Announcements

Deadlines today, 11:59 PM:
- Colab 0, Colab 1
- You can submit many times and will get immediate feedback

Deadlines next Thu, 11:59 PM:
- HW1, Colab 2

How to find teammates for project?
- Ed Discussion Board
- Make sure you have a good dataset accessible
Recap: Finding similar documents

- **Task:** Given a large number ($N$ in the millions or billions) of documents, find “near duplicates”

- **Problem:**
  - Too many documents to compare all pairs

- **Solution:** Hash documents so that similar documents hash into the same bucket
  - Documents in the same bucket are then **candidate pairs** whose similarity is then evaluated
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.
Recap: Shingles

- A \textit{k-shingle} (or \textit{k-gram}) is a sequence of \textit{k} tokens that appears in the document
  - Example: \( k=2; \ D_1 = abcab \)
  - Set of 2-shingles: \( C_1 = S(D_1) = \{ab, bc, ca\} \)
- Represent a doc by a set of hash values of its \textit{k}-shingles
- A natural \textbf{similarity measure} is then the \textbf{Jaccard similarity}:
  \[
  sim(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}
  \]
  - Similarity of two documents is the Jaccard similarity of their shingles
Recap: Minhashing

- **Min-Hashing**: Convert large sets into short signatures, while preserving similarity: $\Pr[h(C_1) = h(C_2)] = \text{sim}(D_1, D_2)$

<table>
<thead>
<tr>
<th>Permutation $\pi$</th>
<th>Input matrix (Shingles x Documents)</th>
<th>Signature matrix $M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 4 3 1 6 5 4 5 7 1 7 6 3 2 1 6 6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\text{Col/Col}$: 

<table>
<thead>
<tr>
<th>1-3</th>
<th>2-4</th>
<th>1-2</th>
<th>3-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>0.75</td>
<td>0.67</td>
<td>1.00</td>
</tr>
</tbody>
</table>

$\text{Sig/Sig}$: 0 0 0 0
Recap: LSH

- **Hash columns of the signature matrix $M$:** Similar columns likely hash to same bucket
  - Divide matrix $M$ into $b$ bands of $r$ rows ($M=b\cdot r$)
  - *Candidate* column pairs are those that hash to the same bucket for $\geq 1$ band
Design a **locality sensitive** hash function (for a given distance metric)

---

**Signatures:** short integer signatures that reflect point similarity

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

---

Apply the “**Bands**” technique
The S-Curve

- The S-curve is where the “magic” happens

The S-curve is where the “magic” happens

Remember: Probability of equal hash-values = similarity

Pr[h_π(C_1) = h_π(C_2)] = sim(D_1, D_2)

Threshold s

Probability=1 if t>s

No chance if t<s

Similarity t of two sets

This is what we want!

This is what 1 hash-code gives you

How to get a step-function?

By choosing r and b!
How Do We Make the S-curve?

- **Remember:** \( b \) bands, \( r \) rows/band
- Let \( \text{sim}(C_1, C_2) = s \)

What’s the prob. that at least 1 band is equal?

- Pick some band (\( r \) rows)
  - Prob. that elements in a single row of columns \( C_1 \) and \( C_2 \) are equal = \( s \)
  - Prob. that all rows in a band are equal = \( s^r \)
  - Prob. that some row in a band is not equal = \( 1 - s^r \)
  - Prob. that all bands are not equal = \( (1 - s^r)^b \)
  - Prob. that at least 1 band is equal = \( 1 - (1 - s^r)^b \)

\[ P(C_1, C_2 \text{ is a candidate pair}) = 1 - (1 - s^r)^b \]
Picking $r$ and $b$: The S-curve

- Picking $r$ and $b$ to get the best S-curve
  - 50 hash-functions ($r=5$, $b=10$)
Given a fixed threshold $s$.

