Finding Similar Items 2

EE412: Foundation of Big Data Analytics

Announcements

- Homeworks
  - HW0 (due: 09/21)
  - HW1 (due: 10/05)

Recap: Shingling and Minhashing

↳ doc → words

document

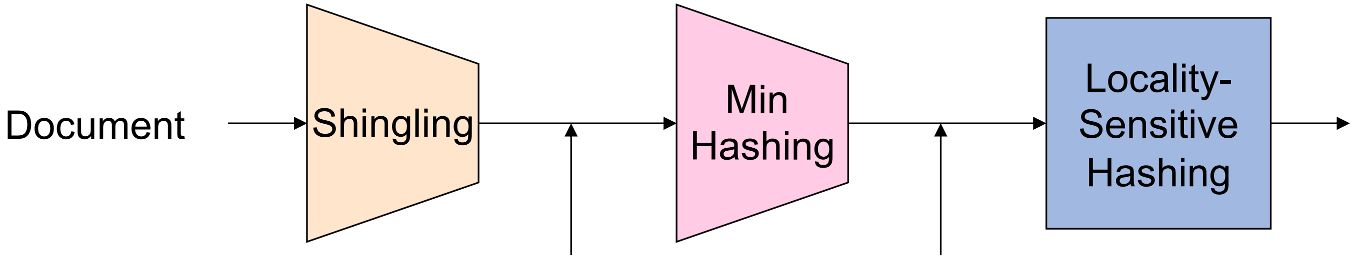
- k-shingles (document → set)
- Jaccard similarity
  - $\mathrm{sim}(S, T) = |S \cap T| / |S \cup T|$
- Minhash
  - Pick a random permutation of rows
  - The minhash value is the first row that has 1
  - E.g., $h(S_1) = 0$, $h(S_2) = 2$, and $h(S_3) = 1$
  - $\Pr(\text{minhash is same}) = $ Jaccard similarity
- Computing minhash signatures

⇓ multiple permutation ⇒ accurate similarity

Row	*Element*	S_1	S_2	S_3
0	"The plane"	1	0	0
1	"The cow j"	0	0	1
2	"Alice and"	0	1	0
3	"Roses are"	1	0	1

Row	S_1	S_2	S_3
1	0	0	1
0	1	0	0
3	1	0	1
2	0	1	0

Recap: The Big Picture



The set of strings of length k that appear in the document

Signatures: Short integer vectors that represent the sets, and reflect their similarity

Candidate Pairs: Those pairs of signatures that we need to test for similarity

Source: Stanford CS246 (2018)

Outline

1. **Locality-Sensitive Hashing (LSH)**
2. LSH Families
3. LSH with Other Distance Measures

Locality-Sensitive Hashing

- The number of document pairs can be large after minhashing
  - What if we have 1M documents with 1μs for each comparison?
  - 6 days to compute all similarities
- **Locality-sensitive hashing (LSH)** can reduce # of candidate pairs
  - Allows us to focus on pairs that are likely to be similar

From Signatures to Buckets

여러번 hash 하기

- **General approach:** Hash items several times
  - Similar items are more likely to hash to same bucket than dissimilar items
    - **Why?** Consider the property of minhashing

x hold for every hash func

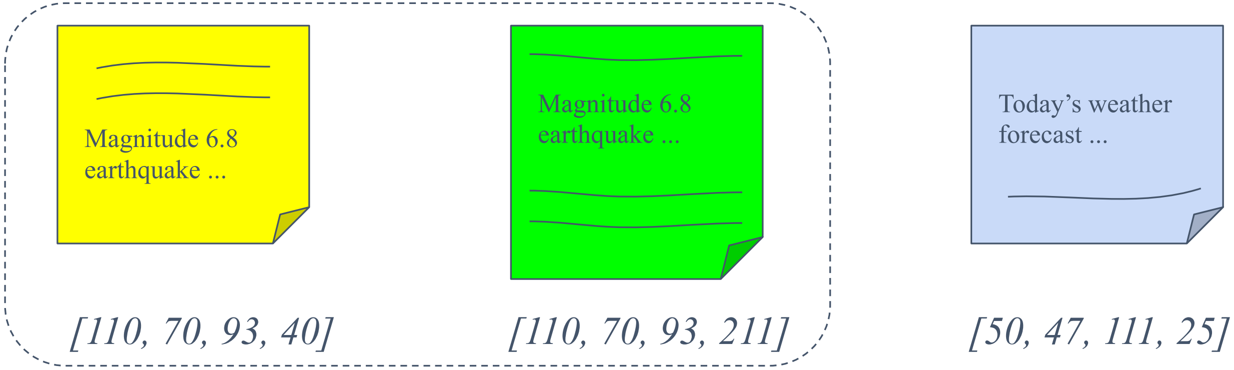
by some hash func

random hashing도 적용되어야

- Any pair that is hashed to same bucket is a candidate
  - **False positive:** Dissimilar pairs in the same bucket // possible
  - **False negative:** Similar pairs not in any same bucket // ⇒ problem

근데 예에서는 그렇게 중요하지 x

└ why??

