Binary Code Learning via Iterative Distance Adjustment

Zhen-fei Ju, Xiao-jiao Mao, Ning Li, and Yu-bin Yang

State Key Laboratory for Novel Software Technology,
Nanjing University, 210093, China
yangyubin@nju.edu.cn

Abstract. Binary code learning techniques have recently been actively studied for hashing based nearest neighbor search in computer vision applications due to its merit of improving hashing performance. Currently, hashing based methods can obtain good binary codes but some data may suffer from the problem of being mapped to inappropriate Hamming codes. To address this issue, this paper proposes a novel binary code learning method via iterative distance adjustment to improve traditional hashing methods, in which we utilize very short additional binary bits to correct the spatial relationship among data points and thus enhance the similarity-preserving power of binary codes. We carry out image retrieval experiments on the well-recognized benchmark datasets to validate the proposed method. The experimental results have shown that the proposed method achieves better hashing performance than the state-of-the-art binary code learning methods.

Keywords: Hashing, nearest neighbor search, binary codes, iterative distance adjustment.

1 Introduction

Efficient similarity search in large image databases is a significant challenge in many computer vision applications. Nearest neighbor search(NNS), usually adopted to find similar objects, is one of the core technical issue involved in these applications. NNS is also a fundamental problem in data mining, machine learning and computer vision. However, the traditional way of searching nearest neighbors by scanning all the data has a linear time complexity, which is very inefficient and expensive for large databases. Fortunately in many applications, it is acceptably sufficient to return approximated nearest neighbors(ANN) instead of the exact ones.

Tree-based methods, such as KD-tree [4], metric tree [19], have been widely studied and used for NNS. These techniques attempt to decrease the complexity of nearest neighbor search, but they may degrade into linear search in the worst case [10]. Unfortunately, these technologies may not be appropriate for high dimensional data because the construction of tree structure is time-consuming and requires large memory space. Hence, hashing based techniques have been actively studied for mapping data to compact binary codes, which are then

X. He et al. (Eds.): MMM 2015, Part I, LNCS 8935, pp. 83–94, 2015.

used to establish hash tables for large databases efficiently. Since the Hamming distance between two binary codes can be computed via exclusive OR operation which is extremely fast, compact binary codes are particularly appropriate for approximating nearest neighbor search.

Hashing based methods usually preserve data similarity when mapping data points to the Hamming space appropriately. Current existing hashing based techniques can be divided into three categories based on their learning strategy: 1)unsupervised, 2)semi-supervised and 3)supervised. For unsupervised methods, Locality-Sensitive Hashing (LSH) [5] [1], which maps data to a low dimensional Hamming space via random projection, is a basic but widely used hashing based technique. After that, many other unsupervised methods have been proposed, such as Spectral Hashing SH [21], KLSH [9], ITQ [6] and AGH [12]. For semi-supervised methods, SSH [20] and Weakly-Supervised Hashing [14] are typical representatives. For supervised hashing including semantic hashing (RBM) [17], BRE [8], MLH [15] and others, they improve hashing performance by incorporating supervised information. Recently, a number of optimized hashing methods have also been proposed, such as Weighted Hamming Ranking [22], Hash Bit Selection [13] and JSD [2]. These methods further improve hashing performance based on the pre-existing methods.

In this paper, we present an iterative distance adjusting method to improve the similarity preserving power of binary codes. Motivated by the optimized methods, we propose a hashing method by correcting the Hamming spatial relationship between the base codes generated by other hashing methods. Although the traditional hashing-based method can obtain good hashing performance,there are still some data mapped to inappropriate Hamming codes. To correct these errors, we introduce a proper adjustment on the Hamming distance between the inaccurate binary code pairs. Moreover, aiming to overcome the inefficiency of building a binary code pool which consists of a number of many unnecessary codes, we resort to providing some additional bits for the base codes to enhance hashing performance, rather than selecting good bits from a binary code pool [13] [2]. It also guarantees that our method doesn't need any redundant candidate codes. An iterative distance adjustment step, which shrinks the Hamming distance of neighboring data and increase the Hamming distance of non-neighbor data, is adopted to reduce the Hamming approximation error between the Euclidean space and the Hamming space. Our method achieves good performance and outperforms several state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 provides a brief introduction to the related work. Section 3 describes the details of our method. The experimental results are then provided in Section 4. Finally, Section 5 provides concluding remarks.

