Similarity Analysis and Distance
Min-Hashing
Locality Sensitive Hashing

DATA MINING

Thanks to:
Tan, Steinbach, and Kumar, “Introduction to Data Mining”
Rajaraman and Ullman, “Mining Massive Datasets”
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.
**Similarity**

- Numerical measure of how **alike** two data objects are.
  - A function that maps pairs of objects to real values
  - Higher when objects are more alike.
- Often falls in the range \([0,1]\), sometimes in \([-1,1]\)

**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**)
Consider the following documents

- apple releases new ipod
- apple releases new ipad
- new apple pie recipe

Which ones are more similar?

How would you quantify their similarity?
Similarity: Intersection

- Number of words in common

apple releases new ipod
apple releases new ipad
new apple pie recipe

- Sim(D,D) = 3, Sim(D,D) = Sim(D,D) = 2
- What about this document?

Vefa releases new book with apple pie recipes

- Sim(D,D) = Sim(D,D) = 3
The Jaccard similarity (Jaccard coefficient) of two sets $S_1$, $S_2$ is the size of their intersection divided by the size of their union.

- $JSim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$. 

3 in intersection.
8 in union.
Jaccard similarity
$= \frac{3}{8}$

- Extreme behavior:
  - Jsim($X,Y$) = 1, iff $X = Y$
  - Jsim($X,Y$) = 0 iff $X,Y$ have no elements in common
- JSim is symmetric
**Similarity: Intersection**

- Number of words in common
  - Apple releases new iPod
  - Apple releases new iPad
  - New apple pie recipe
  - Vefa re-releases new book with apple pie recipes

- \( \text{JSim}(D,D) = \frac{3}{5} \)
- \( \text{JSim}(D,D) = \text{JSim}(D,D) = \frac{2}{6} \)
- \( \text{JSim}(D,D) = \text{JSim}(D,D) = \frac{3}{9} \)
Documents (and sets in general) can also be represented as vectors.

<table>
<thead>
<tr>
<th>document</th>
<th>Apple</th>
<th>Microsoft</th>
<th>Obama</th>
<th>Election</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>10</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>30</td>
<td>60</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

How do we measure the similarity of two vectors?

How well are the two vectors aligned?
Documents $D_1$, $D_2$ are in the “same direction”
Document $D_3$ is orthogonal to these two
Cosine Similarity

- \( \text{Sim}(X,Y) = \cos(X,Y) \)
  - The cosine of the angle between \( X \) and \( Y \)

- If the vectors are aligned (correlated) angle is zero degrees and \( \cos(X,Y) = 1 \)
- If the vectors are orthogonal (no common coordinates) angle is 90 degrees and \( \cos(X,Y) = 0 \)

- Cosine is commonly used for comparing documents, where we assume that the vectors are normalized by the document length.
If $d_1$ and $d_2$ are two vectors, then
\[
\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \|d_2\|},
\]
where $\cdot$ indicates vector dot product and $\|d\|$ is the length of vector $d$.

Example:

\[
d_1 = 3205000200 \\
d_2 = 1000000102
\]

\[
d_1 \cdot d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5
\]

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

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

\[
\cos(d_1, d_2) = 0.3150
\]
Similarity between vectors

cos(D1, D2) = 1

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{document} & \text{Apple} & \text{Microsoft} & \text{Obama} & \text{Election} \\
\hline
D1 & 10 & 20 & 0 & 0 \\
\hline
D2 & 30 & 60 & 0 & 0 \\
\hline
D2 & 0 & 0 & 10 & 20 \\
\hline
\end{array}
\]
Distance

- Numerical measure of how **different** two data objects are
  - A function that maps pairs of objects to real values
    - Lower when objects are more alike
  - Minimum distance is 0, when comparing an object with itself.
- Upper limit varies
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.
Distances for real vectors

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

- \( L_p \) 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:
  \[
  L_1(x, y) = |x_1 - y_1| + \cdots + |x_d - y_d|
  \]

- \( L_\infty \) norm:
  \[
  L_\infty(x, y) = \max\{|x_1 - y_1|, \ldots, |x_d - y_d|\}
  \]
  - The limit of \( L_p \) as \( p \) goes to infinity.

