Locality Sensitive Hashing and its Application

Rice University

Anshumali Shrivastava

anshumali At rice.edu

27th Jan 2016

Pairwise Comparisons Everywhere

- Near Duplicate Detections over web. (mirror pages)
- Plagiarism Detection
- Find Customers With Similar Taste.
- Movie Recommendations. (Find Similar profiles)

Activity : Exact Duplicates

Remove all repeated items in an array
example {1,2,3,8,2,7,3,3,4,8,9}

Activity : Exact Duplicates

Remove all repeated items in an array
example $\{1,2,3,8,2,7,3,3,4,8,9\}$

$O(n)$ or $O(n^2)$

Activity : Exact Duplicates

Remove all repeated items in an array
example $\{1,2,3,8,2,7,3,3,4,8,9\}$

$O(n)$ or $O(n^2)$

Array of vectors instead of numbers ?

Subroutine of Interest : Similarity Search

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

$$p = \arg\max_{x \in \mathcal{C}} \quad sim(q, x)$$

Subroutine of Interest : Similarity Search

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

$$p = \arg\max_{x \in \mathcal{C}} \quad sim(q, x)$$

- *sim* is the similarity, like Cosine Similarity, Resemblance, etc.
- Worst case $O(N)$ for any query. N is huge.
- Querying is a very frequent operation.

Subroutine of Interest : Similarity Search

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

$$p = \arg\max_{x \in \mathcal{C}} \quad sim(q, x)$$

- *sim* is the similarity, like Cosine Similarity, Resemblance, etc.
- Worst case $O(N)$ for any query. N is huge.
- Querying is a very frequent operation.

Our goal is to find sub-linear query time algorithm.

Subroutine of Interest : Similarity Search

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

$$p = \arg\max_{x \in \mathcal{C}} \quad sim(q, x)$$

- *sim* is the similarity, like Cosine Similarity, Resemblance, etc.
- Worst case $O(N)$ for any query. N is huge.
- Querying is a very frequent operation.

Our goal is to find sub-linear query time algorithm.

1. Approximate answer suffices.
2. We are allowed to pre-process $\mathcal{C}$ once. (offline costly step)

Locality Sensitive Hashing

Hashing: Function (randomized) h that maps a given data vector $x \in \mathbb{R}^D$ to an integer key $h : \mathbb{R}^D \mapsto \{0, 1, 2, ..., N\}$

Locality Sensitive Hashing

Hashing: Function (randomized) h that maps a given data vector $x \in \mathbb{R}^D$ to an integer key $h : \mathbb{R}^D \mapsto \{0, 1, 2, ..., N\}$

Locality Sensitive: Additional property

$$Pr_h\big[h(x) = h(y)\big] = f(sim(x, y)),$$

where f is monotonically increasing. *sim* is any similarity of interest.

Locality Sensitive Hashing

Hashing: Function (randomized) h that maps a given data vector $x \in \mathbb{R}^D$ to an integer key $h : \mathbb{R}^D \mapsto \{0, 1, 2, ..., N\}$

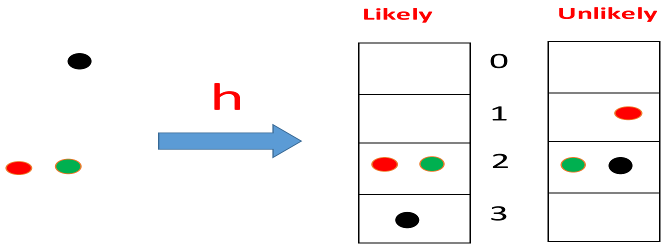
Locality Sensitive: Additional property

$$Pr_h\Big[h(x) = h(y)\Big] = f(sim(x, y)),$$

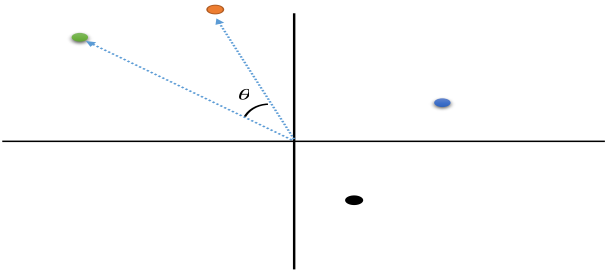
where f is monotonically increasing. *sim* is any similarity of interest.

Similar points are more likely to have the same hash value (hash collision).

Question: Does this definition implies the definition given in the book ?



