Complementary Binary Quantization for Joint Multiple Indexing

Qiang Fu¹, Xu Han¹, Xianglong Liu¹*, Jingkuan Song², Cheng Deng³

¹ Beihang University, Beijing, China
² University of Electronic Science and Technology of China, Chengdu, China
³ Xidian University, Xi’an, China
Outline

- **Introduction**
  - Hash-based Nearest Neighbor Search (NNS) Solution
  - Multi-table Indexing

- **Complementary Binary Quantization (CBQ)**
  - Complementary Multi-Table Quantization Formulation
  - Joint table learning
  - Algorithm details

- **Experiments**
  - Euclidean Nearest Neighbor Search
  - Semantic Nearest Neighbor Search

- **Reference**
Introduction: Hash-based NNS Solution

- **Hash-based solution**
  - Encode data to binary codes
  - Compressed storage and efficient computation
  - Widely-used in many applications like image search, feature match...

- **Locality Sensitive Hashing (LSH)**
  - Close points in the original space have similar hash codes

- **Projection based hashing**
  - Leverage the information contained in the data
  - ITQ, SH, AGH...

- **Prototype based hashing**
  - Characterize the natural data relations better
  - SPH, ABQ...
**Introduction: Multi-table Indexing**

**Multi-table indexing**
- Build multiple hash tables and probe multiple buckets to improve the search performance
- Complementary multi-table method
  - maximally cover the nearest neighbors using fewer tables

**Problems**
- Suffer from the table redundancy still
- Describe the data distribution and relation not well
Complementary Binary Quantization

- **Goal and Motivation**
  - **Complementary**: jointly learn the multiple hash tables
  - **Informative**: use prototype based hashing quantization

- **Formulation**
  \[
  \min_{\{P(l)\},\{C(l)\},\lambda} L = L_{quan} + \mu L_{align}
  \]

- Capture the data distribution better, improve the table complementarity more.
CBQ: Formulation

- **Multi-Table Quantization Loss**

  - Consider the nature of the multi-index search, only at least one of its nearest prototypes in different tables needed for the correct search results
  
  - Select the nearest prototype from all tables

\[
L_{quan} = \frac{1}{n} \sum_{i=1}^{n} d_{o}^{2} (x_i, p_{k^*}^{(l^*)})
\]

\[
d_{o} (x_i, p_{k^*}^{(l^*)}) \leq d_{o} (x_i, p_{k}^{(l)}), (l, k) \neq (l^*, k^*)
\]

\[y = 100\]
**Space Alignment Loss**

- Concentrate on the distance consistence so that codes in Hamming space will be aligned with the original data distribution.

\[
L_{\text{align}} = \frac{1}{nM} \sum_{i=1}^{n} \sum_{l=1}^{L} \sum_{k=1}^{P^{(l)}} \left\| \lambda d_o (x_i, p_{k}^{(l)}) - d_h (c_{k}^{(l')}, c_{k}^{(l)}) \right\|^2
\]

- \(P^{(l)}\) is the number of prototypes for the \(l\)-th table.
- \(M = \sum_{l=1}^{L} |P^{(l)}|\)

Using a small subset of binary codes.

The original space

The Hamming space
CBQ: Joint Table Learning

▪ Reformulation for Joint Table Learning
  - construct a one-to-one mapping that converts the original prototype index \((l, k)\) to a uniform one \(m \in \{1, 2, \ldots, M\}\)

\[
\min_{P, C, \lambda} L = \frac{1}{n} \sum_{i=1}^{n} d_o^2 (x_i, p_{m_{i}^*}) + \frac{\mu}{nM} \sum_{i=1}^{n} \sum_{m=1}^{M} \|\lambda d_o (x_i, p_m) - d_h (c_{m_{i}^*}, c_m)\|^2 \\
\text{s.t.} \quad c_m \in \{-1, 1\}^b \ ; \pi(c_m) \leq L, d_o (x_i, p_{m_{i}^*}) \leq d_o (x_i, p_m), m \neq m_{i}^*
\]

▪ Alternating Optimization
  (1) **Incomplete Coding** (optimize \(C\))
  • Find a sub-codebook most consistent with the prototypes
  (2) **Prototype Pursuit** (optimize \(P\))
  • Find a prototype set that will shrink
  (3) **Rescaling** (optimize \(\lambda\))
  • Find a proper space scaling

---

**Algorithm 1** Complementary Binary Quantization (CBQ).

**Input:** Training set \(X\), hash table number \(L\), code length \(b\) per table.

**Output:** Hash functions \(\{h^{(l)}\}_{l=1}^{L}\)

1: Initialize prototype set \(P\) and the assignment index \(m_{i}^*\) for \(X\) using K-means.
2: Initialize the scale variable \(\lambda\) according to (14).
3: repeat
4:   for \(m' = 1, 2, \ldots, M\) do
5:     Update the code set \(C\) using the local optimal binary code \(c_{m'}\) for \(p_{m'}\) by solving (8).
6:   end for
7: Update \(P\) by iteratively solving (10) and (11).
8: Update \(\lambda\) according to (13).
9: until convergence
10: Assign \(P\) and \(C\) to \(L\) hash tables, generating \(\{h^{(l)}\}_{l=1}^{L}\).
**CBQ: Algorithm Details**

- **Table Assignment**
  - Use a random assignment strategy
    - no identical hash codes in the same hash table
    - the prototype numbers of each hash table should be balanced

- **Initialization**
  - $P$: classical k-means clustering
  - $\lambda$: $M(L \times 2^b)$ prototypes and codes