\( L_p \) norms are known to be distance metrics.
Example of Distances

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

$x = (5,5)$

$L_1$-norm:
\[ \text{dist}(x, y) = 4 + 3 = 7 \]

$y = (9,8)$

$L_\infty$-norm:
\[ \text{dist}(x, y) = \max\{3,4\} = 4 \]
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 $x$
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.
Similarities into distances

- Jaccard distance:
  \[ JDist(X,Y) = 1 - JSim(X,Y) \]

- Jaccard Distance is a metric

- Cosine distance:
  \[ Dist(X,Y) = 1 - \cos(X,Y) \]

- Cosine distance is a metric
Why Jaccard Distance Is a Distance Metric

- $J\text{Dist}(x,x) = 0$
  - since $J\text{Sim}(x,x) = 1$
- $J\text{Dist}(x,y) = J\text{Dist}(y,x)$
  - by symmetry of intersection
- $J\text{Dist}(x,y) \geq 0$
  - since intersection of $X,Y$ cannot be bigger than the union.
- **Triangle inequality:**
  - Follows from the fact that $J\text{Sim}(X,Y)$ is the probability of randomly selected element from the union of $X$ and $Y$ to belong to the intersection
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 3\textsuperscript{rd} and 4\textsuperscript{th} 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 $y$.

- For binary vectors if follows from the fact that $L_1$ norm is a metric
How do we define similarity between strings?

- weird vs. wierd
- intelligent vs. unintelligent
- Athena vs. Athina

Important for recognizing and correcting typing errors and analyzing DNA sequences.
The edit distance of two strings is the number of inserts and deletes of characters needed to turn one into the other.

Example: \(x = \text{abcde} \); \(y = \text{bcduve}\).
- 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 \( y \). The minimum is no more than that
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

<table>
<thead>
<tr>
<th>document</th>
<th>Apple</th>
<th>Microsoft</th>
<th>Obama</th>
<th>Election</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>0.35</td>
<td>0.5</td>
<td>0.1</td>
<td>0.05</td>
</tr>
<tr>
<td>D2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>D2</td>
<td>0.05</td>
<td>0.05</td>
<td>0.6</td>
<td>0.3</td>
</tr>
</tbody>
</table>

- 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)

\[
JS(P, Q) = \frac{1}{2} D_{KL}(P\|Q) + \frac{1}{2} D_{KL}(Q\|P)
\]
Min-Hashing
LOCALITY SENSITIVE HASHING

Thanks to:
Rajaraman and Ullman, “Mining Massive Datasets”
Evimaria Terzi, slides for Data Mining Course.
Why is similarity important?

- We saw many definitions of similarity and distance
- How do we make use of similarity in practice?
- What issues do we have to deal with?
An important problem

- **Recommendation systems**
  - When a user buys an **item** (initially books) we want to recommend other items that the user may like
  - When a user rates a **movie**, we want to recommend movies that the user may like
  - When a user likes a **song**, we want to recommend other songs that they may like

- A big success of data mining
- Exploits the long tail
Content-based:

- Represent the items into a feature space and recommend items to customer C similar to previous items rated highly by C
  - Movie recommendations: recommend movies with same actor(s), director, genre, ...
  - Websites, blogs, news: recommend other sites with “similar” content
Plan of action

recommend

likes

match

Item profiles

build

Red
Circles
Triangles
User profile
Limitations of content-based approach

- Finding the appropriate features
  - e.g., images, movies, music
- Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests
- Recommendations for new users
  - How to build a profile?
Collaborative Filtering (user – user)

- Consider user c
- Find set D of other users whose ratings are “similar” to c’s ratings
- Estimate user’s ratings based on ratings of users in D
Collaborative filtering (item-item)
- For item s, find other similar items
- Estimate rating for item based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model
- In practice, it has been observed that item-item often works better than user-user
Pros and cons of collaborative filtering

- Works for any kind of item
  - No feature selection needed
- New user problem
- New item problem
- Sparsity of rating matrix
  - Cluster-based smoothing?
Another important problem

- Find 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
  - Find replicated news stories and cluster them under a single story.
  - Identify plagiarism

- What if we wanted exact duplicates?
Both the problems we described have a common component
  - We need a quick way to find highly similar items to a query item
  - OR, we need a method for finding all pairs of items that are highly similar.
  - Also known as the Nearest Neighbor problem, or the All Nearest Neighbors problem

We will examine it for the case of near-duplicate web documents.
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 (why?)
- 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** of 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 (LSH)**: focus on pairs of signatures likely to be similar.
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.
A \textit{k-shingle} (or \textit{k-gram}) for a document is a sequence of \textit{k} characters that appears in the document.