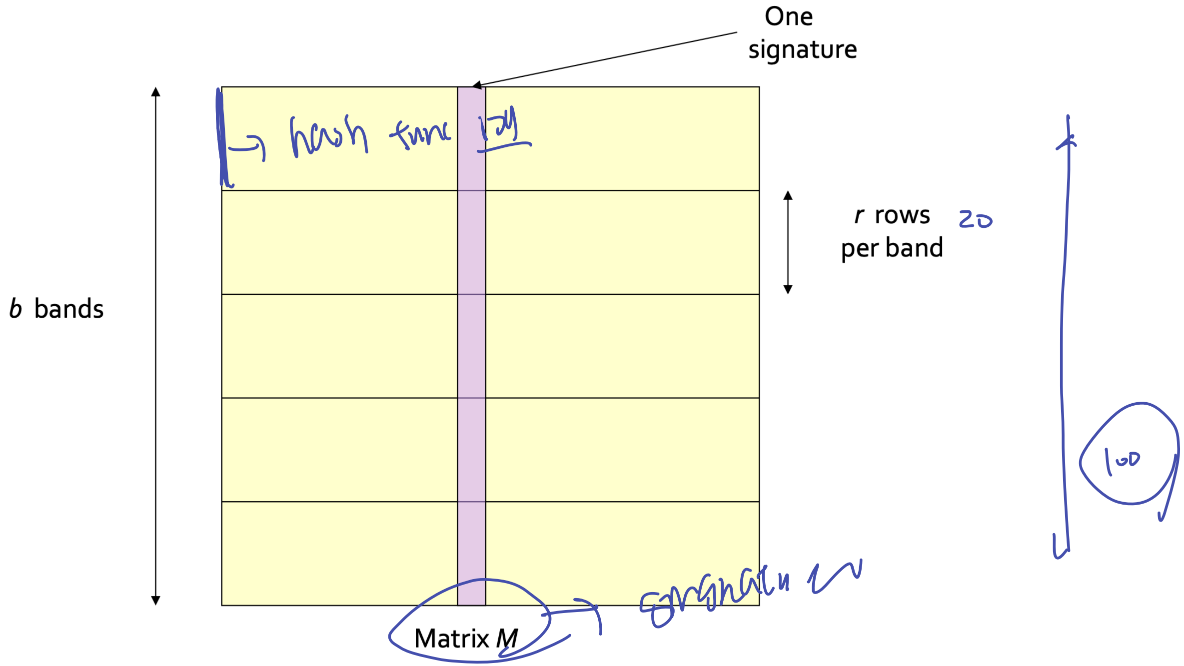


Partition into Bands

- **Trick:** Divide the signature matrix M into b **bands** of r rows
  - For each band, hash each column to a hash table with k buckets
    - Make k as large as possible (no collision)
    - Use a different hash table for each band
  - Candidate column pairs if hashed to the same bucket for ≥ 1 band
  - Tune b and r to catch most similar pairs, but few non-similar pairs

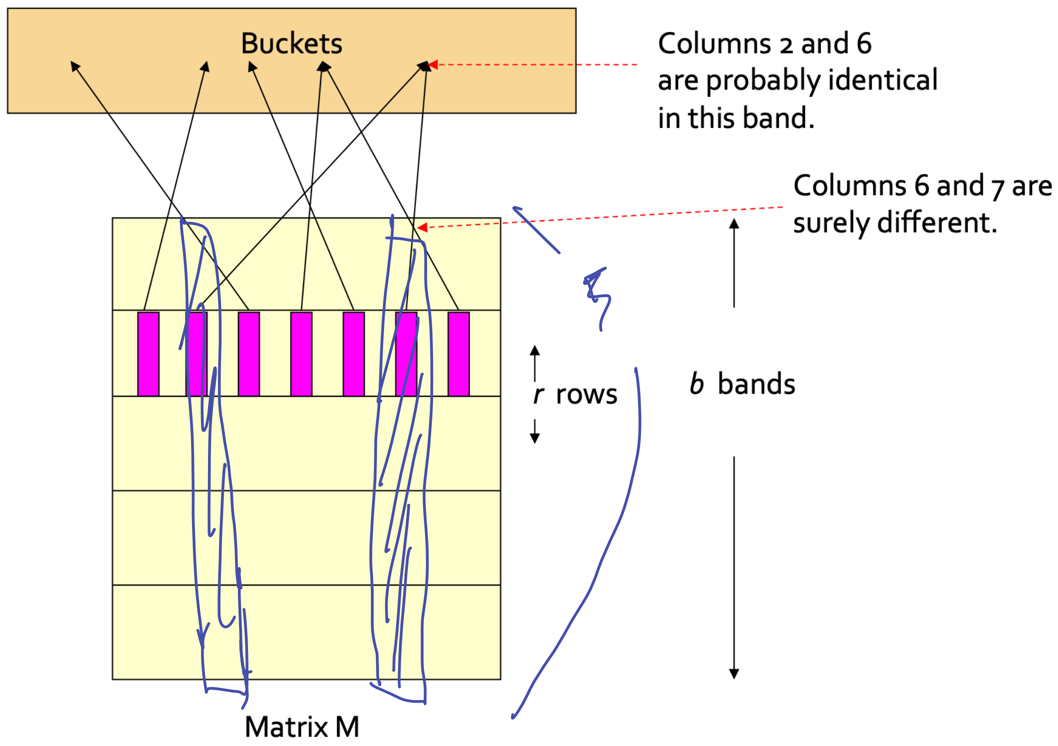
Partition into Bands

- **Trick:** Divide the signature matrix M into b **bands** of r rows



Source: Stanford CS246 (2022)

