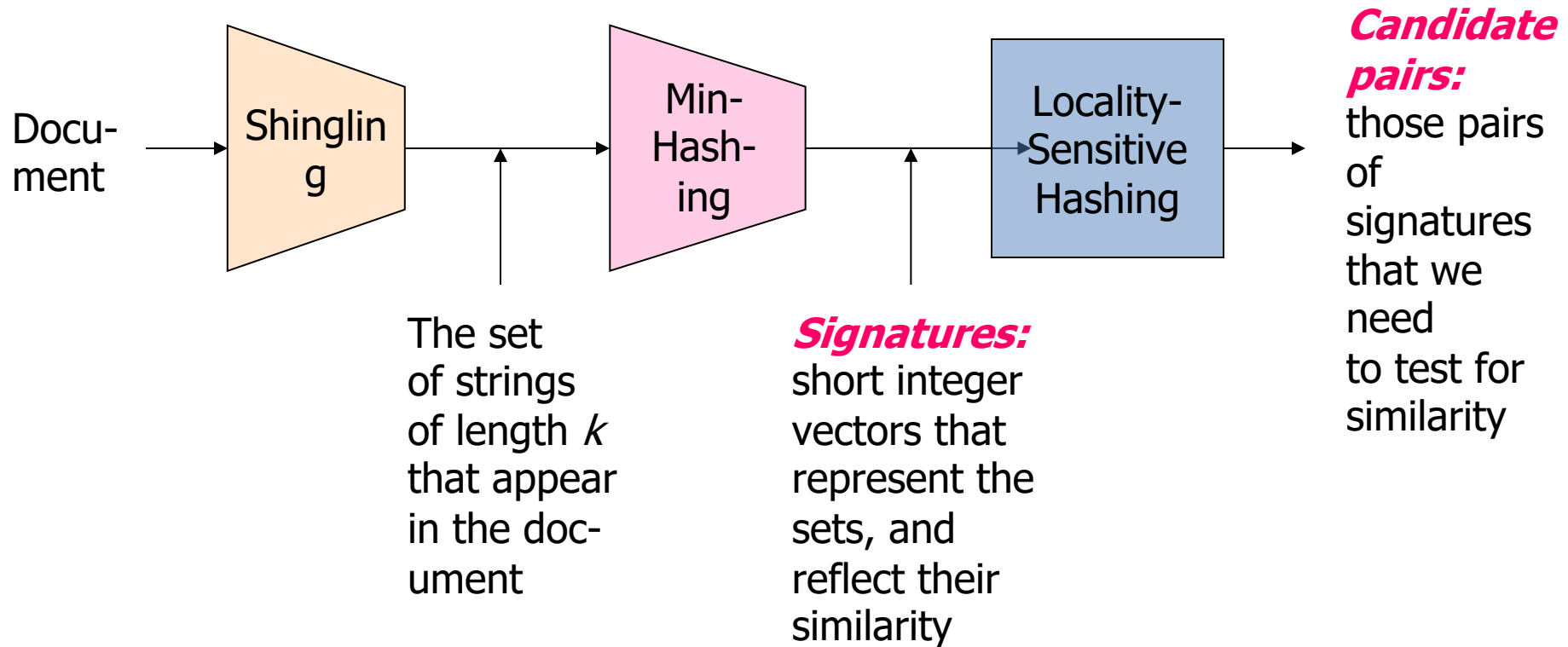


Outline

- Three steps for finding similar items
 1. Shingling: documents -> sets
 2. Min-hashing: sets -> signatures
 3. Locality-sensitive-hashing: signatures -> similarity
- Locality-Sensitive-Hashing (LSH)
- Characteristics of LSH
- Two Applications
 - ❑ Finding similar fingerprints
 - ❑ Finding similar news articles



Locality Sensitive Hashing

Step 3: *Locality-Sensitive Hashing:*

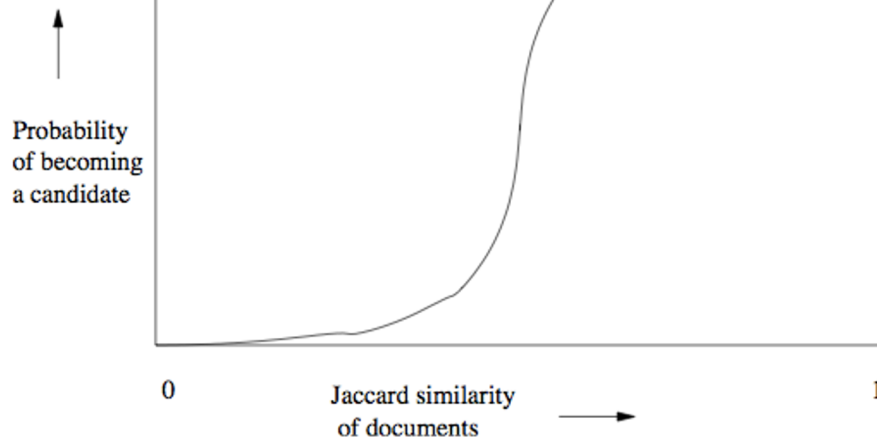
Focus on pairs of signatures likely to be from similar documents

Motivation for Locality Sensitive Hashing

- *Used **k-shingles** to create sets that summarize documents*
- *Used **Minhashing** to generate signatures that represent sets of shingles, reflect their similarity*
- Suppose we need to find near-duplicate documents among a million 10^6 documents
- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of signatures**
 - ❑ 10^6 choose 2
 - ❑ Recall: for large n , $\binom{n}{2}$ is approximately $n^2/2$
 - ❑ $\approx 5 \cdot 10^{11}$ comparisons
 - ❑ At 10^5 secs/day and 10^6 comparisons/sec, it would take **6 days**

Locality Sensitive Hashing

- *Or Near-neighbor search*
- Minhashing is one example of a **family of functions** (the minhash functions) **that can be combined** (by the banding technique) **to distinguish strongly between pairs at a low distance from pairs at a high distance**
- Steepness of the S-curve reflects how effectively we can **avoid false positives and false negatives** among the candidate pairs
- Section 3.6: more general theory of Locality Sensitive Functions



Locality Sensitive Hashing Overview

- **Hash items several times**
 - In a way that **similar items are more likely to be hashed to the same bucket than dissimilar items**
- **Candidate Pair:** Any pair that hashes to the same bucket for **any** of the hashings
- **Check only the candidate pairs for similarity**
- **False positives:** dissimilar pairs that hash to the same bucket
- **False negatives:** truly similar pairs do not hash to the same bucket for at least one of the hash functions

LSH: First Cut

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

- **Goal: Find documents with Jaccard similarity at least s** for some similarity threshold s (e.g. $s=0.8$)
- **LSH – General idea:** Use a function $f(x,y)$ that tells whether x and y are a **candidate pair**: a pair of elements whose similarity must be evaluated
- **For Min-Hash matrix:**
 - ❑ Hash columns of **signature matrix M** to many buckets
 - ❑ Each pair of documents that hashes into the same bucket is a **candidate pair**

