SketchSort: Fast All Pairs Similarity Search Method

Yasuo Tabei (JST Minato ERATO Project)
Joint work with Takeaki Uno (NII),
Masashi Sugiyama (TITECH), Koji Tsuda (AIST)
Outline

- **Motivation**
  - Development of large-scale data
  - Needs for all pairs similarity search method
  - Single sorting method and its drawbacks

- **Method**
  - Multiple sorting method
  - Locality sensitive hashing
  - SketchSort

- **Experiments**
  - Comparison of other state-of-the-art methods
  - Use large-scale image datasets
Data is dramatically increasing...

- Images
  - flicker, image search etc
  - 80 million tiny images (Torralba et al., 2008)

- Chemical Compounds
  - NCBI PubChem
  - 28 million chemical compounds

- Genome Sequences
  - NCBI Sequence Read Archive
  - A large-scale genome sequences from various organisms
Vector Representation

Image

Chemical Compound

Text, Protein, DNA/RNA etc
Locality Sensitive Hashing (LSH) (Gionis et al., 1999)

- Mapping vector to binary string (sketch)
  - Conserve the distance in the original space

\[ x = (0.3, 0.1, 0.5, 0.6, 0.7, 1.2, -0.2, \ldots) \]
\[ s = 1010010001110001010\ldots \]

- Advantage
  - Can keep giga-scale data in main memory
  - Accelerate various algorithms
Finding all neighbor pairs from vector data
- Given a set of data points
- Find all pairs within a distance \( \Delta(x_i, x_j) \leq \varepsilon \)
Usefulness

- Can build a neighborhood graph
  - Vertex: a data-point
  - Edge: a neighbor pair

Applications: semi-supervised learning, spectral clustering, ROI detection in images, retrieval of protein sequences, etc
Single Sorting Method (SSM) (Charikar, 2002)

- Finding neighbor pairs by sorting
- Map vector data to sketches

(a) Input
1: 101111
2: 110101
3: 110010
4: 010000
5: 101000
6: 111100
7: 000000
8: 010110
9: 110110
10: 100100

(b) Sort
7: 000000
4: 010000
8: 010110
10: 100100
5: 101000
1: 101111
3: 110010
2: 110101
9: 110110
6: 111100

(c) Scan neighbors
7: 000000
4: 010000
8: 010110
10: 100100
5: 101000
1: 101111
3: 110010
2: 110101
9: 110110
6: 111100
Drawbacks of Single Sorting Method

- Need a large number of distance calculation for achieving reasonable accuracy
- Can not derive an analytical estimation of the fraction of missing neighbors
Outline

- Motivation
  - Large-scale data
  - Needs for all pairs similarity search method
  - Single sorting method, and its drawbacks

- Method
  - Multiple sorting method
  - Locality sensitive hashing
  - SketchSort

- Experiments
  - Comparison of other state-of-the-art methods
  - Use large-scale image datasets
Brief overview of Multiple Sorting Method (Uno,08)

- Input: set of fixed-length strings $S=\{s_1,\ldots,s_n\}$
- Output: all pairs of strings within a Hamming distance $d$
- By applying radixsort, enumerate all pairs in $O(n+m)$. 
  - $n$: number of strings, $m$: output pairs
- Introduce block-wise masking technique for acceleration
Input fixed length strings
s1=01001001001010100
s2=11010010010101010
s3=00010100100100100
...

Enumerate all pairs which share matching block combinations by MSM
s6: 0011 0111
s8: 0011 0111

Block level duplication?
Yes → Trash
No → Check Hamming Distance?

Check Hamming Distance?
Yes → Trash
No → ≤d

Output
Special case: detection of identical strings \((d=0)\)

- Sort strings by radixsort, divide strings into equivalence classes \(O(n)\)
- Draw edges within all strings in an equivalence class \(O(m)\)
- Computational Complexity: \(O(n+m)\)
Multiple Sorting Method (d > 0)

- Mask d characters in all possible ways
- Performe radixsort \( \binom{l}{d} \) times
- Linear time to the number of strings
- Time exponential to d, polynomial to the length of strings | l
- Ex) d=2

```
7: 0000 0001 0011 1110
4: 0100 0001 1101 1100
8: 0100 1001 0111 1000
10: 1001 0011 1001 0111
5: 1010 0010 1110 1010
3: 1100 1000 1101 1100
6: 1111 0011 1001 0111
1: 1011 1111 0011 1110
2: 1101 0111 0111 0001
9: 1101 1000 1101 1110
10: 1001 0011 1001 0111
```
Mask d blocks in all possible ways
Can reduce the number of sorting operations
Non-neighbor might be detected
Filter out by calculating actual Hamming distance
Ex) d=2
Step 1. Perform radixsort in a block, and detect equivalence classes.