Hash Functions to One Bucket



Source: Stanford CS246 (2022)

Example: Bands

- Suppose a signature matrix of size $100 \times 100{,}000$
  - 100,000 columns (documents)
  - Signatures consist of 100 integers
- **Goal:** Find all 80%-similar pairs V
  - 5,000,000,000 pairs of signatures can take a while to compare
  - Choose $b = 20$ bands of $r = 5$ integers

Example: False Negatives

- Suppose C_1 and C_2 are 80% similar (i.e., positive pair)
- Probability identical in any one particular band: $(0.8)^5 = 0.328$
- Probability not similar in all 20 bands: $(1 - 0.328)^{20} = .00035$
    - **False negatives:** About 1/3000th of all 80%-similar pairs

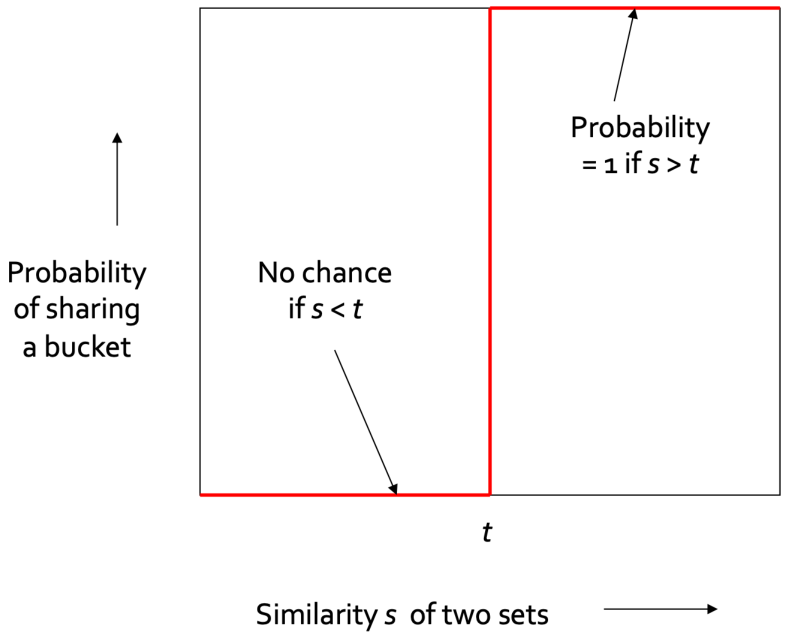
Example: False Positives

- Suppose C_1 and C_2 are 40% similar (i.e., negative pair)
- Probability identical in any one particular band: $(0.4)^5 = 0.01$
- Probability identical in ≥ 1 of 20 bands: $1 - (1 - 0.01)^{20} < 0.2$
  - **False positives:** Less than 1/5th of all 40%-similar pairs
  - This means that we remove more than 80% of all 40%-similar pairs!

22.

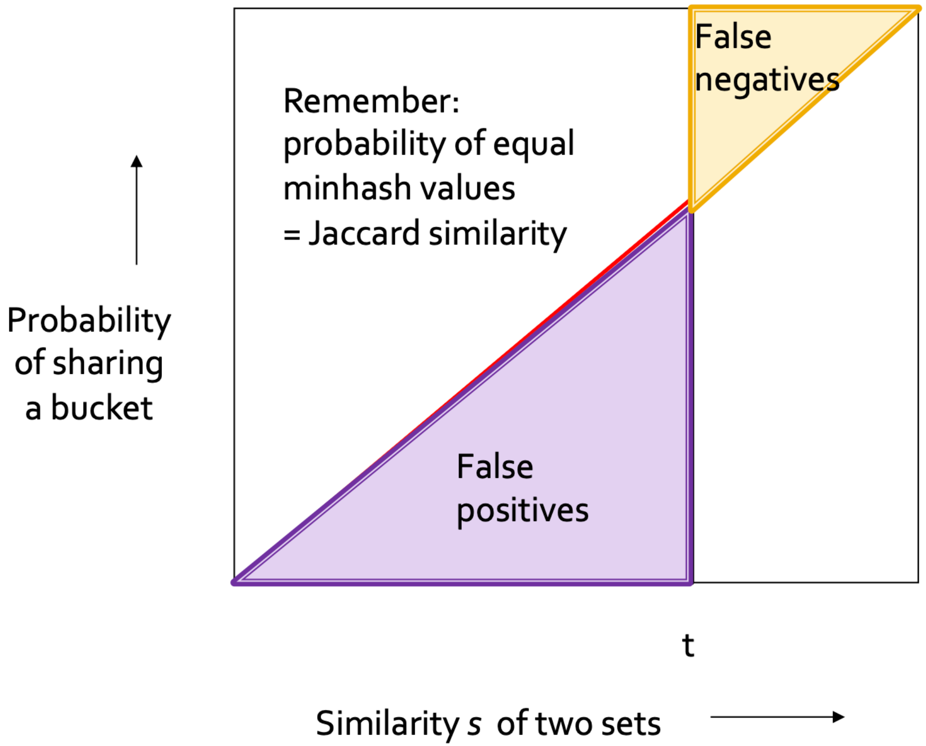
Analysis of LSH: What We Want

- Let s be a **similarity**, and t be a **similarity threshold** for the query
- What we want from LSH is:



Source: Stanford CS246 (2022)

Analysis of LSH: $b = 1$ and $r = 1$

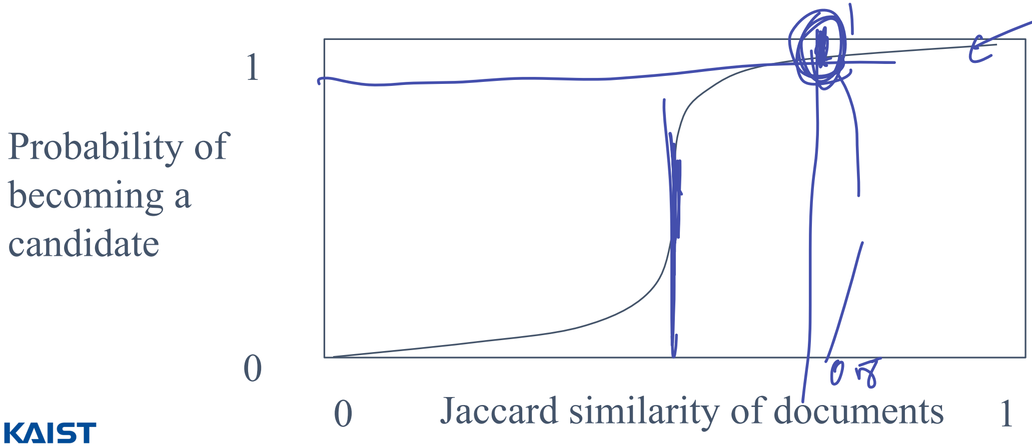


Say “yes” if you are below the line.

Source: Stanford CS246 (2022)

Analysis of LSH: S-Curve

- Suppose b bands of r rows each, and a pair of docs has similarity s
  - Pr(signatures agree in all rows on one band) $= s^r$
  - Pr(signatures do not agree in at least one row of one band) $= 1 - s^r$
  - Pr(signatures do not agree in at least one row of each band) $= (1 - s^r)^b$
  - Pr(signatures agree in all rows of at least one band) $= 1 - (1 - s^r)^b$

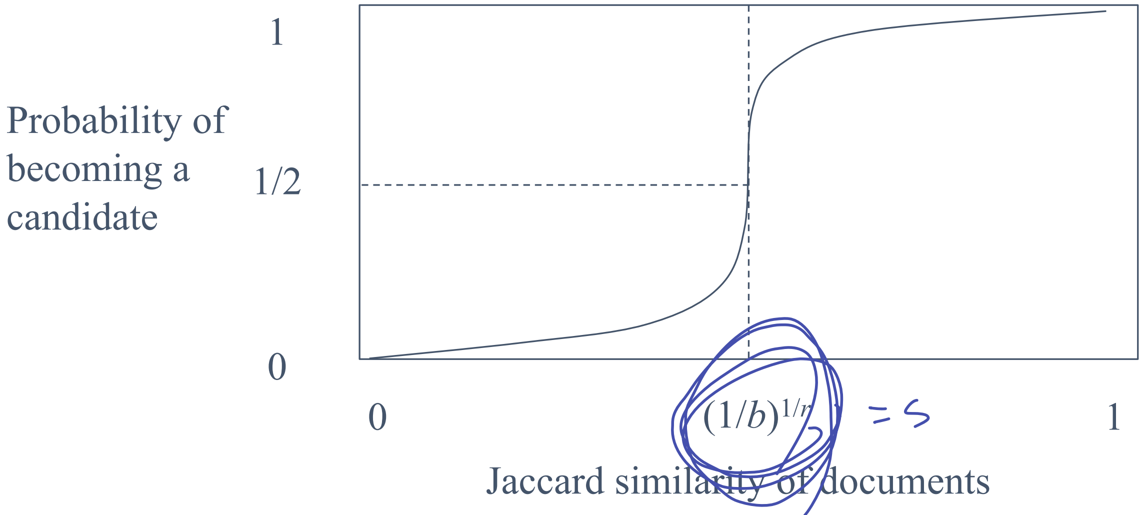


s	$1-(1-s^r)^b$
.2	.006
.4	.186
.6	.802
.8	.9996

$b = 20, r = 5$

Analysis of LSH: S-Curve

- Regardless of b and r, the function has the form of an S-curve
- If $s \approx (1/b)^{1/r}$, the probability of becoming a candidate is $1/2$
  - This is where the rise is steepest

