C Príncipe. 2007. Correntropy: properties and applications in non-Gaussian signal processing. *IEEE Transactions on Signal Processing* 55, 11 (2007), 5286–5298.

[27] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Conference on Computer Vision and Pattern Recognition*. 2074–2081.

[28] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with graphs. In *International Conference on Machine Learning*. 1–8.

[29] Yadan Luo, Yang Yang, Fumin Shen, Zi Huang, Pan Zhou, and Heng Tao Shen. 2018. Robust discrete code modeling for supervised hashing. *Pattern Recognition* 75 (2018), 128–135.

[30] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[31] Mark Manasse, Frank McSherry, and Kunal Talwar. 2010. *Consistent Weighted Sampling*. Technical Report MSR-TR-2010-73. Microsoft Research.

[32] Feiping Nie, Heng Huang, Xiao Cai, and Chris Ding. 2010. Efficient and robust feature selection via joint $l_{2,1}$-norms minimization. In *Neural Information Processing Systems*. 1813–1821.

[33] Mohammad Norouzi, David J Fleet, and Ruslan R Salakhutdinov. 2012. Hamming distance metric learning. In *Neural Information Processing Systems*. 1061–1069.

[34] Aude Oliva and Antonio Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision* 42, 3 (2001), 145–175.

[35] Mohammad Rastegari, Jonghyun Choi, Shobeir Fakhraei, Hal Daumé III, and Larry S Davis. 2013. Predictable dual-View hashing. In *International Conference on Machine Learning*. 1328–1336.

[36] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[37] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *Conference on Computer Vision and Pattern Recognition*. 37–45.

[38] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton Van Den Hengel, and Zhenmin Tang. 2013. Inductive hashing on manifolds. In *Conference on Computer Vision and Pattern Recognition*. 1562–1569.

[39] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. 2011. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM International Conference on Multimedia*. 423–432.

[40] Ryan Spring and Anshumali Shrivastava. 2017. Scalable and sustainable deep learning via randomized hashing. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 445–454.

[41] Christoph Strecha, Alex Bronstein, Michael Bronstein, and Pascal Fua. 2012. LDAHash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 1 (2012), 66–78.

[42] Antonio Torralba, Rob Fergus, and William T Freeman. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 11 (2008), 1958–1970.

[43] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 12 (2012), 2393–2406.

[44] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Neural Information Processing Systems*. 1753–1760.

[45] Dan Zhang, Fei Wang, and Luo Si. 2011. Composite hashing with multiple information sources. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 225–234.