Candidates from Min-Hash

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

- Pick a similarity threshold s ($0 < s < 1$)
- Columns \mathbf{x} and \mathbf{y} of \mathbf{M} are a **candidate pair** if their signatures agree on at least fraction s of their rows:
 $M(i, \mathbf{x}) = M(i, \mathbf{y})$ for at least frac. s values of i
 - We expect documents \mathbf{x} and \mathbf{y} to have the same (Jaccard) similarity as their signatures

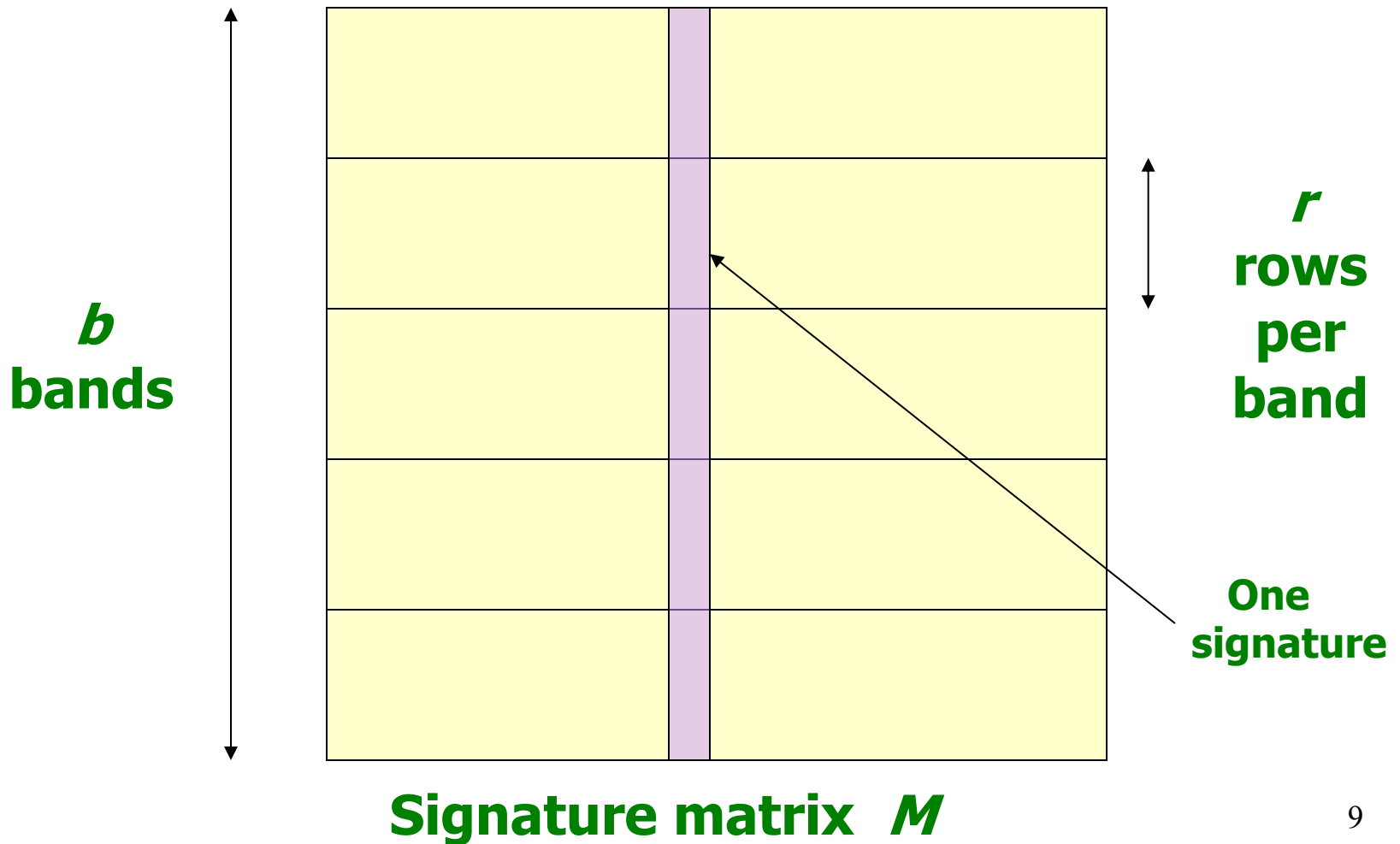
LSH for Min-Hash

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

- **Big idea:** Hash columns of signature matrix M several times
- Arrange that (only) **similar columns** are likely to **hash to the same bucket**, with high probability
- Candidate pairs are those that hash to the same bucket

