DATA MINING
LECTURE 4

Similarity and Distance

Sketching, Locality Sensitive Hashing

SIMILARITY AND DISTANCE

Thanks to:

Tan, Steinbach, and Kumar, “Introduction to Data Mining”

Rajaraman and Ullman, “Mining Massive Datasets”

Similarity and Distance

- For many different problems we need to quantify how close two objects are.
- Examples:
  - For an item bought by a customer, find other similar items
  - Group together the customers of site so that similar customers are shown the same ad.
  - Group together web documents so that you can separate the ones that talk about politics and the ones that talk about sports.
  - Find all the near-duplicate mirrored web documents.
  - Find credit card transactions that are very different from previous transactions.
- To solve these problems we need a definition of similarity, or distance.
  - The definition depends on the type of data that we have

What is Data?

- Collection of data objects and their attributes
- An attribute is a property or characteristic of an object
  - Examples: eye color of a person, temperature, etc.
  - Attribute is also known as variable, field, characteristic, or feature
- A collection of attributes describe an object
  - Object is also known as record, point, case, sample, entity, or instance

Attributes

Objects

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Dimensionality: Number of attributes

Types of Attributes

- There are different types of attributes
  - Nominal – Categorical
    - Examples: ID numbers, eye color, zip codes
    - There is no known ordering or comparison
  - Ordinal
    - Examples: rankings (e.g, good, fair, bad), grades (A,B,C), height in {tall, medium, short}
    - We can order, but not always clear how to compare

Numeric

  - Interval
    - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
    - We can take the difference in order to compare
  - Ratio
    - Examples: temperature in Kelvin, length, time, counts
    - We can take differences as well as ratios.

Discrete and Continuous Attributes

- Discrete Attribute
  - Has only a finite or countably infinite set of values
  - Examples: zip codes, counts, or the set of words in a collection of documents
  - Often represented as integer variables.
  - Note: binary attributes are a special case of discrete attributes

- Continuous Attribute
  - Has real numbers as attribute values
  - Examples: temperature, height, or weight.
  - Practically, real values can only be measured and represented using a finite number of digits.
  - Continuous attributes are typically represented as floating-point variables.

Numeric Data

- If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute

- Such data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Categorical Data

- Data that consists of a collection of records, each of which consists of a fixed set of categorical attributes

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	High	**No**
2	No	Married	Medium	**No**
3	No	Single	Low	**No**
4	Yes	Married	High	**No**
5	No	Divorced	Medium	**Yes**
6	No	Married	Low	**No**
7	Yes	Divorced	High	**No**
8	No	Single	Medium	**Yes**
9	No	Married	Medium	**No**
10	No	Single	Medium	**Yes**

Document Data

- Each document becomes a `term' vector,
  - each term is a component (attribute) of the vector,
  - the value of each component is the number of times the corresponding term occurs in the document.
  - Bag-of-words representation – no ordering

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

Transaction Data

- Each record (transaction) is a set of items.

TID	*Items*
1	**Bread, Coke, Milk**
2	**Beer, Bread**
3	**Beer, Coke, Diaper, Milk**
4	**Beer, Bread, Diaper, Milk**
5	**Coke, Diaper, Milk**

- A set of items can also be represented as a binary vector, where each attribute is an item.
- A document can also be represented as a set of words (no counts)

Ordered Data

- Genomic sequence data

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

- Data is a long ordered string

Types of data

- Numeric data: Each object is a point in a multidimensional space
- Categorical data: Each object is a vector of categorical values
- Set data: Each object is a set of values (with or without counts)
  - Sets can also be represented as binary vectors, or vectors of counts
- Ordered sequences: Each object is an ordered sequence of values.

Similarity and Distance

- Similarity
  - Numerical measure of how alike two data objects are.
    - A function that maps pairs of objects to real values
  - Is higher when objects are more alike.
  - Often falls in the range [0,1]
  - Sometimes in [-1,1]
- Distance
  - Numerical measure of how different are two data objects
    - A function that maps pairs of objects to real values
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies
- Closeness refers to a similarity or distance

Similarity/Dissimilarity for Simple Attributes

p and *q* are the attribute values for two data objects.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Table 5.1. Similarity and dissimilarity for simple attributes

Distance Metric

- A distance function d is a distance metric if it is a function from pairs of objects to real numbers such that:
  1. $d(x,y) \geq 0$. (non-negativity)
  2. $d(x,y) = 0$ iff $x = y$. (identity)
  3. $d(x,y) = d(y,x)$. (symmetry)
  4. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality).

Triangle Inequality

- Triangle inequality guarantees that the distance function is well-behaved.
  - The direct connection is the shortest distance

- It is useful also for proving properties about the data
  - For example, suppose I want to find an object that minimizes the sum of distances to all points in my dataset
  - If I select the best point from my dataset, the sum of distances I get is at most twice that of the optimal point.

Properties of Similarity

- Desirable properties for similarity
  1. s(p, q) = 1 (or maximum similarity) only if p = q. (Identity)
  2. s(p, q) = s(q, p) for all p and q. (Symmetry)

Distances for real vectors

- Vectors $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$

- **L_p** norms or Minkowski distance:

$$L_p(x, y) = \left[|x_1 - y_1|^p + \cdots + |x_d - y_d|^p\right]^{1/p}$$

- **L_2** norm: Euclidean distance:

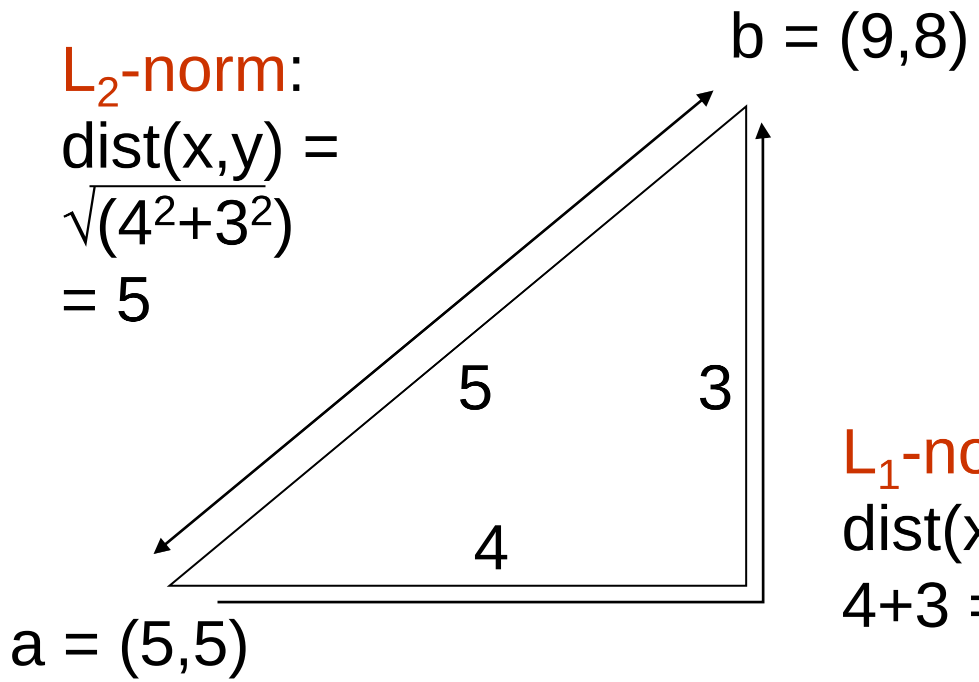
$$L_2(x, y) = \sqrt{|x_1 - y_1|^2 + \cdots + |x_d - y_d|^2}$$

