上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目 基于哈希的大规模图像检索

学生姓名 孔维昊

学生学号 5090309041

指导教师 李武军

专　　业 计算机科学与工程

学院（系）电子信息与电气工程

基于哈希的大规模图像检索

摘　　要

哈希被用来学习数据的二进制码表达方法，以期望它能够保存原始特征空间的距离关系。因为它非常快的搜索速度和极少的存储空间，哈希已经被广泛的用于了在很多种不同的数据集，包括文本检索、图像检索，的高效近邻搜索上。因为通过解整数规划问题直接计算最优的二进制编码非常困难的，所以主流的哈希方法通常采用两步的策略。在第一步中，许多维数据被投影函数投影出来。然后在第二步中，那些实数值被通过截断的方法量子化到二进制串。在这篇论文中，我们在这两方面都作出了贡献。在第一步，一般不同投影维的方差是不同的，比如说主成份分析。使用相同的比特数给不同的数据维进行编码显然是不合理的，因为大方差的维度往往携带更多的信息。尽管这个观点已经被大多数研究者所接受了，它还没有被试验或理论证明，因为目前还没有方法能够保证找到使投影结果能够各个维度的方差相等。本文中，为了学习到能使得投影出的维度能具有等方性的投影函数，我们提出了一种新的哈希方法————等方性哈希。在真实数据集上的实验结果表明我们提出的等方性哈希能够超过许多其他的非等方性的哈希方法。这也证明了各向同性比各向异性方法要好。在第二步，目前大部分的哈希方法使用海明距离来度量哈希码间的距离。然而，海明距离的一个问题是它有可能破坏原空间上的距离关系，这违背了哈希的本来想法。本文提出了使用曼哈顿距离的曼哈顿哈希的方法来解决海明哈希的问题。曼哈顿哈希的基本思想是使用多比特的自然二进制编码来作为哈希码。在距离的计算上，曼哈顿哈希采用曼哈顿距离来度量哈希码之间的距

离，并用其来进行近邻搜索。曼哈顿哈希能够高效的保存数据中的近邻结构，来达到哈希的目的。就我们目前所知的，这是第一个使用曼哈顿距离和自然二进制来哈希的方法。试验表明我们的曼哈顿哈希能大幅度超过其他先进方法。

关键词： 哈希, 图像检索, 近似近邻搜索，等方性哈希，舒尔霍恩定理，逆特征值问题，海明距离，曼哈顿距离

Large-Scale Image Retrieval Based on Hashing

ABSTRACT

Hashing is used to learn binary-code representation for data with expectation of preserving the neighborhood structure in the original feature space. Due to its fast query speed and reduced storage cost, hashing has been widely used for efficient nearest neighbor search in a large variety of applications like text and image retrieval. Because it is usually hard to directly compute the best binary codes for a given data set as an integer programing problem, mainstream hashing methods typically adopt the two-stage strategy. In the first stage, several projected dimensions of real values are generated by projection functions. Then in the second stage, the real values will be quantized into binary codes by thresholding. In this paper, we also make contributions in these two parts. In the first stage, typically the variances of different projected dimensions are different for existing projection functions such as principle component analysis (PCA). Using the same number of bits for different projected dimensions is unreasonable because larger-variance dimensions will carry more information. Although this viewpoint has been widely accepted by many researchers, it is still not verified by either theory or experiment because no methods have been proposed to find a projection with equal variances for different dimensions. In this paper, we propose a novel method, called isotropic hashing (IsoHash), to learn projection functions which can produce projected dimensions with isotropic variances (equal variances). Experimental results on real data

sets show that IsoHash can outperform its counterpart with different variances for different dimensions, which verifies the viewpoint that projections with isotropic variances will be better than those with anisotropic variances. In the second stage, most existing hashing methods adopt Hamming distance to measure the similarity (neighborhood) between points in the hashcode space. However, one problem with Hamming distance is that it may destroy the neighborhood structure in the original feature space, which violates the essential goal of hashing. In this paper, Manhattan hashing (MH), which is based on Manhattan distance, is proposed to solve the problem of Hamming distance based hashing. The basic idea of MH is to encode each projected dimension with multiple bits of natural binary code (NBC), based on which the Manhattan distance between points in the hashcode space is calculated for nearest neighbor search. MH can effectively preserve the neighborhood structure in the data to achieve the goal of hashing. To the best of our knowledge, this is the first work to adopt Manhattan distance with NBC for hashing. Experiments show that our MH method can significantly outperform other state-of-the-art methods.

Keywords: Hashing, Image retrieval, Approximate nearest neighbor search, Isotropic variances, Schur-horn theorem, Inverse eigenvalue problem, Hamming distance, Manhattan distance

Contents

Chapter 1 Introduction

Nearest neighbor (NN) search [1] has been widely used in machine learning and related application areas, such as information retrieval, data mining, and computer vision. Recently, with the explosive growth of data on the Internet, there has been increasing interest in NN search in massive (large-scale) data sets. Traditional brute force NN search requires scanning all the points in a data set whose time complexity is linear to the sample size. Hence, it is computationally prohibitive to adopt brute force NN search for massive data sets which might contain millions or even billions of points. Another challenge facing NN search in massive data sets is the excessive storage cost which is typically unacceptable if traditional data formats are used.

To solve these problems, researchers have proposed to use hashing techniques for efficient approximate nearest neighbor (ANN) search [2–6]. The goal of hashing is to learn binary-code representation for data which can preserve the neighborhood (similarity) structure in the original feature space. The basic idea of hashing is to learn similarity-preserving binary codes for data representation. More specifically, each data point will be encoded as a compact binary string in the hashcode space, and similar points in the original feature space should be mapped to close points in the hashcode space. Compared with the original feature representation, hashing has two advantages. On one hand, we can achieve constant or sub-linear search time complexity [7]. On the other hand, the storage needed to store the binary codes will be dramatically reduced. For example, if each point is represented by a vector of 1024 bytes in the original space, a data set of 1 million points will cost 1G memory. On the contrary, if we hash each point into a vector of 128 bits, the memory needed to store the data set of 1 million points will be reduced to 16M. Therefore, hashing provides a very effective way to achieve fast query speed with low storage cost, which makes it a promising choice for efficient ANN search in massive data sets [2, 3, 5, 8–13].

To avoid the NP-hard solution which directly computes the best binary codes for a given data set [8], most existing hashing methods adopt a learning strategy containing

two stages: projection stage and quantization stage. In the projection stage, several projected dimensions of real values are generated. In the quantization stage, the real values generated from the projection stage are quantized into binary codes by thresholding. For example, the widely used single-bit quantization (SBQ) strategy adopts one single bit to quantize each projected dimension. More specifically, given a point $\mathbf{x}$ from the original space, each projected dimension i will be associated with a real-valued projection function $f_i(\mathbf{x})$. The ith hash bit of $\mathbf{x}$ will be 1 if $f_i(\mathbf{x}) \geq \theta$. Otherwise, it will be 0. Here, θ is a threshold, which is typically set to 0 if the data have been normalized to have zero mean. Although a lot of projection methods have been proposed for hashing, there exist only two quantization methods. One is the SBQ method stated above, and the other is the hierarchical quantization (HQ) method in anchor graph hashing (AGH) [13]. Rather than using one bit, HQ divides each dimension into four regions with three thresholds and uses two bits to encode each region. Hence, HQ will associate each projected dimension with two bits.

The main contributions of this paper are briefly outlined as follows:

- In this paper, a novel hashing method, called isotropic hashing (IsoHash), is proposed to learn a projection function which can produce projected dimensions with isotropic variances (equal variances). To the best of our knowledge, this is the first work which can learn projections with isotropic variances for hashing. Experimental results on real data sets show that IsoHash can outperform its counterpart with anisotropic variances for different dimensions, which verifies the intuitive viewpoint that projections with isotropic variances will be better than those with anisotropic variances. Furthermore, the performance of IsoHash is also comparable, if not superior, to the state-of-the-art methods.

- To further improve the performance of our hashing method, we proposed a novel quantization method called Manhattan Quantization(MQ) to encodes each projected dimension with multiple bits of *natural binary code* (NBC), based on which the Manhattan distance between points in the hashcode space is calculated for nearest neighbor search.

The rest of this paper is organized as follows. We introduce the related work in Chapter 2, Chapter 3 describes the details of our IsoHash method. Chapter 4 present the motivation and details of MQ. Experimental results are presented in Chapter 5. Finally, we conclude the whole paper in Chapter 6.