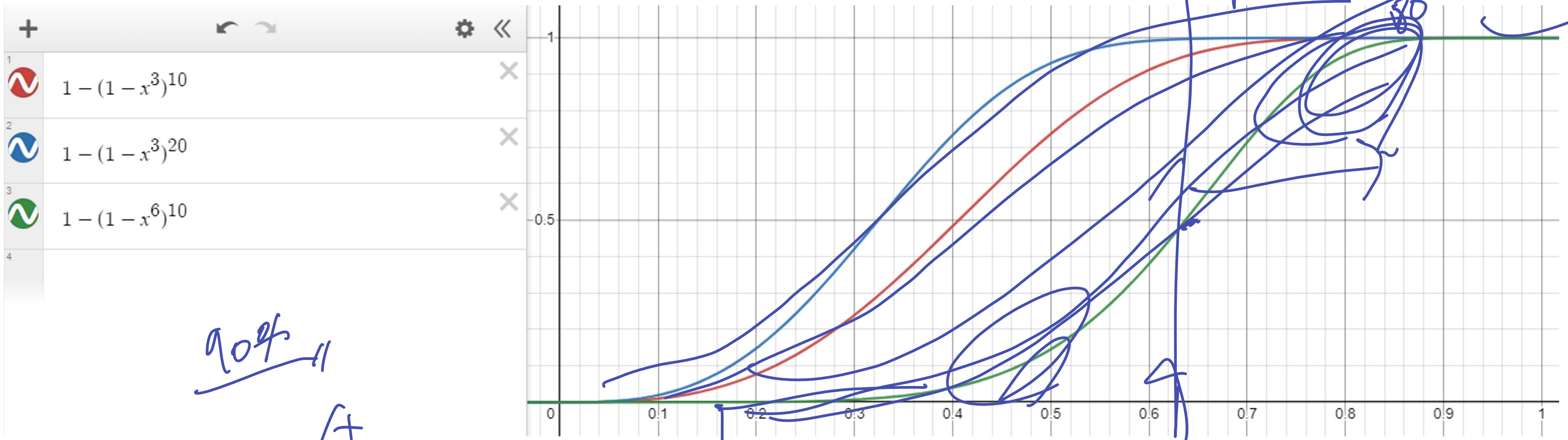


Combining the Techniques

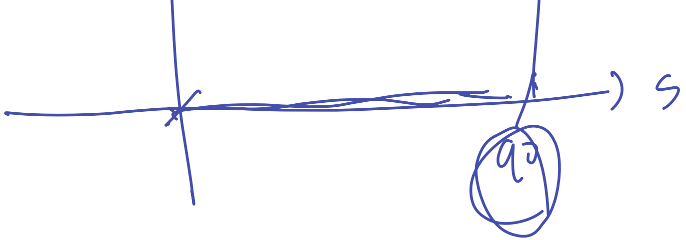
- Pick k and construct from each document a set of k-shingles
- Pick length n for minhash signatures and compute signatures
- Pick similarity threshold t and then # of bands b and # of rows r
  - If $(1/b)^{1/r} > t$, there will be fewer false positives
  - If $(1/b)^{1/r} < t$, there will be fewer false negatives
- Construct candidate pairs and compare their signatures using t
- (Optionally) Check if the documents are actually similar

Example: S-Curves

- Compare the S-curve $1 - (1 - s^r)^b$ when r and b are:
  - $r = 3$ and $b = 10$
  - $r = 3$ and $b = 20$
  - $r = 6$ and $b = 10$



Outline



1. Locality-Sensitive Hashing (LSH)
2. **<u>LSH Families</u>**
3. LSH with Other Distance Measures

LSH Families

- **Q:** What kind of hash functions can we use for LSH?
- There are three conditions for **a family of hash functions**
    1. Closer pairs should more likely be candidates
    2. Functions must be statistically independent
    3. Functions must be efficient
        - Identify candidates faster than looking at all pairs
        - Must be combinable to build better functions

LSH Families

- Let $d_1 < d_2$ be two distances according to a distance function d
  - E.g., d can be Jaccard distance
- A family F of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if
  - For every $f \in F$ and points x and y:

    If $d(x, y) \leq d_1$, then the probability that $f(x) = f(y)$ is at least p_1

    If $d(x, y) \geq d_2$, then the probability that $f(x) = f(y)$ is at most p_2

Meanings of Sensitivity Values

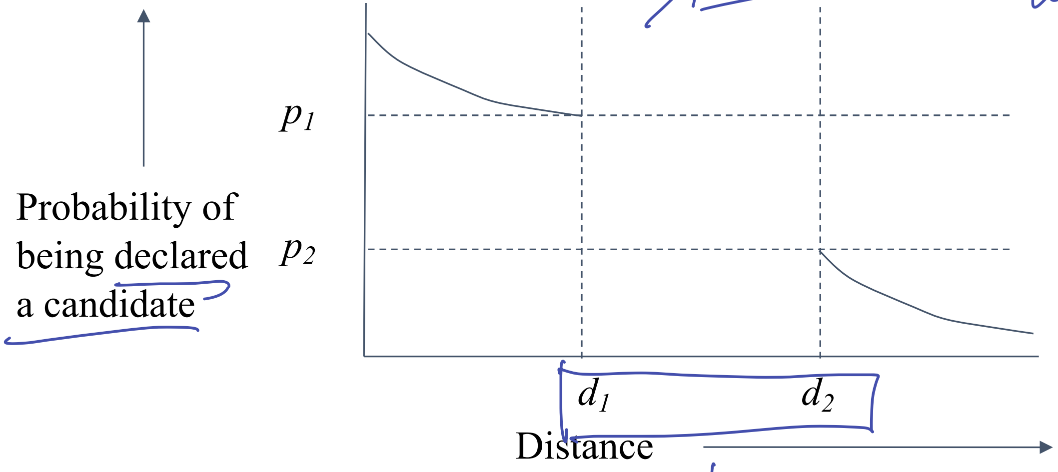
- If F is (d_1, d_2, p_1, p_2)-sensitive,

  F is better with higher d_1, lower d_2, higher p_1, and lower p_2

don't care

stack multiple hash function

to draw S-curve



Reflection of the previous S-curve across the vertical axis

붙어있는 쪽은 그 특성을 잘 알수있다.

LSH Families for Jaccard Similarity

- Minhashing gives a $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive family
  - For any d_1 and d_2 such that $0 \leq d_1 < d_2 \leq 1$
- **Proof**
  - Let d be the Jaccard distance $(= 1 - \boxed{\text{sim}(x, y)})$
  - Then, $d(x, y) \leq d_1 \Rightarrow 1 - \text{sim}(x, y) \leq d_1 \Rightarrow 1 - d_1 \leq \text{sim}(x, y)$
  - Similar argument applies to d_2

$p_1 \leq p$

증명 이용

Amplifying an LSH Family

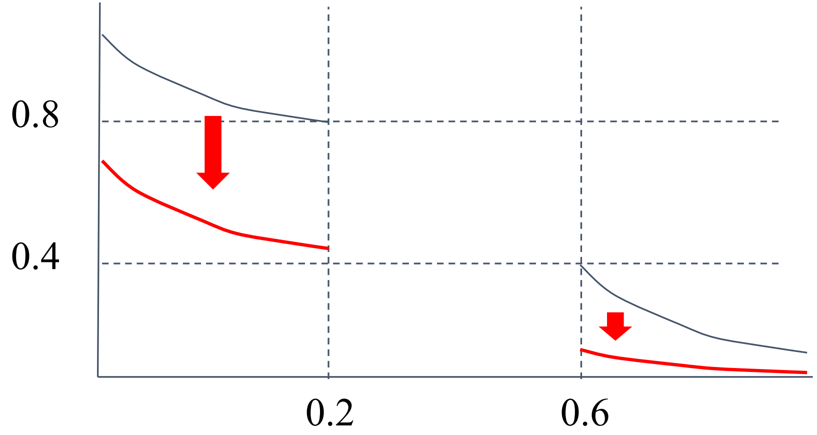
- The banding technique carries over to this more general setting
  - **Goal:** Make the S-curve effect seen there
  - **AND construction** like "rows in a band."
  - **OR construction** like "many bands."
- Given a (d_1, d_2, p_1, p_2)-sensitive family F, create a new family F'
  - Each member of F' consists of n members of F

AND Construction

- If
  - f in F' is constructed from $\{f_1, f_2, \ldots, f_r\}$ of F, and
  - $f(x) = f(y)$ if and only if $f_i(x) = f_i(y)$ **for all** $i = 1, 2, \ldots, r$
- Then, F' is (d_1, d_2, p_1^r, p_2^r)-sensitive

F is a (0.2, 0.6, 0.8, 0.4)-sensitive family

If $r = 4$, $0.8^4 =$ $0.4^4 =$
F' is (0.2, 0.6, 0.4096, 0.0256)-sensitive



OR Construction

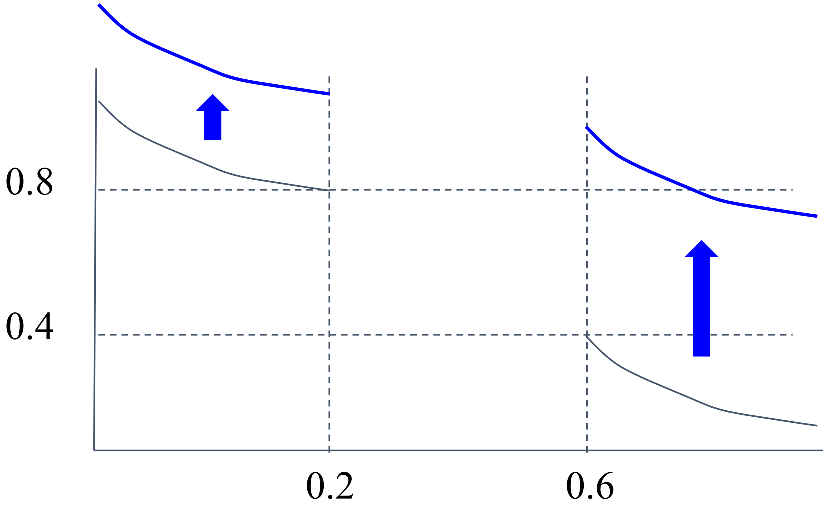
- If
  - f in F' is constructed from $\{f_1, f_2, \ldots, f_b\}$ of F, and
  - $f(x) = f(y)$ if and only if $f_i(x) = f_i(y)$ **for at least one** $i = 1, 2, \ldots, b$
- Then, F' is $\left(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b\right)$-sensitive