2 Related Work

In this section we present some existing studies related to hash-based nearest neighbor search techniques. Given a dataset $X = [x_1, x_2, \ldots, x_n], x_i \in \mathbb{R}^d$, nearest neighbor search aims to find the nearest neighbors for a query q. The objective of

hash-based NNS methods is mapping data to the Hamming space while appropriately preserving the relative distances among them.

2.1 Unsupervised Hashing

LSH [5], a basic but widely used hashing technique, maps data to a low dimensional Hamming space via random projection. The main idea of LSH is mapping similar data to the same hash bucket with a high probability. A typical LSH function is denoted as:

$$h(x) = sign(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{1}$$

where $\boldsymbol{w}$ is a hyperplane and b is a random intercept. $\boldsymbol{w}$ is usually randomly sampled from a specific distribution, such as *p-stable* distribution. KLSH (Kernelized LSH) [9]incorporates LSH with kernel learning so that it can generalize similarity search from standard metric space to kernel space. Since hyperplane $\boldsymbol{w}$ is independently sampled to data, the number of hash bits may be large in order to maintain the original distances.

Motivated by spectral graph partition, Spectral Hashing improves performance over LSH, especially for compact bit lengths. But it has an assumption of underlying distribution of data. Iterative Quantization (ITQ) is motivated by the idea of data rotation to minimize quantization loss. ITQ explains that if the hyperplane coefficients $\boldsymbol{W}$ is an optimal solution, so is $\widetilde{\boldsymbol{W}} = \boldsymbol{WR}$ for any orthogonal matrix R. The goal of ITQ is to minimize the quantization error of mapping data to the vertices of a zero-centered binary hypercube.

Anchor Graph Hashing(AGH) [12] utilizes Anchor Graphs to obtain tractable low-rank adjacency matrices. The hash functions of AGH are learned by thresholding the lower eigenfunctions of the Anchor Graph Laplacian. Meanwhile, a hierarchical hashing method are further proposed to overcome the issue that neighboring points close to the boundary are assigned to different bits due to inappropriate thresholding.

2.2 Supervised Hashing

Supervised hashing methods such as RBM [17], BRE [8] and MLH [15] have shown higher search accuracy than unsupervised ones, but they are more difficult to be optimized and slower to be trained. Semantic Hashing (RBM) leverages Restricted Boltzmann Machines [3] to construct a multi-layer autoencoder to encode a low dimensional binary code. Binary reconstructive embedding (BRE) makes a loss function that penalizes the squared error between the original distances and the reconstructed distances. Since the objective function is non-convex, a coordinate-descent algorithm is proposed for optimization. Minimal Loss Hashing (MLH) advocates a loss function similar to hinge loss used in the SVMs to learn binary hash functions.

2.3 Optimized Hashing

Some hashing methods have been proposed to improve performance of binary codes by optimizing the pre-existing ones. [22] explained that there are often

lots of results sharing the same Hamming distance to a query, which makes this distance measure ambiguous. They proposed a weighted Hamming distance ranking algorithm to rank the generated binary codes so that it can differentiate the ambiguous codes. The algorithm learns both data-adaptive and query-sensitive weight for each hash bit. Hash Bit Selection [13] builds a large pool of over-complete hash bits encoded by various hashing methods with different features. Then, good hash bits are selected through two criteria: 1)similarity preserving and 2)independence. JSD [2] seeks a set of hash functions that minimizes the total probability of Bayes decision errors. A sequential learning algorithm based on LSH [5] [1] is also provided to obtain the projection minimizing Bayes decision errors from the candidate projections.