Chapter 2 Related Work

Due to the promising performance in terms of either speed or storage, hashing has been widely used for efficient ANN search in a large variety of applications with massive data sets, such as text retrieval [6, 14], image retrieval [15, 16], audio retrieval [17], and near-duplicate video retrieval [18]. As a consequence, many hashing methods have been proposed by researchers. In general, the existing methods can be roughly divided into two main classes [6, 15]: data-independent methods and data-dependent methods [1].

The most representative data-independent methods are locality-sensitive hashing (LSH) [2, 3] and its extensions [19–23]. These methods use simple random projections which are independent of the training data for hash functions. Shift invariant kernel hashing (SIKH) [23] chooses projection vectors similar to those of LSH, but SIKH uses a shifted cosine function to generate hash values. Both LSH and SIKH have an important property that points with high similarity in the original space will be most likely to have similar projected values. Many applications, such as image retrieval [23] and cross-language information retrieval [5], have adopted these data-independent hashing methods for ANN. Generally, data-independent methods need longer codes than data-dependent methods to achieve satisfactory performance [15]. Longer codes means higher storage and computational cost. Hence, the data-independent methods are less efficient than data-dependent methods.

More and more recent works have focused on data-dependent methods whose hash functions are learned from the training data. Semantic hashing [9, 24] adopts a deep generative model based on restricted Boltzmann machine (RBM) [25] to learn the hash functions. Experiments on text retrieval demonstrate that semantic hashing can achieve better performance than the original TF-IDF representation [26] and LSH. AdaBoost [27] is adopted by [17] to learn hash functions from weakly labeled positive

[1] In [6], data-independent is called data-oblivious while data-dependent is called data-aware. It is obvious that they have the same meanings.

samples. The resulting hashing method achieves better performance than LSH for audio retrieval. Spectral hashing (SH) [8] uses spectral graph partitioning for hashing with the graph constructed from the data similarity relationships. Binary reconstruction embedding (BRE) [10] learns the hash functions by explicitly minimizing the reconstruction error between the original distances and the Hamming distances of the corresponding binary codes. Anchor graph hashing (AGH) [13] adopts anchor graphs to speed up the computation of graph Laplacian eigenvectors, based on which the Nyström method is used to compute projection functions. Sequential projection learning (SPL) [11] leans the projection functions in a sequential way that each function is designed to correct the errors caused by the previous one. Principle component analysis [28] based hashing (PCAH) [15] adopts principle component analysis (PCA) to learn the projection functions. Semi-supervise hashing (SSH) [11] exploits some labeled data to help hash function learning. Self-taught hashing [6] uses supervised learning algorithms for hashing based on self-labeled data. Complementary hashing [29] exploits multiple complementary hash tables learned sequentially in a boosting manner to effectively balance the precision and recall. Composite hashing [5] integrates multiple information sources for hashing. Minimal loss hashing (MLH) [12] formulates the hashing problem as a structured prediction problem. Both accuracy and time are jointly optimized to learn the hash functions in [30]. Hypergraph hashing [31] extends SH to hypergraph to model the high-order relationships between social images. Active hashing [32] is proposed to actively select the most informative labels for hash function learning. One of the most recent data-dependent methods is iterative quantization (ITQ) [15] which finds an orthogonal rotation matrix to refine the initial projection matrix learned by principal component analysis (PCA) [28] so that the quantization error of mapping the data to the vertices of binary hypercube is minimized. Experimental results show that it can achieve better performance than most state-of-the-art methods.Compared to the data-dependent methods, the data-independent methods need longer codes to achieve satisfactory performance [15].

For most existing projection functions such as those mentioned above, the variances of different projected dimensions are different. Many researchers [13, 15, 33] have

argued that using the same number of bits for different projected dimensions with unequal variances is unreasonable because larger-variance dimensions will carry more information. Some methods [15, 33] use orthogonal transformation to the PCA-projected data with the expectation of balancing the variances of different PCA dimensions, and achieve better performance than the original PCA based hashing. However, to the best of our knowledge, there exist no methods which can guarantee to learn a projection with equal variances for different dimensions. Hence, the viewpoint that using the same number of bits for different projected dimensions is unreasonable has still not been verified by either theory or experiment.

In terms of quantization stage, few of the existing methods discussed above have studied the effect of quantization. Because existing quantization strategies can not effectively preserve the neighborhood structure under the constraint of Hamming distance, most of existing hashing methods still can not achieve satisfactory performance even though sophisticated projection functions have been designed. The work in this paper tries to study these important factors which have been ignored by existing works.

Chapter 3 Isotropic Hashing

In this section, we first introduce the problem of learning projection functions for hashing. Then we will formulate the IsoHash model which can learn projections with isotropic variances. After that, we will describe the learning algorithms of IsoHash. Finally, the complexity of IsoHash will be analyzed.

3.1 Problem Statement

Assume we are given n data points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^d$, which form the columns of the data matrix $X \in \mathbb{R}^{d \times n}$. Without loss of generality, in this paper the data are assume to be zero centered which means $\sum_{i=1}^{n} \mathbf{x}_i = 0$. The basic idea of hashing is to map each point $\mathbf{x}_i$ into a binary string $\mathbf{y}_i \in \{0, 1\}^m$ with m denoting the code size. Furthermore, close points in the original space $\mathbb{R}^d$ should be hashed into similar binary codes in the code space $\{0, 1\}^m$ to preserve the similarity structure in the original space. In general, we compute the binary code of $\mathbf{x}_i$ as $\mathbf{y}_i = [h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \cdots, h_m(\mathbf{x}_i)]^T$ with m binary hash functions $\{h_k(\cdot)\}_{k=1}^{m}$.

Because it is NP hard to directly compute the best binary functions $h_k(\cdot)$ for a given data set [8], most hashing methods adopt a two-stage strategy to learn $h_k(\cdot)$. In the projection stage, m real-valued *projection functions* $\{f_k(\mathbf{x})\}_{k=1}^{m}$ are learned and each function can generate one real value. The projection functions are typically defined as follows: $f_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x}$, and $\mathbf{w}_k$ is the kth column of projection matrix $W \in \mathbb{R}^{d \times m}$. Hence, we have m *projected dimensions* each of which corresponds to one projection function. In the quantization stage, the real-values are quantized into a binary string by thresholding.

Currently, most methods used single-bit quantization (SBQ) strategy adopts one single bit to quantize each projected dimension. More specifically, $h_k(\mathbf{x}_i) = sgn(f_k(\mathbf{x}_i))$ where $sgn(x)$ = 1 if $x \geq 0$ and -1 otherwise. The only exception of the quantization methods is AGH [13], which uses two bits to quantize each dimension. In sum, all of

these methods adopt the same number (either one or two) of bits for different projected dimensions. However, the variances of different projected dimensions are unequal, and larger-variance dimensions typically carry more information. Hence, using the same number of bits for different projected dimensions with unequal variances is unreasonable, which has also been argued by many researchers [13, 15, 33]. Unfortunately, there exist no methods which can learn projection functions with equal variances for different dimensions. In the following content of this section, we present a novel model to learn projections with isotropic variances.

3.2 Model Formulation

The idea of our IsoHash method is to learn an orthogonal matrix to rotate the PCA projection matrix. Here, we first briefly present the PCA projection methods.

As stated in [34], a reasonable standard of designing hash function to produce efficient code is that first, the variance of each bit is maximized, second, the bits are pairwise uncorrelated. They can be described in the following optimization problem form :

$$max \qquad \sum_{k=1}^{m} var(sgn(X^T \mathbf{w}_k)) \tag{3-1}$$

$$\frac{1}{n} sgn(W^T X) sgn(X^T W) = I \tag{3-2}$$

As shown in [34], the variance is maximized by producing exactly balanced bits, i.e., $sgn({\mathbf{x}_i}^T \mathbf{w}_k)$=1 for exactly half of the data points and -1 for the other half. However, the requirement of exact balancedness makes the above optimization problem intractable(NP-hard [8]). Alternatively, we can do signed magnitude relaxation as in [34] to get the following continuous objective function:

$$max \qquad \frac{1}{n} trace(W^T X X^T W) \tag{3-3}$$

$$W^T W = I \tag{3-4}$$

This above formulation is exactly the same as PCA. So to generate a code of m bits, PCAH performs PCA on X, and then use the top m eigenvectors of the covariance matrix XX^T as columns of the projection matrix $W \in \mathbb{R}^{d\times m}$. Here, top m eigenvectors are those corresponding to the m largest eigenvalues $\{\lambda_k\}_{k=1}^m$, generally arranged with the non-increasing order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. Hence, the projection functions of PCAH are defined as follows: $f_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x}$, where $\mathbf{w}_k$ is the kth column of W.

