Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

Qin (Christine) Lv
Stony Brook University

Joint work with Zhe Wang, William Josephson, Moses Charikar, Kai Li (Princeton University)
Motivations

- Massive amounts of feature-rich data
  - Audio, video, digital photos, sensor data, ...

- Fuzzy & high-dimensional
  - Similarity search in high dimensions
  - KNN or ANN in feature-vector space

- Important in various areas
  - Databases, data mining, search engines ...
Ideal Indexing for Similarity Search

- **Accurate**
  - Return results that are close to brute-force search

- **Time efficient**
  - $O(1)$ or $O(\log N)$ query time

- **Space efficient**
  - Small space usage for index
  - May fit into main memory even for large datasets

- **High-dimensional**
  - Work well for datasets with high dimensionality
Previous Indexing Methods

- K-D tree, R-tree, X-tree, SR-tree ...
  - “curse of dimensionality”
  - Linear scan outperforms when $d > 10$ [WSB98]

- Navigating nets [KL04], cover tree [BKL06]
  - Based on “intrinsic dimensionality”
  - Do not perform well with high intrinsic dimensionality

- Locality sensitive hashing (LSH)
Outline

- Motivations
- Locality sensitive hashing (LSH)
  - Basic LSH, entropy-based LSH
- Multi-probe LSH indexing
  - Step-wise probing, query-directed probing
- Evaluations
- Conclusions & future work
LSH: Locality Sensitive Hashing

(r, cr, p₁, p₂)-sensitive [IM98]
- If $D(q,p) < r$, then $Pr [h(q)=h(p)] \geq p₁$
- If $D(q,p) > cr$, then $Pr [h(q)=h(p)] \leq p₂$
- i.e. closer objects have higher collision probability

LSH based on $p$-stable distributions [DIIM04]
- $w$: slot width

$$h_{a,b}(v) = \left\lfloor \frac{a \cdot v + b}{w} \right\rfloor$$
LSH for Similarity Search

- **False positive**
  - Intersection of multiple hashes

- **False negative**
  - Union of multiple hashes
Basic LSH Indexing

- **[IM98, GIM99, DIIM04]**
- **$M$ hash functions per table**
  \[ g_i(v) = (h_{i,1}(v), ..., h_{i,M}(v)) \]
- **$L$ hash tables**
  \[ G = \{ g_1, ..., g_L \} \]
- **Issues:**
  - Large number of tables
  - $L > 100$ in **[GIM99]**
  - $L > 500$ in **[Buhler01]**

Impractical for large datasets
Entropy-Based LSH Indexing

Randomly perturb $q$ at distance $R$

Check hash buckets of perturbed points

Issues:
- Difficult to choose $R$
- Duplicate buckets

Inefficient probing

[Panigrahy, SODA’06]
Outline

Motivations

Locality Sensitive Hashing (LSH)
  - Basic LSH, entropy-based LSH
  - Multi-probe LSH indexing
    - Step-wise probing, query-directed probing

Evaluations

Conclusions & future work
Multi-Probe LSH Indexing

- Probes multiple hash buckets per table

- Perturbs directly on hash values
  - Check left and right slots
  - Perturbation vector $\Delta$
    - $g(q) = (2, 5, 3), \Delta = (-1, 1, 0)$,
    - $g(q) + \Delta = (1, 6, 3)$

- Systematic probing
  - $(\Delta_1, \Delta_2, \Delta_3, \Delta_4, \ldots)$
Multi-Probe LSH Indexing

- A carefully derived probing sequence

Advantages

- Fast probing sequence generation
- No duplicate buckets
- More effective in finding similar objects
Step-Wise Probing

Given $q$’s hash values

1-step buckets: $g(q) = (3,2,5)$

$\Delta = (0,0,1)$

2-step buckets:

- $(2,2,5)$, $(4,2,5)$, $(3,2,6)$
- $\Delta = (-1,-1,0)$

- $(2,1,5)$, $(2,2,6)$, $(3,3,6)$

Intuitions

- 1-step buckets better than 2-step buckets
- All 1-step buckets are equally good

WRONG!
Success Probability Estimation

- Hashed position within slot matters!

