



HOKKAIDO UNIVERSITY

Title	SketchSort: An Efficient Nearest Neighbor Graph Construction Method
Author(s)	Tabei, Yasuo
Description	ERATO 湊離散構造処理系プロジェクト春のワークショップ（キックオフシンポジウム）．2010年5月28日（金）～29日（土）．ERATO湊プロジェクト研究室．
Relation	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．p.307-310.
Issue Date	2011-06
Doc URL	https://hdl.handle.net/2115/48402
Type	conference presentation
File Information	10.tabei.pdf



ERATO Minato project kickoff meeting@Hokkaido University

SketchSort: An Efficient Nearest Neighbor Graph Construction Method

Yasuo Tabei
JST Minato Project, Sapporo, Japan

SketchSort: An Efficient Nearest Neighbor Graph Construction Method

- 高速な近傍グラフ構築手法の提案
 - Input: データ点の集合 Output: 距離 ϵ 以内の点ペア
- LSH + Multiple Sorting Method (Uno 08)
 - LSH: ベクトルデータを距離関係を保持したまま、バイナリーの文字列に射影する
 - MSM: 文字列集合から、ハミング距離 d 以内の文字列ペアを列挙する
- Missing Neighborの理論的な見積もり

$$E \left[\frac{|F_2|}{|E^*|} \right] \leq \left(1 - \sum_{k=0}^{\lfloor d \rfloor} \binom{\ell}{k} p^k (1-p)^{\ell-k} \right)^Q,$$

- 大規模画像上で既存手法と比較することにより、提案手法の有効性を示した。

Outline

- Motivation
- Method
- Experiments and Results

Data represented as vector

Text



Vector

$$\mathbf{x}_i = (1, 0, 1, 0, 0, \dots)$$

Image



$$\mathbf{x}_i = (0.2, -0.3, -1.3, 1.2, 2.2, \dots)$$

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

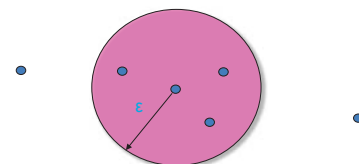
$$\mathbf{x} = (0.2, -0.3, -1.3, 1.2, 2.2, \dots)$$



$$\mathbf{s} = 10101011101010101$$

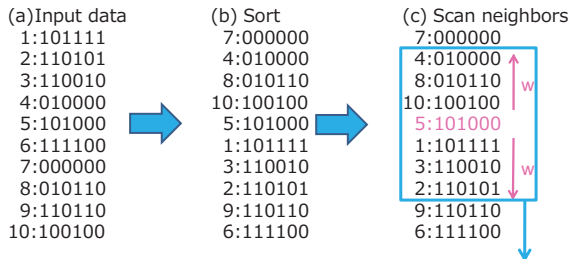
All Pairs Similarity Search

- Finding all neighbor pairs from sketches
 - Find all pairs (i, j) , $i < j$, $\Delta(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon$
- 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



Drawbacks of Single Sorting

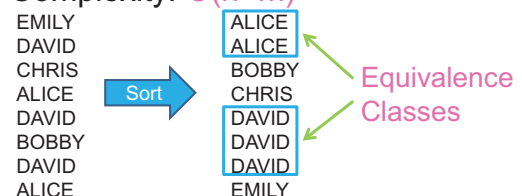
- Need a large number of distance calculation for achieving reasonable accuracy.
- Can not derive an analytic estimate of the fraction of missing neighbors.

Overview of SketchSort

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

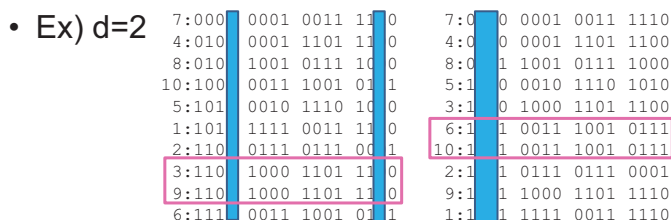
Special case: Finding identical strings ($d=0$)

- 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 ($d > 0$)

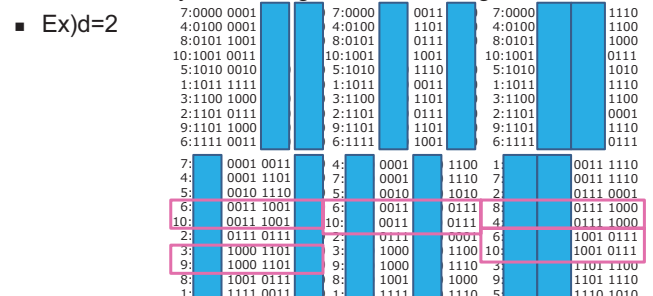
- Mask d characters in all possible ways
- Perform radix sort $\binom{l}{d}$ times
- Time exponential to d , polynomial to the string length l
- Still linear to the number of strings!!



