universidade de aveiro

Mining Large Scale Datasets

Locality-Sensitive Hashing

(Adapted from CS246@Starford.edu; http://www.mmds.org)

Sérgio Matos - aleixomatos@ua.pt

Locality-Sensitive Hashing

- Family of related techniques
- Allows to only examine pairs that are likely to be similar

  Avoids quadratic growth in computation time

LSH: general idea

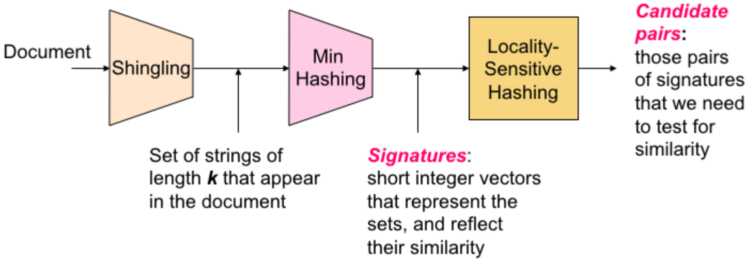
- Hash items into buckets using many different hash functions

  ↪ Functions are designed so that similar items are more likely to hash into same bucket
- Only pairs that share a bucket for at least one of the hash functions need to be examined

  ↪ There may be false negatives – pairs of similar items may not be considered at all

  ↪ There may be false positives – pairs of items may be erroneously found as similar

Similar Documents: Steps



- **Shingling**: Converts a document into a set representation
- **Min-Hashing**: Convert large sets to short signatures, while preserving similarity
- **Locality-Sensitive Hashing**: Focus on pairs of signatures likely to be from similar documents

Min-Hashing

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

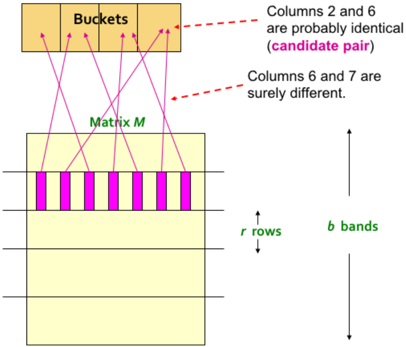
Signature matrix

2	1	2	1
2	1	4	1
1	2	1	2

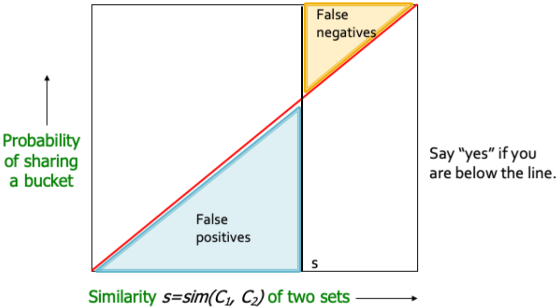
Similarities

1-2	1-3	2-4	3-4	
0	.75	.75	0	Original
0	.67	1	0	Signatures

Locality-Sensitive Hashing

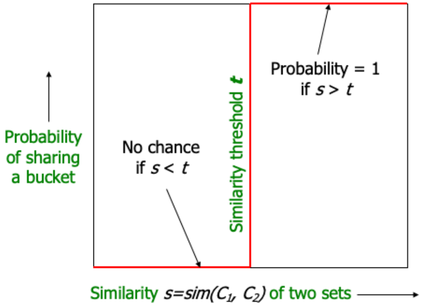


Locality-Sensitive Hashing



LSH with a single min hash function (one band of one row)

Locality-Sensitive Hashing

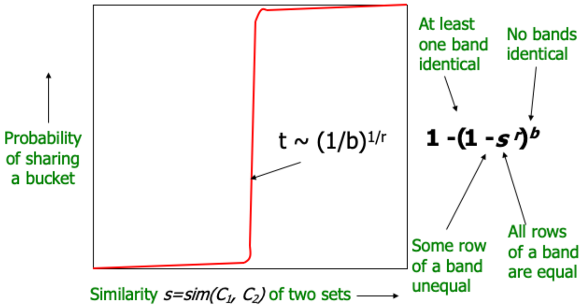


LSH – Optimal scenario: only pairs of sets with similarity $>$ t are selected as candidates