3 Correctional Hashing

3.1 Formulation

It has been well investigated in [13] [2] [22] that some bits learned by the existing hashing methods usually carry little or redundant information. To deal with this issue, they either selected good project directions from a candidate projection pool, or set different weights to different bits [22]. To build a candidate projection pool, many candidate binary codes are needed to be learned, which is very inefficient. In this paper, we address this problem in a totally different manner by investigating whether it is possible to improve the similarity preserving power of the codes by providing some additional bits.

Learning compact binary codes has been commonly treated as a solution to similarity-preserving problem, i.e. maping similar data points to similar binary codes in hashing. But there still exists a gap between the distances among binary codes in Hamming space and the distances among features in Euclidean space. In traditional binary learning methods, most data can be mapped to proper binary codes, but some of them are usually mapped to inappropriate ones, which are either neighboring data in the original space mapping to distant codes in Hamming space, or distant data in the original space mapping to neighboring codes in Hamming space. We design an algorithm to correct such kind of error via providing very compact additional bits. This adjustment can reduce the approximating error between Euclidean space and Hamming space, with which, the learned codes may achieve higher similarity preserving power and the hashing performance can be consequently improved.

Afterwards, we propose a novel method to correct the spatial relationship between pairwise binary codes iteratively. At each iteration, additional binary bits are introduced to minimize the Hamming approximating error. The basic principle is that the Hamming distance between the points which have been assigned to inaccurate base codes can be modified and corrected through additional bits. In a word, among the inaccurate pairs of binary codes, similar additional bits are used to shrink the distance between the neighboring pairs in the original

space, while opposite additional bits are adopted to increase the distance between the distant pairs in the original space. A Hamming approximating error function is defined to determine which pair of binary codes is inaccurate. The Hamming approximating error is minimized iteratively by learning the additional bits incrementally. Specifically, a few bits are learned for the base codes in each iteration and they are concatenated as the new base codes in next iteration.

3.2 Iterative Distance Adjustment

Given a dataset of n points, denoted as $X = [x_1, x_2, \ldots, x_n], x_i \in \mathbb{R}^d$, the objective of hashing method is to learn a hash function family to map X to a binary matrix $B \in \{-1, 1\}^{n \times b}$, where b denotes the length of a binary code, and kth hash function of binary bit is defined as

$$h_k(\boldsymbol{x_i}) = sgn(\boldsymbol{w}_k^\top \boldsymbol{x_i} + b_k). \tag{2}$$

Let $\boldsymbol{H} = [h_1, h_2, \ldots, h_k]$ be a sequence of k hash functions and $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k] \in \mathcal{R}^{d \times k}$. Without loss of generality, let $\boldsymbol{X}$ be normalized to have zero mean. To make hash bit carry as much information as possible, we should balance the hash function to meet $\sum_{i=1}^{n} H_k(\boldsymbol{x}_i) = 0$. Like [11], we denote b_k as the median of $\{\boldsymbol{w}_k^\top \boldsymbol{x}_i\}_{i=1}^n$. By further choosing a fast alternative to the median, we calculate the mean as $b = \sum_{i=1}^{n} \boldsymbol{w}_k^\top \boldsymbol{x}_i / n$. We have $b = 0$ because $\boldsymbol{X}$ is zero-mean.

Given a set of base binary codes $B^{base} \in \{-1, 1\}^{n \times b}$, to adjust the Hamming distance between base codes, our objective is to learn the corresponding additional codes $B^{add} \in \{-1, 1\}^{n \times k}$. The result of final binary matrix is then denoted as $B \in \{-1, 1\}^{n \times (b+k)}$. We learn t bits at each iteration, thus the number of iterations is b/t. We use linear projection mentioned earlier as the hash function for additional codes.

