Today's topics

- FAQs
- Minhash
- Minhash signature
- Calculating Minhash with MapReduce
- Locality Sensitive Hashing

FAQs

- Demo for PA1 is going on
- PA2 is available

Applications of Near-neighbor search

Near Neighbor (NN) Search

- Also known as proximity search, similarity search
- Finding closest or most similar points
- The post-office problem
- Assigning to a residence the nearest post office

Hashing n-grams

- Creating buckets and use the bucket numbers as the shingles
- For the set of 9-grams,
  - Map each of those 9-grams to a bucket number in the range 0 to $2^9 - 1$
  - 9 bytes of data is compacted to 4 bytes
- Can hash-based approaches provide NN search results?
Applications of Near-neighbor search: Minhashing

Similarity-preserving summaries of set

- Signatures
  - Replacing large sets of n-grams by much smaller representations
- We should be able to compare the signatures of two sets and estimate the Jaccard similarity
  - Of the underlying sets from the signatures alone

Matrix representations of sets

<table>
<thead>
<tr>
<th>Element</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>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\(S_1 = \{a,d\} \)
\(S_2 = \{c\} \)
\(S_3 = \{b,d,e\} \)
\(S_4 = \{a,c,d\} \)

Minhashing

- Signature generating algorithm

Minhash of the characteristic matrix

\(\pi = (b,e,a,d,c)\)

\(X = \text{number of rows with 1 for both } S \text{ and } T \text{ (e.g. } x = 1)\)
\(Y = \text{number of rows with either } S \text{ or } T \text{ have 1, but not both (e.g. } y = 2)\)
\(Z = \text{number of rows with both 0 (e.g. } z = 2)\)

\[P(\text{minhash}(S) = \text{minhash}(T)) = \frac{X}{X+Y} = JaccardSIM(S,T)\]
Minhash and permuting the order

- What if we change the order of Elements?
- Can we simulate the effect of a random permutation?
- **It is NOT feasible to permute a large characteristic matrix explicitly**
  - N element will need N! permutations!

Minhash Signatures

- Pick (at random) some number \( n \) of permutations of the rows of the characteristic matrix \( M \)
  - E.g. 100 permutations or several hundred permutations
- The minhash functions are determined by these permutations \( h_1, h_2, h_3, \ldots, h_n \)
- Minhash signature for \( S \)
  - Vector \([h_1(S), h_2(S), h_3(S), \ldots, h_n(S)]\)

Representing the Minhash Signatures

- Form a signature matrix
  - The \( i \)th column of \( M \) is replaced by the min hash signature of the \( i \)th column
- Compressed form for a sparse matrix

\[
\begin{array}{c|cccc}
\text{Element} & S1 & S2 & S3 & S4 \\
\hline
b & 0 & 0 & 1 & 0 \\
e & 0 & 0 & 1 & 0 \\
d & 1 & 0 & 1 & 1 \\
c & 0 & 1 & 0 & 1 \\
\end{array}
\]

>10^9 elements?

Using a random hash function

- A Random hash function
  - Maps row numbers to as many buckets as there are rows
  - 0, 1, …, k-1 to bucket numbers 0 ~ k-1
  - Maps some pairs of integers to the same bucket
  - Leaves other buckets unfilled
  - However, not too many collisions

Using a random hash function

- Pick \( n \) randomly chosen hash functions \( h_1, h_2, \ldots, h_n \) on the rows
  - In the signature matrix, let \( SIG(i,c) \) be the element of the signature matrix for the \( i \)th hash function and column \( c \)
  - Initially, the set \( SIG(i,c) \) to \( \infty \) for all \( i \) and \( c \)

- For each row \( r \)
  - Compute \( h_1(r), h_2(r), \ldots, h_n(r) \)
  - For each column \( c \)
    - If \( r \) has 0 in row, do nothing
    - If \( r \) has 1 in row, then for each \( i = 1, 2, \ldots, n \) set \( SIG(i,c) \) to the smaller of the current value of \( SIG(i,c) \) and \( h_i(r) \)

Example

\[
\begin{array}{c|cccc|c|c|c|c|c|c|c|c|c}
\text{Row} \rightarrow \text{(element)} & S1 & S2 & S3 & S4 & \text{Hash 1} & \ldots & \text{Hash M} \\
\hline
0 & 1 & 0 & 0 & 1 & 1 & \ldots & 1 \\
1 & 0 & 0 & 1 & 0 & 2 & \ldots & 4 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
N-2 & 1 & 0 & 1 & 1 & 4 & \ldots & 0 \\
N-1 & 0 & 0 & 1 & 0 & 0 & \ldots & 3 \\
\end{array}
\]
Example

<table>
<thead>
<tr>
<th>Row (element)</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hash 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Hash M-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Hash M</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Example continued

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>h2</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

- $h_1(x) = (x + 1) \mod 5$
- $h_2(x) = (3x + 1) \mod 5$
- In slide #19, $h_1(0)$ and $h_2(0)$ are both 1
- The row numbered 0 has 1’s in $S1$ and $S4$

Example continued

- Row number 1
  - Only in $S3$ is 1
  - Hash value
    - $H_1(1)=2$ and $h_1(1)=4$

Example continued

- Row number 3
  - $S1$, $S3$ and $S4$ have 1
  - Hash value
    - $h_1(3)=4$ and $h_1(3)=0$
Calculating a single MinHash value with MapReduce

- Suppose you have some documents and have stored n-grams of these documents in a large table
- Each column
  - n-grams for a single document
- Each row
  - r-th n-gram for all the documents
- Compute a minhash value for each of your documents using a single hash function