\textbf{Example:} document = \textit{abcab}. \textit{k}=2

- Set of 2-shingles = \{ab, bc, ca\}.
- \textbf{Option:} regard shingles as a \textit{bag}, and count \textit{ab} twice.

Represent a document by its set of \textit{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
  rose is a
  rose is a
  rose is a
  rose is a
  rose is a
  rose is a
  rose is a

  a rose is
  a rose is
  a rose is
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.

- Extreme case $k = 1$: all documents are the same
- $k = 5$ is OK for short documents; $k = 10$ is better for long documents.

**Alternative ways to define shingles:**
- Use words instead of characters
- Anchor on stop words (to avoid templates)
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.

From now on we will assume that shingles are integers

- Collisions are possible, but very rare
Fingerprinting

- **Hash shingles to 64-bit integers**

  **Set of Shingles**

  - 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

  **Hash function** (Rabin’s fingerprints)

  - 1111
  - 2222
  - 3333
  - 4444
  - 5555
  - 6666
  - 7777
  - 8888
  - 9999
  - 0000

  **Set of 64-bit integers**
Basic Data Model: Sets

- **Document**: A document is represented as a set of shingles (more accurately, hashes of shingles).

- **Document similarity**: Jaccard similarity of the sets of shingles.
  - Common shingles over the union of shingles
  - \[ \text{Sim} (C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \].

- 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 items.
Signatures

- **Problem**: shingle sets are too large to be kept in memory.
- **Key idea**: “hash” each set $S$ to a small signature $\text{Sig}(S)$, such that:
  1. $\text{Sig}(S)$ is small enough that we can fit a signature in main memory for each set.
  2. $\text{Sim}(S_1, S_2)$ is (almost) the same as the “similarity” of $\text{Sig}(S_1)$ and $\text{Sig}(S_2)$. (signature preserves similarity).

- **Warning**: This method can produce false negatives, and false positives (if an additional check is not made).
  - **False negatives**: Similar items deemed as non-similar
  - **False positives**: Non-similar items deemed as similar
Represent the data as a boolean matrix $M$
- **Rows** = the universe of all possible set elements
  - In our case, shingle fingerprints take values in $[0...2^{64}-1]$  
- **Columns** = the sets
  - In our case, documents, sets of shingle fingerprints
- $M(r,S) = 1$ in row $r$ and column $S$ if and only if $r$ is a member of $S$.

- Typical matrix is sparse.
  - We do not really materialize the matrix
Example

- Universe: $U = \{A, B, C, D, E, F, G\}$
- $X = \{A, B, F, G\}$
- $Y = \{A, E, F, G\}$
- $\text{Sim}(X, Y) = \frac{3}{5}$
Example

- Universe: \( U = \{A, B, C, D, E, F, G\} \)
- \( X = \{A, B, F, G\} \)
- \( Y = \{A, E, F, G\} \)
- \( \text{Sim}(X, Y) = \frac{3}{5} \)

At least one of the columns has value 1
Example

- **Universe**: $U = \{A, B, C, D, E, F, G\}$

- $X = \{A, B, F, G\}$

- $Y = \{A, E, F, G\}$

- $\text{Sim}(X, Y) = \frac{3}{5}$

Both columns have value 1
Minhashing

- Pick a random permutation of the rows (the universe U).
- Define “hash” function for set S
  - $h(S) = \text{the index of the first row (in the permuted order)}$ in which column $S$ has 1.
  - OR
  - $h(S) = \text{the index of the first element of } S \text{ in the permuted order.}$
- Use k (e.g., k = 100) independent random permutations to create a signature.
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S_1</th>
<th>S_2</th>
<th>S_3</th>
<th>S_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A transformation process is shown, and the resulting matrix is:

<table>
<thead>
<tr>
<th></th>
<th>S_1</th>
<th>S_2</th>
<th>S_3</th>
<th>S_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

With minhash signatures: 1 2 1 2
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S(_1)</th>
<th>S(_2)</th>
<th>S(_3)</th>
<th>S(_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Output matrix**

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Minhash values:** 3 1 3 1
### Example of minhash signatures

**Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Signature matrix**

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$h_2$</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$h_3$</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Sig($S$)** = vector of hash values
  - e.g., $\text{Sig}(S_2) = [2,1,1]$
- **Sig($S$,i)** = value of the i-th hash function for set $S$
  - E.g., $\text{Sig}(S_2,3) = 1$
Hash function Property

\[ \text{Pr}(h(S_1) = h(S_2)) = \text{Sim}(S_1, S_2) \]

- where the probability is over all choices of permutations.

