SketchSort: An Efficient Nearest Neighbor Graph Construction Method

Tabei, Yasuo

2010年度科学技術振興機構
ERATO湊離散構造処理系プロジェクト講究録

2011-06

http://hdl.handle.net/2115/48360

conference presentation

ERATO 湊離散構造処理系プロジェクトシンポジウム（第1回）「第1回情報科学技術フォーラム」イベント企画セッション
2010年9月8日（水）九州大学伊都キャンパス
SketchSort: An Efficient Nearest Neighbor Graph Construction Method

Yasuo Tabei
JST Minato Project, Sapporo, Japan

Outline

• Motivation
• Method
• Experiments and Results

Data represented as vector

Text

Vector

\[ x_t = (1, 0, 1, 0, 0, \ldots) \]

Image

\[ x_i = (0.2, -0.3, -1.3, 1.2, 2.2, \ldots) \]

Chemical Compound, Protein, DNA/RNA etc

Locality Sensitive Hashing

(Gionis et al, 99)

• Mapping vector to binary string (sketch)
• Conserve the distance in the original space
  - Enable to store gigascale data in main memory
  - Speed up learning algorithms

\[ x = (0.2, -0.3, -1.3, 1.2, 2.2, \ldots) \]

\[ s = 10101011101010101 \]

All Pairs Similarity Search

• Finding all neighbor pairs from sketches
  - Find all pairs \((i, j), i < j, \) \[ \Delta(x_i, x_j) \leq \epsilon \]
• Enable to build a neighborhood graph
  - semi-supervised learning, spectral clustering, ROI detection in images, retrieval of protein sequences

Single Sorting Method (SSM)

• Find neighbors by sorting sketches
  - Various applications ex) google news

<table>
<thead>
<tr>
<th>(a) Input data</th>
<th>(b) Sort</th>
<th>(c) Scan neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:101111</td>
<td>7:000000</td>
<td>8:010110</td>
</tr>
<tr>
<td>2:110101</td>
<td>4:010000</td>
<td>10:100100</td>
</tr>
<tr>
<td>3:110010</td>
<td>8:010110</td>
<td>5:101000</td>
</tr>
<tr>
<td>4:010000</td>
<td>10:100100</td>
<td>5:101000</td>
</tr>
<tr>
<td>5:101000</td>
<td>5:101000</td>
<td>3:110010</td>
</tr>
<tr>
<td>6:111110</td>
<td>1:101111</td>
<td>9:110110</td>
</tr>
<tr>
<td>7:000000</td>
<td>3:110010</td>
<td>9:110110</td>
</tr>
<tr>
<td>8:010110</td>
<td>2:110101</td>
<td>2:110101</td>
</tr>
<tr>
<td>9:110110</td>
<td>9:110110</td>
<td>6:111100</td>
</tr>
<tr>
<td>10:100100</td>
<td>6:111100</td>
<td></td>
</tr>
</tbody>
</table>
Drawbacks of Single Sorting

- Need a large number of distance calculation for achieving reasonable accuracy.
- Can not derive an analytic estimate of the fraction of missing neighbors.

Overview of SketchSort

- Employ the multiple sorting method (MSM) as a building block
  - Enumerate all pairs within Hamming distance $d$ from a string pool $S = \{s_1, ..., s_n\}$
  - A number of distance calculation is significantly reduced
  - A bound of the expected fraction of missing neighbors can be obtained.

Special case: Finding identical strings ($d=0$)

- Radix sort, and partition the strings into equivalence classes: $O(n)$
- Build edges between all pairs in equivalent classes: $O(m)$
- Complexity: $O(n+m)$

Multiple sorting method ($d > 0$)

- Mask $d$ characters in all possible ways
- Perform radix sort $\binom{l}{d}$ times
- Time exponential to $d$, polynomial to the string length!
- Still linear to the number of strings!!
- $\text{Ex}) d=2$

Blockwise masking

- Mask $d$ blocks in all possible ways
- The number of sorting operations reduced
- Non-neighbors might be detected
  - Filtered out by calculating actual Hamming distances
  - $\text{Ex} d=2$

Recursive Algorithm

Figure 5: Updating equivalence classes in block concatenation. Strings in a block are sorted and equivalence classes (shown as square frames) are detected. A next block is concatenated to each equivalence class and sorted again.
SketchSort

- Basic idea: Map vectors to strings and apply MSM
- Not good: Create long strings and apply MSM at once
- Replication:
  - Create $Q$ independent string pools of length $l$
  - apply MSM to each string pool
- Report the pairs less than a threshold $\varepsilon$
  $\Delta(x_i, x_j) \leq \varepsilon$

Duplication Checks

- Block-level duplication check
  - Define dictionary order of blocks, and take only minimum combinations of blocks.
  - Example: $d=2$
    $(1,2)<(1,3)<(1,4)<(2,3)<(2,4)<(3,4)$
- Chunk-level duplication check
  - Take only minimum chunks.

Two types of errors

- True edges $E^*$, Our results $E$
- Type-I error (false positive): A non-neighbor pair has a Hamming distance within $d$ in at least one replicate
  $F_1 = \{(i, j) \mid (i, j) \in E, (i, j) \notin E^*\}$
- Type II-error (false negative): A neighbor pair has a Hamming distance larger than $d$ in all replicates
  $F_2 = \{(i, j) \mid (i, j) \notin E, (i, j) \in E^*\}$

Bound of type-II error: Missing edge ratio

- Basically, type-II error is more crucial
- Type-I errors are filtered out by distance calculations
- Missing edge ratio (type-II error) is bounded as
  $$E \left[ \frac{|F_2|}{|E^*|} \right] \leq \left( 1 - \sum_{k=0}^{d-1} \binom{\ell}{k} p^k (1-p)^{\ell-k} \right)^Q,$$
  where $p$ is an upper bound of the non-collision probability of neighbors
  $$p = \frac{\arccos(1 - \varepsilon)}{\pi}.$$

Results for All Pairs Similarity Search

Faster and more accurate than recent methods

Results for 5-nearest neighbor search

Error rate for 5-nearest neighbor search on MNIST and TinyImage datasets
All Pairs Similarity Search in 1.6 Million Images

- Set parameters so as to keep missing edge ratio no more than $1.0 \times 10^{-6}$
- Enable to detect similar pairs nearly exactly
- Take only 4.3 hours for 1.6 million images

Near duplication detection in up to 1.6 million images at threshold $0.05\Pi$ (left), $0.10\Pi$ (middle) and $0.15\Pi$ (right)

A C++ implementation of SketchSort is available from 
http://code.google.com/p/sketchsort/

Format of input file

Usage

Quick Start

Introduction