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



Recursive Algorithm

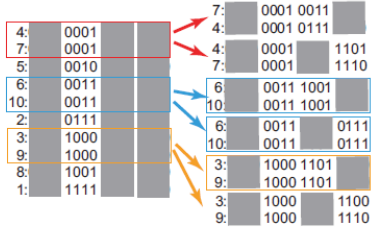


Figure 5: Updating equivalence classes in block concatenation. Strings in a block are sorted and equivalence classes (shown as square frames) are detected. A next block is concatenated to each equivalence class and sorted again.

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 ϵ

$$\Delta(x_i, x_j) \leq \epsilon$$

Duplication Checks

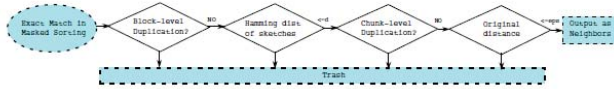


Figure 2: Global flow of our approach.

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

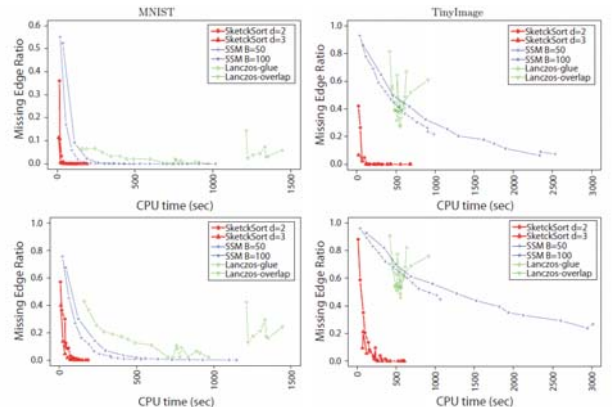
$$E \left[\frac{|F_2|}{|E^*|} \right] \leq \left(1 - \sum_{k=0}^{\lfloor d \rfloor} \binom{\ell}{k} p^k (1-p)^{\ell-k} \right)^Q,$$

where p is an upper bound of the non-collision probability of neighbors

$$p = \frac{\arccos(1 - \epsilon)}{\pi}.$$

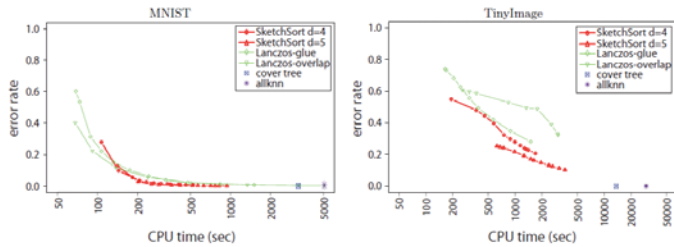
Results for All Pairs Similarity Search

Faster and more accurate than recent methods



All pairs similarity search on MNIST and TinyImage datasets for cosine distance thresholds 0.10π (top) and 0.15π (bottom).

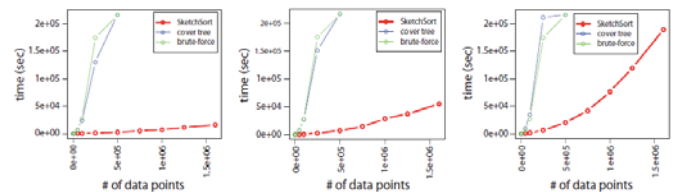
Results for 5-nearest neighbor search



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 1.0×10^{-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 (left), 0.10 (middle) and 0.15 (right)

A C++ implementation of SketchSort is available from

<http://code.google.com/p/sketchsort/>

The screenshot shows the SketchSort project page on Code.google.com. The page has a header with the project name and a search bar. Below the header, there is a navigation bar with links for Project Home, Downloads, Wiki, Issues, Source, and Administrator. The main content area is divided into several sections: Introduction, Quick Start, Usage, and Format of input file. The Introduction section describes the project's goal and the SketchSort algorithm. The Quick Start section provides instructions on how to compile and run the program. The Usage section lists the command-line options for the program. The Format of input file section describes the format of the input data file. The right sidebar contains links for downloading the project, viewing the source code, and viewing the project's history.