We want choose $r$ and $b$ such that the $P(\text{Candidate pair})$ has a “step” right around $s$.

$$\text{prob} = 1 - (1 - t^r)^b$$
Min-Hashing

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

Locality-sensitive Hashing

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

Theory of LSH

general hashing

locality-sensitive hashing
Theory of LSH

- We have used LSH to find similar documents
  - More generally, we found similar columns in large sparse matrices with high Jaccard similarity

- Can we use LSH for other distance measures?
  - e.g., Euclidean distances, Cosine distance
  - Let’s generalize what we’ve learned!
Distance Metric

- **d()** is a **distance metric** if it is a function from pairs of points \( x, y \) to real numbers such that:
  - \( d(x, y) \geq 0 \)
  - \( d(x, y) = 0 \) if and only if \( x = y \)
  - \( d(x, y) = d(y, x) \)
  - \( d(x, y) \leq d(x, z) + d(z, y) \) (triangle inequality)

- **Jaccard distance** for sets = 1 - Jaccard similarity
- **Cosine distance** for vectors = angle between the vectors
- **Euclidean distances**:
  - \( L_2 \) norm: \( d(x, y) = \) square root of the sum of the squares of the differences between \( x \) and \( y \) in each dimension
    - The most common notion of “distance”
  - \( L_1 \) norm: sum of absolute value of the differences in each dimension
    - **Manhattan distance** = distance if you travel along coordinates only
Families of Hash Functions

- For Min-Hashing signatures, we got a Min-Hash function for each permutation of rows
- A “hash function” is any function that allows us to say whether two elements are “equal”
  - Shorthand: $h(x) = h(y)$ means “$h$ says $x$ and $y$ are equal”
- A *family* of hash functions is any set of hash functions from which we can *pick one at random efficiently*
  - Example: The set of Min-Hash functions generated from permutations of rows
Locality-Sensitive (LS) Families

- Suppose we have a space \( S \) of points with a **distance** metric \( d(x,y) \)

- 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 the probability over all \( h \in H \), that \( h(x) = h(y) \) is at least \( p_1 \)
  2. If \( d(x, y) \geq d_2 \), then the probability over all \( h \in H \), that \( h(x) = h(y) \) is at most \( p_2 \)

**Critical assumption**

With a LS Family we can do LSH!
A \((d_1, d_2, p_1, p_2)\)-sensitive function

For all \(h \in H\),
\[
\begin{align*}
P[h(x) = h(y_1)] & \geq p_1 \\
P[h(x) = h(y_2)] & \leq p_2
\end{align*}
\]
A $(d_1, d_2, p_1, p_2)$-sensitive function

Small distance, high probability

Large distance, low probability of hashing to the same value

Notice distance on x-axis, not similarity, hence the S-curve is mirrored!
A \((d_1, d_2, p_1, p_2)\)-sensitive function

Distance \(d(x, y)\)

Small distance, high probability

Large distance, low probability of hashing to the same value

Notice distance on x-axis, not similarity, hence the S-curve is mirrored!
A \((d_1, d_2, p_1, p_2)\)-sensitive function

Small distance, high probability

Distance threshold \(t\)

Large distance, low probability of hashing to the same value

Notice distance on x-axis, not similarity, hence the S-curve is mirrored!
Example of LS Family: Min-Hash

- Let:
  - \( S = \) space of all sets,
  - \( d = \) Jaccard distance,
  - \( H \) is family of Min-Hash functions for all permutations of rows

- Then for any hash function \( h \in H \):
  \[
  \Pr[h(x) = h(y)] = 1 - d(x, y)
  \]

- Simply restates theorem about Min-Hashing in terms of distances rather than similarities
Example: LS Family – (2)

- **Claim:** Min-hash $H$ is a $(1/3, 2/3, 2/3, 1/3)$-sensitive family for $S$ and $d$.

  - If distance $\leq 1/3$ (so similarity $\geq 2/3$)

- Then probability that Min-Hash values agree is $\geq 2/3$

- For Jaccard similarity, Min-Hashing gives a $(d_1, d_2, (1-d_1), (1-d_2))$-sensitive family for any $d_1 < d_2$
Amplifying a LS-Family

- Can we reproduce the “S-curve” effect we saw before for any LS family?

- The “bands” technique we learned for signature matrices carries over to this more general setting.

- Can do LSH with any $(d_1, d_2, p_1, p_2)$-sensitive family!