- **L_1** norm: Manhattan distance:

$$L_1(x, y) = |x_1 - y_1| + \cdots + |x_d - y_d|$$

Example of Distances

L$_2$-norm:
dist(x,y) = $\sqrt{(4^2+3^2)}$
= 5



L$_1$-norm:
dist(x,y) =
4+3 = 7

Another Minkowski distance

- Vectors $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$

- $\mathbf{L_p}$ norms or Minkowski distance:

$$L_p(x, y) = \left[|x_1 - y_1|^p + \cdots + |x_d - y_d|^p\right]^{1/p}$$

- $\mathbf{L_\infty}$ norm:

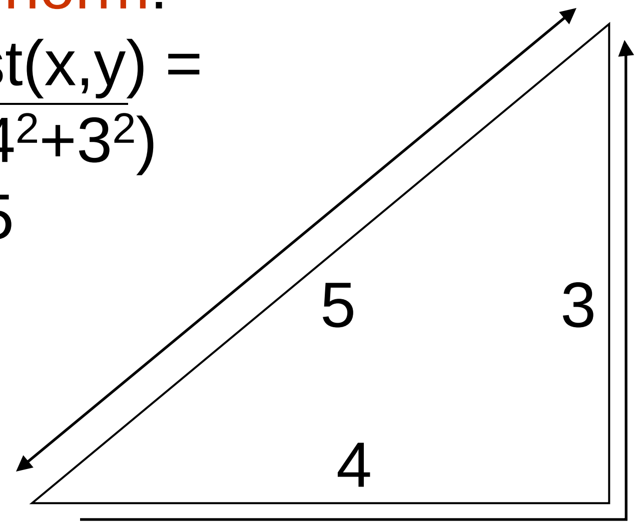
$$L_\infty(x, y) = \max\{|x_1 - y_1|, \dots, |x_d - y_d|\}$$

  - The limit of $\mathbf{L_p}$ as p goes to infinity.

Example of Distances

L_2-norm:
dist(x,y) =
$\sqrt{(4^2+3^2)}$
= 5

b = (9,8)

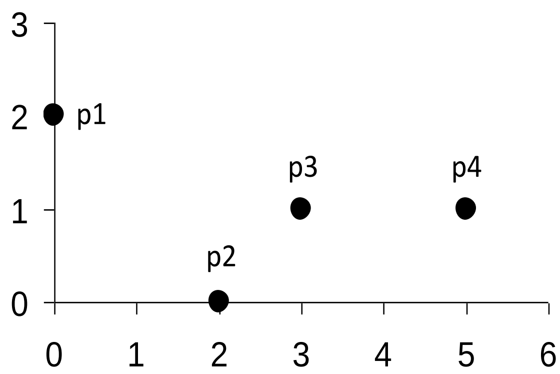


a = (5,5)

L_1-norm:
dist(x,y) =
4+3 = 7

L_∞-norm:
dist(x,y) =
max{3,4} = 4

Minkowski Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

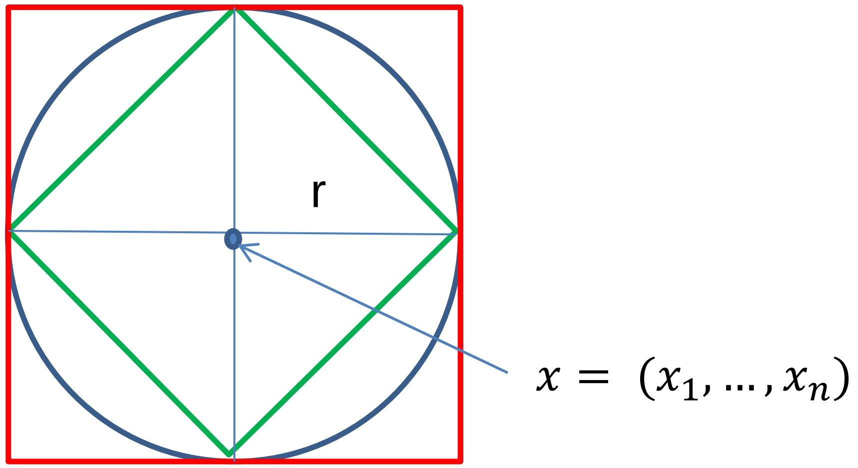
L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix

Example



Green: All points y at distance $L_1(x,y) = r$ from point x

Blue: All points y at distance $L_2(x,y) = r$ from point x

Red: All points y at distance $L_\infty(x,y) = r$ from point x

L_p distances for sets

- We can apply all the L_p distances to the cases of sets of attributes, with or without counts, if we represent the sets as vectors
  - E.g., a transaction is a 0/1 vector
  - E.g., a document is a vector of counts.

Cosine Similarity

- If d_1 and d_2 are two vectors, then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / ||d_1||\ ||d_2|| ,$$

where • indicates vector dot product and || *d* || is the length of vector *d*.

- Example:

d_1 = **3 2 0 5 0 0 0 2 0 0**
d_2 = **1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2$ = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

$||d_1||$ = $(3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5}$ = $(42)^{0.5}$ = 6.481

$||d_2||$ = $(1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5}$ = $(6)^{0.5}$ = 2.245

$\cos(d_1, d_2)$ = .3150

Cosine Similarity

- Geometric Interpretation

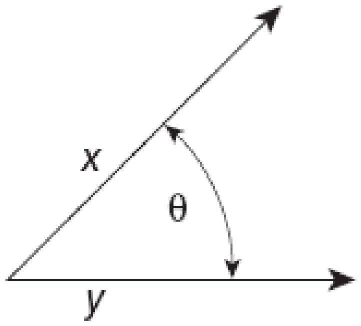


Figure 2.16. Geometric illustration of the cosine measure.

- If the vectors are correlated angle is zero degrees and cosine is 1
- If the vectors are orthogonal (no common coordinates) angle is 90 degrees and cosine is 0
- Note that if one vector is a multiple of another cosine is still 1 (maximum)
- Cosine is commonly used for comparing documents, where we assume that the vectors are normalized by the document length.

