Learning to Hash for Big Data

李武军

LAMDA Group
南京大学计算机科学与技术系
软件新技术国家重点实验室

liwujun@nju.edu.cn

May 09, 2015
1. Introduction
   - Problem Definition
   - Existing Methods

2. Scalable Graph Hashing with Feature Transformation
   - Motivation
   - Model and Learning
   - Experiment

3. Conclusion

4. Reference
Outline

1. Introduction
   - Problem Definition
   - Existing Methods

2. Scalable Graph Hashing with Feature Transformation
   - Motivation
   - Model and Learning
   - Experiment

3. Conclusion

4. 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

- $h(\text{Statue of Liberty}) = 10001010$
- $h(\text{Napoléon}) = 01100001$
- $h(\text{Napoléon}) = 01100101$

Should be very different

Should be similar

flipped bit
Reduce Dimensionality and Storage Cost

10 million images → 20 GB → 128 bits → 160 MB
Querying

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

Query Image

Dataset
Fast Query Speed

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

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

Easy or hard?

Hard: discrete optimization problem

Easy by approximation: two stages of hash function learning

- Projection stage (dimensionality 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

However, there exist essential differences between metric learning (dimensionality reduction) and learning to hash. Simply adapting traditional metric learning is not enough.
Data-Independent Methods

The hash 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).

Hash function: *random projections.*
Data-Dependent Methods

Hash functions are learned from a given training dataset (learning to hash).

- 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**: eigenfunctions computed from the data similarity graph (Weiss et al., 2008).
- **ITQ**: orthogonal rotation matrix to refine the initial projection matrix learned by PCA (Gong and Lazebnik, 2011).
- **AGH**: graph-based hashing (Liu et al., 2011).
- **IsoHash**: projected dimensions with isotropic variances (Kong and Li, 2012b).
- **DGH**: discrete graph hashing (Liu et al., 2014).
- etc.
(Unimodal) Supervised (semi-supervised) Methods

Class labels or pairwise constraints:

- **SSH**: semi-supervised hashing (SSH) exploits both labeled data and unlabeled data for hash function learning (Wang et al., 2010a,b).
- **MLH**: minimal loss hashing (MLH) based on the latent structural SVM framework (Norouzi and Fleet, 2011).
- **KSH**: kernel-based supervised hashing (Liu et al., 2012)
- **LDAHash**: linear discriminant analysis based hashing (Strecha et al., 2012)
- **LFH**: supervised hashing with latent factor models (Zhang et al., 2014)
- etc.

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)
- etc.
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)
- Semantic Correlation Maximization (SCM) (Zhang and Li, 2014)
- etc.
Our Contribution

- Unsupervised Hashing:
  1. Isotropic hashing [NIPS 2012]
  2. Scalable graph hashing with feature transformation [IJCAI 2015]

- Supervised Hashing [SIGIR 2014]:
  Supervised hashing with latent factor models

- Multimodal Hashing [AAAI 2014]:
  Large-scale supervised multimodal hashing with semantic correlation maximization

- Multiple-Bit Quantization:
  1. Double-bit quantization [AAAI 2012]
  2. Manhattan quantization [SIGIR 2012]

- 《大数据哈希学习：现状与趋势》[《科学通报》, 2015]
Outline

1 Introduction
   - Problem Definition
   - Existing Methods

2 Scalable Graph Hashing with Feature Transformation
   - Motivation
   - Model and Learning
   - Experiment

3 Conclusion

4 Reference
(Unsupervised) Graph Hashing

- Guide the hashing code learning procedure by directly exploiting the pairwise similarity (neighborhood structure).

\[
\min \sum_{ij} S_{ij} \| b_i - b_j \|^2 = \text{tr}(B^T \mathcal{L} B)
\]

subject to:
\[
b_i \in \{-1, 1\}^c
\]
\[
\sum_i b_i = 0
\]
\[
\frac{1}{n} \sum_i b_i b_i^T = I
\]

- Should be expected to achieve better performance than non-graph based methods if the learning algorithms are effective.
Motivation

However, it is difficult to design effective algorithms because both the memory and time complexity are at least $O(n^2)$.

- **SH** (Weiss et al., 2008): Use an eigenfunction solution of 1-D Laplacian with uniform assumption
- **BRE** (Kulis and Darrell, 2009): Subsample a small subset for training
- **AGH** (Liu et al., 2011), **DGH** (Liu et al., 2014): Use anchor graph to approximate the similarity graph
Contribution

How to utilize the whole graph and simultaneously avoid $O(n^2)$ complexity?