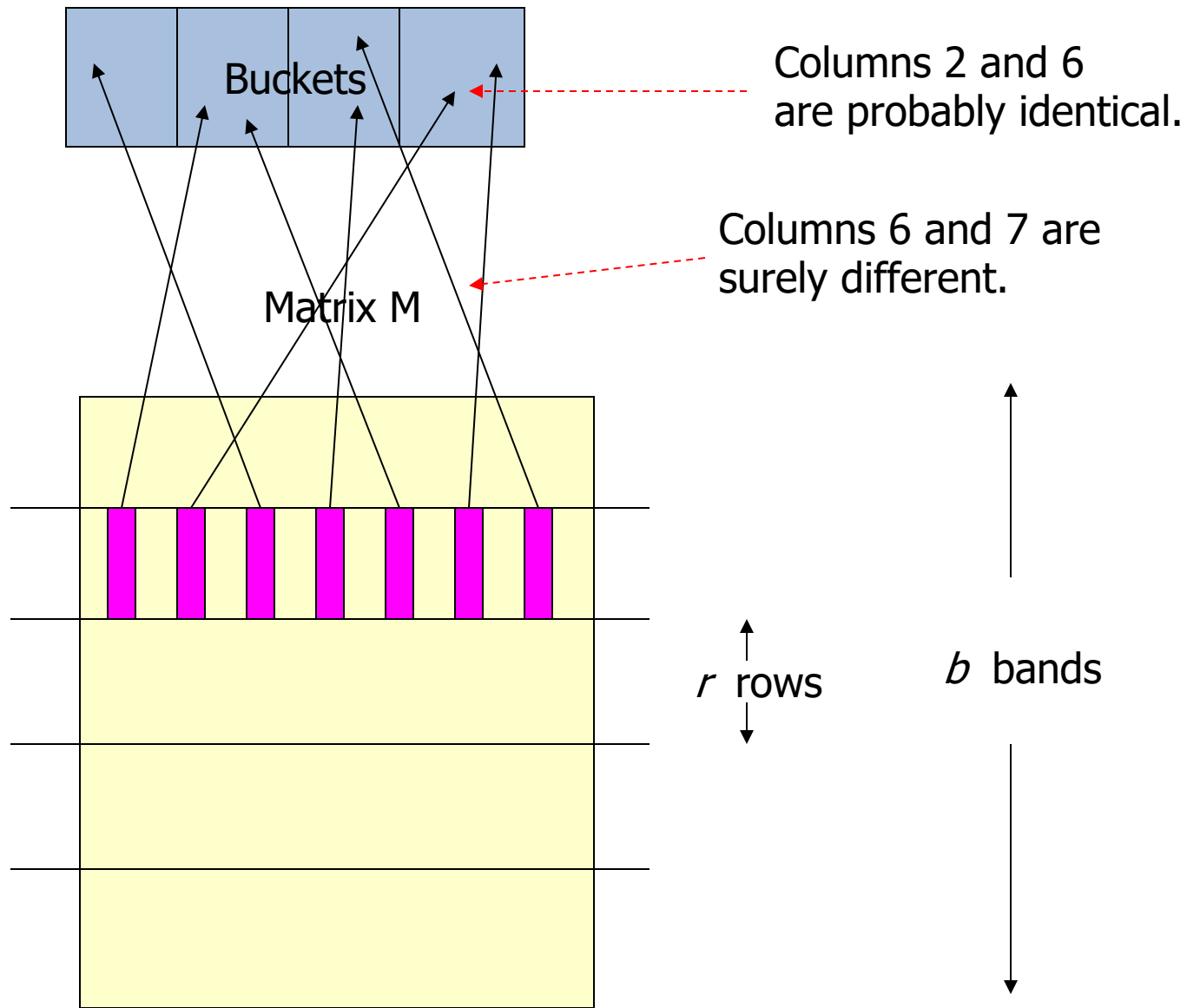
Partition M into b Bands

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



Partition Signatures M into Bands

- Divide matrix M into b bands of r rows
 - ❑ Signatures are still too big, so we check band by band
 - ❑ So, check “similar signatures” becomes check “similar bands”
- For each band, hash its portion of each column to a hash table with k buckets
 - ❑ Make k as large as possible
 - ❑ Use a separate bucket array for each band so columns with the same vector in different bands don’t hash to same bucket
- **Candidate column pairs** are those that hash to the same bucket for ≥ 1 band
- Tune b and r to catch most similar pairs, but few non-similar pairs



Example of Bands

Assume the following case:

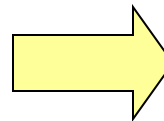
- Suppose 100,000 columns of M
 - Correspond to signatures for 100,000 documents
- Signatures of 100 integers (rows)
 - Correspond to 100 hash functions used in minhashing
- 4 bytes per integer
- Therefore, signatures take 40Mb (4 x 100 x 100,000)
- Choose $b = 20$ bands of $r = 5$ rows of integers/band
- **Goal:** Find pairs of documents that are at least $s = 0.8$ or 80% similar

Recall: Minhashing Example

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

Input 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 M

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

Analysis of Banding Technique

- Use b bands of r rows each
- **Pair of documents have Jaccard similarity t**
 - Probability that minhash signatures for the documents agree in any one particular row of the signature matrix is t
- Columns C_1 and C_2 in signature matrix have similarity t
- Pick any band (r rows)
 - Prob. that **all rows in band are equal** = t^r
 - Prob. **that not all r rows are equal** (some row in band is unequal) = $1 - t^r$
- Prob. that **no band has rows that are all equal** = $(1 - t^r)^b$
- Prob. that **at least 1 band has rows that are all equal** (which is the **probability of being a candidate pair**) = $1 - (1 - t^r)^b$

C_1, C_2 are 80% Similar

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

- Find pairs of $\geq s=0.8$ similarity, set $b=20$, $r=5$
- **Assume:** $\text{sim}(C_1, C_2) = 0.8$
 - ☐ Since $\text{sim}(C_1, C_2) \geq s$, we want C_1, C_2 to be a **candidate pair**
 - ☐ We want them to hash to at **least 1 common bucket** (at least one band is identical)
- **Probability C_1, C_2 identical in one particular band:**
 $t^r = (0.8)^5 = 0.328$
- Probability C_1, C_2 are **not** similar in all of the 20 bands:
 $(1 - t^r)^b = (1 - 0.328)^{20} = 0.00035$
 - ☐ i.e., about .035% of the 80%-similar column pairs are **false negatives** (truly similar pairs that we miss)
 - ☐ We would find **99.965% pairs of truly similar documents**

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

C_1, C_2 are 30% Similar

- Find pairs of $\geq s=0.3$ similarity, set $b=20, r=5$
- Assume:** $\text{sim}(C_1, C_2) = 0.3$
 - Since $\text{sim}(C_1, C_2) < s$ we want C_1, C_2 to hash to **NO common buckets** (all bands should be different)
 - Should **NOT** be a candidate pair!
- Probability C_1, C_2 identical in one particular band:**

$$t^r = (0.3)^5 = 0.00243$$

- Will identify C_1, C_2 as candidate pair if they are **identical in at least one band**
- Probability C_1, C_2 identical in at least 1 of 20 bands:

$$1 - (1 - t^r)^b = 1 - (1 - 0.00243)^{20} = 0.0474$$

- Approximately 4.74% pairs of docs with similarity 0.3% end up becoming **candidate pairs**

- They are **false positives** (dissimilar documents that must be examined as candidate pairs but will have similarity below

Other False Positives and False Negatives

For $b=20$, $r=5$, we may compute ALL the false positives and false negatives as follows:

For false negatives: (using the procedure on slide#15 to compute "?" below):

- How many true 100%-similar pairs we may miss: 0.0% (obvious)
- How many true 90%-similar pairs we may miss: ?.?%
- How many true 80%-similar pairs we may miss: 0.035% (see slide#15)
- How many true 70%-similar pairs we may miss: ?.?%
- How many true 60%-similar pairs we may miss: ?.?%
- How many true 50%-similar pairs we may miss: ?.?%

For false positives: (using the procedure on slide#16 to compute "?" below):

- How many 40%-similar pairs we may mistakenly taken as positive: ?.?%
- How many 30%-similar pairs we may mistakenly taken: 4.74% (see slide#16)
- How many 20%-similar pairs we may mistakenly taken: ?.?%
- How many 10%-similar pairs we may mistakenly taken: ?.?%
- How many 0%-similar pairs we may mistakenly taken: 0.0% (obvious)

LSH Involves a Tradeoff

- **Pick:**

- ☐ The number of Min-Hashes (rows of M)

