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

李武军

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

Joint work with 孔维昊, 张东擎, 过敏意

Dec 21, 2013
Outline

1 Introduction
   - Problem Definition
   - Existing Methods

2 Isotropic Hashing
   - Model
   - Learning
   - Experiment

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

\[ h(\text{Statue of Liberty}) = 10001010 \]

\[ h(\text{Napoléon}) = 01100001 \]

\[ h(\text{Napoléon}) = 01100101 \] (flipped bit)

Should be very different  
Should be similar
Reduce Dimensionality and Storage Cost

1 million images → 2 GB → 512 values → Binary reduction → 16 MB
Querying

Hamming distance:

- \[ ||01101110, 00101101||_H = 3 \]
- \[ ||11011, 01011||_H = 1 \]

[Image of Query Image and Dataset with Hamming distances visualized]
Querying
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

Seminal papers: (Salakhutdinov and Hinton, 2007, 2009; Torralba et al., 2008; Weiss et al., 2008)

Two categories:

- Unimodal
  - Supervised methods
given the labels $y_i$ or triplet $(x_i, x_j, x_k)$
  - Unsupervised methods

- Multimodal
  - Supervised methods
  - Unsupervised methods
(Unimodal) Unsupervised Methods

No labels to denote the categories of the training points.

- **PCA-H**: principal component analysis.
- **SH**: (Weiss et al., 2008) eigenfunctions computed from the data similarity graph.
- **ITQ**: (Gong and Lazebnik, 2011) orthogonal rotation matrix to refine the initial projection matrix learned by PCA.
- **AGH**: Graph-based hashing (Liu et al., 2011).
(Unimodal) Supervised (semi-supervised) Methods

Class labels or pairwise constraints:

- **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.
- **KSH**: Kernel-based supervised hashing (Liu et al., 2012)
- **LDAHash**: Linear discriminant analysis based hashing (Strecha et al., 2012)

Triplet-based methods:

- Hamming Distance Metric Learning (HDML) (Norouzi et al., 2012)
- Column Generation base Hashing (CGHash) (Li et al., 2013)
Multimodal Methods

- Multi-Source Hashing
- Cross-Modal Hashing
Multi-Source Hashing

- Aims at learning better codes by leveraging auxiliary views than unimodal hashing.
- Assumes that all the views provided for a query, which are typically not feasible for many multimedia applications.

- Multiple Feature Hashing (Song et al., 2011)
- Composite Hashing (Zhang et al., 2011)
Cross-Modal Hashing

Given a query of either image or text, return images or texts similar to it.

- Cross View Hashing (CVH) (Kumar and Udupa, 2011)
- Multimodal Latent Binary Embedding (MLBE) (Zhen and Yeung, 2012a)
- Co-Regularized Hashing (CRH) (Zhen and Yeung, 2012b)
- Inter-Media Hashing (IMH) (Song et al., 2013)
- Relation-aware Heterogeneous Hashing (RaHH) (Ou et al., 2013)
国内的工作

FDU: Yugang Jiang, Xuanjing Huang
HKUST: Dit-Yan Yeung
IA-CAS: Cheng-Lin Liu, Yan-Ming Zhang
ICT-CAS: Hong Chang
MSRA: Kaiming He, Jian Sun, Jingdong Wang
NUST: Fumin Shen
SYSU: Weishi Zheng
Tsinghua: Peng Cui, Shiqiang Yang, Wenwu Zhu
ZJU: Jiajun Bu, Deng Cai, Xiaofei He, Yueting Zhuang
......
Outline

1 Introduction
   - Problem Definition
   - Existing Methods

2 Isotropic Hashing
   - Model
   - Learning
   - Experiment

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

4 Conclusion

5 Reference
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
PCA Hash

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

$$h(x) = \text{sgn}(W^T x)$$
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

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

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

and

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

Problem

The problem of IsoHash is to find an orthogonal matrix \(Q\) making \(Q^T W^T X X^T W Q \in \mathcal{T}(a)\).
IsoHash Formulation

Because $Q^T \Lambda Q = Q^T [W^T X X^T W] Q$, let

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

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 $T \in \mathcal{T}(a)$, $Z \in M(\Lambda)$, $|| \cdot ||_F$ 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 $T(a)$ and $M(\Lambda)$. 
Learning Methods

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

\[ M (\Lambda) \]

\[ Z^{(k)} \]

\[ T^{(k)} \]

\[ T^{(k+1)} \]

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

- Objective function:

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

- Objective function:

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

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

- The gradient $\nabla F$ at $Q$:
  \[
  \nabla F(Q) = 2\Lambda \beta(Q),
  \]

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

- 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 $\mathcal{O}(m)$ for function $F(Q)$. Letting $Z = Q^T \Lambda Q$ and $\alpha(Z) = \beta(Q)$, we get

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

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

Li (http://www.cs.sjtu.edu.cn/~liwujun) Learning to Hash CSE, SJTU 32 / 49
<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># bits</td>
</tr>
<tr>
<td>IsoHash</td>
<td></td>
</tr>
<tr>
<td>PCAH</td>
<td></td>
</tr>
<tr>
<td>ITQ</td>
<td></td>
</tr>
<tr>
<td>SH</td>
<td></td>
</tr>
<tr>
<td>SIKH</td>
<td></td>
</tr>
<tr>
<td>LSH</td>
<td></td>
</tr>
</tbody>
</table>