Let $\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T$ and $\Lambda = \text{diag}(\lambda)$, where $\text{diag}(\lambda)$ denotes the diagonal matrix whose diagonal entries are formed from the vector λ. It is easy to prove that $W^TXX^TW = \Lambda$. Hence, the variance of the values $f_k(\mathbf{x}_i)_{i=1}^n$ on the kth projected dimension, which corresponds to the kth row of W^TX, is λ_k. Obviously, the variances for different PCA dimensions are anisotropic.

To get isotropic projection functions, the idea of our IsoHash method is to learn an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ which makes $Q^TW^TXX^TWQ$ become a matrix with equal diagonal values, i.e., $[Q^TW^TXX^TWQ]_{11} = [Q^TW^TXX^TWQ]_{22} = \cdots = [Q^TW^TXX^TWQ]_{mm}$. Here, A_{ii} denotes the ith diagonal entry of a square matrix A, and a matrix Q is said to be orthogonal if $Q^TQ = \mathbf{I}$ where $\mathbf{I}$ is an identity matrix whose dimensionality depends on the context. The effect of the orthogonal matrix Q is to rotate the coordinate axes while keeping the Euclidean distances between any two points unchanged. It is easy to prove that the new projection functions of IsoHash are $f_k(\mathbf{x}) = (WQ)_k^T\mathbf{x}$ which have the same (isotropic) variance. Here $(WQ)_k$ denotes the kth column of WQ.

If we use $\text{tr}(A)$ to denote the trace of a symmetric matrix A, we have the following Lemma 1.

Lemma 1. *If* $Q^TQ = \mathbf{I}$, $\text{tr}(Q^TAQ) = \text{tr}(A)$.

Based on Lemma 1, we have $\text{tr}(Q^TW^TXX^TWQ) = \text{tr}(W^TXX^TW) = \text{tr}(\Lambda) = \sum_{i=1}^m \lambda_i$. Hence, to make $Q^TW^TXX^TWQ$ become a matrix with equal diagonal values, we should set this diagonal value $a = \frac{\sum_{i=1}^m \lambda_i}{m}$.

Let $\mathbf{a} = [a_1, a_2, \cdots, a_m]$ with $a_i = a = \frac{\sum_{i=1}^m \lambda_i}{m}$, and

$$\mathcal{T}(\mathbf{z}) = \{T \in \mathbb{R}^{m\times m} | \text{diag}(T) = \text{diag}(\mathbf{z})\}, \tag{3-5}$$

where $\mathbf{z}$ is a vector of length m, $\text{diag}(T)$ is overloaded to denote a diagonal matrix with the same diagonal entries of matrix T. The **problem of IsoHash** is to find a Q making $Q^T W^T X X^T W Q \in \mathcal{T}(\mathbf{a})$.

Then, we have the following Theorem 1:

Theorem 1. *Assume $Q^T Q = \mathbf{I}$ and $T \in \mathcal{T}(\mathbf{a})$. If $Q^T \Lambda Q = T$, Q will be a solution to the problem of IsoHash.*

Proof. Because $W^T X X^T W = \Lambda$, we have $Q^T \Lambda Q = Q^T [W^T X X^T W] Q$. It is obvious that Q will be a solution to the problem of IsoHash. □

As in [35], we define

$$\mathcal{M}(\Lambda) = \{Q^T \Lambda Q | Q \in \mathcal{O}(m)\}, \tag{3-6}$$

where $\mathcal{O}(m)$ is the set of all orthogonal matrices in $\mathbb{R}^{m \times m}$, i.e., $Q^T Q = \mathbf{I}$.

According to Theorem 1, the problem of IsoHash is equivalent to finding a Q for the following equation [35]:

$$||T - Z||_F = 0, \tag{3-7}$$

where $T \in \mathcal{T}(\mathbf{a})$, $Z \in \mathcal{M}(\Lambda)$, $||\cdot||_F$ denotes the Frobenius norm. Please note that for ease of understanding, we use the same notations as those in [35].

In the following content, we will use the Schur-Horn lemma [36] to prove that we can always find a solution to problem (3-7).

Lemma 2. *[Schur-Horn Lemma] Let $\mathbf{c} = \{c_i\} \in \mathbb{R}^m$ and $\mathbf{b} = \{b_i\} \in \mathbb{R}^m$ be real vectors in non-increasing order respectively* [1]*, i.e., $c_1 \geq c_2 \geq \cdots \geq c_m$, $b_1 \geq b_2 \geq \cdots \geq b_m$. There exists a Hermitian matrix H with eigenvalues $\mathbf{c}$ and diagonal values $\mathbf{b}$*

[1] Please note in [35], the values are in increasing order. It is easy to prove that our presentation of Schur-Horn lemma is equivalent to that in [35]. The non-increasing order is chosen here just because it will facilitate our following presentation due to the non-increasing order of the eigenvalues in Λ.

if and only if

$$\sum_{i=1}^{k} b_i \leq \sum_{i=1}^{k} c_i, \text{ for any } k = 1, 2, ..., m,$$
$$\sum_{i=1}^{m} b_i = \sum_{i=1}^{m} c_i.$$

Proof. Please refer to Horn's article [36]. □

Base on Lemma 2, we have the following Theorem 2.

Theorem 2. *There exists a solution to the IsoHash problem in (3-7). And this solution is in the intersection point of* $\mathcal{T}(\mathbf{a})$ *and* $\mathcal{M}(\Lambda)$.

Proof. Because $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ and $a_1 = a_2 = \cdots = a_m = \frac{\sum_{i=1}^{m} \lambda_i}{m}$, it is easy to prove that $\frac{\sum_{i=1}^{k} \lambda_i}{k} \geq \frac{\sum_{i=1}^{m} \lambda_i}{m}$ for any k. Hence, $\sum_{i=1}^{k} \lambda_i = k \times \frac{\sum_{i=1}^{k} \lambda_i}{k} \geq k \times \frac{\sum_{i=1}^{m} \lambda_i}{m} = \sum_{i=1}^{k} a_i$. Furthermore, we can prove that $\sum_{i=1}^{m} \lambda_i = \sum_{i=1}^{m} a_i$. According to Lemma 2, there exists a Hermitian matrix H with eigenvalues λ and diagonal values $\mathbf{a}$.
Moreover, we can prove that H is in the intersection of $\mathcal{T}(\mathbf{a})$ and $\mathcal{M}(\Lambda)$, i.e., $H \in \mathcal{T}(\mathbf{a})$ and $H \in \mathcal{M}(\Lambda)$. □

According to Theorem 2, to find a Q solving the problem in (3-7) is equivalent to finding the intersection point of $\mathcal{T}(\mathbf{a})$ and $\mathcal{M}(\Lambda)$, which is just an inverse eigenvalue problem called SHIEP in [35].

3.3 Learning

The problem in (3-7) can be reformulated as the following optimization problem:

$$\underset{Q:T\in\mathcal{T}(\mathbf{a}),Z\in\mathcal{M}(\Lambda)}{\operatorname{argmin}} ||T - Z||_F. \tag{3-8}$$

As in [35], we propose two algorithms to learn Q: one is called *lift and projection (LP)*, and the other is called *gradient flow (GF)*. For ease of understanding, we use the same notations as those in [35], and some proofs of theorems are omitted. The readers can refer to [35] for the details.

3.3.1 Lift and Projection

The main idea of lift and projection (LP) algorithm is to alternate between the following two steps:

- Lift step:
  Given a $T^{(k)} \in \mathcal{T}(\mathbf{a})$, we find the point $Z^{(k)} \in \mathcal{M}(\Lambda)$ such that $||T^{(k)} - Z^{(k)}||_F = dist(T^{(k)}, \mathcal{M}(\Lambda))$, where $dist(T^{(k)}, \mathcal{M}(\Lambda))$ denotes the minimum distance between $T^{(k)}$ and the points in $\mathcal{M}(\Lambda)$.

- Projection step:
  Given a $Z^{(k)}$, we find $T^{(k+1)} \in \mathcal{T}(\mathbf{a})$ such that $||T^{(k+1)} - Z^{(k)}||_F = dist(\mathcal{T}(\mathbf{a}), Z^{(k)})$, where $dist(\mathcal{T}(\mathbf{a}), Z^{(k)})$ denotes the minimum distance between $Z^{(k)}$ and the points in $\mathcal{T}(\mathbf{a})$.

