Locality Sensitive Hashing (LSH)

Huan Sun, CSE@The Ohio State University
Finding Similar Items

MMDS Secs. 3.2-3.4.
Slides adapted from: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets,
http://www.mmds.org

Finding Similar Items

Slides also adapted from Prof. Srinivasan Parthasarathy @OSU
Two Essential Steps for Similar Docs

1. **Shingling:** Convert documents to sets

2. **Min-Hashing:** Convert large sets to short signatures, while preserving similarity

Host of follow up applications
- e.g. Similarity Search
- Data Placement
- Clustering etc.

The Big Picture

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

SHINGLING

Step 1: **Shingling**: Convert documents to sets

The set of strings of length \( k \) that appear in the document
Define: Shingles

- A \textit{k-shingle} (or \textit{k-gram}) for a document is a sequence of \textit{k} tokens that appears in the doc
  - Tokens can be \textit{characters}, \textit{words} or something else, depending on the application
  - Assume tokens = characters for examples

- \textbf{Example:} \textit{k}=2; document \textit{D}_1 = abcab
  
  Set of 2-shingles: \( S(\textit{D}_1) = \{ab, bc, ca\} \)
Similarity Metric for Shingles

- **Document** $D_1$ is a set of its $k$-shingles $C_1 = S(D_1)$

- Equivalently, each document is a 0/1 vector in the space of $k$-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse

- **A natural similarity measure is the Jaccard similarity:**
  \[ \text{sim}(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \]
Motivation for Minhash/LSH

- Suppose we need to find similar documents among \( N = 1 \) million documents

- Naïvely, we would have to compute pairwise Jaccard similarities for every pair of docs
  - \( N(N - 1)/2 \approx 5 \times 10^{11} \) comparisons
  - At \( 10^5 \) secs/day and \( 10^6 \) comparisons/sec, it would take \( 5 \) days

- For \( N = 10 \) million, it takes more than a year...
Step 2: **Minhashing**: Convert large variable length sets to short fixed-length signatures, while preserving similarity.
From Sets to Boolean Matrices

- **Rows** = elements (shingles)
- **Columns** = sets (documents)
  - 1 in row e and column s if and only if e is a valid shingle of document represented by s
  - Column similarity is the Jaccard similarity of the corresponding sets (rows with value 1)
  - Typical matrix is sparse!

**Note: Transposed Document Matrix**

\[\begin{array}{cccc}
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
\end{array}\]

Outline: Finding Similar Columns

- **So far:**
  - A document → a set of shingles
  - Represent a set as a boolean vector in a matrix

- **Next goal:** Find similar columns while computing small signatures

- **Similarity of columns == similarity of signatures**
Next Goal: Find similar columns, Small signatures

Naïve approach:

1) Signatures of columns: small summaries of columns
2) Examine pairs of signatures to find similar columns
   - Essential: Similarities of signatures and columns are related
3) Optional: Check that columns with similar signatures are really similar
Next Goal: Find similar columns, Small signatures

Naïve approach:
1) Signatures of columns: small summaries of columns
2) Examine pairs of signatures to find similar columns
   ■ Essential: Similarities of signatures and columns are related
3) Optional: Check that columns with similar signatures are really similar

Warnings:
Comparing all pairs may take too much time: Job for LSH
   ■ These methods can produce false negatives, and even false positives (if the optional check is not made)
Hashing Columns (Signatures) : LSH principle

- **Key idea:** “hash” each column \( C \) to a small **signature** \( h(C) \), such that:
  1. \( h(C) \) is small enough that the signature fits in RAM
  2. \( \text{sim}(C_1, C_2) \) is the same as the “similarity” of signatures \( h(C_1) \) and \( h(C_2) \)

- **Goal:** Find a hash function \( h(\cdot) \) such that:
  1. If \( \text{sim}(C_1, C_2) \) is high, then with high prob. \( h(C_1) = h(C_2) \)
  2. If \( \text{sim}(C_1, C_2) \) is low, then with high prob. \( h(C_1) \neq h(C_2) \)

- Hash docs into buckets. Expect that “most” pairs of near duplicate docs hash into the same bucket!
Min-Hashing

- **Goal:** Find a hash function $h(\cdot)$ such that:
  - if $sim(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
  - if $sim(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

- **Clearly, the hash function depends on the similarity metric:**
  - Not all similarity metrics have a suitable hash function

- **There is a suitable hash function for the Jaccard similarity:** It is called **Min-Hashing**
Min-Hashing