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

**Accuracy (mAP)**
Training Time

Graph showing the training time for different hash functions as a function of the number of training data points. The x-axis represents the number of training data points (in units of 10^4), and the y-axis represents the training time in seconds. The graph compares IsoHash-GF, IsoHash-LP, ITQ, SH, SIKH, LSH, and PCAH, with IsoHash-GF having the highest training time and PCAH having the lowest.
Outline

1. Introduction
   - Problem Definition
   - Existing Methods

2. Isotropic Hashing
   - Model
   - Learning
   - Experiment

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

4. Conclusion

5. Reference
Double Bit Quantization

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

Li (http://www.cs.sjtu.edu.cn/~liwujun)
# Experiment II

## mAP on LabelMe data set

<table>
<thead>
<tr>
<th># bits</th>
<th>SBQ</th>
<th>HH</th>
<th>DBQ</th>
<th>SBQ</th>
<th>HH</th>
<th>DBQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITQ</td>
<td>0.2926</td>
<td>0.2592</td>
<td>0.3079</td>
<td>0.3413</td>
<td>0.3487</td>
<td>0.4002</td>
</tr>
<tr>
<td>SH</td>
<td>0.0859</td>
<td>0.1329</td>
<td>0.1815</td>
<td>0.1071</td>
<td>0.1768</td>
<td>0.2649</td>
</tr>
<tr>
<td>PCA</td>
<td>0.0535</td>
<td>0.1009</td>
<td>0.1563</td>
<td>0.0417</td>
<td>0.1034</td>
<td>0.1822</td>
</tr>
<tr>
<td>LSH</td>
<td>0.1657</td>
<td>0.105</td>
<td>0.12272</td>
<td>0.2594</td>
<td>0.2089</td>
<td>0.2577</td>
</tr>
<tr>
<td>SIKH</td>
<td>0.0590</td>
<td>0.0712</td>
<td>0.0772</td>
<td>0.1132</td>
<td>0.1514</td>
<td>0.1737</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># bits</th>
<th>SBQ</th>
<th>HH</th>
<th>DBQ</th>
<th>SBQ</th>
<th>HH</th>
<th>DBQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITQ</td>
<td>0.3675</td>
<td>0.4032</td>
<td>0.4650</td>
<td>0.3846</td>
<td>0.4251</td>
<td>0.4998</td>
</tr>
<tr>
<td>SH</td>
<td>0.1730</td>
<td>0.2034</td>
<td>0.3403</td>
<td>0.2140</td>
<td>0.2468</td>
<td>0.3468</td>
</tr>
<tr>
<td>PCA</td>
<td>0.0323</td>
<td>0.1083</td>
<td>0.1748</td>
<td>0.0245</td>
<td>0.1103</td>
<td>0.1499</td>
</tr>
<tr>
<td>LSH</td>
<td>0.3579</td>
<td>0.3311</td>
<td>0.4055</td>
<td>0.4158</td>
<td>0.4359</td>
<td>0.5154</td>
</tr>
<tr>
<td>SIKH</td>
<td>0.2792</td>
<td>0.3147</td>
<td>0.3436</td>
<td>0.4759</td>
<td>0.5055</td>
<td>0.5325</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
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 \( x \).
Manhattan Distance Driven Quantization

- 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.
Manhattan Distance Driven Quantization

- 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 $q = 2$,

$$d_m(000100, 110000) = d_d(00, 11) + d_d(01, 00) + d_d(00, 00)$$

$$= 3 + 1 + 0$$

$$= 4.$$
Experiment 1

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></th>
<th>64</th>
<th></th>
<th>96</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SBQ</td>
<td>HQ</td>
<td>2-MQ</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>
<td>0.4641</td>
<td>0.4745</td>
<td><strong>0.5087</strong></td>
</tr>
<tr>
<td>SIKH</td>
<td>0.0394</td>
<td>0.0217</td>
<td><strong>0.0570</strong></td>
<td>0.2027</td>
<td>0.0822</td>
<td><strong>0.2356</strong></td>
</tr>
<tr>
<td>LSH</td>
<td>0.1163</td>
<td>0.0961</td>
<td><strong>0.1173</strong></td>
<td>0.2340</td>
<td>0.2815</td>
<td><strong>0.3111</strong></td>
</tr>
<tr>
<td>SH</td>
<td>0.0889</td>
<td>0.2482</td>
<td><strong>0.2771</strong></td>
<td>0.1828</td>
<td>0.3841</td>
<td><strong>0.4576</strong></td>
</tr>
<tr>
<td>PCA</td>
<td>0.1087</td>
<td>0.2408</td>
<td><strong>0.2882</strong></td>
<td>0.1671</td>
<td>0.3956</td>
<td><strong>0.4683</strong></td>
</tr>
</tbody>
</table>
Outline

1 Introduction
   - Problem Definition
   - Existing Methods

2 Isotropic Hashing
   - Model
   - Learning
   - Experiment

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

4 Conclusion

5 Reference

Li (http://www.cs.sjtu.edu.cn/~liwujun)
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)
Q & A

Thanks!

Question?

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

1 Introduction
   - Problem Definition
   - Existing Methods

2 Isotropic Hashing
   - Model
   - Learning
   - Experiment

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

4 Conclusion

5 Reference

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


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


J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for