Locality-Sensitive Hashing

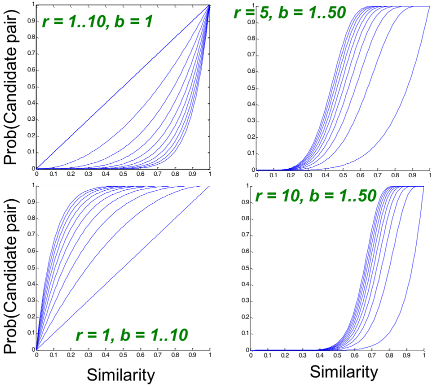
- b bands, r rows/band
- Consider columns C1 and C2 with similarity s
- For any band (r rows):
  - Probability that all rows in band are equal $= s^r$
  - Probability that some row in band is unequal $= 1 - s^r$
  - Probability that no band identical $= (1 - s^r)^b$
  - Probability that at least 1 band identical $= 1 - (1 - s^r)^b$

Locality-Sensitive Hashing: S-curve



LSH – *b* bands of *r* rows: S-curve

Locality-Sensitive Hashing: S-curve



Given a fixed threshold t, we want to choose r and b such that *Prob(Candidate pair)* has a "step" around t.

LSH Families of Hash Functions

- In the context of LSH families, a "hash function" is any function that allows us to say whether two elements are candidates for comparison
  - We use the notation

    $$h(x) = h(y)$$

    to mean "h says x and y are a candidate pair"

- A family of hash functions is any set of hash functions from which we can pick one at random efficiently

  For Min-Hashing signatures, each permutation of rows gives us a different Min-Hash function

LSH Families of Hash Functions

- Consider a space S of points with a distance measure $d(x, y)$

  Can be Jaccard, Cosine, Euclidean, or other distance

(d_1, d_2, p_1, p_2)-sensitive family

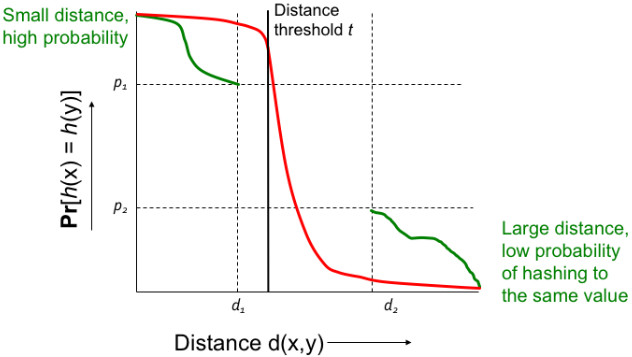
- A family H of hash functions is said to be $(d1, d2, p1, p2)$-sensitive if for any x and y in S:

  If $d(x, y) < d_1$, then the probability that $h(x) = h(y)$, over all $h \in H$, is at least p_1

  If $d(x, y) > d_2$, then the probability that $h(x) = h(y)$, over all $h \in H$, is at most p_2

$\hookrightarrow$ With a LS Family we can do LSH!

(d_1, d_2, p_1, p_2)-sensitive family



For distances d_1 and below, the probability is at least p_1, and for distances d_2 and above, the probability is at most p_2. Between distances d_1 and d_2, we know nothing.

Goal: Minimize difference btw d_1 and d_2 and maximize distance btw p_1 and p_2.

Example of LS Family: Min-Hash

- Consider

  S = space of all sets
  d = Jaccard distance
  H : a family of Min-Hash functions for all permutations of rows

- Then for any hash function $h \in H$

$$Pr\left[h(x) = h(y)\right] = 1 - d(x, y)$$

Example of LS Family: Min-Hash

- For Jaccard distance, Min-Hashing gives a

  $$(d_1, d_2, (1 - d_1), (1 - d_2))\text{-sensitive family for any } d_1 < d_2$$

  Example:
  H is a ($\underline{1/3}$, 2/3, $\underline{2/3}$, 1/3)-sensitive family for S and d

  If distance $\leq 1/3$, similarity $\geq 2/3$

  Then probability that min-hash values are the same is $\geq 2/3$

Amplifying a LS-Family

- Reproduce the “S-curve” effect for any LS family
- The “bands” technique we learned for signature matrices carries over to this more general setting
- Two constructions