F is a (0.2, 0.6, 0.8, 0.4)-sensitive family

If $b = 4$,

F' is (0.2, 0.6, 0.9984, 0.8704)-sensitive

$1\text{-}(1\text{-}0.8)^4 =$ $1\text{-}(1\text{-}0.4)^4 =$



Combining AND and OR Constructions

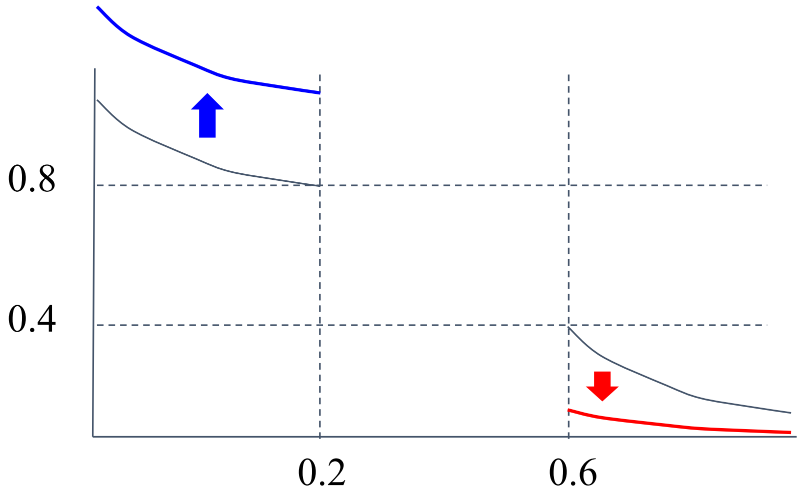
- Make the lower probability goes to 0 while the higher goes to 1
    - By choosing b and r correctly
- AND constructions to decrease the low probability
- OR constructions to increase the high probability

AND-OR Composition

- Do an AND construction and then an OR construction
- Each of the two probabilities p is transformed into $1-(1-p^r)^b$
  - The S-curve studied before

F is a (0.2, 0.6, 0.8, 0.4)-sensitive family

If $r = b = 4$,
F' is (0.2, 0.6, 0.8785, 0.0985)-sensitive



OR-AND Composition

- Do an OR construction and then an AND construction
- Each of the two probabilities p is transformed into $\left(1-(1-p)^b\right)^r$
  - The same S-curve, mirrored horizontally and vertically
- Let's consider the previous example:
  - If we apply a 4-way OR followed by a 4-way AND,
    the constructed family is $(0.2, 0.6, 0.9936, 0.5740)$-sensitive
  - Not ideal because the low probability has increased a lot

Outline

1. Locality-Sensitive Hashing (LSH)
2. LSH Families
3. **LSH with Other Distance Measures**

LSH for Other Distance Measures

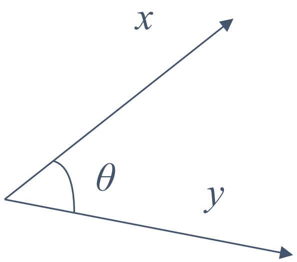
- Generalized LSH is based on "distance" between points
  - Jaccard similarity is not a distance; 1 – Jaccard similarity is
- We cover only **cosine distance** today
- See textbook for other distance measures
  - Euclidean distance, Hamming distance, Edit distance, etc.

Distance Measures

- A function d is a **distance measure** if
  - It takes two points in the space and returns a real number satisfying:
    - $d(x, y) \geq 0$ (non-negative)
    - $d(x, y) = 0$ iff $x = y$ (distances are positive except for $x = y$)
    - $d(x, y) = d(y, x)$ (symmetric)
    - $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

Cosine Distance

- Think of a point as a vector from the origin to its location
- Two points' vectors make an angle
- Cosine of the angle is the normalized dot-product of the vectors