Step 2. For each equivalence class, perform radixsort the next block.
- All neighbor pairs can be enumerated
The same pair can be detected in different block combinations
Ex) (3,9), (6,10)

Naïve method takes $n^2$ memory
Step 1: Make a total order among blocks from left to right
Step 2: Make a total order among block combinations
Step 3: Take the minimum among matching block combinations
Pseudo Code

1: function MULTIPLESORTINGMETHOD
2: \[ I \leftarrow \{1, \ldots, n\} \]
3: \[ B \leftarrow \emptyset \]
4: RECURRENCE(I, B)
5: return
6: end function

7: function RECURRENCE(I, B)
8: if \(|B| = k - d\) then
9: \quad for \((i, j) \in I \times I, i < j\) do
10: \quad \quad if \(s^b_i \neq s^b_j\) for all \(b < \max(B), b \notin B\) then
11: \quad \quad \quad if \(HamDist(s_i, s_j) \leq d\) then
12: \quad \quad \quad \quad Report \((i, j)\) to output file
13: \quad \quad end if
14: \quad end if
15: end for
16: return
17: end if

18: for \(b \in (\max(B) + 1) \ldots (k + |B| + 1)\) do
19: \quad \(J \leftarrow\) Sorted indices based on \(b\)-th block \(\{s^b_i\}_{i \in I}\)
20: \quad \(T \leftarrow\) Intervals of equivalence classes in \(\{s^b_j\}_{j \in J}\)
21: \quad for each interval \((x, y) \in T\) do
22: \quad \quad \(\text{RECURRENCE}(J[x : y], B \cup b)\)
23: \quad end for
24: end for
25: return
26: end function

If the number of blocks is \(k-d\), eliminate duplicate pairs, and calculate Hamming distance.

Call function to equivalence classes.
SketchSort (original)

- Enumerates all neighbor pairs within a distance
  \((x_i, x_j), \ i < j, \ \Delta(x_i, x_j) \leq \varepsilon,\)

- Basic idea
  - Map vector data to sketches by LSH
  - Enumerate all neighbor pairs by MSM

- SketchSort with cosine LSH
  - Enumerate all neighbor pairs within a cosine distance threshold \(\varepsilon\)
  - \((x_i, x_j), \ i < j, \ \Delta(x_i, x_j) = 1 - \frac{x_i^T x_j}{\|x_i\| \|x_j\|} \leq \varepsilon\)
Flowchart

Input vector data
x1=(0.1, 1.2, -0.9, 2.3, …)
x2=(1.5, -0.1, -1.2, -1.2, …)
x3=(-1.6, 1.9, 0.5, -0.6, …)
...

Mapping to sketches by LSH
s1=1000101001...
s2=0110000101…
s3=0101001000…

Enumerating matching block combinations by MSM
s6
s8

Block level duplication?
Yes
Trash
No

Check Hamming Distance?
Yes
Trash
No

Chunk Level Duplication
Yes
Trash
No

Check Cosine Distance
Yes
Trash
No

Output

\[ \geq \epsilon \]

\[ \leq \epsilon \]
Cosine LSH (Gionis et al., 1999)

- Basic idea
  - Generate a random hyperplain centered at 0
  - Map each data-point to ‘1’ if it is above the hyperplain, or else ‘0’

- Repeat \( l \) times

![Diagram of hyperplane with data points]
SketchSort

- Basic idea: Map vector data to sketches and apply MSM
- Not good: create long sketches and apply MSM at once
- Divide long sketches to $Q$ short sketches of length $l$ (chunks)
- Apply MSM to each chunk, obtain neighbor pairs w.r.t Hamming distance $E_q = \{(i, j) \mid HamDist(s^q_i, s^q_j) \leq d, i < j\}$.

- Report neighbor pairs no more than a cosine distance threshold $\varepsilon$
A neighbor pair of sketches can be detected in several chunks within Hamming distance $d$

$S_i$: 1101010011010010 11101001001010 1010100101101101 10101110011101 1110101010001010 1110101010001010

$S_j$: 0101001010010011 0101001010001010 01010101001010 10101100111101 1010101010001010 10101001001010

$\text{HamDist}(s_i^1, s_j^1) > d$

$\text{HamDist}(s_i^3, s_j^3) > d$

$\text{HamDist}(s_i^5, s_j^5) \leq d$

$\text{HamDist}(s_i^2, s_j^2) \leq d$

$\text{HamDist}(s_i^4, s_j^4) > d$

$\text{Duplication!!}$

The same pair is outputted several times (Duplication)
Step 1: Order chunks from left to right.

1 2 3 4 5
1100101010010101 010101010010101 1010101010111111 101010101001001 1101010100110101 0101010101010101 0101000101010101 010101001010100 1110101010011111 111111111010011 ...