Signed Random Projections (SimHash)

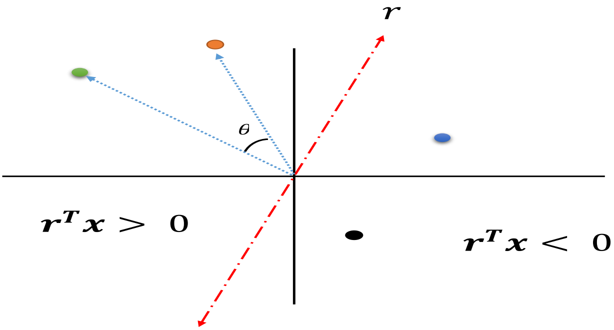


$$h_r(x) = \begin{cases} 1 & \text{if } r^T x \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad r \in \mathbb{R}^D \sim N(0, \mathcal{I})$$

$$Pr_r(h_r(x) = h_r(y)) = 1 - \frac{\theta}{\pi}, \quad \text{monotonic in cosine similarity } \theta = cos^{-1}\mathcal{S}$$

A classical result from Goemans-Williamson (95)

Signed Random Projections (SimHash)



$$h_r(x) = \begin{cases} 1 & \text{if } r^T x \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad r \in \mathbb{R}^D \sim N(0, \mathcal{I})$$

$$Pr_r(h_r(x) = h_r(y)) = 1 - \frac{\theta}{\pi}, \quad \text{monotonic in cosine similarity } \theta = cos^{-1}\mathcal{S}$$

A classical result from Goemans-Williamson (95)

LSH for Estimation

We have

$$Pr_h\big[h(x) = h(y)\big] = f(sim(x, y)),$$

where f is monotonically increasing.

LSH for Estimation

We have

$$Pr_h\big[h(x) = h(y)\big] = f(sim(x, y)),$$

where f is monotonically increasing.

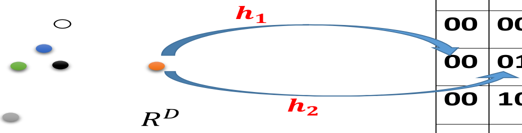
Activity: Design a strategy for estimating $sim(x, y)$ given access to values of $h(x)$ and $h(y)$, with h sampled independently.

Sub-linear Near Neighbor Search: Idea

Given: $Pr_h\big[h(x) = h(y)\big] = f(sim(x,y))$, f is monotonic.

Sub-linear Near Neighbor Search: Idea

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x,y))$, f is monotonic.

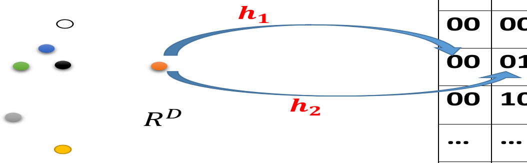


$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	
00	01	
00	10	
…	…	
11	11	

Sub-linear Near Neighbor Search: Idea

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x,y))$, f is monotonic.

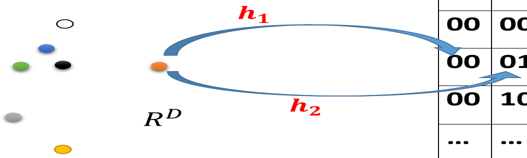


$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

Sub-linear Near Neighbor Search: Idea

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x,y))$, f is monotonic.



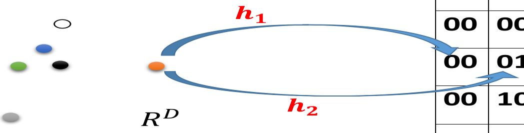
$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

- Given query q, if $h_1(q) = 11$ **and** $h_2(q) = 01$, then probe bucket with index **1101**. It is a good bucket !!

Sub-linear Near Neighbor Search: Idea

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x, y))$, f is monotonic.



$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

- Given query q, if $h_1(q) = 11$ **and** $h_2(q) = 01$, then probe bucket with index **1101**. It is a good bucket !!
- (Locality Sensitive) $h_i(q) = h_i(x)$ implies high similarity.
- Doing better than random !!

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	...
00	...	01	...
00	...	10	Empty
...	...	...	...
11	...	11	...

- We use K concatenation.