Scalable graph hashing (SGH):

- A feature transformation (Shrivastava and Li, 2014) method to effectively approximate the whole graph without explicitly computing it.
- A sequential method for bit-wise complementary learning.
- Linear complexity.
- Outperform state of the art in terms of both accuracy and scalability.
Notation

- \( \mathbf{X} = \{ \mathbf{x}_1, \ldots, \mathbf{x}_n \}^T \in \mathbb{R}^{n \times d} \): \( n \) data points.
- \( \mathbf{b}_i \in \{+1, -1\}^c \): binary code of point \( \mathbf{x}_i \).
- \( \mathbf{b}_i = [h_1(\mathbf{x}_i), \ldots, h_c(\mathbf{x}_i)]^T \), where \( h_k(\mathbf{x}) \) denotes hash function.
- Pairwise similarity metric defined as: \( S_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / \rho} \in (0, 1] \).
Objective Function

\[
\min \sum_{i,j} (\tilde{S}_{ij} - \frac{1}{c} b_i^T b_j)^2
\]

- \(\tilde{S}_{ij} = 2S_{ij} - 1 \in (-1, 1]\).
- Hash function: \(h_k(x_i) = \text{sgn}(\sum_{j=1}^{m} W_{ij} \phi(x_i, x_j) + \text{bias})\)
- \(\forall x_i\), map it into Hamming space as \(b_i = [h_1(x_i), \ldots, h_c(x_i)]^T\)
- \(\forall x\), define: \(K(x) = [\phi(x, x_1) - \sum_{i=1}^{n} \phi(x_i, x_1)/n, \ldots, \phi(x, x_m) - \sum_{i=1}^{n} \phi(x_i, x_m)/n]\)

Objective function with the parameter \(W \in \mathbb{R}^{c \times m}\):

\[
\min_{W} \|c\tilde{S} - \text{sgn}(K(X)W^T)\text{sgn}(K(X)W^T)^T\|^2_F
\]

\(s.t.\) \(WK(X)^T K(X)W^T = I\)
Feature Transformation

\( \forall x, \text{ define } P(x) \text{ and } Q(x): \)

\[
P(x) = \left[ \sqrt{\frac{2(e^2 - 1)}{e\rho}} e^{\frac{-\|x\|_F^2}{\rho}} x; \sqrt{\frac{e^2 + 1}{e}} e^{\frac{-\|x\|_F^2}{\rho}} ; 1 \right]
\]

\[
Q(x) = \left[ \sqrt{\frac{2(e^2 - 1)}{e\rho}} e^{\frac{-\|x\|_F^2}{\rho}} x; \sqrt{\frac{e^2 + 1}{e}} e^{\frac{-\|x\|_F^2}{\rho}} ; -1 \right]
\]

\( \forall x_i, x_j \in X \)

\[
P(x_i)^T Q(x_j) = 2\left[ \frac{e^2 - 1}{2e} \times \frac{2x_i^T x_j}{\rho} + \frac{e^2 + 1}{2e} \right] e^{\frac{-\|x_i\|_F^2 + \|x_j\|_F^2}{\rho}} - 1
\]

\[
\approx 2e^{\frac{-\|x_i\|_F^2 - \|x_j\|_F^2 + 2x_i^T x_j}{\rho}} - 1
\]

\[
= 2e^{\frac{-\|x_i - x_j\|_F^2}{\rho}} - 1 = \tilde{S}_{ij}
\]
Feature Transformation

- Here, we use an approximation \( \frac{e^2 - 1}{2e} x + \frac{e^2 + 1}{2e} \approx e^x \)

- We assume \(-1 \leq \frac{2}{\rho} x_i^T x_j \leq 1\). It is easy to prove that \(\rho = 2 \max \{ \|x_i\|_F^2 \}_{i=1}^n\) can make \(-1 \leq \frac{2}{\rho} x_i^T x_j \leq 1\).