3.3 Learning Additional Binary Codes

First of all, the pairs of base binary codes needed to be corrected should be selected out. Similar to K-means Hashing [7], we choose minimizing the Hamming approximation error as the objective function,which is defined as:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} (d(\boldsymbol{x}_i, \boldsymbol{x}_j) - \lambda d_h(\boldsymbol{x}_i, \boldsymbol{x}_j))^2 \tag{3}$$

This equation minimizes the difference between two n-*by*-n affinity matrices $d(\boldsymbol{x_i}, \boldsymbol{x_j})$ and $d_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the original Euclidean distance between two data points, and $d_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the Hamming distance between two hashing binary codes. We introduce λ here because the Euclidean distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be in arbitrary ranges, while the Hamming distance $d_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is constrained in the range of [0,b] for b bits.

Based on the objective function, we define the Hamming approximating error to distinguish inaccurate pairs which need corrections as:

$$Err_{Ha} = (d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_h(\boldsymbol{x}_i, \boldsymbol{x}_j))^2 \tag{4}$$

In our work, λ has been found hard to determine. Therefore we normalize the two matrices $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $d_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as a compromise. θ is a threshold set to find the binary codes needing correction.

We adopt an intuitive way to learn additional bits, and update a small number of bits at each iteration. We split the inaccurate binary code pairs into two categories: 1) neighbor in the original space, and 2) non-neighbor in the original space. Specifically, a pair $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{N}$ is denoted as a neighboring pair in Euclidean space, and $\mathcal{D}$ is the set of non-neighboring pairs in Euclidean space. In order to correct the spatial relationship of inaccurate binary codes in Hamming space, the Hamming distance between neighboring pairs should be reduced and the Hamming distance between non-neighboring pairs should be increased. We manage to learn a $\boldsymbol{W}$ which provides similar additional bits to the neighboring pairs and different additional bits to the non-neighbor pairs. An objective function measuring the empirical accuracy for the additional hashing functions $[h_1, h_2, \ldots, h_t]$ at each iteration can be defined as:

$$J(\boldsymbol{H}) = \sum_t \left\{ \sum_{(\boldsymbol{x_i}, \boldsymbol{x_j}) \in \mathcal{N}} h_t(\boldsymbol{x_i}) h_t(\boldsymbol{x_j}) - \sum_{(\boldsymbol{x_i}, \boldsymbol{x_j}) \in \mathcal{D}} h_t(\boldsymbol{x_i}) h_t(\boldsymbol{x_j}) \right\}, \tag{5}$$

where t is the length of bits updated at each iteration. We need k/s iterations to learn all the additional projection directions.

Then, we define a matrix $S \in \mathbb{R}^{n \times n}$ to incorporate pairwise Hamming approximation error and the original distance as

$$\boldsymbol{S}_{ij} = \begin{cases} 1 & : \quad Err_{Ha} > \theta, (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{N} \\ -1 & : \quad Err_{Ha} > \theta, (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D} \\ 0 & : \quad Err_{Ha} \leq \theta \end{cases} \tag{6}$$

Suppose $H(\boldsymbol{X}) \in \mathbb{R}^{t \times n}$ maps the points in $\boldsymbol{X}$ to t-bit hash codes at each iteration. Then the objective function can be presented as

$$J(\boldsymbol{H}) = \frac{1}{2} tr\{H(\boldsymbol{X}) \boldsymbol{S} H(\boldsymbol{X})^\top\} \tag{7}$$

that is

$$J(\boldsymbol{W}) = \frac{1}{2} tr\{sgn(\boldsymbol{W}^\top \boldsymbol{X}) \boldsymbol{S} sgn(\boldsymbol{W}^\top \boldsymbol{X})^\top\} \tag{8}$$

where $X = [x_1, x_2, \ldots, x_n], x_i \in \mathbb{R}^d$. We learn the optimal hyperplane matrix $\boldsymbol{W}$ by maximizing objective function $J(\boldsymbol{W})$. Since $J(\boldsymbol{W})$ is nondifferentiable, the above problem is difficult to be solved. We present an intuitive relaxation by replacing the sign of projection with its signed magnitude in Eqn.(7).

