Learning to Hash
with its Application to Big Data Retrieval and Mining

Wu-Jun Li

Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China

Joint work with Weihao Kong and Minyi Guo

Jan 18, 2013
Outline

1 Introduction
   - Problem Definition
   - Existing Methods
   - Motivation and Contribution

2 Isotropic Hashing
   - Model
   - Learning
   - Experimental Results

3 Multiple-Bit Quantization
   - Double-Bit Quantization
   - Manhattan Quantization

4 Conclusion

5 Reference
Nearest Neighbor Search (Retrieval)

- Given a query point \( q \), return the points closest (similar) to \( q \) in the database (e.g. images).
- Underlying many machine learning, data mining, information retrieval problems

Challenge in Big Data Applications:
- Curse of dimensionality
- Storage cost
- Query speed
Similarity Preserving Hashing

- h (dog) = 10001010
- h (Napoléon) = 01100001
- h (Napoléon) = 01100101

Should be very different
Should be similar

flipped bit
Reduce Dimensionality and Storage Cost

1 million images -> 2 GB -> Binary reduction -> 16 MB

Li (http://www.cs.sjtu.edu.cn/~liwujun)
Querying

Hamming distance:
- $||01101110, 00101101||_H = 3$
- $||11011, 01011||_H = 1$

Query Image

Dataset
Querying
Querying

![Diagram showing the concept of querying and hashing]

Li (http://www.cs.sjtu.edu.cn/~liwujun)
Fast Query Speed

- By using hashing scheme, we can achieve constant or sub-linear search time complexity.

- Exhaustive search is also acceptable because the distance calculation cost is cheap now.
Two Stages of Hash Function Learning

- **Projection Stage (Dimension Reduction)**
  - Projected with real-valued projection function
  - Given a point $x$, each projected dimension $i$ will be associated with a real-valued projection function $f_i(x)$ (e.g. $f_i(x) = w_i^T x$)

- **Quantization Stage**
  - Turn real into binary
Data-Independent Methods

The hashing function family is defined independently of the training dataset:

- **Locality-sensitive hashing (LSH):** (Gionis et al., 1999; Andoni and Indyk, 2008) and its extensions (Datar et al., 2004; Kulis and Grauman, 2009; Kulis et al., 2009).

- **SIKH:** Shift invariant kernel hashing (SIKH) (Raginsky and Lazebnik, 2009).

Hashing function: random projections.
Data-Dependent Methods

Hashing functions are learned from a given training dataset.

- Relatively short codes

Two categories:

- Supervised methods
  \[ s(x_i, x_j) = 1 \text{ or } 0 \]

- Unsupervised methods
Unsupervised Methods

No labels to denote the similarity (neighborhood) between points.

- **PCAH**: principal component analysis.
- **ITQ**: (Gong and Lazebnik, 2011) orthogonal rotation matrix to refine the initial projection matrix learned by PCA.
Supervised (semi-supervised) Methods

Training dataset contains additional supervised information, (e.g. class labels or pairwise constraints).

- **SH**: Spectral Hashing (SH) (Weiss et al., 2008) adopts the eigenfunctions computed from the data similarity graph.
- **SSH**: Semi-Supervised Hashing (SSH) (Wang et al., 2010a,b) exploits both labeled data and unlabeled data for hash function learning.
- **MLH**: Minimal loss hashing (MLH) (Norouzi and Fleet, 2011) based on the latent structural SVM framework.
- **AGH**: Graph-based hashing (Liu et al., 2011).
Motivation

Problem:
All existing methods use the same number of bits for different projected dimensions with different variances.

Possible Solutions:
- Different number of bits for different dimensions (Unfortunately, have not found an effective way)
- Isotropic (equal) variances for all dimensions
Contribution

- **Isotropic hashing (IsoHash)**: (Kong and Li, 2012b) hashing with isotropic variances for all dimensions

- **Multiple-bit quantization**:  
  (1) **Double-bit quantization (DBQ)**: (Kong and Li, 2012a) Hamming distance driven 
  (2) **Manhattan hashing (MH)**: (Kong et al., 2012) Manhattan distance driven
Outline

1 Introduction
   • Problem Definition
   • Existing Methods
   • Motivation and Contribution

2 Isotropic Hashing
   • Model
   • Learning
   • Experimental Results

3 Multiple-Bit Quantization
   • Double-Bit Quantization
   • Manhattan Quantization

4 Conclusion

5 Reference

Li (http://www.cs.sjtu.edu.cn/~liwujun)
To generate a code of $m$ bits, PCAH performs PCA on $X$, and then use the top $m$ eigenvectors of the 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 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. Let $\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T$. Then

$$\Lambda = W^T XX^T W = \text{diag}(\lambda)$$

Define hash function

$$h(x) = sgn(W^T x)$$
Weakness of PCA Hash

Using the same number of bits for different projected dimensions is unreasonable because larger-variance dimensions will carry more information.
Weakness of PCA Hash

Using the same number of bits for different projected dimensions is unreasonable because larger-variance dimensions will carry more information.