- Then we have \(\tilde{S} \approx P(X)^T Q(X)\)
Sequential Learning Strategy

- Direct relaxation may lead to poor performance
- We adopt a sequential learning strategy in a bit-wise complementary manner
- Residual definition:
  \[ R_t = c\tilde{S} - \sum_{i=1}^{t-1} \text{sgn}(K(X)w_i)\text{sgn}(K(X)w_i)^T \]
  \[ R_1 = c\tilde{S} \]
- Objective function:
  \[
  \min_{w_t} ||R_t - \text{sgn}(K(X)w_t)\text{sgn}(K(X)w_t)^T||_F^2 \\
  s.t. \quad w_t^T K(X)^T K(X)w_t = 1
  \]

By relaxation, we can get:

\[
\min_{w_t} -\text{tr}(w_t^T K(X)^T R_t K(X)w_t) \\
\text{s.t.} \quad w_t^T K(X)^T K(X)w_t = 1
\]
Sequential Learning Strategy

- Then we obtain a generalized eigenvalue problem:

\[ K(X)^T R_t K(X) w_t = \lambda K(X)^T K(X) w_t \]

- Define \( A_t = K(X)^T R_t K(X) \in \mathbb{R}^{m \times m} \), then:

\[
A_t = A_{t-1} - K(X)^T \text{sgn}(K(X)w_{t-1})\text{sgn}(K(X)w_{t-1})^T K(X)
\]

- Key component:

\[
A_1 = cK(X)^T \tilde{S} K(X)
= c[K(X)^T P(X)^T][Q(X) K(X)] \tilde{S}
\]
Sequential Learning Strategy

- Adopting the residual matrix:

\[ R_t = c\tilde{S} - \sum_{i=1, i \neq t}^{c} \text{sgn}(K(X)w_i)\text{sgn}(K(X)w_i)^T \]

we can learn all the \( \mathbf{W} = \{w_i\}_{i=1}^{c} \) for multiple rounds.

- This can further improve the accuracy.

- We continue it for one more round to get a good tradeoff between accuracy and speed.
Sequential Learning Algorithm

Algorithm 1 Sequential learning algorithm for SGH

Input: Feature vectors $X \in \mathcal{R}^{n \times d}$; code length $c$; number of kernel bases $m$.
Output: Weight matrix $W \in \mathcal{R}^{c \times m}$.

Procedure

Construct $P(X)$ and $Q(X)$ according to (3);
Construct $K(X)$ based on the kernel bases, which are $m$ points randomly selected from $X$;

$A_0 = [K(X)^T P(X)^T][Q(X) K(X)];$
$A_1 = cA_0;$
$Z = K(X)^T K(X) + \gamma I_d;$

for $t = 1 \rightarrow c$ do

Solve the following generalized eigenvalue problem

$A_t w_t = \lambda Z w_t;$
$U = [K(X)^T \text{sgn}(K(X)w_t)][K(X)^T \text{sgn}(K(X)w_t)]^T;$

$A_{t+1} = A_t - U;$

end for

$\hat{A}_0 = A_{c+1}$

Randomly permute $\{1, 2, \cdots, c\}$ to generate a random index set $\mathcal{M}$;

for $t = 1 \rightarrow c$ do

$\hat{t} = \mathcal{M}(t);$

$\hat{A}_0 = \hat{A}_0 + K(X)^T \text{sgn}(K(X)w_{\hat{t}})\text{sgn}(K(X)w_{\hat{t}})^T K(X);$

Solve the following generalized eigenvalue problem

$\hat{A}_0 v = \lambda Z v;$
Update $w_{\hat{t}} \leftarrow v$

$\hat{A}_0 = \hat{A}_0 - K(X)^T \text{sgn}(K(X)w_{\hat{t}})\text{sgn}(K(X)w_{\hat{t}})^T K(X);$

end for
Complexity Analysis

- **Initialization**
  - $P(X)$ and $Q(X)$: $O(dn)$
  - Kernel initialization: $O(dmn + mn)$
  - $A_1$ and $Z$: $O((d + 2)mn)$

- **Main procedure**
  - $O(c(mn + m^2) + m^3)$

Typically, $m, d, c$ will be much less than $n$.

Hence, the time complexity is $O(n)$. Furthermore, the storage complexity is also $O(n)$. 
Dataset

**TINY-1M**
- One million images from the set of 80M tiny images.
- Each tiny image is represented by a 384-dim GIST descriptors.