Example

document	Apple	Microsoft	Obama	Election
D1	10	20	0	0
D2	20	40	0	0
D2	0	0	10	20

cos(D1,D2) = 1

cos(D1,D3) = cos(D2,D3) = 0

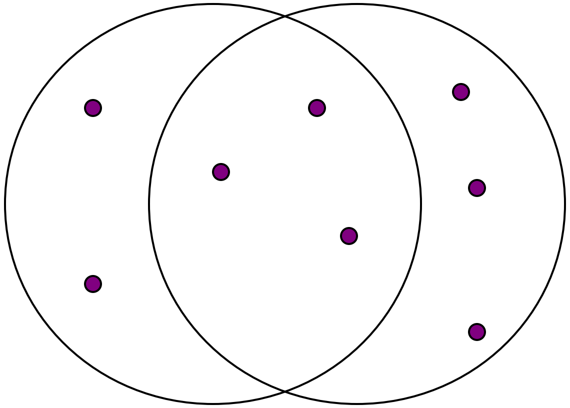
Example

document	Apple	Microsoft	Obama	Election
D1	1/3	2/3	0	0
D2	1/3	2/3	0	0
D2	0	0	1/3	2/3

$$\cos(D1,D2) = 1$$
$$\cos(D1,D3) = \cos(D2,D3) = 0$$

Jaccard Similarity of Sets

- The Jaccard similarity (Jaccard coefficient) of two sets C_1, C_2 is the size of their intersection divided by the size of their union.
  - JSim $(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$.



3 in intersection.
8 in union.
Jaccard similarity
= 3/8

- Jaccard distance Jdist = 1 - JSim

Example with documents

- D1 = {apple, released, new, iPhone}
- D2 = {apple, released, new, iPad}
- D3 = {new, apple, pie, recipie}

- JSim(D1,D2) = 3/5
- JSim(D1,D3) = JSim(D2,D3) = 2/6

Similarity Between Binary Vectors

- Objects, *p* and *q*, have only binary attributes
  - We can view them as sets and compute Jaccard
  - We also compute the Simple Matching Coefficient
- Compute similarities using the following quantities

M_{01} = the number of attributes where p was 0 and q was 1
M_{10} = the number of attributes where p was 1 and q was 0
M_{00} = the number of attributes where p was 0 and q was 0
M_{11} = the number of attributes where p was 1 and q was 1

- Simple Matching and Jaccard Coefficients

SMC = number of matches / number of attributes
$= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

J = number of 11 matches / number of not-both-zero attributes values
$= (M_{11}) / (M_{01} + M_{10} + M_{11})$

Jaccard treats 1's asymmetrically

SMC versus Jaccard: Example

p = 1 0 0 0 0 0 0 0 0 0
q = 0 0 0 0 0 0 1 0 0 1

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)
$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)
$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)
$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$SMC = (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

Hamming Distance

- Hamming distance is the number of positions in which bit-vectors differ.
- Example: p_1 = 10101; p_2 = 10011.
  - $d(p_1, p_2) = 2$ because the bit-vectors differ in the 3rd and 4th positions.
  - The L_1 norm for the binary vectors
- Hamming distance between two vectors of categorical attributes is the number of positions in which they differ.
- Example: x = (married, low income, cheat),
  y = (single, low income, not cheat)
  - d(x,y) = 2

Why Hamming Distance Is a Distance Metric

- d(x,x) = 0 since no positions differ.
- d(x,y) = d(y,x) by symmetry of "different from."
- d(x,y) $\geq$ 0 since strings cannot differ in a negative number of positions.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*.

Edit Distance for strings

- The edit distance of two strings is the number of inserts and deletes of characters needed to turn one into the other.
- Exampe: x = abcde ; y = bcduve.
  - Turn *x* into *y* by deleting a, then inserting u and v after d.
  - Edit distance = 3.
- Minimum number of operations can be computed using dynamic programming
- Common distance measure for comparing DNA sequences

Why Edit Distance Is a Distance Metric

- d(x,x) = 0 because 0 edits suffice.
- d(x,y) = d(y,x) because insert/delete are inverses of each other.
- d(x,y) $\geq$ 0: no notion of negative edits.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*.

Variant Edit Distances

- Allow insert, delete, and mutate.
  - Change one character into another.
- Minimum number of inserts, deletes, and mutates also forms a distance measure.
- Same for any set of operations on strings.
  - Example: substring reversal or block transposition OK for DNA sequences
  - Example: character transposition is used for spelling

Distances between distributions

- We can view a document as a distribution over the words

document	Apple	Microsoft	Obama	Election
D1	0.35	0.5	0.1	0.05
D2	0.4	0.4	0.1	0.1
D2	0.05	0.05	0.6	0.3

- KL-divergence (Kullback-Leibler) for distributions P,Q

$$D_{KL}(P\|Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- KL-divergence is asymmetric. We can make it symmetric by taking the average of both sides
- JS-divergence (Jensen-Shannon)

$$JS(P,Q) = \frac{1}{2} D_{KL}(P\|Q) + \frac{1}{2} D_{KL}(Q\|P)$$

SKETCHING AND LOCALITY SENSITIVE HASHING

Thanks to:

Rajaraman and Ullman, “Mining Massive Datasets”

Evimaria Terzi, slides for Data Mining Course.

Finding near-duplicates documents

- We will now consider the problem of finding duplicate and near-duplicate documents from a web crawl.
- Why is it important:
  - Identify mirrored web pages, and avoid indexing them, or serving them multiple times
  - Identify plagiarism
  - Find replicated stories in news and cluster them under a single story.
- What if we wanted exact duplicates?

Main issues

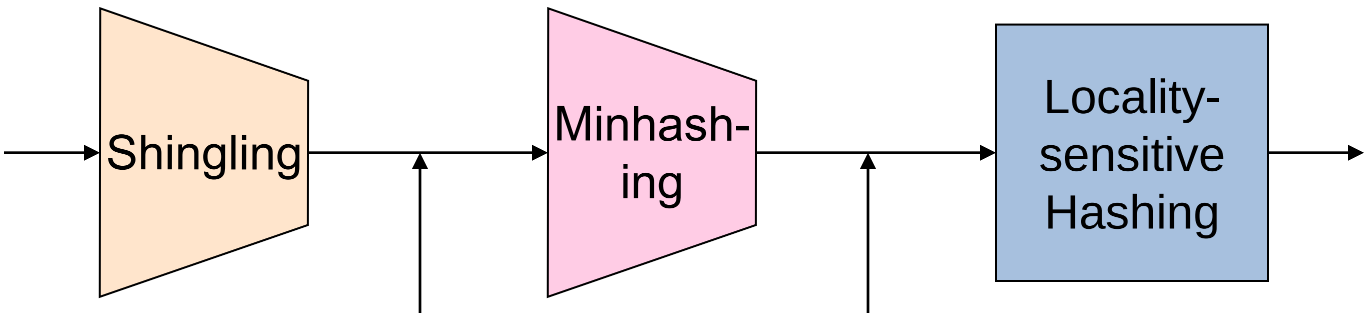
- What is the right representation of the document when we check for similarity?
  - E.g., representing a document as a set of characters will not do
- When we have billions of documents, keeping the full text in memory is not an option.
  - We need to find a shorter representation
- How do we do pairwise comparisons we billions of documents?
  - If exact match was the issue it would be ok, can we replicate this idea?