  AND construction ~ “rows in a band”
  OR construction ~ “many bands”

AND of Hash Functions

- Given family H, construct family H' consisting of r functions from H

  For $h = [h_1, ..., h_r]$ in H',

  $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for <u>all i</u>

  $\hookrightarrow$ corresponds to creating a band of size r

Theorem

If H is (d_1, d_2, p_1, p_2)-sensitive, H' is $(d_1, d_2, {p_1}^r, {p_2}^r)$-sensitive

Lowers probability for large distances (Good)
Also lowers probability for small distances (Bad)

OR of Hash Functions

- Given family H, construct family H' consisting of b functions from H

  For $h = [h_1, ..., h_b]$ in H',

  $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for <u>at least one i</u>

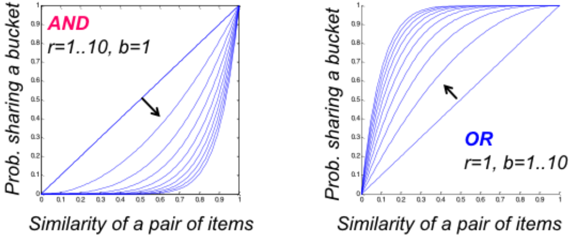
Theorem

If H is (d_1, d_2, p_1, p_2)-sensitive,
H' is $(d_1, d_2, 1 - (1 - p_1)^b, 1 - (1 - p_2)^b)$-sensitive

Raises probability for small distances (Good))
Raises probability for large distances (Bad)

Effect of AND and OR constructions

- **AND** makes all probabilities shrink
  - by choosing *r* correctly, we can make the lower probability approach 0 while the higher does not
- **OR** makes all probabilities grow
  - by choosing *b* correctly, we can make the upper probability approach 1 while the lower does not



Combining AND and OR Constructions

- r-way **AND** followed by b-way **OR** construction
- Same as in Min-Hashing
  - **AND**: If bands match in **all r** (values hash to same bucket)
  - **OR**: Columns that have **at least one** common bucket form a candidate pair

- If for points x and y $Pr[h(x) = h(y)] = s$
  - H will make (x, y) a candidate pair with probability s
  - r***AND** b***OR** construction makes (x, y) a candidate pair with probability $1 - (1 - s^r)^b$

    $\hookrightarrow$ **S-curve**

- Can use OR followed by AND; can combine sequences

AND-OR construction: example

s	$p = 1 - (1 - s^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

r = 4, b = 4 transforms a $(d_1, d_2, 0.8, 0.2)$-sensitive family into a $(d_1, d_2, 0.8785, 0.0064)$-sensitive family.

OR-AND construction: example

s	$p = \left(1 - (1-s)^4\right)^4$
0.1	0.0140
0.2	0.1215
0.3	0.3334
0.4	0.5740
0.5	0.7725
0.6	0.9015
0.7	0.9680
0.8	0.9936

r $=$ 4, b $=$ 4 transforms a $(d_1, d_2, 0.8, 0.2)$-sensitive family into a $(d_1, d_2, 0.9936, 0.1215)$-sensitive family.

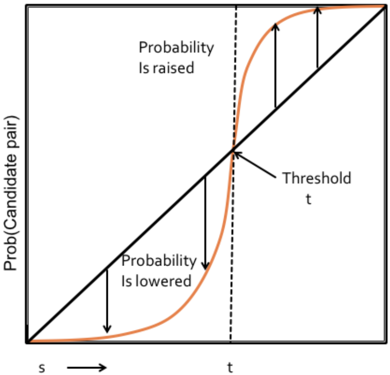
Cascading constructions

- Example: Apply the (4,4) OR-AND construction followed by the (4,4) AND-OR construction

- Transforms a $(d_1, d_2, 0.8, 0.2)$-sensitive family into a $(d_1, d_2, 0.9999996, 0.0008715)$-sensitive family