**MIRFLICKR-1M**
- One million Flickr images.
- Extract 512-dim features from each image.
Evaluation Protocols and Baselines

Evaluation Protocols:

- Ground truth: top 2% nearest neighbors in the training set in terms of Euclidean distance
- Hamming ranking: Top-K precision
- Training time for scalability

Baselines:

- LSH (Andoni and Indyk, 2008)
- PCAH (Gong and Lazebnik, 2011)
- ITQ (Gong and Lazebnik, 2011)
- AGH (Liu et al., 2011)
- DGH (Liu et al., 2014): DGH-I and DGH-R
## Top-1k Precision

<table>
<thead>
<tr>
<th>Method</th>
<th>32 bits</th>
<th>64 bits</th>
<th>96 bits</th>
<th>128 bits</th>
<th>256 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGH</td>
<td>0.4697</td>
<td>0.5742</td>
<td>0.6299</td>
<td>0.6737</td>
<td>0.7357</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.4289</td>
<td>0.4782</td>
<td>0.4947</td>
<td>0.4986</td>
<td>0.5003</td>
</tr>
<tr>
<td>AGH</td>
<td>0.3973</td>
<td>0.4402</td>
<td>0.4577</td>
<td>0.4654</td>
<td>0.4767</td>
</tr>
<tr>
<td>DGH-I</td>
<td>0.3974</td>
<td>0.4536</td>
<td>0.4737</td>
<td>0.4874</td>
<td>0.4969</td>
</tr>
<tr>
<td>DGH-R</td>
<td>0.3793</td>
<td>0.4554</td>
<td>0.4871</td>
<td>0.4989</td>
<td>0.5276</td>
</tr>
<tr>
<td>PCAH</td>
<td>0.2457</td>
<td>0.2203</td>
<td>0.2000</td>
<td>0.1836</td>
<td>0.1421</td>
</tr>
<tr>
<td>LSH</td>
<td>0.2507</td>
<td>0.3575</td>
<td>0.4122</td>
<td>0.4529</td>
<td>0.5212</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>32 bits</th>
<th>64 bits</th>
<th>96 bits</th>
<th>128 bits</th>
<th>256 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGH</td>
<td>0.4919</td>
<td>0.6041</td>
<td>0.6677</td>
<td>0.6985</td>
<td>0.7584</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.5177</td>
<td>0.5776</td>
<td>0.5999</td>
<td>0.6096</td>
<td>0.6228</td>
</tr>
<tr>
<td>AGH</td>
<td>0.4299</td>
<td>0.4741</td>
<td>0.4911</td>
<td>0.4998</td>
<td>0.506</td>
</tr>
<tr>
<td>DGH-I</td>
<td>0.4299</td>
<td>0.4806</td>
<td>0.5001</td>
<td>0.5111</td>
<td>0.5253</td>
</tr>
<tr>
<td>DGH-R</td>
<td>0.4121</td>
<td>0.4776</td>
<td>0.5054</td>
<td>0.5196</td>
<td>0.5428</td>
</tr>
<tr>
<td>PCAH</td>
<td>0.2720</td>
<td>0.2384</td>
<td>0.2141</td>
<td>0.1950</td>
<td>0.1508</td>
</tr>
<tr>
<td>LSH</td>
<td>0.2597</td>
<td>0.3995</td>
<td>0.466</td>
<td>0.5160</td>
<td>0.6072</td>
</tr>
</tbody>
</table>
Top-k Precision @TINY-1M

(a) 64 bit

(b) 128 bit
Top-k Precision @MIRFLICKR-1M

(a) 64 bit

(b) 128 bit

Li (http://cs.nju.edu.cn/lwj)
Training Time (in second)

Training time @TINY-1M. Here, \( t_1 = 1438.60 \)