$$J(\boldsymbol{H}) = \sum_t \left\{ \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{N}} \boldsymbol{w}_t^\top \boldsymbol{x}_i \boldsymbol{x}_j{}^\top \boldsymbol{w}_t - \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \lceil} \boldsymbol{w}_t^\top \boldsymbol{x}_i \boldsymbol{x}_j{}^\top \boldsymbol{w}_t \right\}, \tag{9}$$

The above function $J(\boldsymbol{W})$ can be represented in a matrix form:

$$J(\boldsymbol{W}) = \frac{1}{2} tr\{\boldsymbol{W}^\top \boldsymbol{X}_k \boldsymbol{S} \boldsymbol{X}^\top \boldsymbol{W}\} \tag{10}$$

Algorithm 1. Correctional Hashing

Input:

The set of training sample, $\boldsymbol{X} = \boldsymbol{x}_i \in \mathbb{R}^{d}{}_{i=1}^{n}$; The set of base binary codes of training set, $\boldsymbol{B}^{base} \in \mathbb{R}^{n \times b}$; The length of additional correctional bits, k; The length of bits updated at each iteration, t; The threshold defining "neighbor" in Euclidean space, θ_{eu}; The threshold defining the Hamming approximating error, θ_{ha}

for $i = 1, \ldots, k/t$ **do**
  $\boldsymbol{B}^{add} \leftarrow \boldsymbol{B}^{base}$
  for $j = 1, \ldots, n$ **do**
    for $l = 1, \ldots, n$ **do**
      $Err_{Ha} \Leftarrow (d(\boldsymbol{x}_j, \boldsymbol{x}_l) - d_h(\boldsymbol{B}_j^{add}, \boldsymbol{B}_l^{add}))^2$
      if $Err_{Ha} \leq \theta_{ha}$ **then**
        $\boldsymbol{S}_{jl} \Leftarrow 0$
      else
        if $d(\boldsymbol{x}_j, \boldsymbol{x}_k) \leq \theta_{eu}$ **then**
          $\boldsymbol{S}_{jl} \Leftarrow 1$
        else
          $\boldsymbol{S}_{jl} \Leftarrow -1$
        end if
      end if
    end for
  end for
  $\boldsymbol{M} \leftarrow \boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^{\top}$;
  $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_t] \leftarrow$ the eigenvectors corresponding to top-t eigenvalues of matrix $\boldsymbol{M}$;
  $\boldsymbol{B}^{new} \leftarrow \boldsymbol{W}^{\top}\boldsymbol{X}$;
  $\boldsymbol{B}^{add} \leftarrow \boldsymbol{B}^{add}$ attach $\boldsymbol{B}^{new}, \boldsymbol{B}^{add} \in \mathbb{R}^{n \times (b+t)}$;
end for

Output:

n hash codes $\boldsymbol{B}^{add} \in \mathbb{R}^{n \times (b+k)}$;

Then, the objective function is solved by using eigenvalue decomposition on matrix $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^{\top}$:

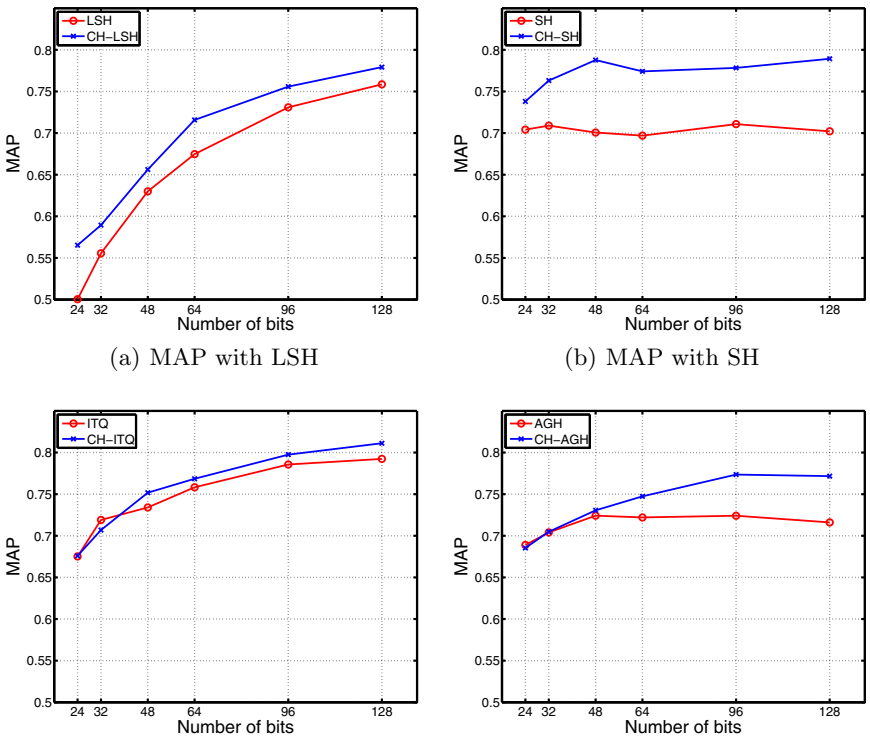
$$\max_{\boldsymbol{W}} J(\boldsymbol{W}) = \sum_{s=1}^{t} \lambda_s \tag{11}$$

where $[\lambda_1, \lambda_2, \ldots, \lambda_t]$ are the top-t eigenvalues of $\boldsymbol{M}$, and $\boldsymbol{w}_k$ are the corresponding eigenvectors.

4 Experiments

4.1 Datasets and Protocols

We evaluate our method on the well-recognized MNIST digit dataset and CIFAR-10 dataset, and make comparisons with several existing state-of-the-art hash-based method.



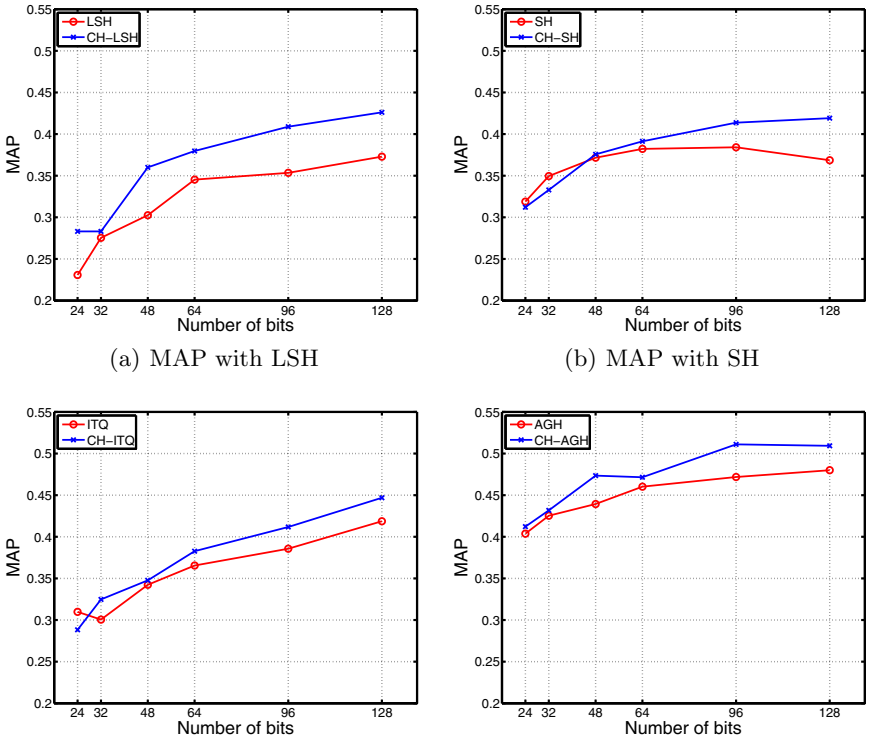
(a) MAP with LSH (b) MAP with SH

(c) MAP with ITQ (d) MAP with AGH

Fig. 1. Performance comparison of CH with SH, LSH, ITQ and AGH on MNIST dataset with Euclidean neighbourhood