Three Essential Techniques for Similar Documents

1. Shingling : convert documents, emails, etc., to sets.
2. Minhashing : convert large sets to short signatures, while preserving similarity.
3. Locality-sensitive hashing : focus on pairs of signatures likely to be similar.

The Big Picture

Docu-
ment



Candidate pairs : those pairs of signatures that we need to test for similarity.

The set of strings of length *k* that appear in the doc- ument

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

Shingles

- A k -shingle (or k -gram) for a document is a sequence of k characters that appears in the document.
- Example: k=2; doc = abcab. Set of 2-shingles = {ab, bc, ca}.
  - Option: regard shingles as a bag, and count ab twice.
- Represent a doc by its set of k-shingles.

Shingling

- Shingle: a sequence of k contiguous characters

```
a rose is a rose is a rose
a rose is 
 rose is a
  rose is a 
   ose is a r
    se is a ro
     e is a ros
       is a rose
       is a rose 
        s a rose i
          a rose is
          a rose is 
```

Working Assumption

- Documents that have lots of shingles in common have similar text, even if the text appears in different order.
- Careful: you must pick k large enough, or most documents will have most shingles.
  - $k = 5$ is OK for short documents; $k = 10$ is better for long documents.

Shingles: Compression Option

- To compress long shingles, we can hash them to (say) 4 bytes.
- Represent a doc by the set of hash values of its *k*-shingles.
- Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared.

Rabin's fingerprinting technique

- Comparing two strings of size n

```
a = 10110                    a=b?
b = 11010
```

O(n) too expensive!

`f(a)=f(b)?`

$$A = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$$

$$B = 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0$$

```
f(a)= A mod p       p = small random prime
f(b)= B mod p       size O(logn loglogn)
```

- if **a=b** then **f(a)=f(b)**

  if **f(a)=f(b)** then **a=b** with high probability

Thought Question

- Why is it better to hash 9-shingles (say) to 4 bytes than to use 4-shingles?
- Hint: How random are the 32-bit sequences that result from 4-shingling?

Basic Data Model: Sets

- Document: A document is represented as a set shingles (more accurately, hashes of shingles)
- Document similarity: Jaccard similarity of the sets of shingles.
  - Common shingles over the union of shingles
  - $Sim\ (C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$.

- Although we use the documents as our driving example the techniques we will describe apply to any kind of sets.
  - E.g., similar customers or products.

From Sets to Boolean Matrices

- Rows = elements of the universal set (shingles)
- Columns = sets (documents)
- 1 in row *e* and column *S* if and only if *e* is a member of *S*.
- Column similarity is the Jaccard similarity of the sets of their rows with 1.
- Typical matrix is sparse.

Example: Jaccard Similarity of Columns

C_1	C_2		
0	1		*
1	0		*
1	1	*	*
0	0		
1	1	*	*
0	1		*

Sim (C_1, C_2) =

2/5 = 0.4

Aside

- We might not really represent the data by a boolean matrix.
- Sparse matrices are usually better represented by the list of places where there is a non-zero value.
- But the matrix picture is conceptually useful.

Outline: Finding Similar Columns

1. Compute signatures of columns = small summaries of columns.
2. Examine pairs of signatures to find similar signatures.
   - Essential: similarities of signatures and columns are related. The signatures preserve similarity.
3. Optional: check that columns with similar signatures are really similar.

Warnings

1. Comparing all pairs of signatures may take too much time, even if not too much space.
   - A job for Locality-Sensitive Hashing.
2. These methods can produce false negatives, and even false positives (if the optional check is not made).

Signatures

- Key idea: "hash" each column *C* to a small signature *Sig* (C), such that:

1. *Sig* (C) is small enough that we can fit a signature in main memory for each column.
2. *Sim* (C_1, C_2) is (almost) the same as the "similarity" of *Sig* (C_1) and *Sig* (C_2).

Four Types of Rows

- Given documents X and Y,
- Rows may be classified as:

type	X bit	Y bit
R_{11}	1	1
R_{10}	1	0
R_{01}	0	1
R_{00}	0	0

X	Y
1	1
1	0
0	0
0	0
0	0
1	1
1	1

Four Types of Rows

- Given documents X and Y,

- Rows may be classified as:

type	X bit	Y bit	
R_{11}	1	1	3
R_{10}	1	0	
R_{01}	0	1	
R_{00}	0	0	

X	Y
1	1
1	0
0	0
0	0
0	0
1	1
1	1

Four Types of Rows

- Given documents X and Y,

- Rows may be classified as:

type	X bit	Y bit	
R_{11}	1	1	3
R_{10}	1	0	1
R_{01}	0	1	
R_{00}	0	0	

X	Y
1	1
1	0
0	0
0	0
0	0
1	1
1	1

Four Types of Rows

- Given documents X and Y,
- Rows may be classified as:

type	X bit	Y bit	
R_{11}	1	1	3
R_{10}	1	0	1
R_{01}	0	1	0
R_{00}	0	0	

X	Y
1	1
1	0
0	0
0	0
0	0
1	1
1	1

Four Types of Rows

- Given documents X and Y,

- Rows may be classified as:

type	X bit	Y bit	
R_{11}	1	1	3
R_{10}	1	0	1
R_{01}	0	1	0
R_{00}	0	0	3

X	Y
1	1
1	0
0	0
0	0
0	0
1	1
1	1

Four Types of Rows

- Given documents X and Y,

- Rows may be classified as:

type	X bit	Y bit	
R_{11}	1	1	3
R_{10}	1	0	1
R_{01}	0	1	0
R_{00}	0	0	3

X	Y
1	1
1	0
0	0
0	0
0	0
1	1
1	1

- Also, R_{11} = # rows of type R_{11} , etc.
- Note $Sim(X,Y) = R_{11} / (R_{11} + R_{10} + R_{01})$.

Minhashing

- Imagine the rows permuted randomly.
- Define "hash" function $h(C)$ = the number of the first (in the permuted order) row in which column C has 1.
- Use several (e.g., 100) independent hash functions to create a signature.

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

A
C
G
F
B
E
D

	x1	x2	x3	X4
A	1	0	1	0
C	0	1	0	1
G	1	0	1	0
F	1	0	1	0
B	1	0	0	1
E	0	1	0	1
D	0	1	0	1

1	**2**	**1**	**2**

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

D
B
A
C
F
G
E

	x1	x2	x3	X4
D	0	1	0	1
B	1	0	0	1
A	1	0	1	0
C	0	1	0	1
F	1	0	1	0
G	1	0	1	0
E	0	1	0	1

2	**1**	**3**	**1**

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

C
D
G
F
A
B
E

	x1	x2	x3	X4
C	0	1	0	1
D	0	1	0	1
G	1	0	1	0
F	1	0	1	0
A	1	0	1	0
B	1	0	0	1
E	0	1	0	1

3	**1**	**3**	**1**

Surprising Property

- The probability (over all permutations of the rows) that $h(C_1) = h(C_2)$ is the same as $Sim(C_1, C_2)$.
- Both are $R_{11} / (R_{11} + R_{10} + R_{01})$!
- Why?
  - Look down the permuted columns C_1 and C_2 until we see a 1.
  - If it's a type-R_{11} row, then $h(C_1) = h(C_2)$. If a type-R_{10} or type-R_{01} row, then not.