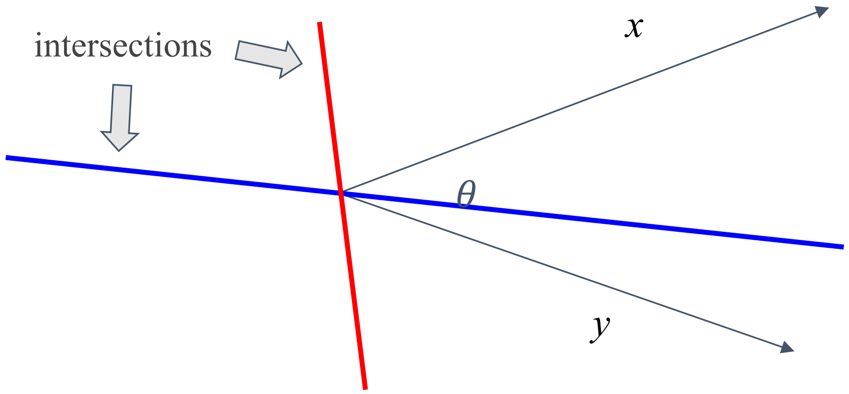
$$\cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

only care about angle

Cosine distance

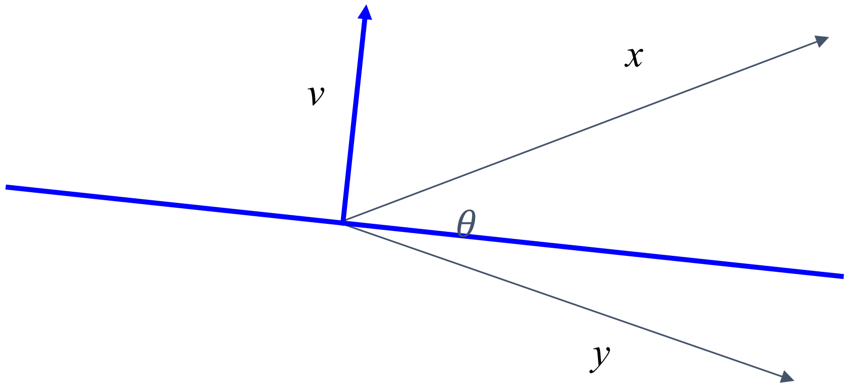
Random Hyperplanes

- **Random hyperplanes:** Technique analogous to minhashing
- Two vectors x and y that have angle θ define a plane
- Two types of hyperplanes through the origin:
  - Red hyperplane where x and y are on the same side
  - Blue hyperplane where x and y are on different sides



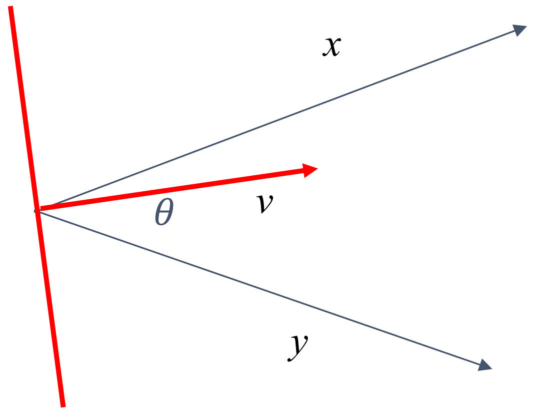
Random Hyperplanes

- Consider a vector v that is normal to a blue-type hyperplane
- Then, $v \cdot x$ and $v \cdot y$ will have different signs
  - Since x and y are on different sides
- Even if v extends in opposite direction, signs are still different



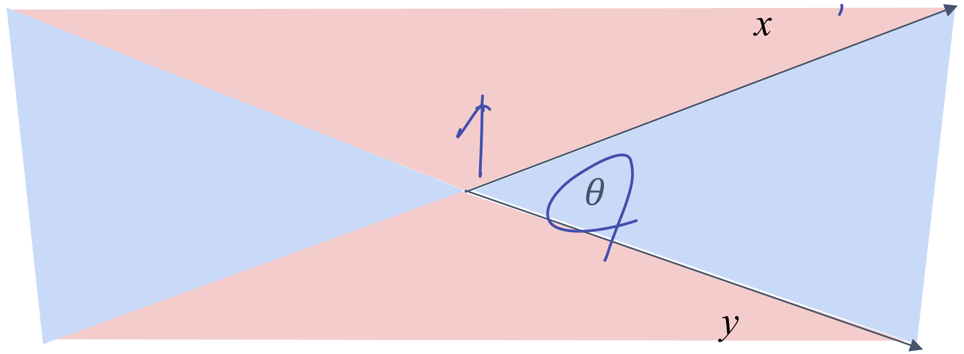
Random Hyperplanes

- Suppose that v is normal to a red-type hyperplane
- Then, $v \cdot x$ and $v \cdot y$ have the same sign



Random Hyperplanes

- Suppose that v is randomly chosen
- Then, probability that hyperplane will be a red type: $1 - \theta/180$



Random Hyperplanes

$d \Rightarrow \theta$

- If
  - Hash function $f \in F$ is built from a randomly-chosen vector v_f
  - Given two vectors x and y such that
    $f(x) = f(y)$ iff $v_f \cdot x$ and $v_f \cdot y$ have the same sign (i.e., a red type)
- Then, F is $(d_1, d_2, 1 - d_1/180, 1 - d_2/180)$-sensitive
- We can amplify F as we wish, just like the minhash-based family

Summary

1. Locality-Sensitive Hashing (LSH)
   - Banding technique
   - S-curve
2. LSH Families
   - (d_1, d_2, p_1, p_2)-sensitivity
   - AND and OR constructions
3. LSH with Other Distance Measures
   - Cosine similarity
   - Random hyperplanes