Solve it by making variances equal (isotropic)!
Idea of IsoHash

- Learn an orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ which makes $Q^T W^T X X^T W Q$ become a matrix with equal diagonal values.

- Effect of $Q$: to make each projected dimension has the same variance while keeping the Euclidean distances between any two points unchanged.
**Problem Definition**

\[
\text{tr}(Q^T W^T X X^T W Q) = \text{tr}(W^T X X^T W) = \text{tr}(\Lambda) = \sum_{i=1}^{m} \lambda_i
\]

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

and

\[\mathcal{T}(z) = \{ T \in \mathbb{R}^{m \times m} \mid \text{diag}(T) = \text{diag}(z) \},\]

**Problem**

*The problem of IsoHash is to find an orthogonal matrix $Q$ making $Q^T W^T X X^T W Q \in \mathcal{T}(a)$.*
Because $Q^T \Lambda Q = Q^T [W^T X X^T W] Q$, let

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

where $\mathcal{O}(m)$ is the set of all orthogonal matrices in $\mathbb{R}^{m \times m}$.

Then, the IsoHash problem is equivalent to:

$$||T - Z||_F = 0,$$

where $T \in \mathcal{T}(a)$, $Z \in \mathcal{M}(\Lambda)$, $|| \cdot ||_F$ denotes the Frobenius norm.
Existence Theorem

Lemma

[Schur-Horn Lemma (Horn, 1954)] Let \( c = \{c_i\} \in \mathbb{R}^m \) and \( b = \{b_i\} \in \mathbb{R}^m \) be real vectors in non-increasing order respectively, i.e.,
\[
c_1 \geq c_2 \geq \cdots \geq c_m, \quad b_1 \geq b_2 \geq \cdots \geq b_m.
\]
There exists a Hermitian matrix \( H \) with eigenvalues \( c \) and diagonal values \( b \) if and only if
\[
\sum_{i=1}^{k} b_i \leq \sum_{i=1}^{k} c_i, \quad \text{for any } k = 1, 2, \ldots, m,
\]
\[
\sum_{i=1}^{m} b_i = \sum_{i=1}^{m} c_i.
\]

So we can prove:
There exists a solution to the IsoHash problem. And this solution is in the intersection of \( \mathcal{T}(\mathbf{a}) \) and \( \mathcal{M}(\Lambda) \).
Learning Methods

Two methods: (Chu, 1995)
- Lift and projection (LP)
- Gradient Flow (GF)
Lift and projection (LP)
Gradient Flow

- Objective function:

\[
\min_{Q \in O(m)} F(Q) = \frac{1}{2} \| \text{diag}(Q^T \Lambda Q) - \text{diag}(a) \|_F^2.
\]
Gradient Flow

- Objective function:
  \[
  \min_{Q \in \mathcal{O}(m)} F(Q) = \frac{1}{2} \| \text{diag}(Q^T \Lambda Q) - \text{diag}(a) \|_F^2.
  \]

- The gradient \( \nabla F \) at \( Q \):
  \[
  \nabla F(Q) = 2 \Lambda \beta(Q),
  \]
  where \( \beta(Q) = \text{diag}(Q^T \Lambda Q) - \text{diag}(a) \).
Gradient Flow

- Objective function:

$$\min_{Q \in O(m)} F(Q) = \frac{1}{2} \| \text{diag}(Q^T \Lambda Q) - \text{diag}(a) \|^2_F.$$ 

- The gradient $\nabla F$ at $Q$:

$$\nabla F(Q) = 2 \Lambda \beta(Q),$$

where $\beta(Q) = \text{diag}(Q^T \Lambda Q) - \text{diag}(a)$.

- The projection of $\nabla F(Q)$ onto $O(m)$

$$g(Q) = Q [Q^T \Lambda Q, \beta(Q)]$$

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

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

$$\dot{Z} = [Z, [\alpha(Z), Z]],$$

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

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># bits</td>
</tr>
<tr>
<td>IsoHash</td>
<td>0.2249</td>
</tr>
<tr>
<td>PCAH</td>
<td>0.0319</td>
</tr>
<tr>
<td>ITQ</td>
<td><strong>0.2490</strong></td>
</tr>
<tr>
<td>SH</td>
<td>0.0510</td>
</tr>
<tr>
<td>SIKH</td>
<td>0.0353</td>
</tr>
<tr>
<td>LSH</td>
<td>0.1052</td>
</tr>
</tbody>
</table>

Li (http://www.cs.sjtu.edu.cn/~liwujun)
Training Time

The chart shows the training time (in seconds) for various hashing methods as the number of training data points increases. The methods compared include IsoHash−GF, IsoHash−LP, ITQ, SH, SIKH, LSH, and PCAH. As the number of training data points increases, the training time for all methods increases as well, but IsoHash−LP and PCAH show significantly higher training times compared to the others.