Similarity for Signatures

- The similarity of signatures is the fraction of the hash functions in which they agree.

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

≈

x1	x2	x3	X4
1	2	1	2
2	1	3	1
3	1	3	1

	actual	Sig
(x1,x2)	0	0
(x1,x3)	0.75	2/3
(x1,x4)	1/7	0
(x2,x3)	0	0
(x2,x4)	0.75	1
(x3,x4)	0	0

Minhash algorithm

- Pick **k** (e.g., 100) permutations of the rows

- Think of **Sig(x)** as a new vector

- Let **Sig(x)[i]**: in the **i**-th permutation, the index of the **first row that has 1** for object **x**

Is it now feasible?

- Assume a billion rows
- Hard to pick a random permutation of 1…billion
- **Even representing a random permutation requires 1 billion entries!!!**
- How about accessing rows in permuted order?
- ☹

Being more practical

- Approximating row permutations: pick **k=100** (?) hash functions **($h_1,\ldots,h_k$)**

for each row **r**

  for each column **c**

    if **c** has **1** in row **r**

      for each hash function **h_i** **do**

        if **h_i (r)** is a smaller value than **M(i,c)** **then**

          M (i,c) = h_i (r);

M(i,c) will become the smallest value of **h_i(r)** for which column **c** has **1** in row **r**; *i*.e., **h_i (r)** gives order of rows for **i-**th permutation

Example

Row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

$h(x) = x \bmod 5$
$g(x) = 2x+1 \bmod 5$

	Sig1	Sig2
$h(1) = 1$	1	-
$g(1) = 3$	3	-
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	1	2
$g(3) = 2$	2	0
$h(4) = 4$	1	2
$g(4) = 4$	2	0
$h(5) = 0$	1	0
$g(5) = 1$	2	0

Implementation – (4)

- Often, data is given by column, not row.
  - E.g., columns = documents, rows = shingles.
- If so, sort matrix once so it is by row.
- And always compute $h_i(r)$ only once for each row.

Finding Similar Pairs

- Suppose we have, in main memory, data representing a large number of objects.
  - May be the objects themselves .
  - May be signatures as in minhashing.
- We want to compare each to each, finding those pairs that are sufficiently similar.

Checking All Pairs is Hard

- While the signatures of all columns may fit in main memory, comparing the signatures of all pairs of columns is quadratic in the number of columns.
- Example: 10^6 columns implies $5*10^{11}$ column-comparisons.
- At 1 microsecond/comparison: 6 days.

Locality-Sensitive Hashing

- General idea: Use a function f(x,y) that tells whether or not x and y is a candidate pair: a pair of elements whose similarity must be evaluated.

- For minhash matrices: Hash columns to many buckets, and make elements of the same bucket candidate pairs.

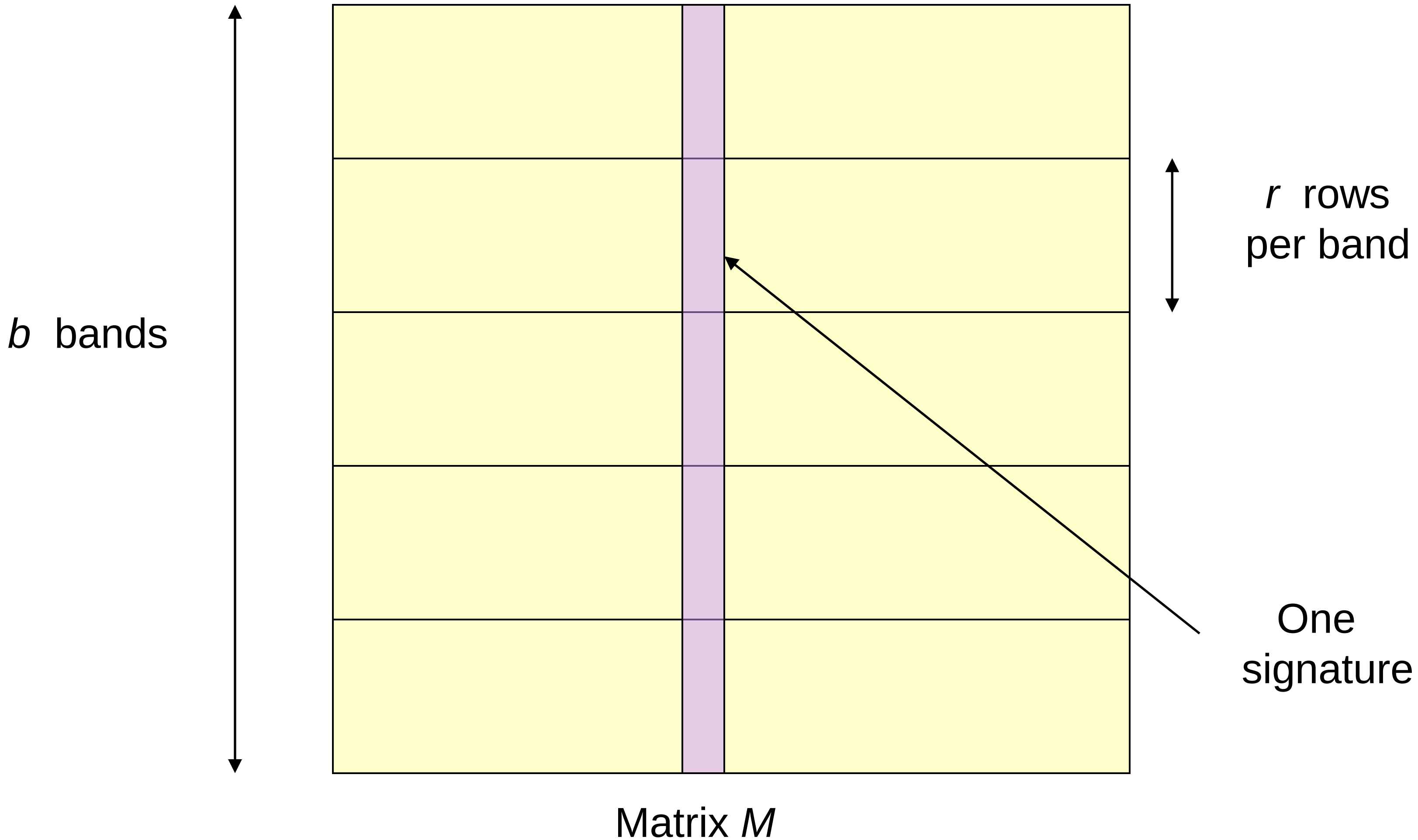
Candidate Generation From Minhash Signatures

- Pick a similarity threshold s, a fraction < 1.
- A pair of columns x and y is a candidate pair if their signatures agree in at least fraction s of the rows.
  - I.e., $M(i, c) = M(i, d)$ for at least fraction s values of i.