- ☐ The number of bands b , and

- ☐ The number of rows r per band

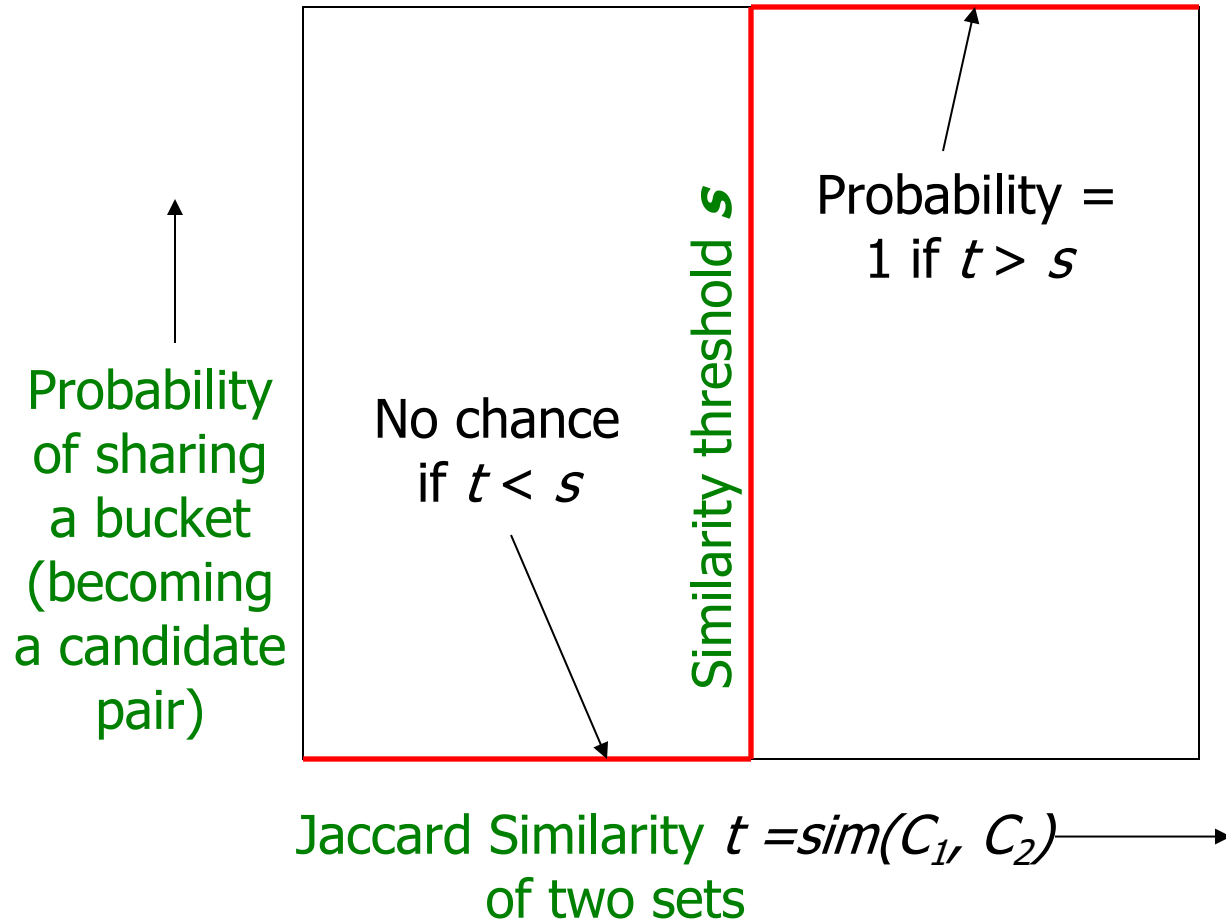
to balance false positives/negatives

- **Example:** If we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up

Example of Tradeoffs

- Previous example: 20 bands of 5 rows each
 - ❓ Probability of false negatives when C1, C2 are 80% similar:
0.00035
 - ❓ Probability of false positives when C1, C2 are 30% similar:
0.0474
- What if we use 15 rows of 5 bands each (smaller signature matrix)?
 - ❓ **Probability of false negatives higher** when C1, C2 are 80% similar:
 - $(1 - t^r)^b = (1 - 0.8^{15})^5 = (1 - 0.035)^5 = 0.837$
 - ❓ **Probability of false positives lower** when C1, C2 are 30% similar:
 - $1 - (1 - t^r)^b = 1 - (1 - 0.3^{15})^5 = 7.174^{-8}$