MNIST is a well-known greyscale image dataset of handwritten digits consisting of 70,000 784-dimension digit samples from '0' to '9'. The original handwriting digit image size is 28×28 . CIFAR-10 dataset is a labeled subset of an 80-million tiny images collection [18]. It consists of 60,000 32×32 color images of 10 classes, each of which has 6,000 images. Each image in CIFAR-10 is represented by a 512-dimension GIST feature vector [16]. We randomly sample 2,000 images for training and 1000 images for testing for each dataset.

We evaluate our Correctional Hashing(CH) method by comparing the generated base binary codes with those of other four state-of-the-art methods, that is, LSH [1], SH [21], ITQ [6] and AGH [12]. For LSH, we randomly select projections from a Gaussian distribution with zero-mean to construct hash functions. To run AGH, we take the two-layer AGH(2-AGH) and fix the number of anchors as 300. The Correctional Hashing codes consists of base binary codes and additional codes as:



(a) MAP with LSH (b) MAP with SH

(c) MAP with ITQ (d) MAP with AGH

Fig. 2. Performance comparison of CH with SH, LSH, ITQ and AGH on CIFAR-10 dataset with Euclidean neighbourhood

Base Bits	16	24	32	48	64	96
Additional Bits	8	8	16	16	32	32
CH Bits	24	32	48	64	96	128

4.2 Result

We compare the codes generated by Correctional Hashing and the other four methods at the same length. The number of iterations is fixed as 2. The performance is measured by Mean Average Precision (MAP) defined as the mean precision rate of the Top n points in the ranked list of each testing query. All the points in the testing set are ranked according to the values of their Hamming distances to the query point, and top-50 points of the ranked list are taken to calculate MAP. The complexity of Hamming ranking is linear, but it is adequately fast for binary codes. We query all points in the testing set and list the mean precisions of them.

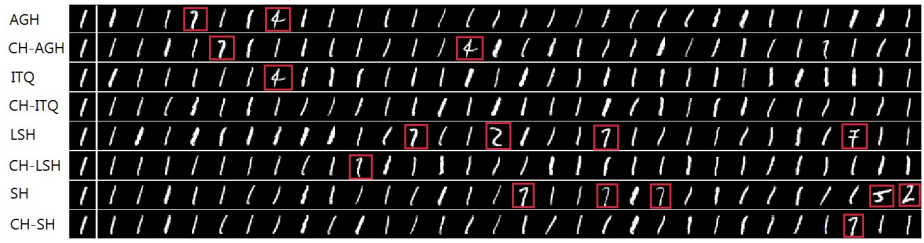


Fig. 3. The 30 nearest neighbors of an example digit output by all tested methods on the MNIST dataset. The images on the left is the query sample. From top to bottom, it shows the nearest neighbor samplesof AGH, CH-AGH, ITQ, CH-ITQ, LSH, CH-LSH, SH and CH-SH using 64-bit codes. The digit in red box is the incorrect result of the query.

Fig.1 and Fig.2 show the Hamming ranking performance measured by MAP of Correctional Hashing with LSH [1], SH [21], ITQ [6], AGH [12] on MNIST and CIFAR-10. The number of binary bits varies from 24 to 128. We take top 50 points to obtain the MAP of Hamming Ranking based on Euclidean neighborhood.

Our method adjusts the Hamming distance between inaccurate pairs so that the similarity preserving power of binary codes is enhanced. As shown in Fig.1 and Fig.2, our method outperforms other methods on generating base codes. But in some cases with short bits, our method didn't perform well, which may caused by the mediocre performance of its base binary codes. Overall, Correctional Hashing algorithm obtains higher MAP of Hamming ranking in all the tested methods.