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	● ● ...
00	...	01	● ○ ...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	● ● ...
00	...	01	● ● ...
00	...	10	○ ● ●
...	...	...	...
11	...	11	Empty

- We use K concatenation.
- Repeat the process L times. (L Independent Hash Tables)

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	...
00	...	01	...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	...
00	...	01	...
00	...	10	
...	...	...	...
11	...	11	Empty

- We use K concatenation.
- Repeat the process L times. (L Independent Hash Tables)
- **Querying :** Probe one bucket from each of L tables. Report union.

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	...
00	...	01	...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	...
00	...	01	...
00	...	10	
...	...	...	...
11	...	11	Empty

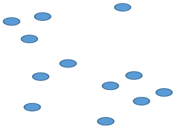
- We use K concatenation.
- Repeat the process L times. (L Independent Hash Tables)
- **Querying :** Probe one bucket from each of L tables. Report union.

1. Two knobs K and L to control.
2. **Theory says we have a sweet spot.** Provable sub-linear algorithm. (Indyk & Motwani 98)

A Real Problem: Avoiding Quadratic

Dataset of around 250,000 Syrian death records from 7 sources.

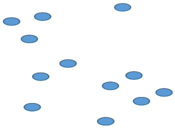
- A very short noisy text description of who died.
- Arabic suffixes and prefixes have many ambiguities.
- Selection biases.

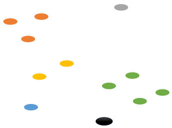


A Real Problem: Avoiding Quadratic

Dataset of around 250,000 Syrian death records from 7 sources.

- A very short noisy text description of who died.
- Arabic suffixes and prefixes have many ambiguities.
- Selection biases.

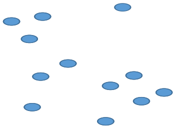


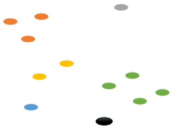


A Real Problem: Avoiding Quadratic

Dataset of around 250,000 Syrian death records from 7 sources.

- A very short noisy text description of who died.
- Arabic suffixes and prefixes have many ambiguities.
- Selection biases.





Many records correspond to the same individual.

Problem: Can we estimate how many people died ? (**Record Linkage**)

A Real Problem: Avoiding Quadratic

Dataset of around 250,000 Syrian death records from 7 sources.

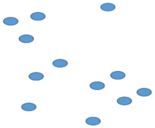
- A very short noisy text description of who died.
- Arabic suffixes and prefixes have many ambiguities.
- Selection biases.

Many records correspond to the same individual.

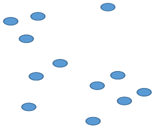
Problem: Can we estimate how many people died ? (**Record Linkage**)

Reasonable Idea: Try predicting match/mismatch given a pair.
Concern: Just too many pairs ! (3.1×10^{10})

Reducing Potential Pairs via Hashing



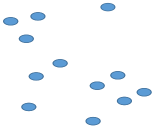
Reducing Potential Pairs via Hashing



h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

h_3	h_4	Buckets (pointers only)
00	00	...
00	01	...
00	10	
...	...	...
11	11	Empty

Reducing Potential Pairs via Hashing

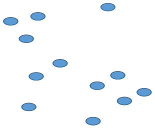


h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

h_3	h_4	Buckets (pointers only)
00	00	...
00	01	...
00	10	
...	...	...
11	11	Empty

- Co-occurrence in bucket mean high resemblance between records.

Reducing Potential Pairs via Hashing

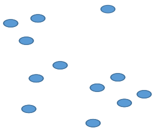


h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

h_3	h_4	Buckets (pointers only)
00	00	...
00	01	...
00	10	
...	...	...
11	11	Empty

- Co-occurrence in bucket mean high resemblance between records.
- Only form pairs within each bucket.

Reducing Potential Pairs via Hashing



h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

h_3	h_4	Buckets (pointers only)
00	00	...
00	01	...
00	10	
...	...	...
11	11	Empty

- Co-occurrence in bucket mean high resemblance between records.
- Only form pairs within each bucket.
  1. All operations near linear.
  2. **99% recall** and only evaluate **1% of the total pairs**.

Brain Storm Activity : Graph Matching !

- Given a collection of n graphs find a reasonable routine to remove isomorphic (identical or duplicates) graphs
- Assume you have an subroutine $isIsomorphic(G_1, G_2)$. Try to avoid quadratic call to this subroutine.

Brain Storm Activity : Graph Matching !

- Given a collection of n graphs find a reasonable routine to remove isomorphic (identical or duplicates) graphs
- Assume you have an subroutine $isIsomorphic(G_1, G_2)$. Try to avoid quadratic call to this subroutine.

Any real application ?