Data Mining
Learning from Large Data Sets

Lecture 3 – Locality Sensitive Hashing

263-5200-00L
Andreas Krause
Announcement

- No class next week
Review:

Fast near neighbor search in high dimensions
Locality sensitive hashing

- **Idea:** Create hash function that maps “similar” items to same bucket

- **Key problem:** Is it possible to construct such hash functions??
  - Depends on the distance function
  - Possible for Jaccard distance!! 😊
  - Some other distance functions work as well
Recall: Shingle Matrix

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

<table>
<thead>
<tr>
<th>documents</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>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Min-hashing

Simple hash function, constructed in the following way:

- Use random permutation $\pi$ to reorder the rows of the matrix
  - Must use same permutation for all columns $C$!!
- $h(C) =$ minimum row number in which permuted column contains a 1

$$h(C) = h_{\pi}(C) = \min_{i : C(i) = 1} \pi(i)$$
Min-hashing property

- Want that similar documents (columns) have same value of hash function (with high probability)

- Turns out it holds that

\[ \Pr[h(C_1) = h(C_2)] = \text{Sim}(C_1, C_2) \]

- Need to control false positives and misses.
### Min-hash signatures

<table>
<thead>
<tr>
<th>Input matrix</th>
<th>Signature matrix ( M )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4 3 3 1 0 1 0</td>
<td>2 1 2 1 2</td>
</tr>
<tr>
<td>3 2 4 1 0 1 0 1</td>
<td>2 1 4 1 2</td>
</tr>
<tr>
<td>7 1 7 0 1 0 1 1</td>
<td>1 2 1 2 2</td>
</tr>
<tr>
<td>6 3 6 0 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>2 6 1 0 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>5 7 2 0 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>4 5 5 0 1 0 1 1</td>
<td></td>
</tr>
</tbody>
</table>

#### Similarities:

<table>
<thead>
<tr>
<th></th>
<th>1-3</th>
<th>2-4</th>
<th>1-2</th>
<th>3-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col/Col</td>
<td>0.75</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sig/Sig</td>
<td>0.67</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Hashing bands of $M$
One hash function

![Line graph showing the relationship between similarity and the probability of a hash hit. The graph indicates a linear increase from 0 to 1 as similarity increases from 0 to 1.]

- $r=1$
- $b=1$
100 hash functions

\[ P(\text{hash hit}) \]

Similarity

\[ r=10 \]
\[ b=10 \]
100 hash functions
1000 hash functions
10000 hash functions
So far we have considered

- **Min-hashing** for computing compact document signatures representing Jaccard similarity
- **Locality Sensitive Hashing (LSH)** for decreasing false negatives and false positives
- Let’s us do duplicate detection without requiring pairwise comparisons!

Can we generalize what we learned?

- Other data types (e.g., real vectors ➔ images)
- Other distance functions (Euclidean? Cosine?)
Key insight behind LSH

- LSH allows to **boost** the gap between similar \((\text{Sim}(C1,C2)>s)\) non-similar \((\text{Sim}(C1,C2)<s' \text{ for } s'<s)\) pairs

![Graphs showing the P(hash hit) vs Similarity for different values of r and b.](image)
LSH more generally

- Consider a metric space \((S, d)\), and a family \(F\) of hash functions \(h: S \rightarrow B = \{\ldots, \gamma\}\).

- \(F\) is called \((d_1, d_2, p_1, p_2)\)-sensitive if

\[
\forall x, y \in S : d(x, y) \leq d_1 \Rightarrow \Pr_{h}[h(x) = h(y)] \geq p_1
\]

\[
\forall x, y \in S : d(x, y) \geq d_2 \Rightarrow \Pr_{h}[h(x) = h(y)] \leq p_2
\]

\(h\) drawn uniformly at random

from \(F\)
Example: Jaccard-distance

Recall, we want:

\[ \forall x, y \in S : d(x, y) \leq d_1 \Rightarrow \Pr[h(x) = h(y)] \geq p_1 \]

\[ \forall x, y \in S : d(x, y) \geq d_2 \Rightarrow \Pr[h(x) = h(y)] \leq p_2 \]

\[ F = \{ \text{IT permutations used in n.n. hashing} \} \]

\[ \text{is } (d_1, 1-d_1, d_2, 1-d_2) \text{-sensitive} \]

\[ \Pr(h_i (f)) = 1-d \text{ boosting} \]
Boosting a LS hash family

- Can we reduce false positives and false negatives (create “S-curve effect”) for arbitrary LS hash functions??