- Why?
  - The first row where one of the two sets has value 1 belongs to the union.
    - Recall that union contains rows with at least one 1.
  - We have equality if both sets have value 1, and this row belongs to the intersection.
The similarity of signatures is the fraction of the hash functions in which they agree.

With multiple signatures we get a good approximation.
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?
- 😞
Approximating row permutations: pick $k=100$ hash functions $(h_1, \ldots, h_k)$

for each row $r$

for each hash function $h_i$

compute $h_i(r)$

for each column $S$ that has 1 in row $r$

if $h_i(r)$ is a smaller value than $\text{Sig}(S,i)$ then

$\text{Sig}(S,i) = h_i(r)$;

$\text{Sig}(S,i)$ will become the smallest value of $h_i(r)$ among all rows (shingles) for which column $S$ has value 1 (shingle belongs in $S$); i.e., $h_i(r)$ gives the min index for the $i$-th permutation
**Algorithm – All sets, k hash functions**

Pick $k=100$ hash functions $(h_1, \ldots, h_k)$

for each row $r$

for each hash function $h_i$

compute $h_i(r)$

for each column $S$ that has 1 in row $r$

if $h_i(r)$ is a smaller value than $\text{Sig}(S, i)$ then

$\text{Sig}(S, i) = h_i(r)$;
### Example

<table>
<thead>
<tr>
<th>x</th>
<th>Row</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( h(x) )</th>
<th>( g(x) )</th>
<th>( \text{Sig}_1 )</th>
<th>( \text{Sig}_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

\( h(x) = x+1 \text{ mod } 5 \)
\( g(x) = 2x+3 \text{ mod } 5 \)

<table>
<thead>
<tr>
<th>Row</th>
<th>( \text{Sig}_1 )</th>
<th>( \text{Sig}_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>E 0 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A 1 0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B 0 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C 1 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>D 1 0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>E 0 1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A 1 0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>B 0 1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>C 1 1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>D 1 0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>E 0 1</td>
<td></td>
</tr>
</tbody>
</table>

\( h(0) = 1 \)
\( g(0) = 3 \)
\( h(1) = 2 \)
\( g(1) = 0 \)
\( h(2) = 3 \)
\( g(2) = 2 \)
\( h(3) = 4 \)
\( g(3) = 4 \)
\( h(4) = 0 \)
\( g(4) = 1 \)
Finding similar pairs

- Problem: Find all pairs of documents with similarity at least $t = 0.8$
- 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 $5 \times 10^{11}$ column-comparisons.
- At 1 microsecond/comparison: 6 days.
What we want: 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.

A simple idea: $X$ and $Y$ are a candidate pair if they have the same min-hash signature.
- Easy to test by hashing the signatures.
- Similar sets are more likely to have the same signature.
- Likely to produce many false negatives.
  - Requiring full match of signature is strict, some similar sets will be lost.

Improvement: Compute multiple signatures; candidate pairs should have at least one common signature.
- Reduce the probability for false negatives.
Signature matrix reminder

\[ \text{Prob}(\text{Sig}(S, i) == \text{Sig}(S', i)) = \text{sim}(S, S') \]

Matrix \( M \)

Signature for set \( S' \)

Signature for set \( S \):

\( \text{Sig}(S, i) \):

\( \text{Sig}(S', i) \):

\( n \) hash functions

Hash function \( i \)
Partition into Bands – (1)

- Divide the signature matrix $\text{Sig}$ into $b$ bands of $r$ rows.
  - Each band is a mini-signature with $r$ hash functions.
Partitioning into bands

\[ n = b^r \] hash functions

\( b \) bands

\( b \) mini-signatures

Matrix \( \text{Sig} \)

One signature

\( r \) rows per band
Partition into Bands – (2)

- Divide the signature matrix $\text{Sig}$ into $b$ bands of $r$ rows.
  - Each band is a mini-signature with $r$ hash functions.
- For each band, hash the mini-signature to a hash table with $k$ buckets.
  - Make $k$ as large as possible so that mini-signatures that hash to the same bucket are almost certainly identical.
Columns 2 and 6 are (almost certainly) identical.