We call $Z^{(k)}$ a lift of $T^{(k)}$ onto $\mathcal{M}(\Lambda)$ and $T^{(k+1)}$ a projection of $Z^{(k)}$ onto $\mathcal{T}(\mathbf{a})$. The projection operation is easy to complete. Suppose $Z^{(k)} = [z_{ij}]$, then $T^{(k+1)} = [t_{ij}]$ must be given by

$$t_{ij} = \begin{cases} z_{ij}, \text{ if } i \neq j \\ a_i, \text{ if } i = j \end{cases} \tag{3-9}$$

For the lift operation, we have the following Theorem 3.

Theorem 3. *Suppose $T = Q^T D Q^T$ is an eigen-decomposition of T where $D = diag(\mathbf{d})$ with $\mathbf{d} = [d_1, d_2, ..., d_m]^T$ being T's eigenvalues which are ordered as $d_1 \geq d_2 \geq \cdots \geq d_m$. Then the nearest neighbor of T in $\mathcal{M}(\Lambda)$ is given by*

$$Z = Q^T \Lambda Q. \tag{3-10}$$

Proof. See Theorem 4.1 in [37]. □

Since in each step we minimize the distance between T and Z, we have

$$||T^{(k)} - Z^{(k)}||_F \geq ||T^{(k+1)} - Z^{(k)}||_F \geq ||T^{(k+1)} - Z^{(k+1)}||_F.$$

It is easy to see that $(T^{(k)}, Z^{(k)})$ will converge to a stationary point. The whole IsoHash algorithm based on LP, abbreviated as IsoHash-LP, is briefly summarized in Algorithm 1.

Algorithm 1 Lift and projection based IsoHash (IsoHash-LP)

Input: $X \in \mathbb{R}^{d\times n}, m \in \mathbb{N}^+, t \in \mathbb{N}^+$

- $[\Lambda, W] = PCA(X, m)$, as stated in Section 3.2.
- Generate a random orthogonal matrix $Q_0 \in \mathbb{R}^{n\times n}$.
- $Z^{(0)} \leftarrow Q_0^T \Lambda Q_0$.
- **for** $k = 1 \rightarrow t$ **do**
  Calculate $T^{(k)}$ from $Z^{(k-1)}$ by equation (3-9).
  Perform eigen-decomposition of $T^{(k)}$ to get $Q_k^T D Q_k = T^{(k)}$.
  Calculate $Z^{(k)}$ from Q_k and Λ by equation (3-10).
- **end for**
- $Y = sgn(Q_t^T W^T X)$.

Output: Y

Because $\mathcal{M}(\Lambda)$ is not a convex set, the stationary point we find is not necessarily inside the intersection of $\mathcal{T}(\mathbf{a})$ and $\mathcal{M}(\Lambda)$. For example, if we set $Z^{(0)} = \Lambda$, the lift and projection learning algorithm would get no progress because the Z and T are already in a stationary point. To solve this problem of degenerate solutions, we initiate Z as a transformed Λ with some random orthogonal matrix Q_0, which is illustrated in Algorithm 1.

3.3.2 Gradient Flow

Another learning algorithm is a continuous one based on the construction of a gradient flow (GF) on the surface $\mathcal{M}(\Lambda)$ that moves towards the desired intersection point. Be-

cause there always exists a solution for the problem in (3-7) according to Theorem 2, the objective function in (3-8) can be reformulated as follows [35]:

$$\min_{Q\in\mathcal{O}(n)} F(Q) = \frac{1}{2}||\text{diag}(Q^T\Lambda Q) - \text{diag}(\mathbf{a})||_F^2. \tag{3-11}$$

The details about how to optimize (3-11) can be found in [35]. We just show some key steps of the learning algorithm in the following content.

The gradient ∇F at Q can be calculated as

$$\nabla F(Q) = 2\Lambda\beta(Q), \tag{3-12}$$

where $\beta(Q) = \text{diag}(Q^T\Lambda Q) - \text{diag}(\mathbf{a})$. Once we have computed the gradient of F, it can be projected onto the manifold $\mathcal{O}(m)$ according to the following Theorem 4.

Theorem 4. *The projection of $\nabla F(Q)$ onto $\mathcal{O}(m)$ is given by*

$$g(Q) = Q[Q^T\Lambda Q, \beta(Q)] \tag{3-13}$$

where $[A, B] = AB - BA$ is the Lie bracket.

Proof. See the formulas (20), (21) and (22) in [37]. □

The vector field $\dot{Q} = -g(Q)$ defines a steepest descent flow on the manifold $\mathcal{O}(m)$ for function $F(Q)$. Letting $Z = Q^T\Lambda Q$ and $\alpha(Z) = \beta(Q)$, we get

$$\dot{Z} = [Z, [\alpha(Z), Z]], \tag{3-14}$$

where $\dot{Z}$ is an isospectral flow that moves to reduce the objective function $F(Q)$.

As stated by Theorems 3.3 and 3.4 in [35], a stable equilibrium point of (3-14) must be combined with $\beta(Q) = 0$, which means that $F(Q)$ has decreased to zero. Hence, the gradient flow method can always find an intersection point as the solution. The whole IsoHash algorithm based on GF, abbreviated as IsoHash-GF, is briefly summarized in Algorithm 2.

Algorithm 2 Gradient flow based IsoHash (IsoHash-GF)

Input: $X \in \mathbb{R}^{d\times n}, m \in \mathbb{N}^+$

- $[\Lambda, W] = PCA(X, m)$, as stated in Section 3.2.
- Generate a random orthogonal matrix $Q_0 \in \mathbb{R}^{m\times m}$.
- $Z^{(0)} \leftarrow Q_0^T \Lambda Q_0$.
- Start integration form $Z = Z^{(0)}$ with gradient computed from equation (3-14).
- Stop integration when reaching a stable equilibrium point.
- Perform eigen-decomposition of Z to get $Q^T \Lambda Q = Z$.
- $Y = sgn(Q^T W^T X)$.

Output: Y

We now discuss some implementation details of IsoHash-GF. Since all diagonal matrices in $\mathcal{M}(\Lambda)$ result in $\dot{Z} = \mathbf{0}$, one should not use Λ as the starting point. In our implementation, we use the same method as that in IsoHash-LP to avoid this degenerate case, i.e., a random orthogonal transformation matrix Q_0 is use to rotate Λ. To integrate Z with gradient (3-14), we use Adams-Bashforth-Moulton PECE solver in [38] where the parameter RelTol is set to 10^{-3}. The relative error of the algorithm is computed by comparing the diagonal entries of Z to the target $\mathrm{diag}(\mathbf{a})$. The whole integration process will be terminated when their relative error is below 10^{-7}.

3.4 Complexity Analysis

The learning of our IsoHash method contains two phases: the first phase is PCA and the second phase is LP or GF. The time complexity of PCA is $O(\min(n^2 d, nd^2))$. The time complexity of LP after PCA is $O(m^3 t)$, and that of GF after PCA is $O(m^3)$. In our experiments, t is set to 100 because good performance can be achieved at this setting. Because m is typically set to be a very small number like 64 or 128, the main time complexity of IsoHash is from the PCA phase. In general, the training of IsoHash-GF will be faster than IsoHash-LP in our experiments.

One promising property of both LP and GF is that the time complexity after PCA is independent of the number of training data, which makes them scalable to large-scale data sets.

3.5 Relation to Existing Works

The most related method of IsoHash is ITQ [15], because both ITQ and IsoHash have to learn an orthogonal matrix. However, IsoHash is different from ITQ in many aspects: firstly, the goal of IsoHash is to learn a projection with isotropic variances, but the results of ITQ cannot necessarily guarantee isotropic variances; secondly, IsoHash directly learns the orthogonal matrix from the eigenvalues and eigenvectors of PCA, but ITQ first quantizes the PCA results to get some binary codes, and then learns the orthogonal matrix based on the resulting binary codes; thirdly, IsoHash has an explicit objective function to optimize, but ITQ uses a two-step heuristic strategy whose goal cannot be formulated by a single objective function; fourthly, to learn the orthogonal matrix, IsoHash uses Lift and Projection or Gradient Flow, but ITQ uses Procrustean method which is much slower than IsoHash. From the experimental results which will be presented in the next section, we can find that IsoHash can achieve accuracy comparable to ITQ with much faster training speed.

3.6 Summary

In this chapter, we firstly formulate the IsoHash model, secondly use Schur-Horn Lemma to prove the existence of solution, thirdly present two algorithms to find the solution, fourthly discuss the computational complexity and relation to other existing works.

