Mining Massive Datasets:

Week 3: Locality-Sensitive Hashing

Ian Quah (itq)

June 14, 2017

Topic 1

Finding Similar Sets

1. **Applications:**

   Given a body of docs, find pairs of documents with a lot of text in common

   (a) Mirror Sites, or approximate mirrors: don't want to show both in a search

   (b) Plagiarism, including large quotations

   (c) Similar news sites (cluster articles by "same story ")

2. **The process:** Shingling → Min-hashing → Locality-Sensitive hashing

   (a) **Shingling**: Convert docs, emails, etc. to sets

   (b) **MinHasing**: convert large sets to short signatures, preserving similarity

   (c) **Locality-Sensitive Hashing** focus on pairs of signatures likely to be similar

3. **Shingles**

   A k-shingle (or k-gram) for a document is a sequence of k chars that appear in the document
   Example: k = 2, doc = abcab, set of 2-shingles = {ab, bc, ca}

   Shingles and Similarity

   (a) Documents that are intuitively similar will have many shingles

   (b) Changing a word only affects k-shingles within dist k from the word

   (c) reordering paragraphs: affects 2k shingles that cross para boundaries

   Shingles: Compression

   (a) Compress long shingles (called tokens)

   (b) Represent a doc by the tokens, the set of hash values of its k-shingles

Topic 2

Min-Hashing

1. **Jaccard Similarity**

   (a) Size of intersection divided by size of the union

   (b) Sim(C_1,C_2) = $\frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$

2. **Sets to Boolean Matrices**

   (a) Rows = elems of universal set, e.g: set of all k-shingles

   (b) Col = sets

   (c) 1 in row e and col S iff e is a member of S

   (d) Col similarity is Jaccard of sets of rows with at least 1 in either of the cols' rows

   (e) Typical matrix is sparse

Topic 2 continued on next page...

3. **Consider the following example**

C_1	C_2
0	1
1	0
1	1
0	0
1	1
0	1

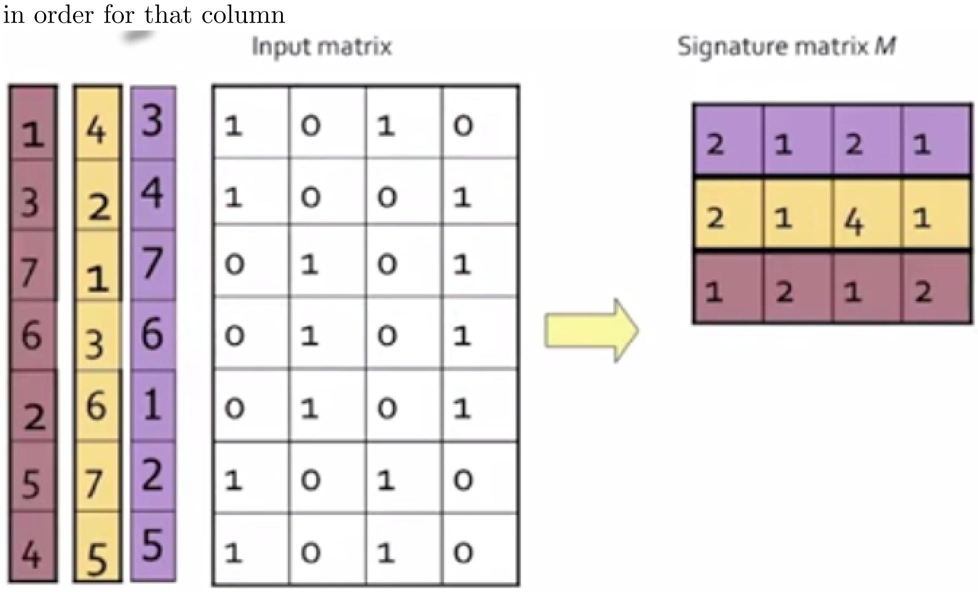
- 6 rows, but only 5 of them have at least a single 1 in them. Thus, denominator is 5
- Sim(C_1,C_2) == 2/5 == 0.4

4. **Minhashing**

   (a) Defn: minhash function h(C) = # of first row in which column C has 1

   (b) Use several independent hash functions to create signature for each column

   (c) Signature matrix: alternate repr for signatures - columns == sets and rows == minhash values, in order for that column



   (d) **Algorithm**

      i. For each hash function
      ii. permute the number of rows and then access the rows in that order
      iii. access the cols (within the row) which have 1's in them and give them the index value (if they haven't been accessed before, else keep the min)
      iv. Continue until we've gotten a hash for all columns

   (e) Similarity

      i. Columns and sets: jacard similarity
      ii. Signatures: fraction of components in which the two signatures agree

   (f) **Simulating permutations w/o actually permuting**

A good approximation to permuting rows: pick, say, 100 hash functions.
For each column c and each hash function h_i, keep a "slot" $M(i, c)$.
Intent: $M(i, c)$ will become the smallest value of $h_i(r)$ for which column c has 1 in row r.

- I.e., $h_i(r)$ gives order of rows for i^{th} permutation.

```
for each row r do begin
   for each hash function h_i do
      compute h_i(r);
   for each column c
      if c has 1 in row r
         for each hash function h_i do
            if h_i(r) is smaller than M(i, c) then
               M(i, c) := h_i(r);
end;
```

Row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

$h(x) = x \bmod 5$
$g(x) = (2x+1) \bmod 5$

	Sig1	Sig2
$h(1) = 1$	1	∞
$g(1) = 3$	3	∞
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	1	2
$g(3) = 2$	2	0
$h(4) = 4$	1	2
$g(4) = 4$	2	0
$h(5) = 0$	1	0
$g(5) = 1$	2	0