LSH for Minhash Signatures

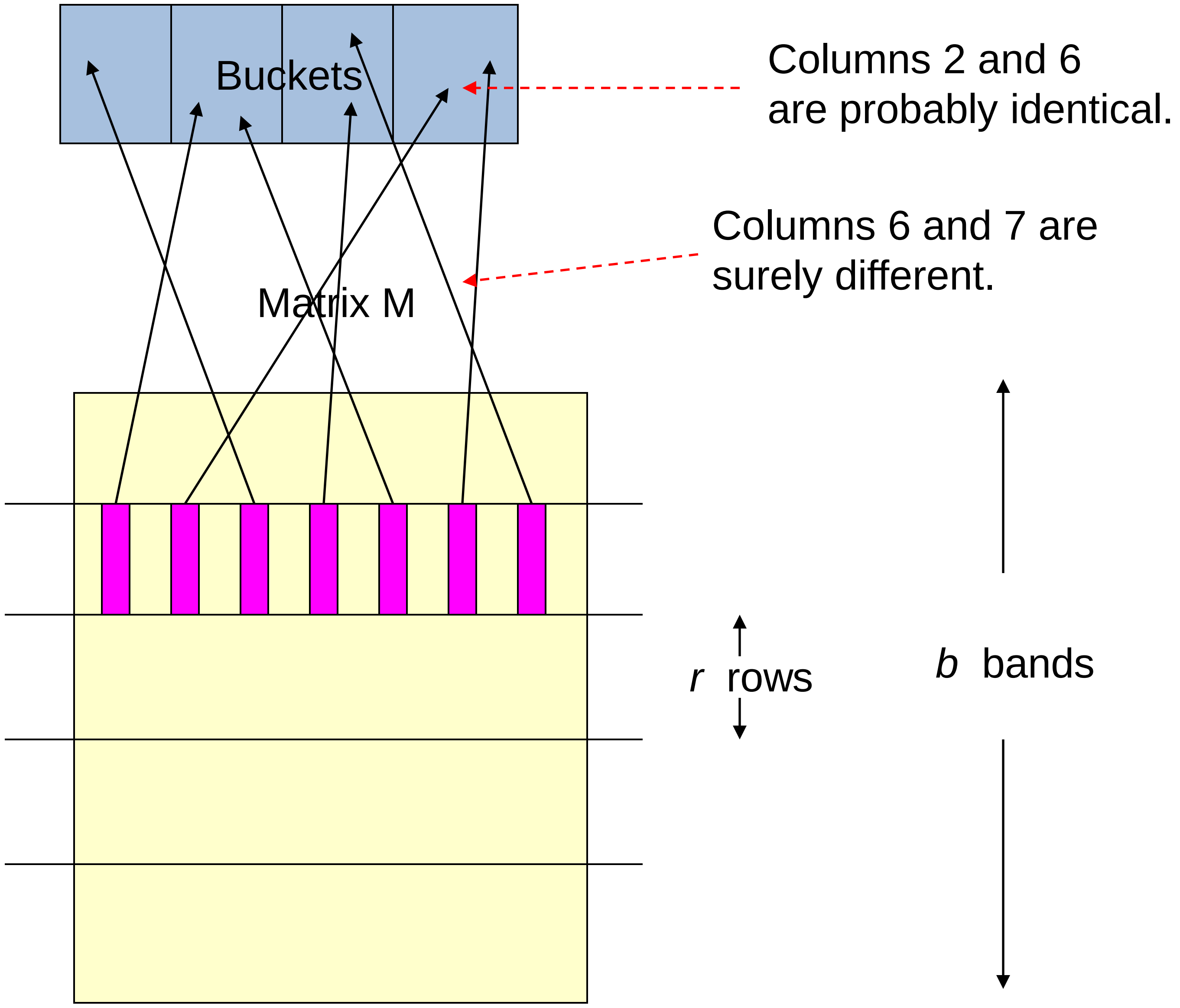
- Big idea: hash columns of signature matrix M several times.
- Arrange that (only) similar columns are likely to hash to the same bucket.
- While dissimilar columns are less likely to hash to the same bucket
- Candidate pairs are those that hash at least once to the same bucket.

Partition Into Bands



Partition into Bands – (2)

- Divide matrix M into b bands of r rows.
- For each band, hash its portion of each column to a hash table with k buckets.
  - Make k as large as possible.
- 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.


Buckets
Columns 2 and 6
are probably identical.
Columns 6 and 7 are
surely different.
Matrix M
r rows
b bands

Simplifying Assumption

- There are enough buckets that columns are unlikely to hash to the same bucket unless they are identical in a particular band.
- Hereafter, we assume that "same bucket" means "identical in that band."

Example: Effect of Bands

- Suppose 100,000 columns.
- Signatures of 100 integers.
- Therefore, signatures take 40Mb.
- Want all 80%-similar pairs.
- 5,000,000,000 pairs of signatures can take a while to compare.
- Choose $\underbrace{20}_{b}$ bands of $\underbrace{5}_{r}$ integers/band.

Suppose C_1, C_2 are 80% Similar

- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$.
- Probability C_1, C_2 are not similar in any of the 20 bands: $(1-0.328)^{20} = .00035$.
  - i.e., about 1/3000th of the 80%-similar column pairs are false negatives.