- Imagine the rows of the boolean matrix permuted under random permutation $\pi$

- Define a “hash” function $h_{\pi}(C) = \text{the index of the first (in the permuted order $\pi$) row in which column } C \text{ has value 1}$:

$$h_{\pi}(C) = \min_{\pi} \pi(C)$$

- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column
Min-Hashing

- Imagine the rows of the boolean matrix permuted under random permutation $\pi$

- Define a "hash" function $h_{\pi}(C) =$ the index of the first (in the permuted order $\pi$) row in which column $C$ has value 1:
  \[ h_{\pi}(C) = \min_{\pi} \pi(C) \]

- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column
Min-Hashing Example

Permutation $\pi$  

Input matrix (Shingles x Documents)  

Signature matrix $M$

$2^{nd}$ element of the permutation is the first to map to a 1

Input matrix:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Signature matrix:

| 2 | 1 | 2 | 1 |
Min-Hashing Example

Permutation $\pi$  

Input matrix (Shingles x Documents)

Signature matrix $M$

2\textsuperscript{nd} element of the permutation is the first to map to a 1

4\textsuperscript{th} element of the permutation is the first to map to a 1

Min-Hashing Example

2\textsuperscript{nd} element of the permutation is the first to map to a 1

4\textsuperscript{th} element of the permutation is the first to map to a 1

Note: Another (equivalent) way is to store row indexes or raw shingles (e.g. mouse, lion):

Permutation $\pi$

Input matrix (Shingles x Documents)

Signature matrix $M$

Min-Hash Signatures

- Pick \( K = 100 \) random permutations of the rows

- Think of \( \text{sig}(C) \) as a column vector
  - \( \text{sig}(C)[i] \) = according to the \( i \)-th permutation, the index of the first row that has a 1 in column \( C \)
    \[
    \text{sig}(C)[i] = \min (\pi_i(C))
    \]

- **Note:** The sketch (signature) of document \( C \) is small \(~100\) bytes!

- We achieved our goal! We “compressed” long bit vectors into short signatures
Key Fact

For two sets $A$, $B$, and a min-hash function $mh_i()$:

$$Pr[mh_i(A) = mh_i(B)] = Sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Unbiased estimator for $Sim$ using $K$ hashes (notation policy – this is a different $K$ from size of shingle)

$$\hat{Sim}(A, B) = \frac{1}{k} \sum_{i=1:k} I[mh_i(A) = mh_i(B)]$$
Key Fact

For two sets $A$, $B$, and a min-hash function $mh_i()$:

$$Pr[mh_i(A) = mh_i(B)] = Sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Unbiased estimator for $Sim$ using $K$ hashes (notation policy – this is a different $K$ from size of shingle)

$$\hat{Sim}(A, B) = \frac{1}{k} \sum_{i=1:k} I[mh_i(A) = mh_i(B)]$$

The similarity of two signatures is the fraction of the hash functions in which they agree
## Min-Hashing Example

<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</td>
<td>1 0 1 0 1 0</td>
<td>2 1 2 1</td>
</tr>
<tr>
<td>3 2 4</td>
<td>1 0 0 1 0 1</td>
<td>2 1 4 1</td>
</tr>
<tr>
<td>7 1 7</td>
<td>0 1 0 1 0 1</td>
<td>1 2 1 2</td>
</tr>
<tr>
<td>6 3 2</td>
<td>0 1 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>1 6 6</td>
<td>0 1 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>5 7 1</td>
<td>1 0 1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>4 5 5</td>
<td>1 0 1 0 0 1</td>
<td></td>
</tr>
</tbody>
</table>

**Similarities:**

<table>
<thead>
<tr>
<th>Col/Col</th>
<th>Sig/Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>2-4</td>
</tr>
<tr>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Min-Hashing Example

Permutation $\pi$

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Input matrix (Shingles x Documents)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Signature matrix $M$

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>4</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Similarities:

<table>
<thead>
<tr>
<th>Col/Col</th>
<th>1-3</th>
<th>2-4</th>
<th>1-2</th>
<th>3-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sig/Sig</td>
<td>0.75</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Choose a random permutation $\pi$

Claim: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$

Why?
The Min-Hash Property

- Choose a random permutation $\pi$
- Claim: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- Why?
  - Let $X$ be a doc (set of shingles), $y \in X$ is a shingle

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
The Min-Hash Property

- Choose a random permutation $\pi$
- Claim: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
- Why?
  - Let $X$ be a doc (set of shingles), $y \in X$ is a shingle
  - Then: $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$
    - It is equally likely that any $y \in X$ is mapped to the $\min$ element

One of the two cols had to have 1 at position $y$
The Min-Hash Property

- Choose a random permutation $\pi$
- **Claim:** $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
- **Why?**
  - Let $X$ be a doc (set of shingles), $y \in X$ is a shingle
  - **Then:** $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$  
    - It is equally likely that any $y \in X$ is mapped to the min element
  - Let $y$ be s.t. $\pi(y) = \min(\pi(C_1 \cup C_2))$
  - **Then either:** $\pi(y) = \min(\pi(C_1))$ if $y \in C_1$, or $\pi(y) = \min(\pi(C_2))$ if $y \in C_2$

One of the two cols had to have 1 at position $y$
Choose a random permutation $\pi$

Claim: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$

Why?