- Two constructions:
  - **AND** construction like “rows in a band”
  - **OR** construction like “many bands”
Amplifying Hash Functions: AND and OR
AND of Hash Functions

- Given family $H$, construct family $H'$ consisting of $r$ independent functions from $H$
- For $h = [h_1,...,h_r]$ in $H'$, we say $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all $1 \leq i \leq r$
  - Note this corresponds to creating a band of size $r$

- Theorem: If $H$ is $(d_1, d_2, p_1, p_2)$-sensitive, then $H'$ is $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive
- Proof: Use the fact that $h_i$'s are independent

Also lowers probability for small distances (Bad)

Lowers probability for large distances (Good)
Subtlety Regarding Independence

- Independence of hash functions (HFs) really means that the prob. of two HFs saying “yes” is the product of each saying “yes”
  - But two particular hash functions could be highly correlated
    - For example, in Min-Hash if their permutations agree in the first one million entries
  - However, the probabilities in definition of a LSH-family are over all possible members of $H, H'$ (i.e., average case and not the worst case)
Given family $H$, construct family $H'$ consisting of $b$ independent functions from $H$

For $h = [h_1, ..., h_b]$ in $H'$, $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for at least 1 $i$

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

**Proof:** Use the fact that $h_i$'s are independent

Raises probability for small distances (Good)  
Raises probability for large distances (Bad)
**Effect of AND and OR Constructions**

- **AND** makes all probs. **shrink**, but by choosing $r$ correctly, we can make the lower prob. approach 0 while the higher does not.

- **OR** makes all probs. **grow**, but by choosing $b$ correctly, we can make the higher prob. approach 1 while the lower does not.

![Graphs showing the effect of AND and OR on probability sharing a bucket and similarity of a pair of items](image-url)
Combining AND and OR Constructions

- By choosing $b$ and $r$ correctly, we can make the lower probability approach 0 while the higher approaches 1.

- As for the signature matrix, we can use the AND construction followed by the OR construction:
  - Or vice-versa
  - Or any sequence of AND’s and OR’s alternating.
Composing Constructions

- $r$-way \textbf{AND} followed by $b$-way \textbf{OR} construction
  - Exactly what we did with Min-Hashing
    - \textbf{AND}: If bands match in all $r$ values hash to same bucket
    - \textbf{OR}: Cols that have $\geq 1$ common bucket $\rightarrow$ Candidate

- Take points $x$ and $y$ s.t. $Pr[h(x) = h(y)] = s$
  - $H$ will make $(x,y)$ a candidate pair with prob. $s$
  - Construction makes $(x,y)$ a candidate pair with probability $1-(1-s^r)^b$ The S-Curve!

- \textbf{Example}: Take $H$ and construct $H'$ by the \textbf{AND} construction with $r = 4$. Then, from $H'$, construct $H''$ by the \textbf{OR} construction with $b = 4$
Table for Function $1-(1-s^4)^4$

<table>
<thead>
<tr>
<th>s</th>
<th>$p=1-(1-s^4)^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.0064</td>
</tr>
<tr>
<td>.3</td>
<td>.0320</td>
</tr>
<tr>
<td>.4</td>
<td>.0985</td>
</tr>
<tr>
<td>.5</td>
<td>.2275</td>
</tr>
<tr>
<td>.6</td>
<td>.4260</td>
</tr>
<tr>
<td>.7</td>
<td>.6666</td>
</tr>
<tr>
<td>.8</td>
<td>.8785</td>
</tr>
<tr>
<td>.9</td>
<td>.9860</td>
</tr>
</tbody>
</table>

$r = 4$, $b = 4$ transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.8785,.0064)-sensitive family.
How to choose $r$ and $b$
Picking \( r \) and \( b \): The S-curve

- **Picking \( r \) and \( b \) to get desired performance**
  - 50 hash-functions \((r = 5, \ b = 10)\)

---

**Yellow area X:** False Negative rate
These are pairs with \( \text{sim} > s \) but the \( X \) fraction won’t share a band and then will never become candidates. This means we will never consider these pairs for (slow/exact) similarity calculation!

**Blue area Y:** False Positive rate
These are pairs with \( \text{sim} < s \) but we will consider them as candidates. This is not too bad, we will consider them for (slow/exact) similarity computation and discard them.
Picking $r$ and $b$: The S-curve

- Picking $r$ and $b$ to get desired performance
  - 50 hash-functions ($r \times b = 50$)

![Graph showing probability of candidate pairs vs. similarity](image)

Thresholds:
- $r=2, b=25$
- $r=5, b=10$
- $r=10, b=5$
OR-AND Composition

- Apply a $b$-way OR construction followed by an $r$-way AND construction
- Transforms similarity $s$ (probability $p$) into $(1-(1-s)^b)^r$
  - The same S-curve, mirrored horizontally and vertically
- Example: Take $H$ and construct $H'$ by the OR construction with $b = 4$. Then, from $H'$, construct $H''$ by the AND construction with $r = 4$
### Table for Function $(1-(1-s)^4)^4$