Chapter 4 Manhattan Quantization

Currently, almost all hashing methods adopt Hamming distance to measure the similarity (neighborhood) between points in the hashcode space. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different [1]. Till now, only two quantization methods have been proposed for hashing. One is single-bit quantization(SBQ) just discussed above, and the other is hierarchical quantization (HQ) which is adopted by only one hashing method AGH [13]. Rather than using one bit, HQ divides each projected dimension into four regions with three thresholds and uses two bits to encode each region. Hence, to get a c-bit code, HQ based hashing need only $c/2$ projection functions. Figure 4.1 (b) illustrates the result of HQ for one projected dimension. However neither SBQ nor HQ can effectively preserve the neighborhood structure under the constraint of Hamming distance. Hence, although the projection functions in the projection stage can preserve the neighborhood structure, the whole hashing procedure will still destroy the neighborhood structure in the original feature space due to the limitation of Hamming distance. This will violate the goal of hashing and consequently satisfactory performance cannot be easily achieved by traditional Hamming distance based hashing methods, even though a large number of sophisticated projection functions have been designed by researchers. We tries to study these important factors which have been ignored by existing works.

To achieve satisfactory performance for ANN, one important requirement of hashing is to preserve the neighborhood structure in the original space. More specifically, close points in the original space $\mathbb{R}^d$ should be mapped to similar binary codes in the code space $\{0,1\}^c$.

We can easily find that with Hamming distance, both SBQ and HQ will destroy the neighborhood structure in the data. As illustrated in Figure 4.1 (a), point 'C' and point 'D' will be quantized into 0 and 1 respectively although they are very close to each other in the real-valued space. On the contrary, point 'D' and point 'F' will be

[1] http://en.wikipedia.org/wiki/Hamming_distance

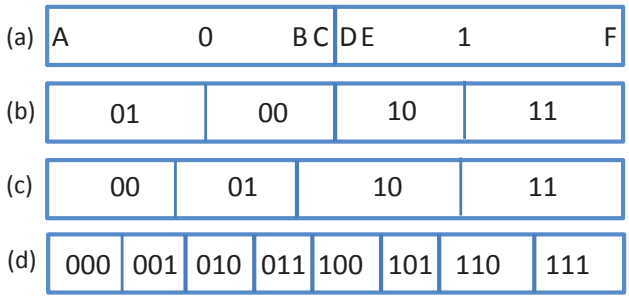
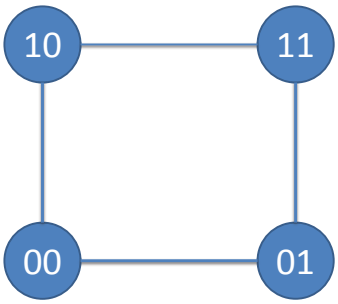


Figure 4.1 Different quantization methods: (a) single-bit quantization (SBQ); (b) hierarchical quantization (HQ); (c) 2-bit Manhattan quantization (2-MQ); (d) 3-bit Manhattan quantization (3-MQ).

quantized into the same code 1 although they are far away from each other. Hence, in the code space of this dimension, the Hamming distance between 'F' and 'D' is smaller than that between 'C' and 'D', which obviously indicates that SBQ can destroy the neighborhood structure in the original space.

HQ can also destroy the neighborhood structure of data. Let $d_h(x, y)$ denote the Hamming distance between binary codes x and y. From Figure 4.1 (b), we can get $d_h(A, F) = d_h(A, B) = d_h(C, D) = d_h(D, F) = 1$, and $d_h(A, D) = d_h(C, F) = 2$. Hence, we can find that the Hamming distance between the two farthest points 'A' and 'F' is the same as that between two relatively close points such as 'A' and 'B'. The even worse case is that $d_h(A, F) < d_h(A, D)$, which is obviously very unreasonable.

The problem of HQ is inevitable under the constraint of Hamming distance. Figure 4.2 (a) shows the Hamming distance between different 2-bit codes, where the distance between two points (i.e., nodes in the graph) is equivalent to the length of the shortest path between them. We can see that the largest Hamming distance between 2-bit codes is 2. However, to keep the relative distances between 4 different points (or regions), the largest distance between two different 2-bit codes should be at least 3. Hence, no matter how we permute the 2-bit codes for the four regions in Figure 4.1 (b), we cannot get any neighborhood-preserving result under the constraint of Hamming distance. One choice to overcome this problem of HQ is to design a new distance measurement.



(a) Hamming distance

(b) Decimal distance with NBC

Figure 4.2 Hamming distance and Decimal distance between 2-bit codes. The distance between two points (i.e., nodes in the graph) is the length of the shortest path between them.

4.1 Manhattan Distance Driven Quantization

As stated above, the problem that HQ cannot preserve the neighborhood structure in the data is essentially from the Hamming distance. Here, we will show that Manhattan distance with natural binary code (NBC) can solve the problem of HQ.

The Manhattan distance between two points is the sum of the differences on their dimensions. Let $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$, $\mathbf{y} = [y_1, y_2, \cdots, y_d]^T$, the Manhattan distance between $\mathbf{x}$ and $\mathbf{y}$ is defined as follows:

$$d_m(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|, \tag{4-1}$$

where $|x|$ denotes the absolute value of x.

To adapt Manhattan distance for hashing, we adopt a q-bit quantization scheme. More specifically, after we have learned the real-valued projection functions, we divide each projected dimension into 2^q regions and then use q bits of *natural binary code* (NBC) to encode the index of each region. For example, if $q = 2$, each projected dimension is divided into 4 regions, and the indices of these regions are $\{0, 1, 2, 3\}$,

the NBC codes of which are $\{00, 01, 10, 11\}$. If $q = 3$, the indices of regions are $\{0, 1, 2, 3, 4, 5, 6, 7\}$, and the NBC codes are $\{000, 001, 010, 011, 100, 101, 110, 111\}$. Figure 4.1 (c) shows the quantization result with $q = 2$ and Figure 4.1 (d) shows the quantization result with $q = 3$. Because this quantization scheme is driven by Manhattan distance, we call it *Manhattan quantization* (MQ). The MQ with q bits is denoted as q-MQ.

Another issue for MQ is about threshold learning. Badly learned thresholds will deteriorate the quantization performance. To achieve the neighborhood-preserving goal, we need to make the points in each region as similar as possible. In this paper, we use k-means clustering algorithm to learn the thresholds from the training data. More specifically, if we need to quantize each projected dimension into q bits, we use k-means to cluster the real values of each projected dimension into 2^q clusters, and the midpoint of the line joining neighboring cluster centers will be used as thresholds.

In our MH, we use the *decimal distance* rather than the Hamming distance to measure the distances between the q-bit codes for each projected dimension. The decimal distance is defined to be the difference between the decimal values of the corresponding NBC codes. For example, let $d_d(\mathbf{x}, \mathbf{y})$ denote the decimal distance between $\mathbf{x}$ and $\mathbf{y}$, then $d_d(10, 00) = |2 - 0| = 2$ and $d_d(010, 110) = |2 - 6| = 4$. Figure 4.2 (b) shows the decimal distances between different 2-bit codes, where the distance between two points (i.e., nodes in the graph) is equivalent to the length of the shortest path between them. We can see that the largest decimal distance between 2-bit codes is 3, which is enough to effectively preserve the relative distances between 4 different points (or regions). Figure 4.1 (c) shows one of the encoding results which can preserve the relative distances between the regions. Figure 4.1 (d) is the results with $q = 3$. It is obvious that the relative distances between the regions are also preserved. In fact, it is not hard to prove that this nice property will be satisfied for any positive integer q. Hence, our MQ strategy with $q \geq 2$ provides a more effective way to preserve the neighborhood structure than SBQ and HQ.

Given two binary codes $\mathbf{x}$ and $\mathbf{y}$ generated by MH, the Manhattan distance between them is computed from (4-1), where x_i and y_i correspond to the ith projected dimension

which should contain q bits. Furthermore, the difference between two q-bit codes of each dimension should be measured with decimal distance. For example, if $q = 2$,

$$\begin{aligned} d_m(000100, 110000) &= d_d(00, 11) + d_d(01, 00) + d_d(00, 00) \\ &= 3 + 1 + 0 \\ &= 4. \end{aligned}$$

If $q = 3$,

$$\begin{aligned} d_m(000100, 110000) &= d_d(000, 110) + d_d(100, 000) \\ &= 6 + 4 \\ &= 10. \end{aligned}$$

It is easy to see that when $q = 1$, the results computed with Manhattan distance are equivalent to those with Hamming distance, and consequently our MH method degenerates to the traditional SBQ-based hashing methods.

