Data mining

Locality Sensitive Hashing

University of Szeged

Similarity of textual documents

- k-shingles: character k-grams of a document, e.g. $k = 2$ and $D =' abdca' \Rightarrow D = \{ab, bd, dc, ca\}$
- Handling whitespaces (e.g. *The plane was ready for touch down.* and *The quarterback scored a touchdown.*)
- choosing k is important (what should be the objectives)
- One way to measure document similarity is by calculating the Jaccard similarity of their k-shingles representation

k-shingles – characters vs. words

- English alphabet has 27 letters $\Rightarrow$ possible 4-shingles $=$ 531,441
- $P(\text{"}than\text{"}) \gg P(\text{"}qyzz\text{"}) \approx P(\text{"}yyyy\text{"})$
- increase value k (e.g. k'=10) and represent documents as the set of hash values (stored on e.g. k bytes) of its k'-shingles
- similarity of news articles: instead of character-based k-shingles of tokens that started with a stopword were utilized. Reason?

Locality Sensitive Hashing (LSH) – Motivation

- Suppose we would like to find duplicate documents in a corpus of size $N = 1,000,000 = 10^6$
- Brute force solution: calculating $\binom{N}{2}$ Jaccard similarities
- carrying out 10^6 comparisons per second, that would take more than 5.5 days

Locality Sensitive Hashing (LSH)

- Find hash function h such that it satisfies with high probability that:
  - $s(A, B) \to 1 \Rightarrow h(A) = h(B)$
  - $s(A, B) \to 0 \Rightarrow h(A) \neq h(B)$
- Since similar documents are likely to share their hash value, comparing those elements for which $h(A) = h(B)$ will be sufficient

Representing sets

- In the form of signatures: characteristic matrix

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

- The entire matrix is naturally not stored in practice $\rightarrow$ sparse representation

Minhash function

- Reorder the rows of the characteristic matrix
- Let the hash value of a set be the index of the first occurrence of a non-zero element
- e.g. $h_{min}(S_2) = 4,\ h_{min}(S_3) = 0$

Item	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Minhash signatures

- Measuring similarity of minhashes based on all the possible reordering of rows of the characteristic matrix gives Jaccard similarity
- Determine minhash values for multiple random permutations of the rows of the characteristic matrix
- Represent sets as a smaller dimension (say 100) vector of minhash values
- For large matrices permutation cannot be carried out effectively. Solution?

Minhash signatures – Example

Item	S_1	S_2	S_3	S_4	$h_1 = x + 1 \mod 5$	$h_2 = 3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Initialization

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

Minhash signatures – Example

Item	S_1	S_2	S_3	S_4	$h_1 = x + 1 \mod 5$	$h_2 = 3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Step 1

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

Minhash signature – Example

Item	S_1	S_2	S_3	S_4	$h_1 = x + 1 \mod 5$	$h_2 = 3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Step 2

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Minhash signature – Example

Item	S_1	S_2	S_3	S_4	$h_1 = x + 1 \mod 5$	$h_2 = 3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Step 3

	S_1	S_2	S_3	S_4
h_1	1	3	2	$\min(1,3)$
h_2	1	2	4	$\min(1,2)$

Minhash signature – Example

Item	S_1	S_2	S_3	S_4	$h_1 = x + 1 \mod 5$	$h_2 = 3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Step 4

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Minhash signature – Example

Item	S_1	S_2	S_3	S_4	$h_1 = x + 1 \mod 5$	$h_2 = 3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Step 5

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Minhash signature – Example

- Final minhash signatures

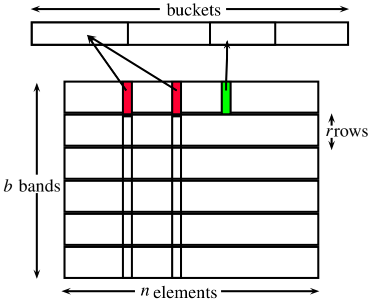
	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

- Estimated (and the true) similarities

	S_1	S_2	S_3	S_4
S_1	1.0 (1,0)	0.0 (0.0)	0.5 (0.25)	1.0 (2/3)
S_2	0.0 (0,0)	1.0 (1.0)	0.0 (0.0)	0.0 (1/3)
S_3	0.5 (0,25)	0.0 (0.0)	1.0 (1.0)	0.5 (0.2)
S_4	1.0 (2/3)	0.0 (1/3)	0.5 (0.2)	1.0 (1.0)

Locality Sensitive Hashing

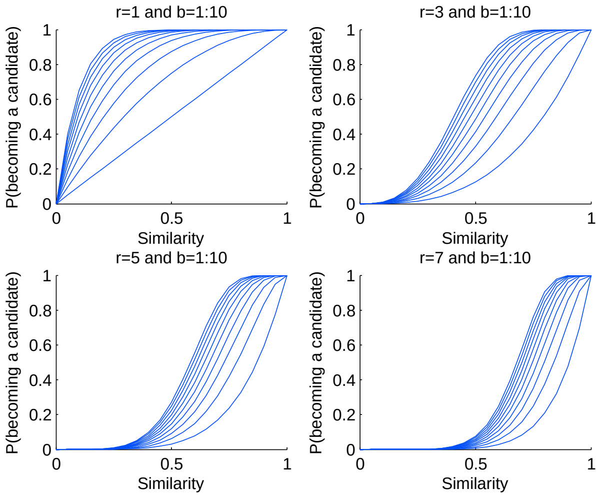
- $M = [m_{i,j}]_{k \times n}$: $m_{i,j}$ = minhash value of the i^{th} permutation of the j^{th} data point
- divide M to b bands each having r rows ($k = br$)
- $\exists$ band such that each r minhash values equal within that band $\Rightarrow$ a, b are regarded as high-similarity candidates