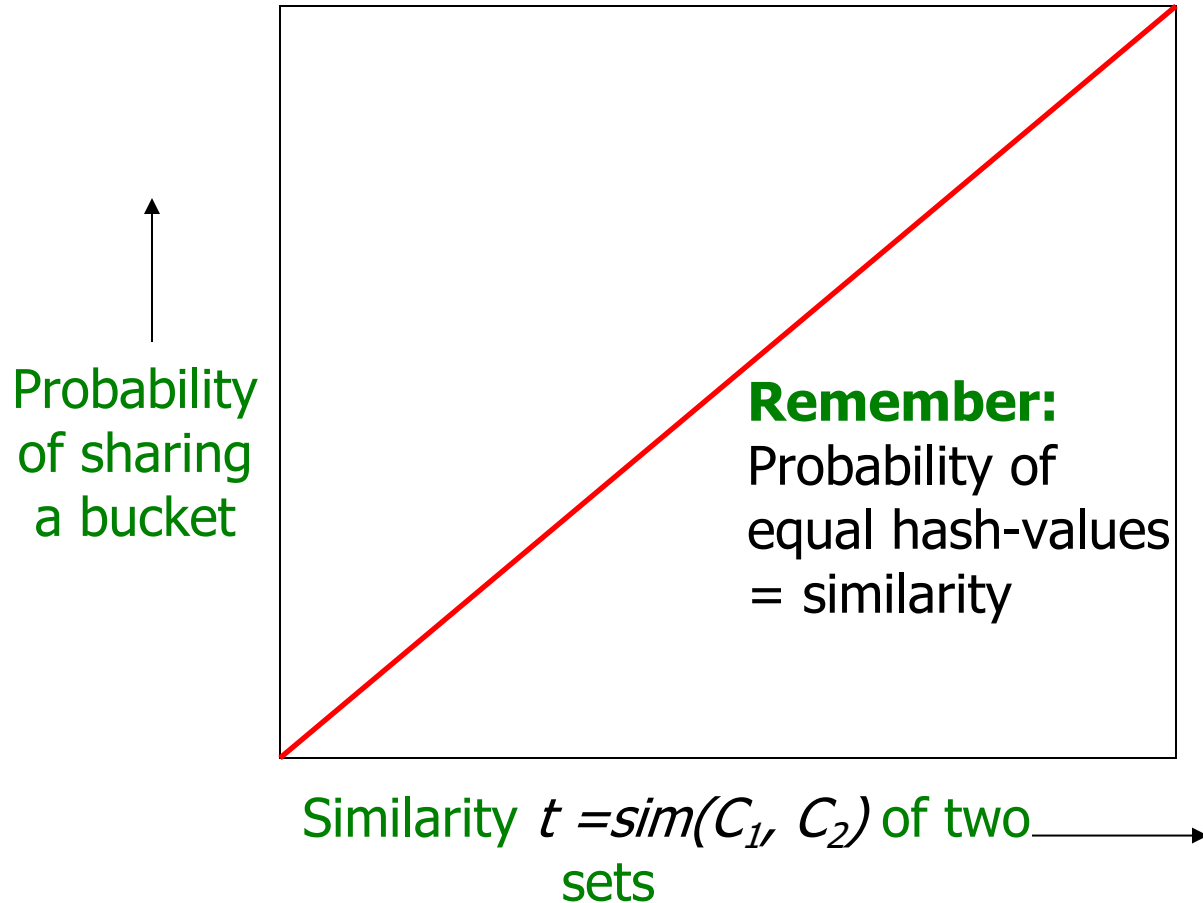
Analysis of LSH – What We Want



What 1 Band (b=1) of 1 Row (r=1) Gives You

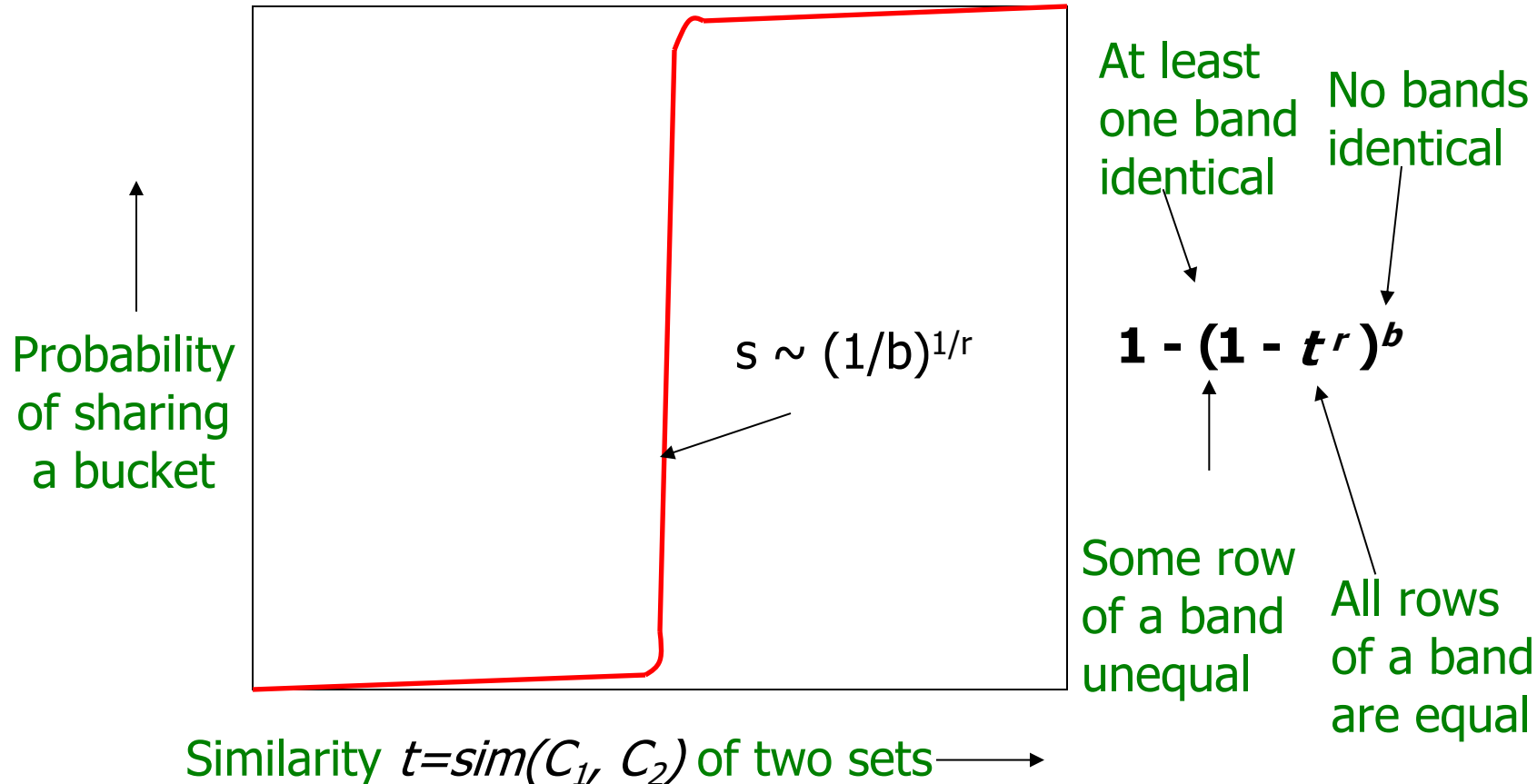
- Compare two values in similarity matrix

$$1 - (1 - t^r)^b = 1 - (1 - t^1)^1 = t$$



What b Bands of r Rows Gives You: $1 - (1 - t^r)^b$

- Form of an S-curve, regardless of values of b and r
- Threshold s is where rise of curve is steepest: approximately $(1/b)^{1/r}$



$r=5, b=20$, for $t=0.9$: $1-(1-t^r)^b=0.99999$; for $t=0.1$: 0.0000199

Example: $b = 20; r = 5$

- Similarity t of two columns
- Prob. that at least 1 band is identical (so a candidate pair):

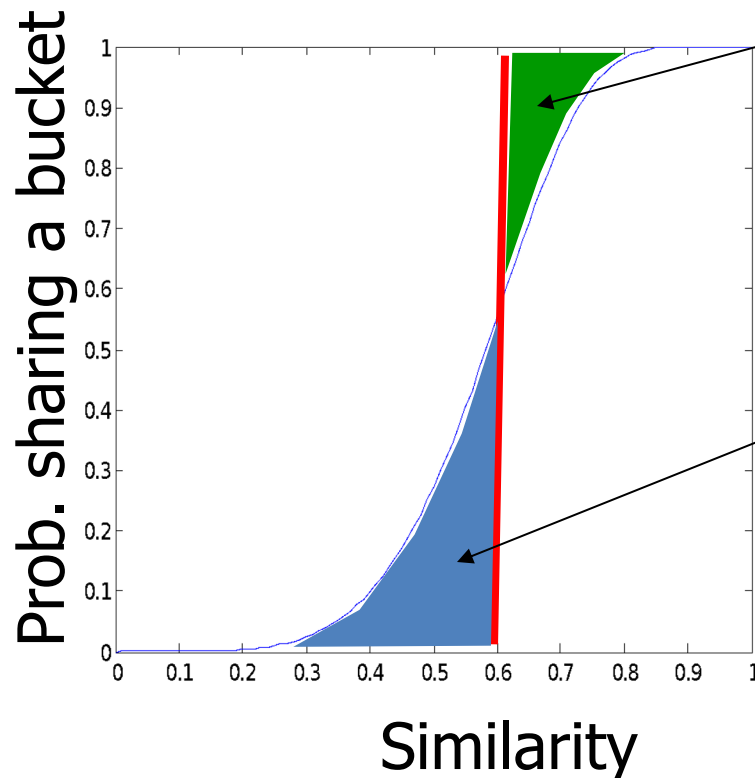
t	$1-(1-t^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