  Note that this family uses 256 (=4*4*4*4) of the original hash functions

Constructions: visualization of threshold t



For an AND-OR S-curve $1-(1-s^r)^b$, the threshold t is where $1-(1-s^r)^b = t$

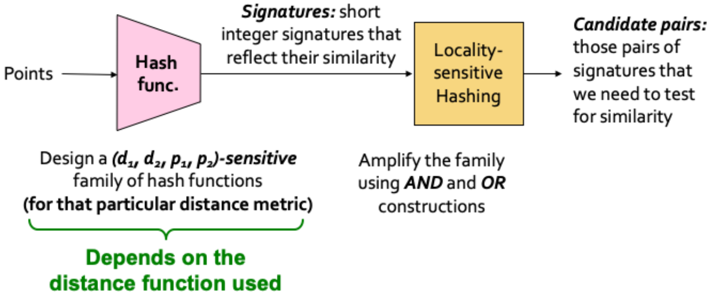
Probabilities p_1 and p_2 should be at opposite sides of t

Summary

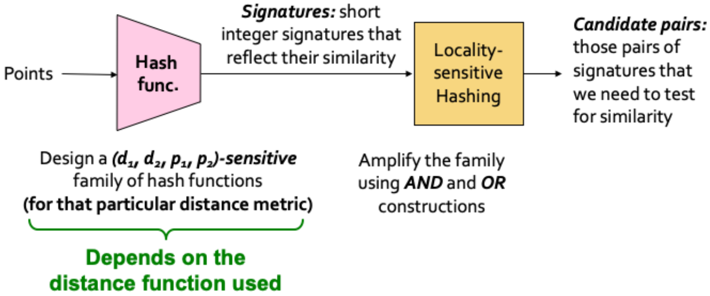
- Pick any two distances $d_1 < d_2$
- Start with a (d_1, d_2, p_1, p_2)-sensitive family
- Apply constructions to amplify (d_1, d_2, p_1^*, p_2^*)-sensitive family, where p_1^* is almost 1 and p_2^* is almost 0

- The closer to 0 and 1 we want to get, the more hash functions must be used!

LSH for other distance metrics



LSH for other distance metrics



- Cosine distance
  - **Random hyperplanes**
- Euclidean distance
  - **Random projections**

LSH for cosine distance

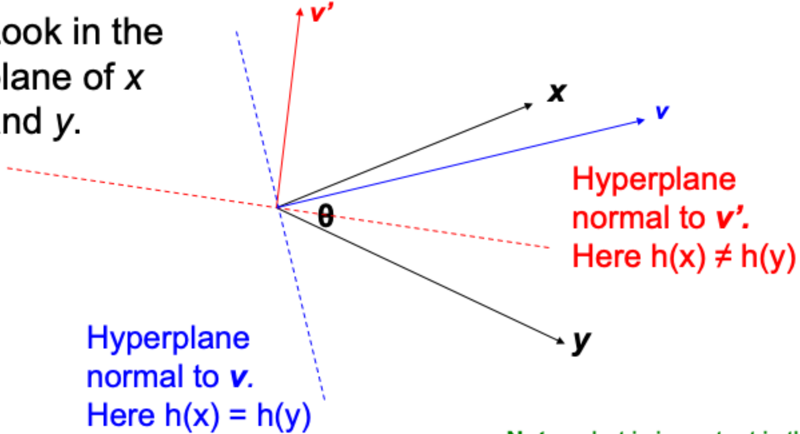
- For cosine distance, there is a technique called **Random Hyperplanes**
  - Technique similar to Min-Hashing

- Random Hyperplanes method is a $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$ - sensitive family for any d_1 and d

Reminder: (d_1, d_2, p_1, p_2)-sensitive
If $d(x, y) < d_1$, then probability that h(x) = h(y) is at least p_1
If $d(x, y) > d_2$, then probability that h(x) = h(y) is at most p_2