4.2 Summary of MH Learning

Given a training set, the whole learning procedure of MH, including both projection and quantization stages, can be summarized as follows:

- Choose a positive integer q, which is 2 in default;
- Choose an existing projection method or design a new projection method, and then learn $\lfloor \frac{c}{q} \rfloor$ projection functions;
- Use k-means to learn 2^q clusters, and compute $2^q - 1$ thresholds based on the centers of the learned clusters;
- Use MQ in Section 4.1 to quantize each projected dimension into q bits of NBC code based on the learned thresholds;

- Concatenate the q-bit codes of all the $\lfloor \frac{c}{q} \rfloor$ projected dimensions into one long code to represent each point.

4.3 Summary

In this chapter, we explain the problem of hashing using Hamming distance and propose our Manhattan Hasing (MH) algorithm.

Chapter 5 Experiment

5.1 Data Sets

We evaluate our methods on two widely used data sets, CIFAR [39] and LabelMe [7].

The first data set is *CIFAR-10* [39] which consists of 60,000 images. These images are collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton as the labeled subset of the original 80 million Tiny Images [40]. TinyImage data set aims to present a visualization of all the nouns in the English language arranged by semantic meaning. A total number of 79,302,017 images were collected by Google's image search engine and other search engines. The original images have the size of 32x32 pixels. These images are manually labeled into 10 classes, which are *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. The size of each image is 32×32 pixels. We represent them with 256-dimensional gray-scale GIST descriptors [41].

The second data set is *22K LabelMe* used in [7, 12]. LabelMe is a web-based tool designed to facilitate image annotation. With the help of this annotation tool, the current LabelMe data set contains as large as 200,790 images which span a wide variety of object categories. Most images in LabelMe contain multiple objects. *22K LabelMe* contains 22,019 images sampled from the large LabelMe data set. As in [7], we scale the images to have the size of 32x32 pixels, and represent each image with 512-dimensional GIST descriptors [41].

5.2 Evaluation Protocols and Baselines

As the protocols widely used in recent papers [8, 12, 15, 23], Euclidean neighbors in the original space are considered as ground truth. More specifically, a threshold of the average distance to the 50th nearest neighbor is used to define whether a point is a true positive or not. Based on the Euclidean ground truth, we compute the precision-recall curve and mean average precision (mAP) [13, 15]. For all experiments, we randomly select 1000 points as queries, and leave the rest as training set to learn the hash func-

tions. All the experimental results are averaged over 10 random training/test partitions.

Although a lot of hashing methods have been proposed, some of them are either supervised [12] or semi-supervised [11]. Our IsoHash method is essentially an *unsupervised* one. Hence, for fair comparison, we select the most representative unsupervised methods for evaluation, which contain PCAH [15], ITQ [15], SH [8], LSH [2], and SIKH [23]. These chosen methods are briefly introduced as follows:

- **PCAH** uses the eigenvectors corresponding to the largest eigenvalues of the covariance matrix for projection.
- **ITQ** uses an iteration method to find an orthogonal rotation matrix to refine the initial projection matrix learned by PCA so that the quantization error of mapping the data to the vertices of binary hypercube is minimized. Experimental results in [15] show that it can achieve better performance than most state-of-the-art methods. We set the iteration number of ITQ to be 100.
- **SH** uses the eigenfunctions computed from the data similarity graph for projection.
- **SIKH** [23] uses random projections to approximate the shift-invariant kernels. As in [15, 23], we use a Gaussian kernel whose bandwidth is set to the average distance to the 50th nearest neighbor.
- **LSH** uses a Gaussian random matrix to perform random projection.

Among these methods, PCAH, ITQ and SH are data-dependent methods, while SIKH and LSH are data-independent methods.

All experiments are conducted on our workstation with Intel(R) Xeon(R) CPU X7560@2.27GHz and 64G memory.

5.3 Evaluation Metrics

We adopt the scheme widely used in recent papers [8, 15, 23] to evaluate our method. More specifically, Euclidean neighbors in the original feature space are considered as

ground truth. A threshold of the average distance to the 50th nearest neighbors is used to define whether a point is a true positive or not. All the experimental results are averaged over 10 random training/test partitions. For each partition, we randomly select 1000 points as queries, and leave the rest as training set to learn the hash functions.

Based on the Euclidean ground truth, we can compute the precision, recall and the mean average precision (mAP) [13, 15] which are defined as follows:

$$
\begin{aligned}
Precision &= \frac{\text{the number of retrieved relevant points}}{\text{the number of all retrieved points}}, \\
Recall &= \frac{\text{the number of retrieved relevant points}}{\text{the number of all relevant points}}, \\
mAP &= \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{n_i}\sum_{k=1}^{n_i} Precision(R_{ik}),
\end{aligned}
$$

where $q_i \in Q$ is a query, n_i is the number of points relevant to q_i in the data set, the relevant points are ordered as $\{x_1, x_2, \cdots, x_{n_i}\}$, R_{ik} is the set of ranked retrieval results from the top result until you get to point x_k.

5.4 Accuracy

5.4.1 Accuracy of IsoHash

Table 5.1, 5.2 show the Hamming ranking performance measured by mAP on LabelMe and CIFAR-10 datasets. It is clear that our IsoHash methods, including both IsoHash-GF and IsoHash-LP, achieve far better performance than PCAH. The main difference between IsoHash and PCAH is that the PCAH dimensions have anisotropic variances while IsoHash dimensions have isotropic variances. Hence, the intuitive viewpoint that using the same number of bits for different projected dimensions with anisotropic variances is unreasonable has been successfully verified by our experiments. Furthermore, the performance of IsoHash is also comparable, if not superior, to the state-of-the-art methods, such as ITQ.

Figure 5.1 illustrates the precision-recall curves on LabelMe data set with different

Table 5.1 mAP on LabelMe and CIFAR data sets.

Method	LabelMe					CIFAR				
# bits	32	64	96	128	256	32	64	96	128	256
IsoHash-GF	0.2580	0.3269	0.3528	0.3662	0.3889	0.2249	0.2969	**0.3256**	**0.3357**	0.3600
IsoHash-LP	0.2534	0.3223	**0.3577**	**0.3826**	0.4274	0.1907	0.2624	0.3027	0.3223	**0.3651**
PCAH	0.0516	0.0401	0.0341	0.0307	0.0232	0.0319	0.0274	0.0241	0.0216	0.0168
ITQ	**0.2786**	**0.3328**	0.3504	0.3615	0.3728	**0.2490**	**0.3051**	0.3238	0.3319	0.3436
SH	0.0826	0.1034	0.1447	0.1653	0.2080	0.0510	0.0589	0.0802	0.1121	0.1535
SIKH	0.0590	0.1482	0.2074	0.2526	**0.4488**	0.0353	0.0902	0.1245	0.1909	0.3614
LSH	0.1549	0.2574	0.3147	0.3375	0.4034	0.1052	0.1907	0.2396	0.2776	0.3432

Table 5.2 Average precision (AP) of top 250 ranked images on LabelMe and CIFAR data sets.

Method	LabelMe					CIFAR				
# bits	32	64	96	128	256	32	64	96	128	256
IsoHash-GF	0.5688	0.6787	**0.7218**	**0.7317**	0.7591	0.4766	**0.5629**	**0.5941**	0.6058	0.6337
IsoHash-LP	0.5043	0.6241	0.6842	0.7096	**0.7593**	0.4623	0.5492	0.5876	**0.6146**	**0.6630**
PCAH	0.1348	0.1392	0.1327	0.1258	0.1115	0.1618	0.1438	0.1318	0.1252	0.1054
ITQ	**0.6259**	**0.6975**	0.7167	0.7256	0.7394	**0.4950**	0.5615	0.5811	0.5950	0.6070
SH	0.1611	0.1955	0.2608	0.3379	0.4221	0.2109	0.2559	0.3287	0.3643	0.4235
SIKH	0.1120	0.2823	0.3528	0.4648	0.6788	0.1609	0.3272	0.4050	0.4624	0.6510
LSH	0.3665	0.5323	0.6140	0.6623	0.7387	0.3491	0.4906	0.5595	0.5832	0.6435

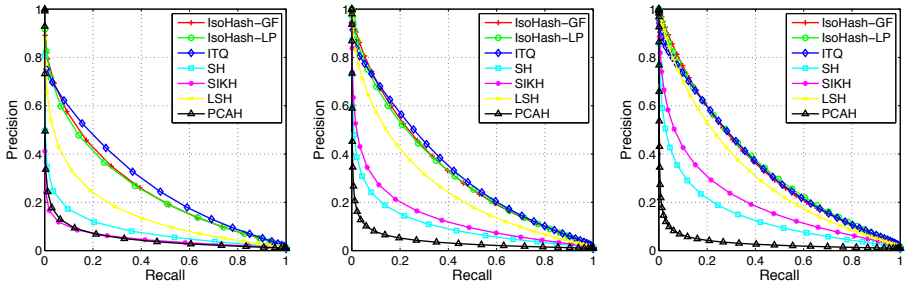
code sizes. The relative performance in the precision-recall curves on CIFAR is similar to that on LabelMe. We omit the results on CIFAR due to space limitation. Once again, we can find that our IsoHash methods can achieve performance which is far better than PCAH and comparable with the state-of-the-art.