<table>
<thead>
<tr>
<th>Method</th>
<th>32 bits</th>
<th>64 bits</th>
<th>96 bits</th>
<th>128 bits</th>
<th>256 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGH</td>
<td>34.49</td>
<td>52.37</td>
<td>71.53</td>
<td>89.65</td>
<td>164.23</td>
</tr>
<tr>
<td>ITQ</td>
<td>31.72</td>
<td>60.62</td>
<td>89.01</td>
<td>149.18</td>
<td>322.06</td>
</tr>
<tr>
<td>AGH</td>
<td>18.60 + ( t_1 )</td>
<td>19.40 + ( t_1 )</td>
<td>20.08 + ( t_1 )</td>
<td>22.48 + ( t_1 )</td>
<td>25.09 + ( t_1 )</td>
</tr>
<tr>
<td>DGH-I</td>
<td>187.57 + ( t_1 )</td>
<td>296.99 + ( t_1 )</td>
<td>518.57 + ( t_1 )</td>
<td>924.08 + ( t_1 )</td>
<td>1838.30 + ( t_1 )</td>
</tr>
<tr>
<td>DGH-R</td>
<td>217.06 + ( t_1 )</td>
<td>360.18 + ( t_1 )</td>
<td>615.74 + ( t_1 )</td>
<td>1089.10 + ( t_1 )</td>
<td>2300.10 + ( t_1 )</td>
</tr>
<tr>
<td>PCAH</td>
<td>4.29</td>
<td>4.54</td>
<td>4.75</td>
<td>5.85</td>
<td>6.49</td>
</tr>
<tr>
<td>LSH</td>
<td>1.68</td>
<td>1.77</td>
<td>1.84</td>
<td>2.55</td>
<td>3.76</td>
</tr>
</tbody>
</table>

Training time @MIRFLICKR-1M. Here, \( t_2 = 1564.86 \)

<table>
<thead>
<tr>
<th>Method</th>
<th>32 bits</th>
<th>64 bits</th>
<th>96 bits</th>
<th>128 bits</th>
<th>256 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGH</td>
<td>41.51</td>
<td>59.02</td>
<td>74.86</td>
<td>97.25</td>
<td>168.35</td>
</tr>
<tr>
<td>ITQ</td>
<td>36.17</td>
<td>64.61</td>
<td>89.50</td>
<td>132.71</td>
<td>285.10</td>
</tr>
<tr>
<td>AGH</td>
<td>17.99 + ( t_2 )</td>
<td>18.80 + ( t_2 )</td>
<td>20.30 + ( t_2 )</td>
<td>19.87 + ( t_2 )</td>
<td>21.60 + ( t_2 )</td>
</tr>
<tr>
<td>DGH-I</td>
<td>85.81 + ( t_2 )</td>
<td>143.68 + ( t_2 )</td>
<td>215.41 + ( t_2 )</td>
<td>352.73 + ( t_2 )</td>
<td>739.56 + ( t_2 )</td>
</tr>
<tr>
<td>DGH-R</td>
<td>116.25 + ( t_2 )</td>
<td>206.24 + ( t_2 )</td>
<td>308.32 + ( t_2 )</td>
<td>517.97 + ( t_2 )</td>
<td>1199.44 + ( t_2 )</td>
</tr>
<tr>
<td>PCAH</td>
<td>7.65</td>
<td>7.90</td>
<td>8.47</td>
<td>9.23</td>
<td>10.42</td>
</tr>
<tr>
<td>LSH</td>
<td>2.44</td>
<td>2.43</td>
<td>2.71</td>
<td>3.38</td>
<td>4.21</td>
</tr>
</tbody>
</table>
Sensitivity to Parameter $\rho$

(a) TINY-1M  
(b) MIRFLICKR-1M

- Red line: 64 bits
- Blue line: 128 bits
Sensitivity to Parameter $m$

(a) TINY-1M

(b) MIRFLICKR-1M
Outline

1. Introduction
   - Problem Definition
   - Existing Methods

2. Scalable Graph Hashing with Feature Transformation
   - Motivation
   - Model and Learning
   - Experiment

3. Conclusion

4. Reference
Conclusion

- Hashing can significantly improve retrieval speed and reduce storage cost.

- It has become a hot research topic in big learning with wide applications.

- We have proposed a series of hashing methods, including unsupervised, supervised, and multimodal methods. Furthermore, some quantization strategies (Kong and Li, 2012a; Kong et al., 2012) are also designed.

- In particular, the details of SGH with feature transformation (Jiang and Li, 2015) are introduced in this talk.
Future Trends

- Discrete optimization and learning
- Scalable training for supervised hashing
- Quantization
- New applications

《大数据哈希学习：现状与趋势》[《科学通报》,2015] (Li and Zhou, 2015)
Thanks!

Question?

Code available at
http://cs.nju.edu.cn/lwj
1 Introduction
   - Problem Definition
   - Existing Methods

2 Scalable Graph Hashing with Feature Transformation
   - Motivation
   - Model and Learning
   - Experiment

3 Conclusion

4 Reference


A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*, 2014.