- Not an ideal step function
- Probability rises by more than 0.6 going from similarity $t = 0.4$ to $t = 0.6$
- Slope in middle > 3.0
- $0.6/(0.6-0.4)=3.0$

Picking r and b : The S-curve

- Picking r and b to get the best S-curve

50 hash-functions ($r=5$, $b=10$)



Green area:

False Negative rate

Similar documents that are not identified as candidate pairs

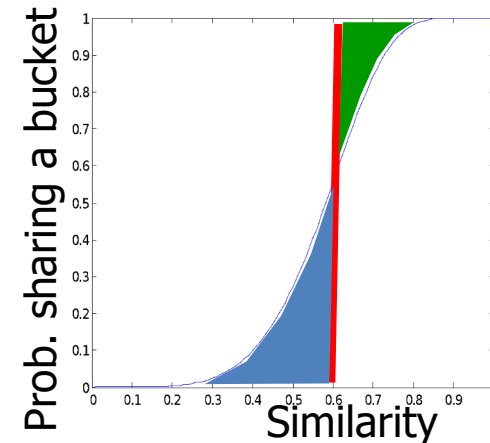
Blue area:

False Positive rate

Dissimilar documents that are identified as candidate pairs

Picking b and r

- Threshold s defines how similar documents have to be for them to be regarded as a similar pair (e.g., $s = 0.8$)
- Length n for minhash signatures
- Pick number of bands b and number of rows r such that $br = n$ and threshold s is approximately $(1/b)^{1/r}$
- To avoid false negatives (green area):
 - ❓ Select b and r to produce a threshold lower than s
- To avoid false positives (blue area):
 - ❓ Select b and r to produce a higher threshold than s



Example
: $n=100$

b	r	$(1/b)^{1/r}$
50	2	0.1414
20	5	0.5493
10	10	0.7943
5	20	0.9227

Example

- $(1/b)^{1/r}$ represents the threshold of the S curve for function
- $1 - (1 - t^r)^b$, the probability of being a candidate pair
- If $s=0.6$ (similarity of documents to be a candidate pair) what values should you choose for b and r to reduce the number of **false negatives**?
- **To avoid false negatives:** Select b and r to produce a threshold lower than s
- **To avoid false positives:** Select b and r to produce a higher threshold than s
- Could choose $(b=20, r=5)$ or $(b=50, r=2)$: both give threshold lower than s
- Better answer probably $b=20, r=5$
- Because $b=50, r=20$ will have a higher rate of false positives: TRADEOFFS

Example
: $n=100$

b	r	$(1/b)^{1/r}$
50	2	0.1414
20	5	0.5493
10	10	0.7943
5	20	0.9227

LSH Summary

- Tune M , b , r to identify almost all candidate pairs with similar signatures, but eliminate most pairs that do not have similar signatures
- Then check in main memory that candidate pairs really do have similar signatures
- **Optional:** In another 2nd pass through data, check that the remaining candidate pairs really represent similar documents

Summary: 3 Steps

- **Shingling:** Convert documents to sets
 - ☐ We used hashing to assign each shingle an ID
- **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
 - ☐ We used **similarity preserving hashing** to generate signatures with property $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
 - ☐ We used hashing to get around generating random permutations
- **Locality-Sensitive Hashing:** Focus on pairs of signatures likely to be from similar documents
 - ☐ We used hashing to find **candidate pairs** of similarity $\geq s$

Combining the techniques (1)

1. Pick a value of k and **construct from each document the set of k -shingles**
 - ☐ Optionally **hash the k -shingles** to shorter bucket numbers
2. **Sort the document-shingle pairs** to order them by shingle
 - ☐ Which sets contain which elements (shingles)
3. Pick a length n for minhash signatures corresponding to n minhash functions and **compute the minhash signatures** for all the documents

Combining the techniques (2)

4. **Choose threshold s** that defines how similar documents have to be for them to be regarded as a “similar pair”
 - ❑ **Pick number of bands b and number of rows r** such that $br = n$
 - ❑ **Adjust b and r to limit false positives or negatives**
5. **Construct candidate pairs with LSH technique**
6. **Examine candidate pair signatures** and determine whether fraction of components where they agree is at least s
7. **Optionally**, if signatures are sufficiently similar, **compare documents** to check they are truly similar

CHARACTERISTICS OF LSH

Locality Sensitive Hashing

- *Or Near-neighbor search*
- Minhashing is one example of a **family of functions** (the minhash functions) **that can be combined** (by the banding technique) **to distinguish strongly between pairs at a low distance from pairs at a high distance**
- Steepness of the S-curve reflects how effectively we can **avoid false positives and false negatives** among the candidate pairs
- Section 3.6: more general theory of Locality Sensitive Functions

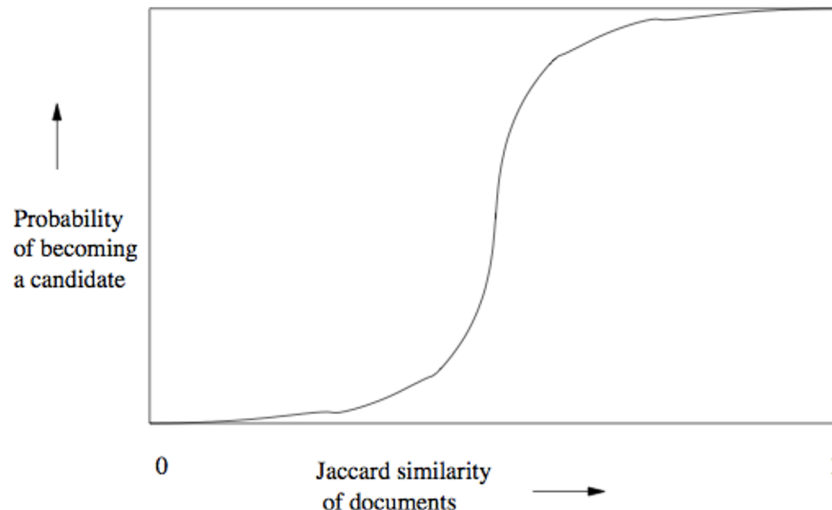


Figure 3.7: The S-curve

Families of Functions for LSH

- **Families of functions** (including minhash functions) that can serve to **produce candidate pairs efficiently**
 - Space of sets and Jaccard distance OR other space and/or distance measure
- **Three conditions for family of functions:**
 1. **More likely to make close pairs be candidate pairs than distant pairs**
 2. **Statistically independent**
 3. **Efficient** in two ways
 1. **Be able to identify candidate pairs in time much less than time to look at all pairs**
 2. **Combinable to build functions better at avoiding false positives and negatives** (e.g., banding technique takes single minhash functions, combines them to produce S-curve shape we want)