Moreover, our method also obtains sound results when the ground-truth labels are used to evaluate semantic consistency of binary codes. We choose MNIST dataset to present the query result for the convenience of observing. The adjacent two rows are base codes learned by other hashing-based techniques and its corresponding codes learned by Correctional Hashing, respectively. As shown in Fig.3, our algorithm promotes the semantic consistency of binary code. The incorrect digits among the top-30 query results, such as digit 7 and 4 at first row, have been adjusted to latter position of the rank list. It is safe to make a conclusion that Correctional Hashing retrieves more accurate than the base methods and is capable of correcting some inaccurate binary codes.

5 Conclusion

This paper presents a novel binary code learning method via iterative distance adjustment. Considering the existence of inaccurate mapping between the data and the mapped Hamming positions, we propose to correct Hamming distance between inaccurate binary pairs to minimize Hamming approximating error. Experimental results show that the proposed method improves the hashing performance of the state-of-the-art techniques.

In the future, we will focus on learning the threshold of Hamming approximating error to achieve better hash performance.

Acknowledgment. This work is supported by the Program for New Century Excellent Talents of MOE China (Grant No.NCET-11-0213), the Natural Science Foundation of China (Grant Nos. 61273257, 61321491, 61035003), National 973 Program of China (Grant No.2010CB327903), the Program for Distinguished Talents of Jiangsu (Grant No. 2013-XXRJ-018), and the Scientific Research Foundation of Graduate School of Nanjing University (Grant No. 2014CL03).

References

1. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262. ACM (2004)
2. Fan, L.: Supervised binary hash code learning with jensen shannon divergence. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 2616–2623. IEEE (2013)
3. Freund, Y., Haussler, D.: Unsupervised learning of distributions of binary vectors using two layer networks. Computer Research Laboratory, University of California, Santa Cruz (1994)
4. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS) 3(3), 209–226 (1977)
5. Gionis, A., Indyk, P., Motwani, R., et al.: Similarity search in high dimensions via hashing. VLDB 99, 518–529 (1999)
6. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 817–824. IEEE (2011)
7. He, K., Wen, F., Sun, J.: K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2938–2945. IEEE (2013)
8. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: Advances in Neural Information Processing Systems, pp. 1042–1050 (2009)
9. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 2130–2137. IEEE (2009)
10. Liu, T., Moore, A.W., Yang, K., Gray, A.G.: An investigation of practical approximate nearest neighbor algorithms. In: Advances in Neural Information Processing Systems, pp. 825–832 (2004)
11. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2074–2081. IEEE (2012)
12. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 1–8 (2011)
13. Liu, X., He, J., Lang, B., Chang, S.F.: Hash bit selection: a unified solution for selection problems in hashing. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1570–1577. IEEE (2013)

14. Mu, Y., Shen, J., Yan, S.: Weakly-supervised hashing in kernel space. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3344–3351. IEEE (2010)
15. Norouzi, M., Blei, D.M.: Minimal loss hashing for compact binary codes. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 353–360 (2011)
16. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision 42(3), 145–175 (2001)
17. Salakhutdinov, R., Hinton, G.E.: Learning a nonlinear embedding by preserving class neighbourhood structure. In: International Conference on Artificial Intelligence and Statistics, pp. 412–419 (2007)
18. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(11), 1958–1970 (2008)
19. Uhlmann, J.K.: Satisfying general proximity/similarity queries with metric trees. Information Processing Letters 40(4), 175–179 (1991)
20. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3424–3431. IEEE (2010)
21. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in neural information processing systems, pp. 1753–1760 (2008)
22. Zhang, L., Zhang, Y., Tang, J., Lu, K., Tian, Q.: Binary code ranking with weighted hamming distance. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1586–1593. IEEE (2013)