Step 2: Check whether left chunks are no more than Hamming distance \( d \)

\[
\begin{align*}
\text{HamDist}(s_i^1, s_j^1) &> d? \\
\text{HamDist}(s_i^3, s_j^3) &> d? \\
\text{HamDist}(s_i^5, s_j^5) &\leq d \\
\text{HamDist}(s_i^2, s_j^2) &> d? \\
\text{HamDist}(s_i^4, s_j^4) &> d?
\end{align*}
\]

- If such chunk is found, trash the pair,
- Or else check cosine distance
Pseudo Code

1: function SKETCHSORT($x_1, \ldots, x_n$)
2: Use LSH to obtain sketches $\{s_{q_i}, \ldots, s_{q_j}\}_{i=1}^n$ from data $\{x_i\}_{i=1}^n$
3: $I \leftarrow \{1, \ldots, n\}$
4: for $q = 1 : Q$ do
5: $B \leftarrow \emptyset$
6: RECURSION($I$, $B$, $q$)
7: end for
8: return
9: end function

10: function RECURSION($I$, $B$, $q$)
11: if $|B| = k - d$ then
12: for $(i, j) \in I \times I, i < j$ do
13: if $s_{qi} \neq s_{qj}$ for all $b < \max(B), b \notin B$ then
14: if $\text{HamDist}(s_{qi}, s_{qj}) \leq d$ then
15: if $\text{HamDist}(s_{ri}, s_{rj}) > d$ for all $r < q$
16: then
17: if $\Delta(x_i, x_j) \leq \epsilon$ then
18: Report $(i, j)$ to output file
19: end if
20: end if
21: end if
22: end for
23: end if
24: end if
25: for $b$ in $(\max(B) + 1) \ldots (k + |B| + 1)$ do
26: $J \leftarrow$ Sorted indices based on $b$-th block $\{s_{qi}\}_{i \in I}$
27: $T \leftarrow$ Intervals of equivalence classes in $\{s_{qj}\}_{j \in J}$
28: for each interval $(x, y) \in T$ do
29: RECURSION($J[x : y], B \cup b, q$)
30: end for
31: end for
32: return
33: end function

- Call function for each chunk
- Check duplication four times
- Divide sketches into equivalence classes
- Call function recursively
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 chunk
  \[ F_1 = \{(i, j) \mid (i, j) \in E, (i, j) \notin E^*\}. \]
- Type II-error (false negative): A neighbor pair has a Hamming distance larger than $d$ in all chunks
  \[ F_2 = \{(i, j) \mid (i, j) \notin E, (i, j) \in E^*\}. \]
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
  \[
  E \left[ \frac{|F_2|}{|E^*|} \right] \leq \left( 1 - \sum_{k=0}^{\left\lfloor \frac{d}{\ell} \right\rfloor} \binom{\ell}{k} p^k (1 - p)^{\ell-k} \right)^Q,
  \]
  where \( p \) is an upper bound of the non-collision probability of neighbors

  \[
  p = \frac{\arccos(1 - \epsilon)}{\pi}.
  \]
Outline

- Motivation
  - Large-scale data
  - Needs for All Pairs Similarity Search Method
  - Single Sorting Method, and its drawbacks
- Method
  - Multiple Sorting Method
  - Cosine Locality Sensitive Hashing
  - SketchSort
- Experiments
  - Comparison of other state-of-the-art methods
  - Use large-scale image datasets
Two image datasets
- MNIST (60,000 data, 748 dimension)
- Tiny Image (100,000 data, 960 dimension)

Use missing edge ratio as an evaluation measure
Set cosine distance threshold of $0.15\pi$
Length of each chunk to 32bit
Hamming distance and number of blocks are set to $(2,5)$ and $(3,6)$.
Number of chunks vary from 2, 4, 6, ..., 50
Compare our method to Lanczos bisection method (JMLR, 2009)
MNIST Threshold 0.15π
Tiny Images Threshold 0.15π
K-nearest neighbor

- K-nearest neighbor graph construction by SketchSort
  - Keep \(k\)-nearest neighbor pairs by priority queue
- Compare SketchSort to
  - Cover Tree (Beygelzimer et al., ICML 2006)
  - AllKNN (Ram et al., NIPS 2009),
  - Lanczos-bisection (JMLR, 2009)
5-nearest neighbor graphs on MNIST competitive
5-nearest neighbor on Tiny Images is competitive.
Experiments on 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
Summary

- Fast all pairs similarity search method
- Applicable to large-scale vector data
- Applicable to Euclidean distance (Raginsky, 10), Jaccard-coefficients (Broder, 00)
- Various applications
- Software
  - [http://code.google.com/p/sketchsort/](http://code.google.com/p/sketchsort/)