- Can apply same partitioning technique!
- AND/OR construction
r-way AND of hash function

- **Goal**: Decrease false positives
- Convert hash family $F$ to new family $F'$
- Each member of $F'$ consists of a “vector” of $r$ hash functions from $F$
- For $h = [h_1,...,h_r]$ in $F'$, $h(x)=h(y) \iff h_i(x)=h_i(y)$ for all $i$.

**Theorem**: Suppose $F$ is $(d_1,d_2,p_1,p_2)$-sensitive. Then $F'$ is $(\ldots,p_r^r,p_2^r)$-sensitive.
b-way OR of hash function

- **Goal:** Decrease false negatives
- Convert hash family $F$ to new family $F'$
- Each member of $F'$ consists of a “vector” of $b$ hash functions from $F$
- For $h = [h_1, ..., h_r]$ in $F'$, $h(x) = h(y) \iff h_i(x) = h_i(y)$ for some $i$.

**Theorem:** Suppose $F$ is $(d_1, d_2, p_1, p_2)$-sensitive. Then $F'$ is $(d_1, d_2, (1-p_1)^b, (1-(1-p_2)^b)$-sensitive.
Suppose we start with a \((d_1, d_2, p_1, p_2)\)-sensitive \(F\).

First apply \(r\)-way AND, then \(b\)-way OR.

This results in \((d_1, d_2, 1 \cdot (1 - p_1)^b, 1 \cdot (1 - p_2)^b)\) sensitive \(F'\).

Can also reverse order of AND and OR.

This results in \((d_1, d_2, (1 \cdot (1 - p_1)^b)^\gamma, (1 \cdot (1 - p_2)^b)^\gamma)\) sensitive \(F'\).
Example
Cascading constructions

- Can also combine all previous constructions
- For example, first apply (4,4) OR-AND construction followed by a (4,4) AND-OR construction.
- Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9999996,.0008715)-sensitive family!

- How many hash functions are used?

\[4^4 = 256\]
Other examples of LS families

- So far: Jaccard distance has a LS hash family
- Several other distance functions do too
  - Cosine distance
  - Euclidean distance
LSH for Cosine Distance

\[ h_w(u) = \text{sign}(w^T u) \]
\[ F = \{ h_w : w \in \mathbb{R}^D, \|w\|_2 = 1 \} \]

\[ w^T x > 0 \]
\[ = \{ w^T x = 0 \} \]
\[ \cap \{ w^T x < 0 \} \]
**Key idea:** Map points to random line

Here, let’s consider 2 dimensions (but generalizes)
LSH for Euclidean distance

Points at distance \( d \)

If \( d < a \), then the chance the points are in the same bucket is

\[
1 - \frac{a}{q}
\]

Bucket width \( a \)

Randomly chosen line
If $d \gg a$, $\theta$ must be close to $90^\circ$ for there to be any chance points go to the same bucket.
LSH for Euclidean distance

- If distance \( d \leq \frac{a}{2} \), \( P(\text{same bucket}) \geq 1 - \frac{d}{a} = \frac{1}{2} \)
- If distance \( d \geq 2a \), then they can end up in the same bucket only if \( d \cos \theta \leq a \)
  - \( \cos \theta \leq \frac{1}{2} \)
  - \( 60 \leq \theta \leq 90 \)
  - This event has probability at most \( \frac{1}{3} \).
- Yields a \((a/2, 2a, 1/2, 1/3)\)-sensitive family of hash functions for any \( a \).
- Can boost using AND-OR constructions
LSH in MapReduce?