Locality-Sensitive Functions

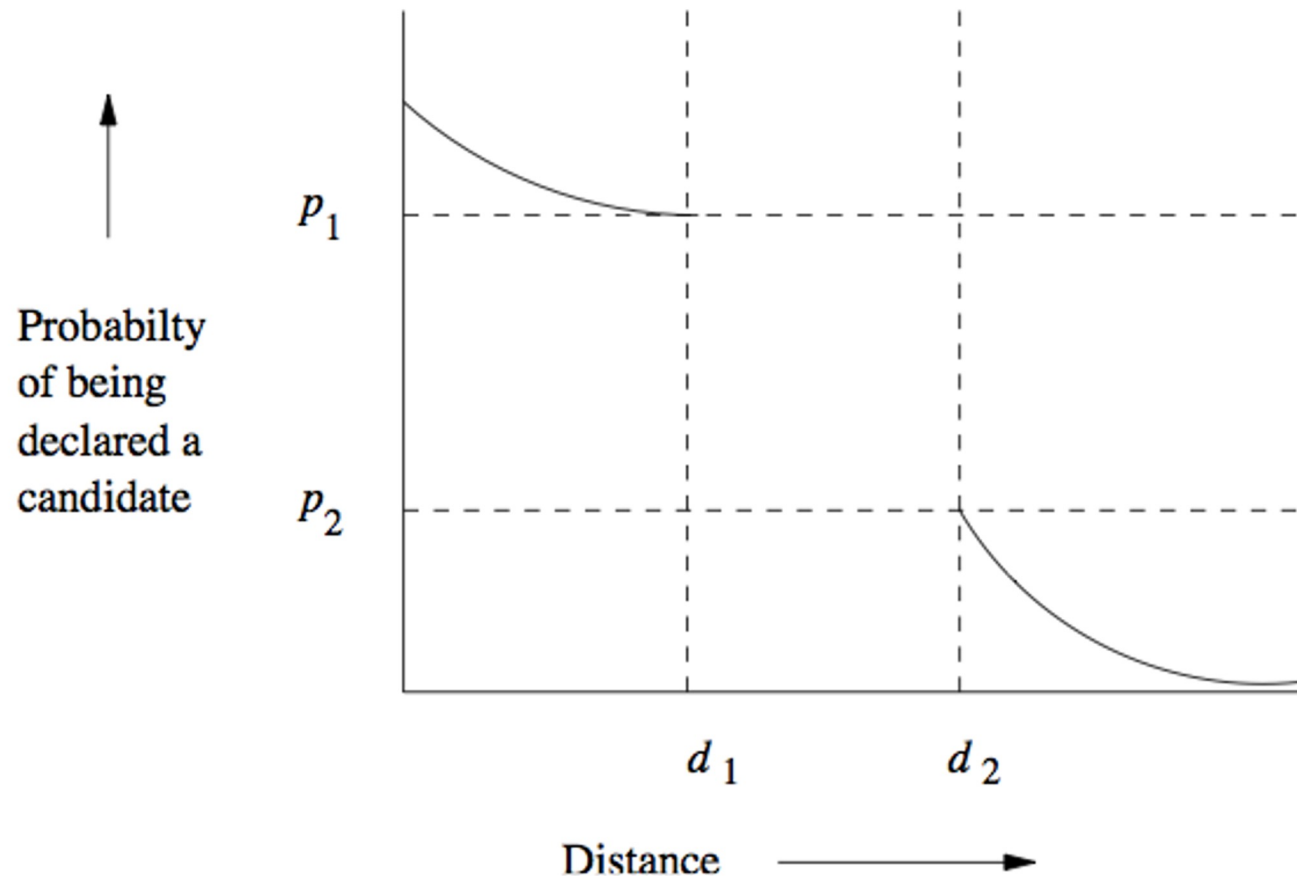


Figure 3.9: Behavior of a (d_1, d_2, p_1, p_2) -sensitive function

LS Families of Hash Functions

- Suppose we have a **space S of points** with a **distance measure d**
- A **family H of hash functions** is said to be **(d_1, d_2, p_1, p_2) -sensitive** if for any x and y in S :
 1. If $d(x, y) \leq d_1$, then prob. over all h in H , that $h(x) = h(y)$ is at least p_1
 2. If $d(x, y) \geq d_2$, then prob. over all h in H , that $h(x) = h(y)$ is at most p_2
- Note: we say nothing about what happens when the distance between items is between d_1 and d_2
 - ❓ But can make d_1 and d_2 as close as we wish
 - ❓ Can drive p_1 and p_2 apart while keeping d_1 and d_2 fixed

Locality Sensitive Hashing for Other Distance Measures

- We focused on **minhashing**, a locality sensitive hashing family that uses **Jaccard distance**
 - ❑ Based on sets representing documents and their Jaccard similarity
- Book covers LSH families for **other distance measures**:
 - ❑ **Euclidean distance**: based on the locations of points in a *Euclidean space* with some number of real-valued dimensions
 - ❑ **Cosine distance**: angle between vectors from the origin to the points in question
 - ❑ **Edit distance**: number of inserts and deletes to change one string into another
 - ❑ **Hamming Distance**: number of positions in which bit vectors differ