Approach 1. Partitioning by rows

- The table must be partitioned across the processors by rows
- Each processor receives a subset of the n-grams for every document
- Can each processor compute the minhash for each document?
  - No. It should pass the hash values and group them based on the document ID (Assume that document ID is stored separately)

Design your map function

- What should be the input <key, value>?
- What should be the output <key, value>?

```java
map(k: docID, v: (row_number, freq)) {
  if (freq > 0) {
    emit_intermediate(ik = k,
   iv = hash(row_number))
  }
}
```

Design your reduce function

- MapReduce system will group the hashes by documentID
- Reduce function will find the minimum hash value within the group

```java
reduce(ik: docID, ivs: hashval[]) {
  var minhash = INFINITY
  for each iv in ivs {
    if iv < minhash {
      minhash = iv
    }
  }
  emit_final(fk = ik, fv = minhash)
}
```

Approach 2. Partitioning by column

- All the k-grams for a document go to the same processor

```java
map(k: docID, v: kgram_row_with_value[]) {
  var minhash = INFINITY
  for each kgram with non 0 value in v {
    var h := hash(kgram_row)
    if h < minhash {
      minhash := h
    }
  }
  emit_intermediate(ik = k, iv = minhash)
}
```

- Do you need a reduce function?
Locality sensitive hashing

Finding the most similar documents

- Number of pairs of documents to compare is too high
- Example
  - 1M documents, signatures of length 250 (4 Byte each)
  - $1M \times 1,000\text{Bytes} = 1\text{GB}$
  - Number of comparisons = $\frac{1M \times (1M - 1)}{2}$ (Half of trillion pairs)
  - $1\text{ms}$ per calculation of similarity
  - 6 days to complete computing

Do we need to calculate the similarity for all of the pairs?

Locality-sensitive hashing (LSH)

- Near-neighbor search
- LSH
  - Hash items several times, so that similar items are more likely to be hashed to the same bucket than dissimilar items
  - Assign the candidate pair to the same bucket
  - Reduce false positives
    - Dissimilar pairs in the same bucket
  - Reduce false negatives
    - Similar pairs in different buckets

Dividing a signature matrix into 4 bands and 3 rows per band

The first band of MinHash signature

Analysis of the Banding Technique [1/2]

- Suppose that we use $b$ bands of $r$ rows each
- Suppose that a particular pair of documents have Jaccard Similarity $s$
  - $P$thh minhash signatures for these documents agree in any one particular row of the signature matrix $= s$
Analysis of the Banding Technique  [2/2]

- The probability that documents (signatures) become a candidate pair (Signatures should match at least in ONE band):
  - $P$(the signatures agree in all rows of one particular band)=$s^r$
  - $P$(the signatures do not agree in at least one row of a particular band)=$1-s^r$
  - $P$(the signatures do not agree in all rows of any of the bands)=$(1-s^r)^b$

\[
P\text{(the signatures agree in all the rows of at least one band)} = 1-(1-s^r)^b
\]

Threshold

- The value of similarity $s$ to determine two documents are similar
- An approximation of the threshold is $(1/b)^r$

Example

- For example
  - If there are 16 bands and each band contains 4 rows
    - $s=0.5$
  - If there are 8 bands and each band contains 8 rows
    - $s=0.77$
  - If there are 4 bands and each band contains 16 rows
    - $s=0.92$

Distance Measures

Distance measure

- A distance measure over a space is a function $d(x,y)$ that takes two points in the space as arguments and produces a real number that satisfies the following axioms:
  1. $d(x,y) \geq 0$ (no negative distance)
  2. $d(x,y) = 0$ if and only if $x = y$
  3. $d(x,y) = d(y,x)$ (distance is symmetric)
  4. $d(x,y) \leq d(x,z) + d(z,y)$ (the triangle inequality)
Distance measures

- **Euclidean distances**
  \[ d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \]

- **Jaccard distance**
  \[ d(x, y) = 1 - \text{SIM}(x, y) \]

- **Cosine distance**
  - Degree between the vectors

- **Hamming distance**
  - The number of components in which they differ
  - 10111 and 11110?

This material is built based on,

- Chapter 5
- [http://infolab.stanford.edu/~ullman/mmds.html](http://infolab.stanford.edu/~ullman/mmds.html)

Early Search Engines

- They worked by crawling the Web and listing the terms
- Words or other strings of characters other than white space
- In an inverted index
- An **inverted index** is a data structure that makes it easy, given a term, to find (pointer to) all the places where that term occurs
Inverted index (1/2)

- Inverted index
  - For given texts:
    
    \[ T[0] = \text{"Colorado State University"} \]
    
    \[ T[1] = \text{"Colorado water source"} \]
    
    \[ T[2] = \text{"University of Colorado"} \]

- We have the following inverted file index

```
"Colorado": \{0,1,2\}
"State": \{0\}
"water": \{1\}
"University": \{0,2\}
```

Inverted index (2/2)

- A term search for the terms, "Colorado", "State", and "University" would give the set

\[ \{0,1,2\} \cap \{0\} \cap \{0,2\} = \{0\} \]

Term spam

- If you were selling shirts on the Web
  - All you care about was that people would see your page
  - You could add a term like "movie" to your page
  - Add thousands of times
    - It does not even need to show
    - Give the same color as background to the letters
  - A search engine would think this page is very important one about "movie"
  - You could go to the search engine and search "movie" and see
    the first listed page
    - Copy that page with the same color as background