- LSH is well suited for MapReduce style computation!
- You’ll find out how in the homework 😊
LSH for nearest neighbor search

- So far we discussed the problem of finding near duplicates.
- How do we implement nearest neighbor search?
Approximate near-neighbor search

- Consider slightly different problem: *approximate near neighbor search*
  - Want to find any point in data set that has distance at most $r$ from query
  - Don’t want to return points of distance more than $(1+\varepsilon) r$

- Pick $(r, (1+\varepsilon) r, p,q)$-sensitive hash family

**Preprocessing**: Hash data set as in duplicate detection

**Query**: Hash query in the same way

Retrieve all candidate pairs (perhaps pick closest)
Can we use approximate near-neighbor search for (approximate) nearest-neighbor search?
use binary search to find approximate nearest neighbor distance
Course organization

- **Retrieval**
  - Given a query, find “most similar” item in a large data set
  - *Applications*: GoogleGoggles, Shazam, ...

- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, ...

- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, “common patterns”; anomaly detection
  - *Applications*: Recommender systems, fraud detection, ...

- **Learning with limited feedback**
  - Learn to optimize a function that’s expensive to evaluate
  - *Applications*: Online advertising, opt. UI, learning rankings, ...
Classification (intuitively)

- Want to assign data points
  - Documents
  - Queries
  - Images
  - Audio
  - ...
  
a label (spam/not-spam; topic such as sports, politics, entertainment, etc.)

- Goal:
  - extract rules (hypotheses) based on training examples.
  - Hope that those rules generalize to previously unseen data
**Input:** Labeled data set (e.g., rep. bag-of-words) with positive (+) and negative (-) examples

**Output:** Decision rule (hypothesis)
Linear classifiers

Data set

\[(x_1, y_1), \ldots, (x_n, y_n)\]

\[x_i \in \mathbb{R}^D, \ y_i \in \{+1, -1\}\]

\[w^T x + b = 0\]

\[w^T x + b > 0\]

\[y \leq \text{sign} (w^T x + b)\]
Which linear classifier is the best one?

- Data set
  \((x_1, y_1), \ldots, (x_n, y_n)\)

- Linear classifier:
  \(\text{sign}(w^T x + b)\)
Large margin classification

- Margin of confidence:
  \[ y_i (w^T x_i + b) \geq \eta_i \]
  \[
  \max \min_{w,b} \eta_i
  \]

- Want to maximize confidence in our prediction!

- Turns out to be the “right” thing to do
  - Larger margin \(\Rightarrow\) Better generalization
\[
\begin{align*}
\max_{w,b,\gamma} \gamma &\geq \infty \\
\text{s.t.} \quad (w^T x_i + b) y_i &\geq \gamma
\end{align*}
\]

So far, our notion of confidence is not yet well defined!
Review: Projection on a plane

\[ \gamma = \| x_i - \bar{x}_i \|_2 \]

\[ w^T x + b = 0 \]

\[ X_i = \bar{X}_i + \gamma \frac{w}{\| w \|} \]

\[ w^T X_i + b = w^T \bar{X}_i + b + \gamma \frac{w^T w}{\| w \|} \]

\[ = 0 + \gamma \frac{w^T w}{\| w \|} \]

\[ = \gamma \frac{w^T w}{\| w \|} \]

\[ w^T W = \| w \|_2^2 \]
For all positive examples
\[ w^T x + b \geq 1 \]
For all negative examples
\[ w^T x + b \leq -1 \]
\[ y (w^T x + b) \geq 1 \]
\[ \gamma = \frac{1}{\lVert w \rVert} \]
Maximizing the normalized margin

\[
\begin{align*}
\max_{w, b, \gamma} \ & \gamma \\
\text{s.t.} \ & (w^T x_i + b) y_i \geq \gamma & \forall i \\
\gamma = \frac{1}{\|w\|_2} \\
\end{align*}
\]

\[
\begin{align*}
\equiv \min_{w, b} \ & \|w\|_2 \\
\text{s.t.} \ & (w^T x_i + b) y_i \geq 1 & \forall i \\
\end{align*}
\]

\[
\begin{align*}
\equiv \min_{w, b} \ & w^T w \\
\text{s.t.} \ & (w^T x_i + b) y_i \geq 1 & \forall i \\
\end{align*}
\]

Support Vector Machines
Acknowledgments

- Several slides adapted from the material accompanying the textbook (Anand Rajaraman, Stanford)