- Estimation based on $x_i(-1)$ and $x_i(1)$

\[
Pr[g(p) = g(q) + \Delta] = \prod_{i=1}^{M} Pr[h_i(p) = h_i(q) + \delta_i] \\
\approx \prod_{i=1}^{M} e^{-Cx_i(\delta_i)^2} = e^{-C \sum_{i} x_i((\delta_i)^2)}
\]

\[
\text{score} (\Delta) = \sum_{i=1}^{M} x_i(\delta_i)^2
\]
Query-Directed Probing

\[ g(q) = (h_1(q), h_2(q), h_3(q)) = (2, 5, 1) \]

\[ \{ 0.2, 0.3, 0.4, 0.6, 0.7, 0.8 \} \]

\[ \{ x_3(-1), x_1(1), x_2(-1), x_2(1), x_1(-1), x_3(1) \} \]

\[ \Delta_1 = (0, 0, -1) \]  \( (2, 5, 0) \)

\[ \Delta_2 = (1, 0, 0) \]  \( (3, 5, 1) \)

\[ \Delta_3 = (1, 0, -1) \]  \( (3, 5, 0) \)

score(\( \Delta \)) = \[ \sum_{i=1}^{M} x_i(\delta_i)^2 \]

\[ \{ 0.2 \} \rightarrow \{ 0.2, 0.3 \} \rightarrow \{ 0.2, 0.3, 0.4 \} \]

\[ \{ 0.3 \} \rightarrow \{ 0.3, 0.4 \} \]

\[ \{ 0.4 \} \]
Outline

- Motivations
- Locality Sensitive Hashing (LSH)
  - Basic LSH, entropy-based LSH
- Multi-probe LSH indexing
  - Step-wise probing, query-directed probing
- Evaluations
- Conclusions & future work
Evaluations

Multi-probe vs. basic vs. entropy-based
- Tradeoff among space, speed and quality
- Space reduction

Query-directed vs. step-wise probing
- Tradeoff between search quality and number of probes
Evaluation Methodology

### Benchmarks
- 100 random queries, top K results

### Evaluation metrics
- Search quality: recall, error ratio
- Search speed: query latency
- Space usage: #hash tables

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#objects</th>
<th>#dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web images</td>
<td>1.3 million</td>
<td>64</td>
</tr>
<tr>
<td>Switchboard audio</td>
<td>2.6 million</td>
<td>192</td>
</tr>
</tbody>
</table>

$$\text{recall} = \frac{|I \cap R|}{|I|}$$
Multi-Probe vs. Basic vs. Entropy

Multi-probe LSH achieves higher recall with fewer hash tables.
Space Savings of Multi-Probe LSH

14x - 18x fewer tables than basic LSH
5x - 8x fewer tables than entropy LSH
Multi-Probe vs. Entropy-Based

Multi-probe LSH uses much fewer number of probes
Query-Directed vs. Step-Wise Probing

Query-directed probing uses 10x fewer number of probes
Conclusions

**Multi-probe LSH indexing**

- Systematically probes multiple buckets per hash table
- More space-efficient than basic LSH (14x-18x) and entropy-based LSH (5x-8x)
- More time-efficient than entropy-based LSH
  - 10x fewer number of probes
- Query-directed probing is far superior to step-wise probing
Future Work

- Multi-probe LSH on larger datasets
  - 60 million images, out-of-core, distributed
- Self-tuning
  - Analytical model, LSH Forest
- Compare with other indexing methods
- Evaluate on other data types, features
  - Genomic data, video data, scientific sensor data …
Thanks!

Princeton CASS Project

- Content-Aware Search Systems

Qin (Christine) Lv at Stony Brook

- http://www.cs.sunysb.edu/~qlv