Columns 6 and 7 are surely different.
Divide the signature matrix $\text{Sig}$ into $b$ bands of $r$ rows.
- Each band is a mini-signature with $r$ hash functions.
- For each band, hash the mini-signature to a hash table with $k$ buckets.
  - Make $k$ as large as possible so that mini-signatures that hash to the same bucket are almost certainly identical.
- Candidate column pairs are those that hash to the same bucket for at least 1 band.
- Tune $b$ and $r$ to catch most similar pairs, but few non-similar pairs.
Analysis of LSH – What We Want

Probability of sharing a bucket

No chance if \( s < t \)

Probability = 1 if \( s > t \)

Similarity \( s \) of two sets

t
Similarity of two sets

Probability of sharing a bucket

Remember: probability of equal hash-values = similarity

Single hash signature

$\text{Prob}(\text{Sig}(S,i) == \text{Sig}(S',i)) = \text{sim}(S, S')$

Similarity $s$ of two sets
What \( b \) Bands of \( r \) Rows Gives You

- Probability of sharing a bucket
  - \( t \sim (1/b)^{1/r} \)

- Similarity \( s \) of two sets
  - \( 1 - (1 - s^r)^b \)

- At least one band identical
- No bands identical
- Some row of a band unequal
- All rows of a band are equal
Example: $b = 20, r = 5$

<table>
<thead>
<tr>
<th>$s$</th>
<th>$1 - (1 - s^r)^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.006</td>
</tr>
<tr>
<td>.3</td>
<td>.047</td>
</tr>
<tr>
<td>.4</td>
<td>.186</td>
</tr>
<tr>
<td>.5</td>
<td>.470</td>
</tr>
<tr>
<td>.6</td>
<td>.802</td>
</tr>
<tr>
<td>.7</td>
<td>.975</td>
</tr>
<tr>
<td>.8</td>
<td>.9996</td>
</tr>
</tbody>
</table>

Figure 3.7: The S-curve
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 \( h: \mathbb{R}^d \rightarrow U \) such that for any pair of points \( p, q \), for distance function \( D \) we have:
  - If \( D(p, q) \leq r \), then \( \Pr[h(p) = h(q)] \geq \alpha \) is high
  - If \( D(p, q) \geq cr \), then \( \Pr[h(p) = h(q)] \leq \beta \) is small
- Then, we can find close pairs by hashing

- LSH is a general framework: for a given distance function \( D \) we need to find the right \( h \)
  - \( h \) is \((r, cr, \alpha, \beta)\)-sensitive
- For cosine distance, there is a technique analogous to minhashing for generating a family of\( (d_1, d_2, (1-d_1/180), (1-d_2/180)) \)-sensitive family for any \( d_1 \) and \( d_2 \).
- Called *random hyperplanes*. 
Random Hyperplanes

- Pick a random vector $v$, which determines a hash function $h_v$ with two buckets.
- $h_v(x) = +1$ if $v \cdot x > 0; = -1$ if $v \cdot x < 0$.
- LS-family $H$ = set of all functions derived from any vector.
- $\text{Prob}[h(x) = h(y)] = 1 - \left(\text{angle between } x \text{ and } y \text{ divided by } 180\right)$.
Signatures for Cosine Distance

- Pick some number of vectors, and hash your data for each vector.
- The result is a signature (sketch) of +1’s and –1’s that can be used for LSH like the minhash signatures for Jaccard distance.
Simplification

- We need not pick from among all possible vectors \( \mathbf{v} \) to form a component of a sketch.
- It suffices to consider only vectors \( \mathbf{v} \) consisting of +1 and −1 components.
Signatures for Cosine Distance

- Pick some number of vectors, and hash your data for each vector.
- The result is a signature (sketch) of +1’s and –1’s that can be used for LSH like the minhash signatures for Jaccard distance.
We need not pick from among all possible vectors $v$ to form a component of a sketch. It suffices to consider only vectors $v$ consisting of +1 and –1 components.
Course Recap

- Coupon Collector Problem
- Reservoir Sampling
- Hadoop = MapReduce + HDFS
- Information Retrieval = TD-IDF
- Association Analysis = Frequent Itemsets + Frequent Rules (sorted by Lift)
- Similarity Analysis = Shingling + Min-Hash + Locality Sensitive Hashing
- Min-Hashing = compact representation of sets that maintains similarity.
- LSH = Find similar candidates using hashing.