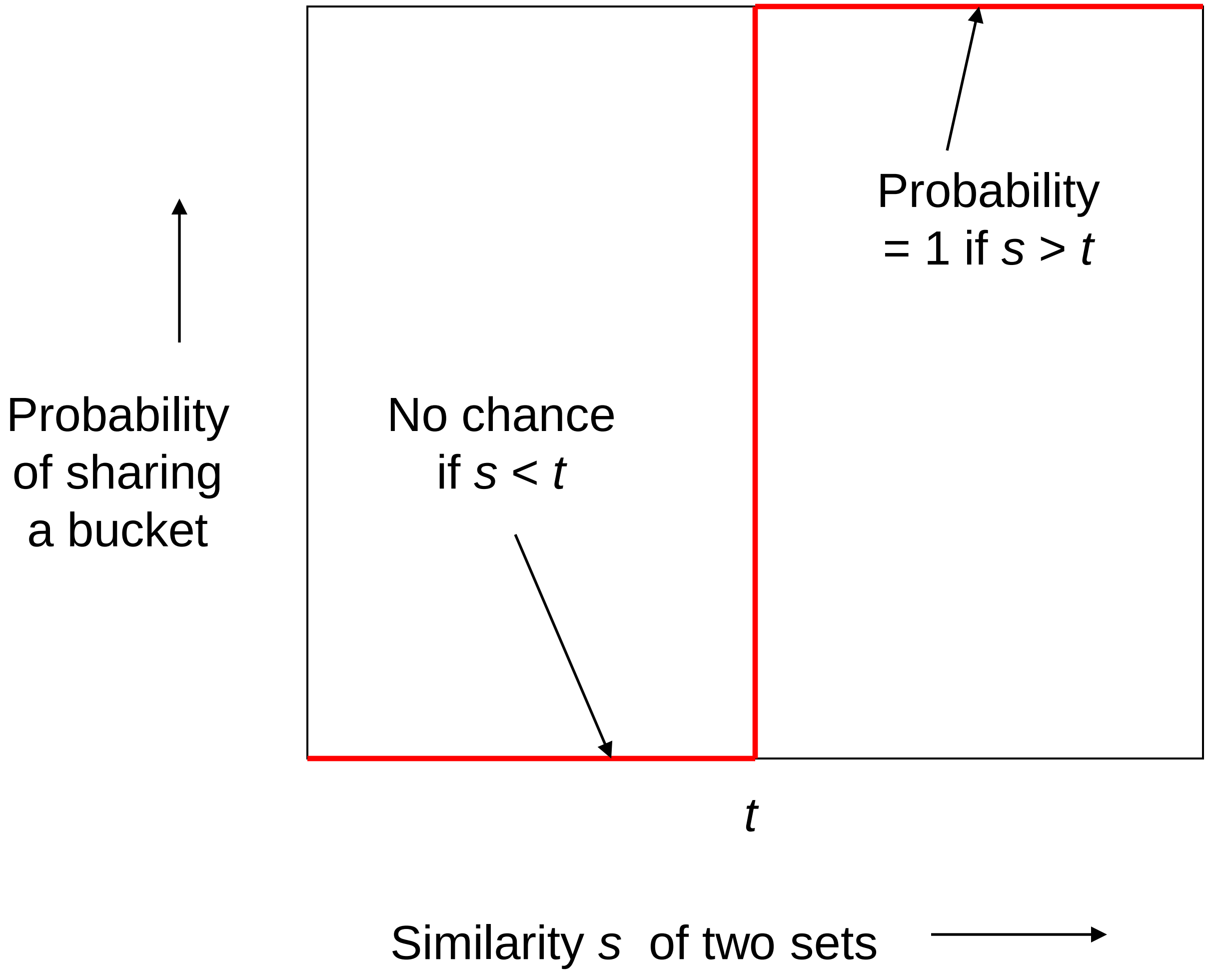
Suppose C_1, C_2 Only 40% Similar

- Probability C_1, C_2 identical in any one particular band: $(0.4)^5 = 0.01$.
- Probability C_1, C_2 identical in $\geq$ 1 of 20 bands: $\leq 20 * 0.01 = 0.2$.
- But false positives much lower for similarities << 40%.

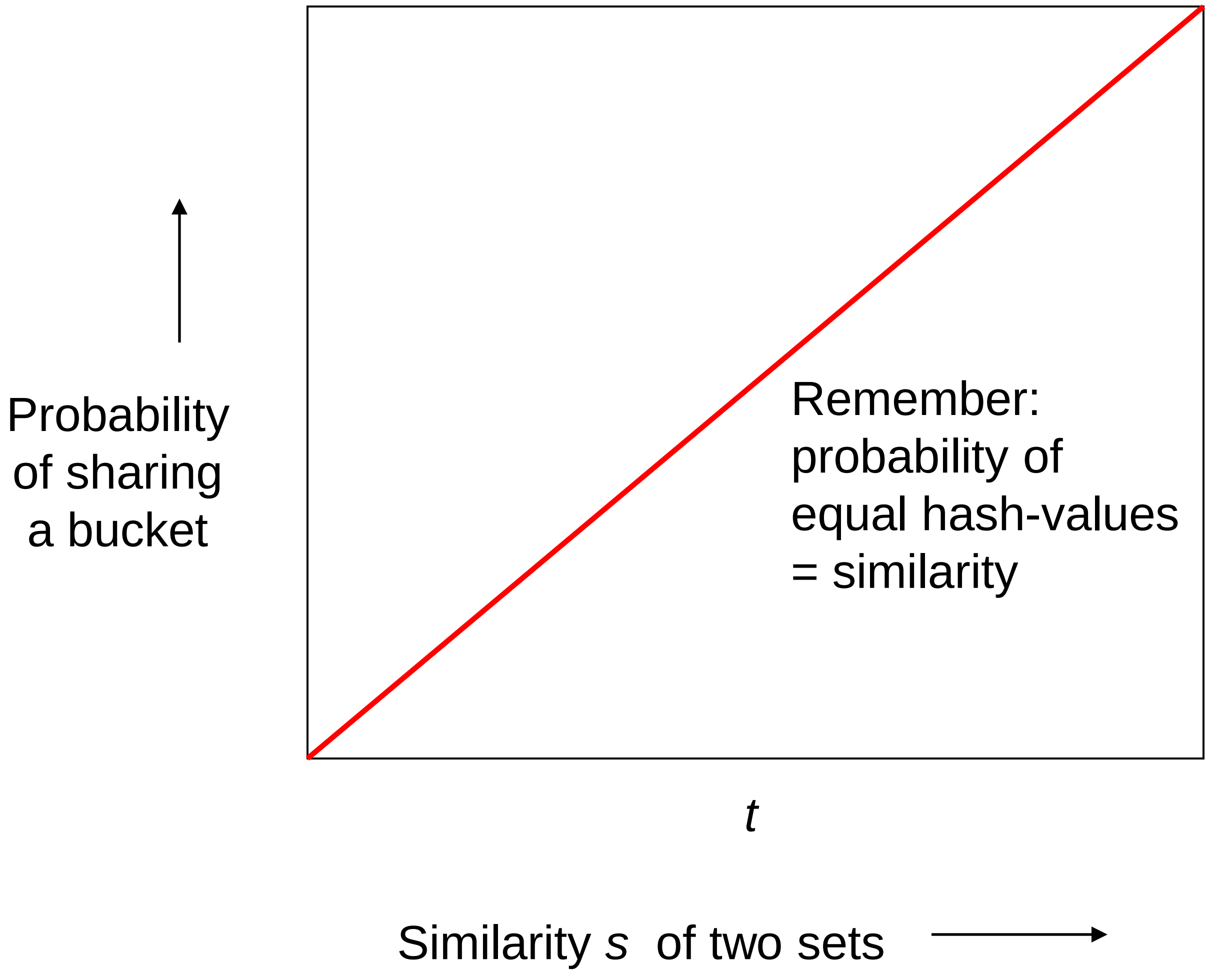
LSH Involves a Tradeoff

- Pick the number of minhashes, the number of bands, and the number of rows 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.