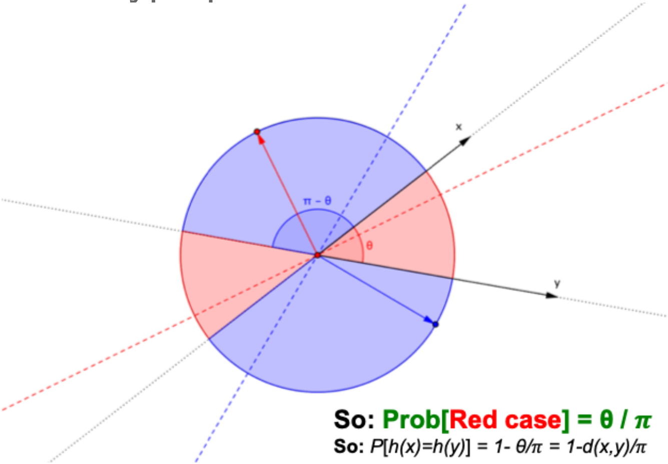
Random hyperplanes visualization

Look in the plane of *x* and *y*.



Note: what is important is that hyperplane is outside the angle, not that the vector is inside.

Random hyperplanes visualization



Signatures for Cosine Distance

- Pick some number of random vectors v_i, and hash your data for each vector

  $$h_{v_i}(x) = +1 \text{ if } v_i \cdot x \geq 0$$
  $$h_{v_i}(x) = -1 \text{ if } v_i \cdot x < 0$$

- Result is a signature (sketch) of $+1$'s and -1's for each data point
- Can be used for LSH, in same way as Min-Hash signatures are used for Jaccard distance
- Amplify using AND/OR constructions

How to select random vectors

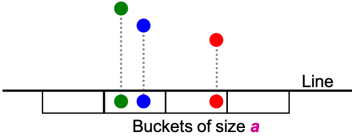
- Expensive to pick random vectors in M dimensions for large M
  Would require generating M random numbers

- It suffices to consider only vectors v_i consisting of $+1$ and –1 components
  Assuming data is random, then vectors of $+/$-1 cover the entire space evenly (no bias)

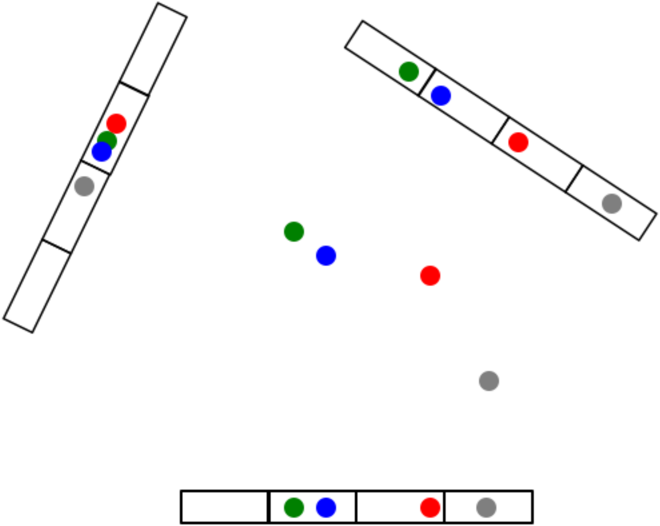
LSH for Euclidean distance

- Hash functions correspond to lines
- Partition the line into buckets (line segments) of size *a*
- Hash each point to the bucket containing its projection onto the line
- An element of the "signature" is a bucket id for that given projection line

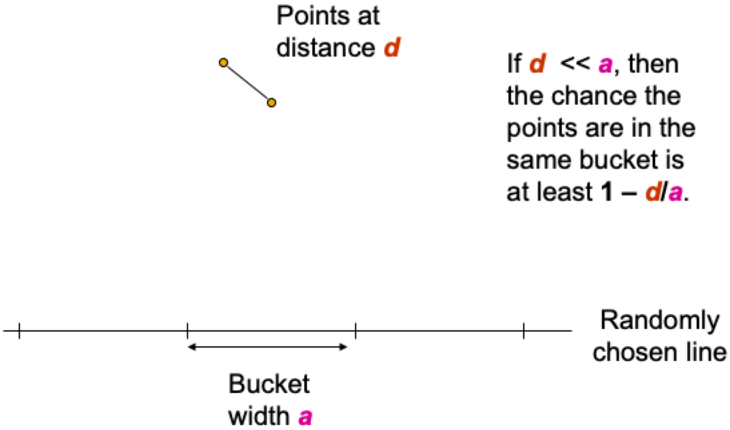
- Nearby points are always close; distant points are rarely in same bucket