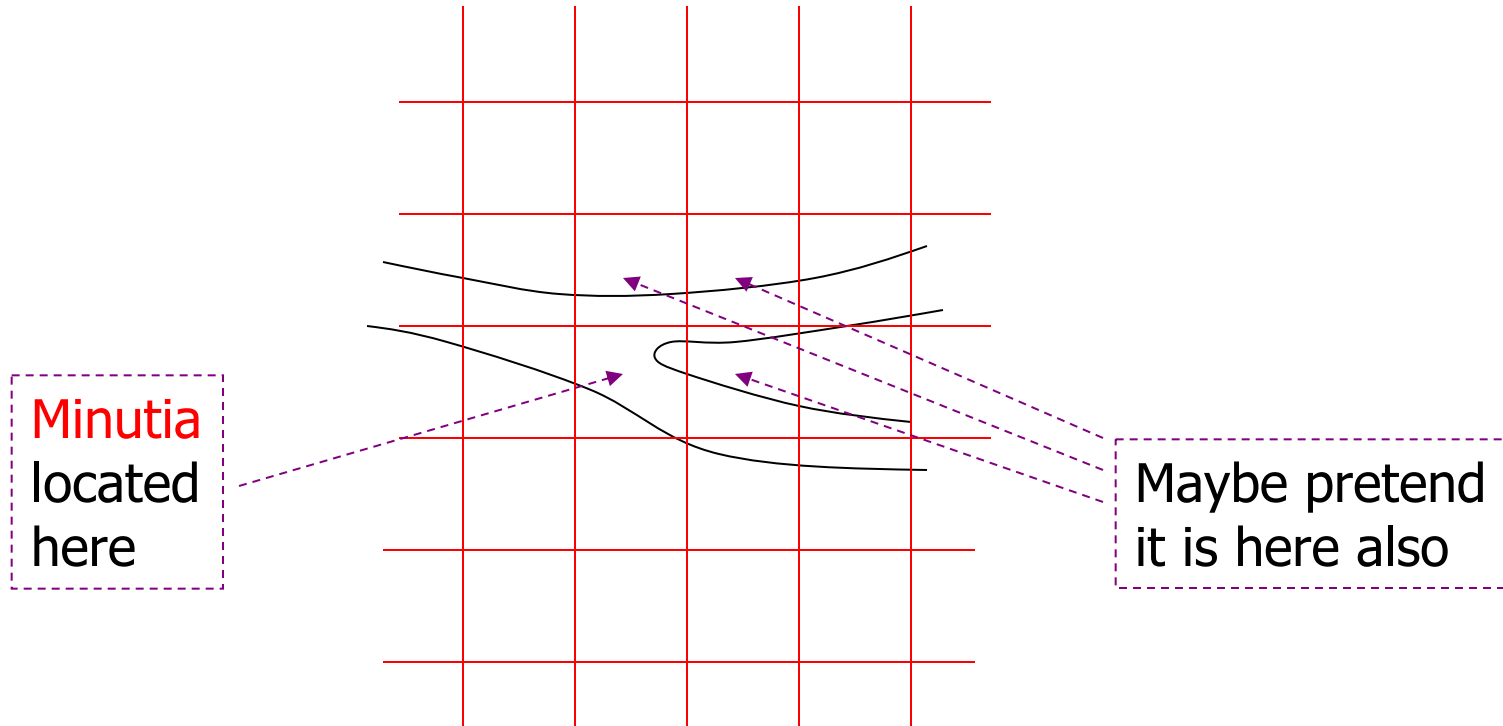
LSH and Shingling Application Examples

- Matching fingerprints
- Identifying similar news articles

LSH for Fingerprints

- Typical representation is not an image, but **set of locations in which minutiae are located**
 - ❑ **Place where something unusual happens**: two ridges merging or a ridge ending
- **Place a grid over a fingerprint**
 - ❑ Normalize for size and orientation so that identical prints will overlap
- **Represent fingerprint by set of grid points where minutiae are located**
 - ❑ Possibly, treat minutiae near a grid boundary as if also present in adjacent grid points

Discretizing Minutiae



Place a **minutia** in several adjacent grid squares if it lies close to the border of the squares

Applying LSH to Fingerprints

- **Make a bit vector for each fingerprint's set of grid points with minutiae**

□ Similar to set representing a document: 1 if the shingle is in the document, 0 otherwise

- We could minhash the bit vectors to obtain signatures

□ But since there probably aren't too many grid points, we can work from the bit-vectors directly

Matching Fingerprints with LSH:

Many-to-many problem

- **Many-to-many version of fingerprint matching:** take an entire database of fingerprints and identify if there are any pairs that represent the same individual
 - ❑ Analogous to finding similar documents among millions of documents
- **Define a locality-sensitive family of hash functions:**
 - ❑ Each function f in the family F is defined by 3 grid squares
 - ❑ Function f says “yes” for two fingerprints if both have minutiae in all three grid squares, otherwise, f says “no”
 - ❑ “Yes” means the two fingerprints are candidate pairs
- **Sort of “bucketization”**
 - ❑ Each set of three points creates one bucket
 - ❑ Function f sends fingerprints to its bucket that have minutiae in all three grid points of f
- **Compare all fingerprints in each of the buckets**

Matching Fingerprints with LSH: Many-to-One Problem

- **Many-to-one version** : A fingerprint has been found at a crime scene, and we want to **compare it with all fingerprints in a large database to see if there is a match**
- Could use many functions f from family F
- **Precompute their buckets of fingerprints to which they answer “yes” on the large database**
- **For a new fingerprint:**
 - ❑ **Determine which buckets it belongs to**
 - ❑ **Compare it with all fingerprints found in any of those buckets**

Example 3.22

- 1024 functions chosen randomly from F
 - Each function f says “yes” for two fingerprints if both have minutiae in all three grid squares, otherwise, f says “no”
- Suppose **typical fingerprints have minutiae in 20% of the grid points**
- Suppose **fingerprints from the same finger agree in at least 80% of their points**
- **Probability two random fingerprints each have 1 in all three points = $(0.2)^6 = .000064$**
 - 2 fingerprints, 3 points each, all independent events

First image
has 1 in a
point

Example: Continued

Second image
of same finger
also has 1

- Probability **two fingerprints from the same finger each have 1's in three given points** =

$$((0.2)(0.8))^3 = .004096$$

(Analogy: t^r)

- Prob. for **at least one of 1024 sets of three points** = $1 - (1 - .004096)^{1024} = .985$

(Analogy:
 $1 - (1 - t^r)^b$)

- But for **random fingerprints**:

- $1 - (1 - .000064)^{1024} = .063$

**6.3% false
positives**

**1.5% false
negatives**

Choosing the number of functions from F

- **Want to use many functions from F , but not too many**
- **Want a good probability of matching fingerprints from the same finger while not having too many false positives**
- Previous example: only 1.5% chance we fail to identify a print on the gun (false negative), but have to look at 6.3% of entire database (due to false positives)
- **Increasing number of functions from F increases number of false positives**
 - Only a small benefit in reducing false negatives below 1.5%
- **Can use constructions/combinations of functions**
 - Several examples in the chapter

Finding Same/Similar News Articles

- **Want to organize large repository of online news articles**
 - ❑ Group together web pages derived from same basic text
- **Scenario:** the same article, say from the Associated Press, appears on the Web site of many newspapers, but looks quite different
- **Each newspaper surrounds the text of the article with:**
 - ❑ Its own logo and text
 - ❑ Ads
 - ❑ Perhaps links to other articles
- **A newspaper may also “crop” the article (delete parts)**

Variation on Shingling

- **Looks like earlier problem:** find documents whose shingles have high Jaccard similarity
- **But: Shingling treats all parts of document equally**
- For this application, **we want to ignore certain parts of the documents** (e.g., ads, links to other articles, etc.)
- **There is a difference between text that appears in prose and text in ads or headlines/links**
 - ❑ Prose contains greater frequency of **stop_words**
 - E.g., common words like “and” or “the”
 - ❑ Common to use list of several hundred most frequent words

New Shingling Technique

- News articles have a lot of stop words, while ads do not
 - ❑ “Buy Sudzo” vs. “I recommend that you buy Sudzo for your laundry.”
- Define a *shingle* to be a stop word plus the next two following words
 - ❑ Shingles are: “I recommend that”, “that you buy”, “you buy Sudzo”, “for your laundry”, “your laundry <nextword>”
- Then compare the similarity of the sets of shingles that represent each document
 - ❑ Don’t use minhashing or LSH in this example

Why it Works

- By requiring each shingle to have a stop word: **bias the mapping from documents to shingles** so it picked more shingles from the article than from the ads
- **Pages with the same article, but different ads, have higher Jaccard similarity** than those with the same ads, but different articles