5.4.2 Accuracy of MQ

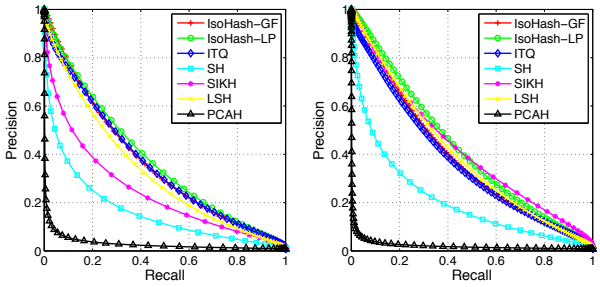
Now we compare the performance of two quantization method, traditional SBQ and our proposed MQ. From Table 5.3, we can clearly seen that our MQ method achieves great performance improvement under most settings. This implies that our MQ with Manhattan distance is indeed very effective.

5.5 Computational Cost

Table 5.4 shows the training time on CIFAR. We can see that our IsoHash methods are much faster than ITQ. The time complexity of ITQ also contains two parts: the first part is PCA which is the same as that in IsoHash, and the second part is an iteration

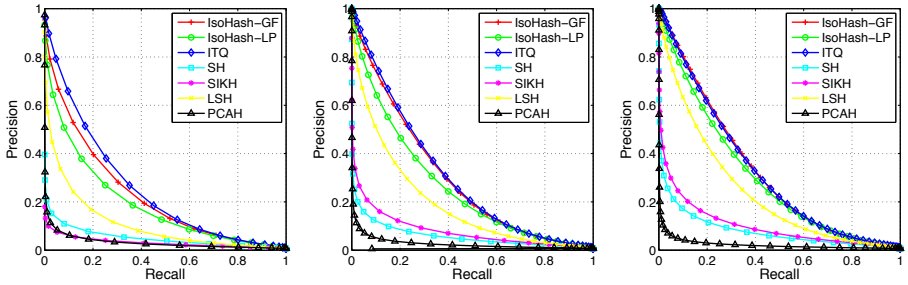


(a) 32 bits (b) 64 bits (c) 96 bits

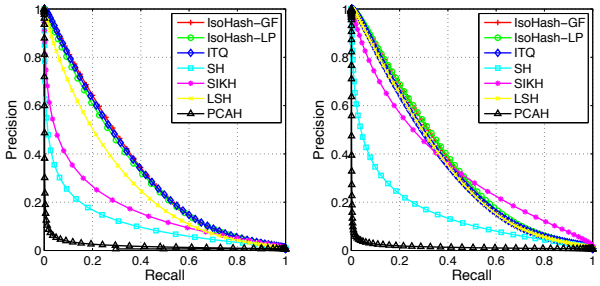


(d) 128 bits (e) 256 bits

Figure 5.1 Precision-recall curves on LabelMe data set.



(a) 32 bits (b) 64 bits (c) 96 bits



(d) 128 bits (e) 256 bits

Figure 5.2 Precision-recall curve on CIFAR data set

Table 5.3 mAP on *22K LabelMe* data set. The best mAP among all the projection methods under the same quantization method is shown in bold face.

# bits	32		64	
	SBQ	2-MQ	SBQ	2-MQ
IsoHash-GF	**0.2838**	0.3391	0.329	0.4831
IsoHash-LP	0.2541	0.3354	**0.3461**	0.4749
ITQ	0.2771	**0.3537**	0.3283	**0.4881**
SIKH	0.0487	0.0722	0.1175	0.1700
LSH	0.1563	0.1382	0.2577	0.2833
SH	0.0802	0.2207	0.0988	0.3237
PCA	0.0503	0.1913	0.0388	0.2233
# bits	128		256	
	SBQ	2-MQ	SBQ	2-MQ
IsoHash-GF	0.3859	0.6017	0.4041	0.6867
IsoHash-LP	**0.3874**	**0.626**	**0.4344**	**0.6984**
ITQ	0.3559	0.5905	0.3731	0.6496
SIKH	0.2673	0.3669	0.4109	0.5704
LSH	0.3310	0.4596	0.3955	0.6115
SH	0.1644	0.4367	0.2027	0.4418
PCA	0.0298	0.2114	0.0226	0.1710

algorithm to rotate the original PCA matrix with time complexity $O(nm^2)$, where n is the number of training points and m is the number of bits in the binary code. Hence, as the number of training data increases, the second part time complexity of ITQ will increase linearly to the number of training points. But the time complexity of IsoHash after PCA is independent of the number of training points. Hence, IsoHash will be much faster than ITQ, particularly in the case with large number of training points. This is clearly shown in Figure 5.3 which illustrates the training time when the numbers of training data are changed.

5.6 Summary

In this chapter, we conduct several experiment to test the performance of our proposed IsoHash and MH algorithm on two widely-used datasets CIFAR-10 and LabelMe and get promising results. Our used evaluation protocols are the general ones used by many researchers even outside image retrieval. The base line methods include state-of-the-art

Table 5.4 Training time (in second) on CIFAR data set.

# bits	32	64	96	128	256
IsoHash-GF	2.48	2.45	2.70	3.00	5.55
IsoHash-LP	2.14	2.43	2.94	3.47	8.83
PCAH	1.84	2.14	2.23	2.36	2.92
ITQ	4.35	6.33	9.73	12.40	29.25
SH	1.60	3.41	8.37	13.66	49.44
SIKH	1.30	1.44	1.57	1.55	2.20
LSH	0.05	0.08	0.11	0.19	0.31

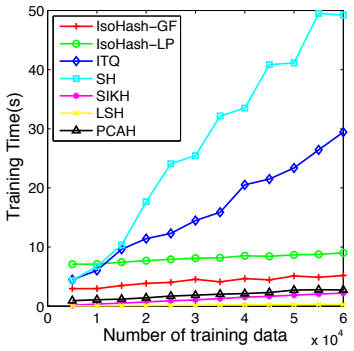


Figure 5.3 Training time comparison on CIFAR data set when changing the number of training data.

method ITQ, the one to verify the power of isotropic variance PCA which make the result more convincing.

Chapter 6 Conclusion

Although many researchers have intuitively argued that using the same number of bits for different projected dimensions with anisotropic variances is unreasonable, this viewpoint has still not been verified by either theory or experiment because no methods have been proposed to find projection functions with isotropic variances for different dimensions. The proposed IsoHash method in this paper is the first work to learn projection functions which can produce projected dimensions with isotropic variances (equal variances). Experimental results on real data sets have successfully verified the viewpoint that projections with isotropic variances will be better than those with anisotropic variances. Furthermore, IsoHash can achieve accuracy comparable to the state-of-the-art methods with faster training speed.

Almost all the existing hashing methods focus on the projection stage while ignoring the quantization stage. To further improve the accuracy, we focus on the quantization stage and find it at least as important as the projection stage. The existing quantization methods, such as SBQ and HQ, will destroy the neighborhood structure in the original space, which violates the goal of hashing. We propose a novel quantization strategy called Manhattan quantization (MQ) to effectively preserve the neighborhood structure among data. The MQ based hashing method, call Manhattan hashing (MH), encodes each projected dimension with multiple bits of *natural binary code* (NBC), based on which the Manhattan distance between points in the hashcode space is calculated for nearest neighbor search. MH can effectively preserve the neighborhood structure in the data to achieve the goal of hashing. Experiments show that our MH method is significantly effective. And the new hashing method which is combined by IsoHash and MQ achieves very promising accuracy.

Chapter 7 Publication

[1] Weihao Kong, Wu-Jun Li. Double-Bit Quantization for Hashing. Twenty-Sixth AAAI Conference on Artificial Intelligence(AAAI), 2012

[2] Weihao Kong, Wu-Jun Li, Minyi Guo. Manhattan hashing for large-scale image retrieval. Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval(SIGIR), 2012.