- Let $X$ be a doc (set of shingles), $y \in X$ is a shingle.
- Then: $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$.
  - It is equally likely that any $y \in X$ is mapped to the min element.

- Let $y$ be s.t. $\pi(y) = \min(\pi(C_1 \cup C_2))$.
- Then either: $\pi(y) = \min(\pi(C_1))$ if $y \in C_1$, or $\pi(y) = \min(\pi(C_2))$ if $y \in C_2$.

So the prob. that both are true is the prob. $y \in C_1 \cap C_2$.

$\Pr[\min(\pi(C_1)) = \min(\pi(C_2))] = |C_1 \cap C_2| / |C_1 \cup C_2| = \text{sim}(C_1, C_2)$.

One of the two cols had to have 1 at position $y$. 

The Min-Hash Property (Take 2: simpler proof)

- **Choose a random permutation** $\pi$
- **Claim:** $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
- **Why?**
  - Given a set $X$, the probability that any one element is the min-hash under $\pi$ is $1/|X|$ $\leftarrow (0)$
    - It is equally likely that any $y \in X$ is mapped to the min element
  - Given a set $X$, the probability that one of any $k$ elements is the min-hash under $\pi$ is $k/|X|$ $\leftarrow (1)$
The Min-Hash Property (Take 2: simpler proof)

- Choose a random permutation $\pi$
- Claim: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
- Why?
  - Given a set $X$, the probability that any one element is the min-hash under $\pi$ is $1/|X| \quad \leftarrow (0)$
    - It is equally likely that any $y \in X$ is mapped to the min element
  - Given a set $X$, the probability that one of any $k$ elements is the min-hash under $\pi$ is $k/|X| \quad \leftarrow (1)$
  - For $C_1 \cup C_2$, the probability that any element is the min-hash under $\pi$ is $1/|C_1 \cup C_2| \quad (\text{from } 0) \quad \leftarrow (2)$
  - For any $C_1$ and $C_2$, the probability of choosing the same min-hash under $\pi$ is $|C_1 \cap C_2|/|C_1 \cup C_2| \quad \leftarrow \text{from (1) and (2)}$
Similarity for Signatures

- We know: $Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- Now generalize to multiple hash functions

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

- **Note:** Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures
**Min-Hashing Example**

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

**Similarities:**

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

Min-Hash Signatures

- Pick $K=100$ random permutations of the rows
- Think of $\text{sig}(C)$ as a $K \times 1$ column vector
- $\text{sig}(C)[i] = \text{according to the } i\text{-th permutation, the index of the first row that has a 1 in column } C$

$$\text{sig}(C)[i] = \min (\pi_i(C))$$
Min-Hash Signatures

- Pick $K=100$ random permutations of the rows
- Think of $\text{sig}(C)$ as a $K*1$ column vector
- $\text{sig}(C)[i] =$ according to the $i$-th permutation, the index of the first row that has a 1 in column $C$
  \[ \text{sig}(C)[i] = \min(\pi_i(C)) \]
- **Note:** The sketch (signature) of document $C$ is small $\sim 100$ bytes!

- **We achieved our goal!** We “compressed” long bit vectors into short signatures
Implementation Trick

- **Permuting rows even once is prohibitive**

- **Row hashing!**
  - Pick $K = 100$ hash functions $k_i$
  - Ordering under $k_i$ gives a random row permutation!

- **One-pass implementation**
  - For each column $C$ and hash-func. $k_i$ keep a “slot” for the min-hash value
  - Initialize all $\text{sig}(C)[i] = \infty$
  - Scan rows looking for 1s
    - Suppose row $j$ has 1 in column $C$
    - Then for each $k_i$:
      - If $k_i(j) < \text{sig}(C)[i]$, then $\text{sig}(C)[i] \leftarrow k_i(j)$

---

How to pick a random hash function $h(x)$?

**Universal hashing:**

$h_{a,b}(x) = ((a \cdot x + b) \mod p) \mod N$

where:

- $a, b$ … random integers
- $p$ … prime number ($p > N$)

Summary: Two Key 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
Backup slides