<table>
<thead>
<tr>
<th>$s$</th>
<th>$p=(1-(1-s)^4)^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1</td>
<td>.0140</td>
</tr>
<tr>
<td>.2</td>
<td>.1215</td>
</tr>
<tr>
<td>.3</td>
<td>.3334</td>
</tr>
<tr>
<td>.4</td>
<td>.5740</td>
</tr>
<tr>
<td>.5</td>
<td>.7725</td>
</tr>
<tr>
<td>.6</td>
<td>.9015</td>
</tr>
<tr>
<td>.7</td>
<td>.9680</td>
</tr>
<tr>
<td>.8</td>
<td>.9936</td>
</tr>
</tbody>
</table>

The example transforms a $(.2,.8,.8,.2)$-sensitive family into a $(.2,.8,.9936,.1215)$-sensitive family.
Cascading Constructions

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

- **Transforms a** \((0.2, 0.8, 0.8, 0.2)\)-sensitive family into a \((0.2, 0.8, 0.9999996, 0.0008715)\)-sensitive family

- **Note this family uses** 256 \((=4*4*4*4)\) of the original hash functions
Summary

- Pick any two distances $d_1 < d_2$

- Start with a $(d_1, d_2, (1-d_1), (1-d_2))$-sensitive family

- Apply constructions to amplify $(d_1, d_2, p_1, p_2)$-sensitive family, where $p_1$ is almost 1 and $p_2$ is almost 0

- The closer to 0 and 1 we want to get, the more hash functions must be used!
LSH for other distance metrics
**LSH for other Distance Metrics**

- **LSH methods for other distance metrics:**
  - **Cosine distance:** Random hyperplanes
  - **Euclidean distance:** Project on lines

Diagram:

- Points → Hash func. → Signatures: short integer signatures that reflect their similarity → Locality-sensitive Hashing → Candidate pairs: those pairs of signatures that we need to test for similarity

Design a \((d_1, d_2, p_1, p_2)\)-sensitive family of hash functions (for that particular distance metric)

Depends on the distance function used

Amplify the family using \(AND\) and \(OR\) constructions

---

Summary of what we will learn

MinHash

Documents

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

MinHash

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

“Bands” technique

Candidate pairs

Random Hyperplanes

Data points

-1 +1 -1 -1
+1 +1 +1 -1
-1 -1 -1 -1

“Bands” technique

Candidate pairs
Cosine Distance

- **Cosine distance** = angle between vectors from the origin to the points in question
  \[ d(A, B) = \theta = \arccos\left(\frac{A \cdot B}{\|A\| \cdot \|B\|}\right) \]
  - Has range \([0, \pi]\) (equivalently \([0,180^\circ]\))
  - Can divide \(\theta\) by \(\pi\) to have distance in range \([0,1]\)
- Cosine similarity = 1 - \(d(A,B)\)
  - But often defined as **cosine sim**: \( \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \)
    - Has range -1…1 for general vectors
    - Range 0..1 for non-negative vectors (angles up to 90°)

![Diagram of Cosine Distance and Similarity](image-url)
LSH for Cosine Distance

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

- **Random Hyperplanes** method is a 
  \((d_1, d_2, (1-d_1/\pi), (1-d_2/\pi))\)-**sensitive** family for any \(d_1\) and \(d_2\)

- **Reminder**: \((d_1, d_2, p_1, p_2)\)-**sensitive**
  1. If \(d(x,y) \leq d_1\), then prob. that \(h(x) = h(y)\) is at least \(p_1\)
  2. If \(d(x,y) \geq d_2\), then prob. that \(h(x) = h(y)\) is at most \(p_2\)
Random Hyperplanes

- Each vector \( \mathbf{v} \) determines a hash function \( h_v \) with two buckets

\[
h_v(x) = +1 \text{ if } \mathbf{v} \cdot \mathbf{x} \geq 0; \quad = -1 \text{ if } \mathbf{v} \cdot \mathbf{x} < 0
\]