Li (http://www.cs.sjtu.edu.cn/~liwujun)
Outline

1 Introduction
   - Problem Definition
   - Existing Methods
   - Motivation and Contribution

2 Isotropic Hashing
   - Model
   - Learning
   - Experimental Results

3 Multiple-Bit Quantization
   - Double-Bit Quantization
   - Manhattan Quantization

4 Conclusion

5 Reference
Point distribution of the real values computed by PCA on 22K LabelMe data set, and different coding results based on the distribution:

- (a) single-bit quantization (SBQ);
- (b) hierarchical hashing (HH) (Liu et al., 2011);
- (c) double-bit quantization (DBQ).
Experiment I

Precision-recall curve on 22K LabelMe data set

SH 32 bits

SH 64 bits

SH 128 bits

SH 256 bits
## Experiment II

**mAP on LabelMe data set**

<table>
<thead>
<tr>
<th># bits</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SBQ</td>
<td>HH</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.2926</td>
<td>0.2592</td>
</tr>
<tr>
<td>SH</td>
<td>0.0859</td>
<td>0.1329</td>
</tr>
<tr>
<td>PCA</td>
<td>0.0535</td>
<td>0.1009</td>
</tr>
<tr>
<td>LSH</td>
<td><strong>0.1657</strong></td>
<td>0.105</td>
</tr>
<tr>
<td>SIKH</td>
<td>0.0590</td>
<td>0.0712</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># bits</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SBQ</td>
<td>HH</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.3675</td>
<td>0.4032</td>
</tr>
<tr>
<td>SH</td>
<td>0.1730</td>
<td>0.2034</td>
</tr>
<tr>
<td>PCA</td>
<td>0.0323</td>
<td>0.1083</td>
</tr>
<tr>
<td>LSH</td>
<td>0.3579</td>
<td>0.3311</td>
</tr>
<tr>
<td>SIKH</td>
<td>0.2792</td>
<td>0.3147</td>
</tr>
</tbody>
</table>
Quantization Stage

Figure 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).
Natural Binary Code (NBC)

(a) Hamming distance

(b) Decimal distance with NBC

Li (http://www.cs.sjtu.edu.cn/~liwujun)
Multiple-Bit Quantization
Manhattan Quantization

Manhattan Distance

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:

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

where $|x|$ denotes the absolute value of $x$. 
We divide each projected dimension into $2^q$ regions and then use $q$ bits of natural binary code to encode the index of each region.
We divide each projected dimension into $2^q$ regions and then use $q$ bits of natural binary code to encode the index of each region.

For example, if $q = 3$, the indices of regions are \{0, 1, 2, 3, 4, 5, 6, 7\} and the natural binary codes are \{000, 001, 010, 011, 100, 101, 110, 111\}.
Manhattan Distance Driven Quantization

- Manhattan quantization (MQ) with $q$ bits is denoted as $q$-MQ.
- For example, if $q = 2$,

\[
\begin{align*}
    d_m(000100, 110000) &= d_d(00, 11) + d_d(01, 00) + d_d(00, 00) \\
                        &= 3 + 1 + 0 \\
                        &= 4.
\end{align*}
\]
Experiment I

Figure: Precision-recall curve on 22K LabelMe data set
## Experiment II

### Table: mAP on ANN-SIFT1M data set. The best mAP among SBQ, HQ and 2-MQ under the same setting is shown in bold face.

<table>
<thead>
<tr>
<th># bits</th>
<th>32</th>
<th>64</th>
<th>96</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SBQ</td>
<td>HQ</td>
<td>2-MQ</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.1657</td>
<td>0.2500</td>
<td><strong>0.2750</strong></td>
</tr>
<tr>
<td>SIKH</td>
<td>0.0394</td>
<td>0.0217</td>
<td><strong>0.0570</strong></td>
</tr>
<tr>
<td>LSH</td>
<td>0.1163</td>
<td>0.0961</td>
<td><strong>0.1173</strong></td>
</tr>
<tr>
<td>SH</td>
<td>0.0889</td>
<td>0.2482</td>
<td><strong>0.2771</strong></td>
</tr>
<tr>
<td>PCA</td>
<td>0.1087</td>
<td>0.2408</td>
<td><strong>0.2882</strong></td>
</tr>
</tbody>
</table>
Outline

1. Introduction
   - Problem Definition
   - Existing Methods
   - Motivation and Contribution

2. Isotropic Hashing
   - Model
   - Learning
   - Experimental Results

3. Multiple-Bit Quantization
   - Double-Bit Quantization
   - Manhattan Quantization

4. Conclusion

5. Reference
Conclusion

- Hashing can significantly **improve searching speed and reduce storage cost**.

- Projections with **isotropic variances** will be **better than** those with anisotropic variances. (IsoHash)

- The **quantization** stage is at least as important as the **projection** stage. (DBQ/MQ)
Conclusion

Q & A

Thanks!

Question?

Code available at
http://www.cs.sjtu.edu.cn/~liwujun
Outline

1 Introduction
   - Problem Definition
   - Existing Methods
   - Motivation and Contribution

2 Isotropic Hashing
   - Model
   - Learning
   - Experimental Results

3 Multiple-Bit Quantization
   - Double-Bit Quantization
   - Manhattan Quantization

4 Conclusion

5 Reference