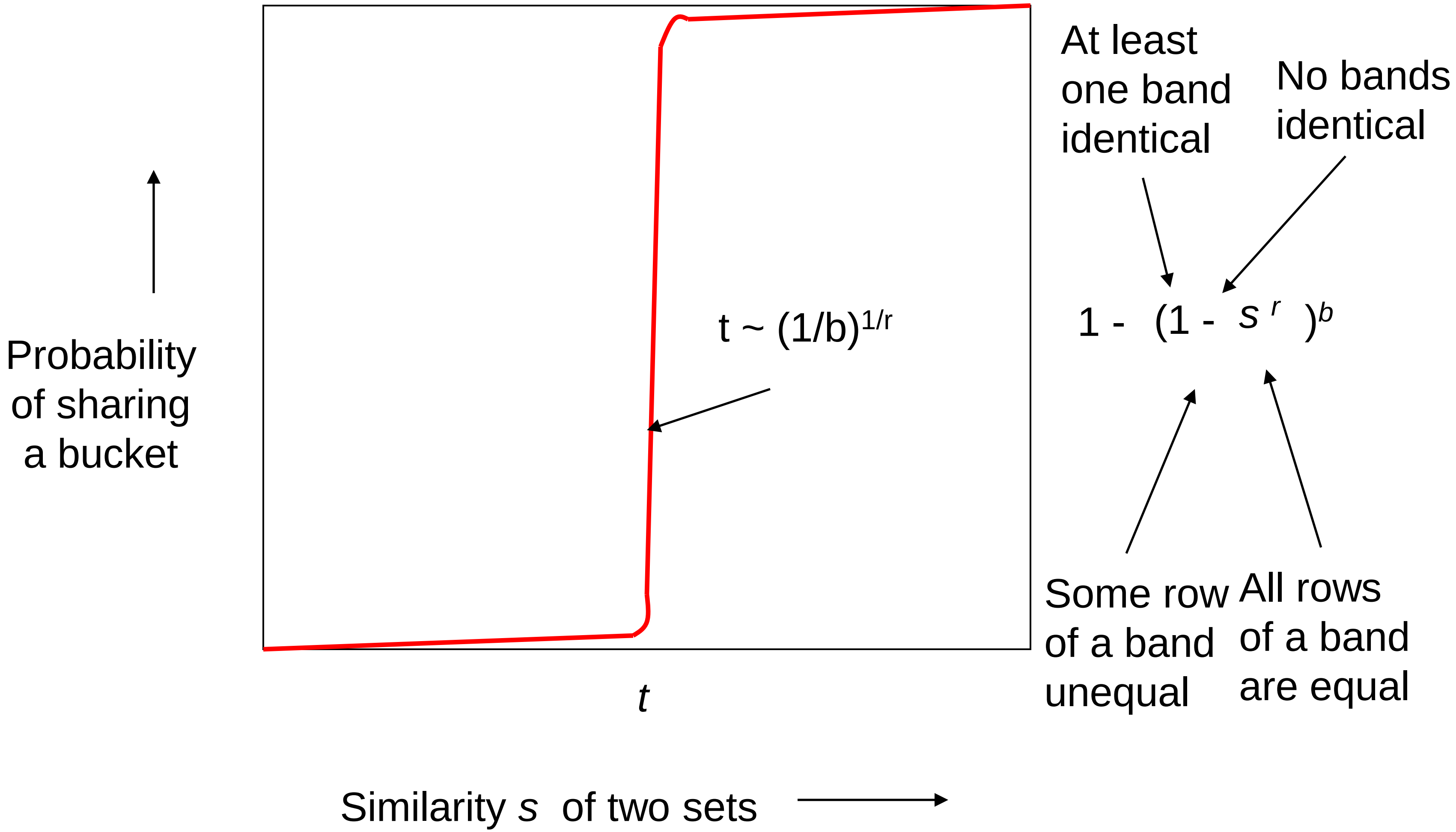
Analysis of LSH – What We Want



What One Band of One Row Gives You



What *b* Bands of *r* Rows Gives You



Example: $b = 20; r = 5$

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

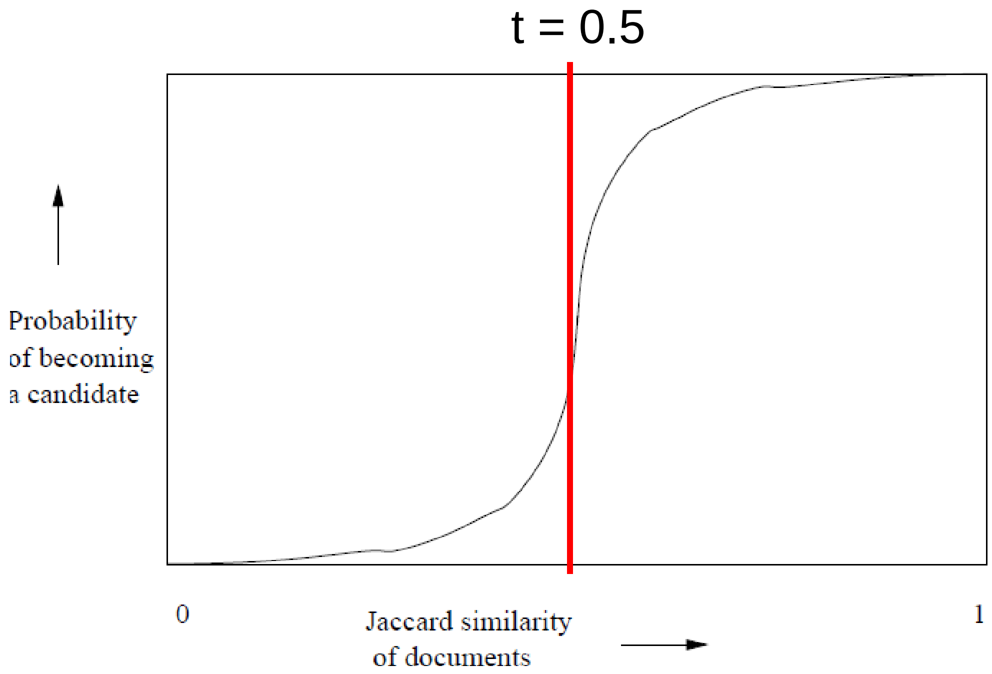


Figure 3.7: The S-curve

LSH Summary

- Tune to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.
- Check in main memory that candidate pairs really do have similar signatures.
- Optional: In another pass through data, check that the remaining candidate pairs really represent similar *sets* .

Locality-sensitive hashing (LSH)

- **Big Picture**: Construct hash functions **$h: R^d \rightarrow U$** such that for any pair of points **p,q**:
  - If **$D(p,q) \leq r$**, then **$Pr[h(p)=h(q)]$** is high
  - If **$D(p,q) \geq cr$**, then **$Pr[h(p)=h(q)]$** is small
- Then, we can solve the “approximate NN” problem by hashing

- LSH is a general framework; for a given distance function **D** we need to find the right **h**