- LS-family \( H \) = set of all functions derived from any vector

**Claim:** For points \( \mathbf{x} \) and \( \mathbf{y} \),

\[
\Pr[h(x) = h(y)] = 1 - \frac{d(x,y)}{\pi}
\]
Proof of Claim

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

\[ \theta \]
Proof of Claim

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

Hyperplane normal to $v$. Here $h(x) = h(y)$.
Proof of Claim

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

Hyperplane normal to $v'$. Here $h(x) \neq h(y)$.
Proof of Claim

So: \( \text{Prob[Red case]} = \frac{\theta}{\pi} \)

So: \( P[h(x)=h(y)] = 1 - \frac{\theta}{\pi} = 1 - \frac{d(x,y)}{\pi} \)
Signatures for Cosine Distance

- Pick some number of random vectors, and hash your data for each vector
- The result is a signature (sketch) of \(+1\)'s and \(-1\)'s for each data point
- Can be used for LSH like we used the Min-Hash signatures for Jaccard distance
- Amplify using \textbf{AND/OR} constructions
How to pick random vectors?

- Expensive to pick a random vector in $M$ dimensions for large $M$
  - Would have to generate $M$ random numbers

- A more efficient approach
  - It suffices to consider only vectors $v$ consisting of +1 and −1 components
    - **Why?** Assuming data is random, then vectors of +/-1 cover the entire space evenly (and does not bias in any way)
LSH for Euclidean Distance

- **Idea:** Hash functions correspond to lines
- Partition the line into buckets of size $a$
- Hash each point to the bucket containing its projection onto the line
  - An element of the “Signature” is a bucket id for that given projection line
- Nearby points are always close; distant points are rarely in same bucket
“Lucky” case:
- Points that are close hash in the same bucket
- Distant points end up in different buckets
Projection of Points

- **“Lucky” case:**
  - Points that are close hash in the same bucket
  - Distant points end up in different buckets

- **Two “unlucky” cases:**
  - **Top:** unlucky quantization
  - **Bottom:** unlucky projection
Multiple Projections
Projection of Points

If \(d << a\), then the chance the points are in the same bucket is at least \(1 - \frac{d}{a}\).

Points at distance \(d\)

Randomly chosen line

Bucket width \(a\)

Exactly \(1 - \frac{d}{a}\) when the randomly chosen line is parallel to the line from \(x\) to \(y\).
Projection of Points

If \( d \gg a \), \( \theta \) must be close to \( 90^\circ \) for there to be any chance points go to the same bucket. Then: \( d \cos \theta \leq a \)
A LS-Family for Euclidean Distance

- If points are distance $d \leq a/2$, prob. they are in same bucket $\geq 1 - d/a = 1/2$
- If points are distance $d > 2a$ apart, then they can be in the same bucket only if $d \cos \theta \leq a$
  - $\cos \theta \leq 1/2$
  - $60 \leq \theta \leq 90$, i.e., at most $1/3$ probability

- Yields a $(a/2, 2a, 1/2, 1/3)$-sensitive family of hash functions for any $a$
- **Amplify using AND-OR cascades**
Data → Hash function → Signatures: short integer signatures that reflect their similarity → Locality-sensitive Hashing

Design a \((d_1, d_2, p_1, p_2)\)-sensitive family of hash functions (for that particular distance metric)

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

Amplify the family using \(\text{AND}\) and \(\text{OR}\) constructions

Documents

| 0 1 0 0 |
| 1 1 1 0 |
| 0 0 0 1 |
| 0 1 0 1 |
| 0 0 1 0 |
| 1 0 0 1 |

MinHash

| 1 | 5 | 1 | 5 |
| 2 | 3 | 1 | 3 |
| 6 | 4 | 6 | 4 |

“Bands” technique

Candidate pairs

Data points

| 0 1 0 0 |
| 1 1 1 0 |
| 0 0 0 1 |
| 0 1 0 1 |
| 0 0 1 0 |

Random Hyperplanes

| -1 | +1 | -1 | -1 |
| +1 | +1 | +1 | -1 |
| -1 | -1 | -1 | -1 |

“Bands” technique

Candidate pairs
Two Important Points

- Property $P(h(C_1)=h(C_2))=\text{sim}(C_1,C_2)$ of hash function $h$ is the essential part of LSH, without which we can’t do anything.

- LS-hash functions transform data to signatures so that the bands technique (AND, OR constructions) can then be applied.
Please give us feedback 😊