[3] Weihao Kong, Wu-Jun Li. Isotropic Hashing. Advances in Neural Information Processing Systems 25(NIPS), 2012

REFERENCE

[1] SHAKHNAROVICH G, DARRELL T, INDYK P. Nearest-Neighbor Methods in Learning and Vision: Theory and Practice[M].[S.l.]: The MIT Press, 2006.

[2] ANDONI A, INDYK P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions[J]. Commun. ACM, 2008, 51(1):117–122.

[3] GIONIS A, INDYK P, MOTWANI R. Similarity Search in High Dimensions via Hashing[C]//Proceedings of International Conference on Very Large Data Bases. .[S.l.]: [s.n.] , 1999.

[4] STEIN B. Principles of hash-based text retrieval[C]//Special Interest Group on Information Retrieval. .[S.l.]: [s.n.] , 2007.

[5] ZHANG D, WANG F, SI L. Composite hashing with multiple information sources[C]//Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval. .[S.l.]: [s.n.] , 2011.

[6] ZHANG D, WANG J, CAI D, et al. Self-taught hashing for fast similarity search[C]//Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval. .[S.l.]: [s.n.] , 2010.

[7] TORRALBA A, FERGUS R, WEISS Y. Small Codes and Large Image Databases for Recognition[C]//Proceedings of Computer Vision and Pattern Recognition. .[S.l.]: [s.n.] , 2008.

[8] WEISS Y, TORRALBA A, FERGUS R. Spectral hashing[C]//Proceedings of Neural Information Processing Systems. .[S.l.]: [s.n.] , 2008.

[9] SALAKHUTDINOV R, HINTON G E. Semantic hashing[J]. Int. J. Approx. Reasoning, 2009, 50(7):969–978.

[10] KULIS B, DARRELL T. Learning to Hash with Binary Reconstructive Embeddings[C]//Proceedings of Neural Information Processing Systems. .[S.l.]: [s.n.] , 2009.

[11] WANG J, KUMAR S, CHANG S F. Sequential Projection Learning for Hashing with Compact Codes[C]//Proceedings of International Conference on Machine Learning. .[S.l.]: [s.n.] , 2010.

[12] NOROUZI M, FLEET D J. Minimal Loss Hashing for Compact Binary Codes[C]//Proceedings of International Conference on Machine Learning. .[S.l.]: [s.n.] , 2011.

[13] LIU W, WANG J, KUMAR S, et al. Hashing with Graphs[C]//Proceedings of International Conference on Machine Learning. .[S.l.]: [s.n.] , 2011.

[14] TURE F, ELSAYED T, LIN J J. No free lunch: brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity[C]//Special Interest Group on Information Retrieval. .[S.l.]: [s.n.] , 2011.

[15] GONG Y, LAZEBNIK S. Iterative Quantization: A Procrustean Approach to Learning Binary Codes[C]//Proceedings of Computer Vision and Pattern Recognition. .[S.l.]: [s.n.] , 2011.

[16] MU Y, SHEN J, YAN S. Weakly-supervised hashing in kernel space[C]//Proceeding of Computer Vision and Pattern Recognition. .[S.l.]: [s.n.] , 2010.

[17] BALUJA S, COVELL M. Learning to hash: forgiving hash functions and applications[J]. Data Min. Knowl. Discov., 2008, 17(3):402–430.

[18] SONG J, YANG Y, HUANG Z, et al. Multiple feature hashing for real-time large scale near-duplicate video retrieval[C]//ACM Multimedia. .[S.l.]: [s.n.] , 2011:423–432.

[19] DATAR M, IMMORLICA N, INDYK P, et al. Locality-sensitive hashing scheme based on p-stable distributions[C]//Proceedings of the ACM Symposium on Computational Geometry. .[S.l.]: [s.n.] , 2004.

[20] KULIS B, GRAUMAN K. Kernelized locality-sensitive hashing for scalable image search[C]//Proceedings of International Conference on Computer Vision. .[S.l.]: [s.n.] , 2009.

[21] KULIS B, JAIN P, GRAUMAN K. Fast Similarity Search for Learned Metrics[J]. IEEE Trans. Pattern Anal. Mach. Intell., 2009, 31(12):2143–2157.

[22] MU Y, YAN S. Non-Metric Locality-Sensitive Hashing[C]//Proceeding of Association for the Advancement of Artificial Intelligence. .[S.l.]: [s.n.] , 2010.

[23] RAGINSKY M, LAZEBNIK S. Locality-Sensitive Binary Codes from Shift-Invariant Kernels[C]//Proceedings of Neural Information Processing Systems. .[S.l.]: [s.n.] , 2009.

[24] SALAKHUTDINOV R, HINTON G. Semantic Hashing[C]//SIGIR workshop on Information Retrieval and applications of Graphical Models. .[S.l.]: [s.n.] , 2007.

[25] HINTON G E. Training Products of Experts by Minimizing Contrastive Divergence[J]. Neural Computation, 2002, 14(8):1771–1800.

[26] SALTON G, BUCKLEY C. Term-Weighting Approaches in Automatic Text Retrieval[J]. Inf. Process. Manage., 1988, 24(5):513–523.

[27] FREUND Y, SCHAPIRE R E. Experiments with a New Boosting Algorithm[C]//Proceeding of International Conference on Machine Learning. .[S.l.]: [s.n.] , 1996.

[28] JOLLIFFE I. Principal Component Analysis[M].[S.l.]: Springer, 2002.

[29] XU H, WANG J, LI Z, et al. Complementary hashing for approximate nearest neighbor search[C]//Proceeding of International Conference on Computer Vision. .[S.l.]: [s.n.] , 2011.

[30] HE J, RADHAKRISHNAN R, CHANG S F, et al. Compact Hashing with Joint Optimization of Search Accuracy and Time[C]//Proceedings of Computer Vision and Pattern Recognition. .[S.l.]: [s.n.] , 2011.

[31] ZHUANG Y, LIU Y, WU F, et al. Hypergraph spectral hashing for similarity search of social image[C]//ACM Multimedia. .[S.l.]: [s.n.] , 2011.

[32] ZHEN Y, YEUNG D Y. Active hashing and its application to image and text retrieval[J]. Data Mining and Knowledge Discovery, 2012.

[33] JEGOU H, DOUZE M, SCHMID C, et al. Aggregating local descriptors into a compact image representation[C]//Proceeing of Computer Vision and Pattern Recognition. .[S.l.]: [s.n.] , 2010.

[34] WANG J, KUMAR S, CHANG S F. Semi-supervised hashing for large-scale image retrieval[C]//Proceedings of Computer Vision and Pattern Recognition. .[S.l.]: [s.n.] , 2010.

[35] CHU M. Constructing a Hermitian matrix from its diagonal entries and eigenvalues[J]. SIAM Journal on Matrix Analysis and Applications, 1995, 16(1):207–217.

[36] HORN A. Doubly stochastic matrices and the diagonal of a rotation matrix[J]. American Journal of Mathematics, 1954, 76(3):620–630.

[37] CHU M, DRIESSEL K. The projected gradient method for least squares matrix approximations with spectral constraints[J]. SIAM Journal on Numerical Analysis, 1990:1050–1060.

[38] SHAMPINE L, GORDON M. Computer solution of ordinary differential equations: the initial value problem[J]. Freeman, San Francisco, California, 1975.

[39] KRIZHEVSKY A. Learning multiple layers of features from tiny images[R].[S.l.]: University of Toronto, 2009.

[40] TORRALBA A, FERGUS R, FREEMAN W T. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition[J]. IEEE Trans. Pattern Anal. Mach. Intell., 2008, 30(11):1958–1970.

[41] OLIVA A, TORRALBA A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope[J]. International Journal of Computer Vision, 2001, 42(3):145–175.

Acknowledgement

I would like to gratefully and sincerely thank Prof. Wu-Jun Li for his guidance of my thesis. Through his mentorship, I not only learnt a lot about the ways to discover a problem, analysis a problem and solve a problem, but also get to know how to be a scientist and how to do research. Prof. Li is really busy on regular teaching and research work, but that does not affect his concern and mentorship on me. When I meet a bottleneck on my research, Prof. Li always discuss with me after a whole day hard work which really impress me. For everything you've done for me, Prof. Li, I thank you. Under the guidance of Prof. Li, My completion of bachelor thesis is not just a thesis but the first step of the way to do fundamental and meaningful research.

I would like to thank Prof. Yong Yu and his creation, the ACM Honored Class. Although Prof. Yu does not directly supervise my thesis, He keep concern about the thesis of our class. And all I can not achieve what I had already attained so far without the hard work of Prof. Yu in the four years education.

I would like to thank all the members of our research group.

Finally, I would like to thank my mother school for the years foster.