- **Generate long hash codes** ($L \times 2^{b'}$)
  - Use product quantization method

- **Complexity**
  - Training: $O \left( \left( L \times 2^b \right)^2 nd \right) = O(4^b L^2 nd)$, almost linear to $n$
  - Testing: $O(2^b Ld)$, close to constant, almost same to LSH and ITQ
Experiments

- **Datasets**
  - Euclidean Nearest Neighbor Search (NNS)
    - **SIFT-1M**: 1M 128-D SIFT, **GIST-1M**: 1M 960-D GIST
  - Semantic Nearest Neighbor Search (NNS)
    - **CIFAR-10**: 60K 384-D GIST, **NUS-WIDE**: 269K 4096-D Conv feature

- **Baselines**
  - State-of-the-art unsupervised hashing
    - Projection-based: **LSH**, **ITQ**, **SH**, **AGH**
    - Prototype-based: **SPH**, **ABQ**
  - Multi-table hashing methods: **CH**, **BCH**

- **Settings**
  - 10,000 training samples and 1,000 queries on each set
  - Hash code length: \( B = 24, b = 3 \) (each subspace)


**Experiments: Euclidean NNS**

- **Groundtruth**
  - the top 5% points with the smallest Euclidean distances

- **Hamming distance ranking**
  \[ d(x_q, x_i) = \min_{l=1, \ldots, L} d_h(y_q^{(l)}, y_i^{(l)}) \]

<table>
<thead>
<tr>
<th>Method</th>
<th>SIFT-1M</th>
<th>GIST-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L=1</td>
<td>L=4</td>
</tr>
<tr>
<td>LSH</td>
<td>27.71 ± 0.63</td>
<td>29.40 ± 0.81</td>
</tr>
<tr>
<td>ITQ</td>
<td>41.06 ± 1.53</td>
<td>30.70 ± 0.33</td>
</tr>
<tr>
<td>SH</td>
<td>48.81 ± 0.84</td>
<td>19.64 ± 3.32</td>
</tr>
<tr>
<td>AGH</td>
<td>31.55 ± 1.71</td>
<td>30.01 ± 1.91</td>
</tr>
<tr>
<td>SPH</td>
<td>39.41 ± 0.89</td>
<td>41.49 ± 0.59</td>
</tr>
<tr>
<td>ABQ</td>
<td>51.53 ± 1.30</td>
<td>32.88 ± 0.57</td>
</tr>
<tr>
<td>CH</td>
<td>50.51 ± 0.94</td>
<td>52.28 ± 0.35</td>
</tr>
<tr>
<td>BCH</td>
<td>45.81 ± 0.93</td>
<td>53.30 ± 0.44</td>
</tr>
<tr>
<td>CBQ(OURS)</td>
<td>52.39 ± 0.71</td>
<td>55.95 ± 0.68</td>
</tr>
</tbody>
</table>

Table 1: The AP @ 100 (%) and time cost (seconds) of different hashing methods on SIFT-1M and GIST-1M.

- **Hash table lookup** (Hamming radius ≤3)

<table>
<thead>
<tr>
<th>Method</th>
<th>SIFT-1M</th>
<th>GIST-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1 Measure</td>
<td>Time Cost</td>
</tr>
<tr>
<td>PQ</td>
<td>23.18</td>
<td>1.22</td>
</tr>
<tr>
<td>CBQ</td>
<td>L=8</td>
<td>19.52</td>
</tr>
<tr>
<td>CBQ</td>
<td>L=16</td>
<td>27.08</td>
</tr>
</tbody>
</table>

< 1. SIFT-1M 2. GIST-1M
Experiments: Semantic NNS

- **Groundtruth**
  - those samples with common tags as the query

- **Hamming distance ranking**
  \[ d(x_q, x_i) = \min_{l=1, \ldots, L} d_h(y_q^{(l)}, y_i^{(l)}) \]

<table>
<thead>
<tr>
<th>METHOD</th>
<th>CIFAR-10</th>
<th>NUS-WIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L=1</td>
<td>L=2</td>
</tr>
<tr>
<td>LSH</td>
<td>17.58 ± 0.76</td>
<td>18.40 ± 1.10</td>
</tr>
<tr>
<td>ITQ</td>
<td>31.01 ± 0.63</td>
<td>27.37 ± 0.84</td>
</tr>
<tr>
<td>SH</td>
<td>18.22 ± 0.87</td>
<td>14.84 ± 0.91</td>
</tr>
<tr>
<td>AGH</td>
<td>32.23 ± 1.39</td>
<td>31.11 ± 2.51</td>
</tr>
<tr>
<td>SPH</td>
<td>22.64 ± 1.38</td>
<td>22.17 ± 0.29</td>
</tr>
<tr>
<td>ABQ</td>
<td>18.94 ± 5.26</td>
<td>10.62 ± 0.88</td>
</tr>
<tr>
<td>CH</td>
<td>18.48 ± 0.27</td>
<td>22.02 ± 0.82</td>
</tr>
<tr>
<td>BCH</td>
<td>18.52 ± 1.10</td>
<td>19.95 ± 1.06</td>
</tr>
<tr>
<td>CBQ(OURS)</td>
<td><strong>39.66 ± 1.79</strong></td>
<td><strong>39.37 ± 1.80</strong></td>
</tr>
</tbody>
</table>

Table 2: MAP (%) and time cost (seconds) of different hashing methods on CIFAR-10 and NUS-WIDE.

- **Hash table lookup** (Hamming radius ≤3)
Reference


Thank you!