Projections

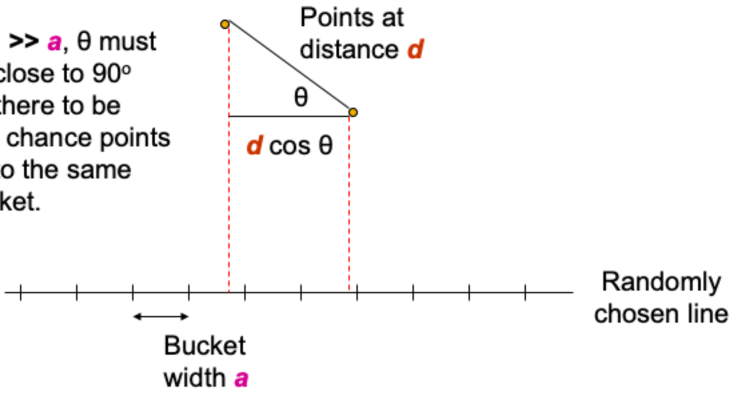


Projections



Projections

If d >> a, θ must be close to 90° for there to be any chance points go to the same bucket.



LS-Family for Euclidean Distance

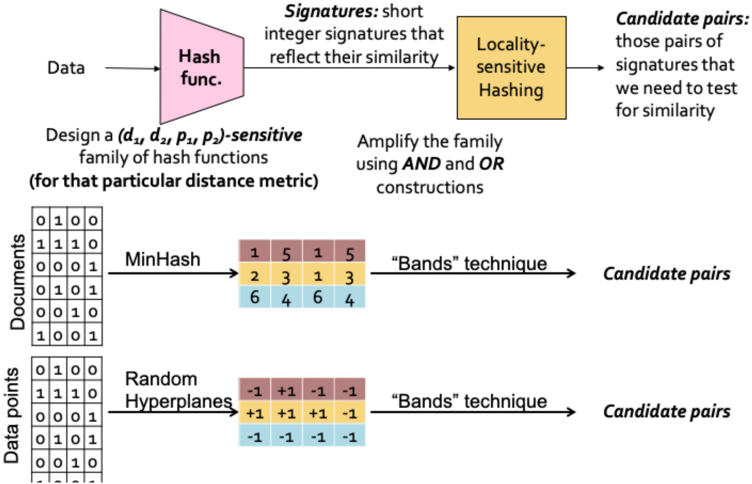
- If points are at distance $d \leq a/2$, then the probability of falling in same bucket is at least $1 - d/a$, or $1/2$
- If points are at distance $d \geq 2a$, then they can fall in the same bucket only if $d \cdot cos\theta \leq a$, which is only true if

  $cos\theta \leq 1/2$

  $60 < \theta < 90$, i.e., at most $1/3$ probability

- Yields a (a/2, 2a, 1/2, 1/3)-sensitive family of hash functions for any a
- Amplify using AND-OR cascades

LS families



Data
Hash func.
Signatures: short integer signatures that reflect their similarity
Locality-sensitive Hashing
Candidate pairs: those pairs of signatures that we need to test for similarity
Design a ***(d_1, d_2, p_1, p_2)-sensitive*** family of hash functions
(for that particular distance metric)
Amplify the family using ***AND*** and ***OR*** constructions
Documents
MinHash
"Bands" technique
Candidate pairs
Data points
Random Hyperplanes
"Bands" technique
Candidate pairs

Important points

- Property $P(h(C1)=h(C2)) = sim(C1,C2)$ of hash function *h* is the essential part of LSH, without it we can't do anything
- LS-hash functions transform data to signatures so that the bands technique (AND, OR constructions) can then be applied

Further fun... [MMDS 3.8, 3.9]

- Application examples

  Entity resolution

  Idea: Score 100 for each full match on name, address, or phone
  Sort on name, score matching lines; repeat for address, phone

  Fingerprint matching

  Fingerprints represented by set of grid squares containing minutiae: can use Jaccard similarity
  LSH family: each hash function is a set of grid squares

  Similar news articles

  Shingles defined by a stopword followed by the next two words

- Sets as strings
  - Length-based filtering
  - Prefix indexing