Locality Sensitive Hashing

- A, B has x lines in common (line of type 1-1), and differ in y lines (on lines of type either 0-1 or 1-0) $\Rightarrow sim(A,B) =$ P(a and b shares a minhash value)$=\frac{x}{x+y}$
- P(A and B differs on at least one line in r rows) $= 1 - sim(A,B)^r$
- P(A and B has at least one band of signatures being equal) $= 1 - (1 - sim(A,B)^r)^b$

Locality Sensitive Hashshing – The effect of r and b



Locality sensitive functions

- Suppose we have a distance metric d over the points of the data space S
- a family of functions H is said to be $(d1, d2, p1, p2)$-sensitive, if for every $h \in H$ and $(A, B) \in S$ pairs of points, it holds that:
  - $d(A, B) < d_1 \Rightarrow P(h(A) = h(B)) \geq p_1$
  - $d(A, B) > d_2 \Rightarrow P(h(A) = h(B)) \leq p_2$
- minhash function is a $(d_1, d_2, (1 - d_1), (1 - d_2))$-sensitive function with respect Jaccard similarity (for $d_1 \leq d_2$). What is the case when $d_1 < d(A, B) < d_2$?
- False positives: $h(A) = h(B)$ but $d(A, B) > t$ for some t
- False negatives: $h(A) \neq h(B)$ but $d(A, B) < t$ for some t
- Let they also be: statistically independent, effective (i.e. cheap and combinable)

Amplification of locality sensitive functions – AND constructions

- Suppose we are given H, a family of (d_1, d_2, p_1, p_2)-sensitive functions
- the family of functions H' is going to be (d_1, d_2, p_1^s, p_2^s)-sensitive in case it holds for all $h' \in H'$ that
  - $h'(A) = h'(B) \Leftrightarrow \forall_{1 \leq i \leq s} h_i(A) = h_i(B) : h_i \in H$

Amplification of locality sensitive functions – OR constructions

- Suppose we are given H, a family of (d_1, d_2, p_1, p_2)-sensitive functions
- the family of functions H' is going to be $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive in case it holds for all $h' \in H'$ that
    - $h'(A) = h'(B) \Leftrightarrow \exists_{1 \leq i \leq b} h_i(A) = h_i(B) : h_i \in H$

Amplification of locality sensitive functions – Combining AND and OR constructions

- Regard minhash function as $(0.2, 0.6, 0.8, 0.4)$-sensitive
- $1-(1-p^4)^4$: apply AND$_{r=4}$ construction followed by OR$_{b=4}$
- $(1-(1-p)^4)^4$: apply OR$_{b=4}$ construction followed by AND$_{r=4}$

d	p	$1-(1-p^4)^4$	$(1-(1-p)^4)^4$
0.9	0.1	0.00039	0.01399
0.8	0.2	0.00638	0.12150
0.7	0.3	0.03201	0.33345
0.6	0.4	0.09854	0.57395
0.5	0.5	0.22752	0.77248
0.4	0.6	0.42605	0.90147
0.3	0.7	0.66655	0.96799
0.2	0.8	0.87850	0.99362
0.1	0.9	0.98601	0.99960

LSH families for cosine distance

- Project vectors being checked for similarity to randomly chosen hyperplanes (let $\boldsymbol{s_i}$ denote the normal vector to them)
  - Let sketch hash functions be defined as $h_{\boldsymbol{s_i}}(\boldsymbol{a}) = sign(\boldsymbol{s_i}^\intercal \boldsymbol{a})$
- $h_{\boldsymbol{s_i}}(\boldsymbol{a}) = h_{\boldsymbol{s_i}}(\boldsymbol{b}) \Leftrightarrow \boldsymbol{a}$ and $\boldsymbol{b}$ are located in the same halfspace defined by the hyperplane selected randomly (i.e. the scalar products of both vectors and the vector being normal to the randomly chosen hyperplane have the same sign)
- $P(h_{\boldsymbol{s_i}}(\boldsymbol{a}) = h_{\boldsymbol{s_i}}(\boldsymbol{b})) = 1 - \frac{d(\boldsymbol{a},\boldsymbol{b})}{180}$
- Random projections define a $(d_1, d_2, (180 - d_1)/180, d_2/180)$-sensitive family for the cosine distance

Heuristic approaches

- Motivation: false negatives might be untolerable in certain applications
- Idea: do not compare those pairs (x_i, x_j) for which $P(s(x_i, x_j) \geq J) = 0$ for some threshold J
- Assumption: the object to be compared are string comprising of sorted characters without any repetition
  - Sorted: $\forall a, b$ character their order should be the same in any representation
  - Not real restrictions only makes discussion more convenient
- Heuristics
  - **Length based filtering**
  - Prefix indexing, using position information, ...

Length based filtering

- Suppose objects are ordered based on their string representations such that $L_s < L_t$
- The overlap between the two string is $\leq L_s \Rightarrow s_{Jacc}(s,t) \leq \frac{L_s}{L_t}$
- Given that we are only interested in pair (s,t) such that $s_{Jacc}(s,t) \geq J$ then $\frac{L_s}{L_t} \geq J \Leftrightarrow L_t \leq \frac{L_s}{J}$ is a necessity condition

Prefix indexing

- Let index $\forall$ object by the characters in their p-character prefixes
- p should be chosen such that $sim(s,t) \geq J \Rightarrow p_s \cap p_t \neq \emptyset$
  - Suppose $sim(s,t) \geq J$ yet $p_s \cap p_t = \emptyset$ $\Rightarrow$ maximum similarity possible $\left(\frac{L_s - p_s}{L_s}\right)$ (iff the suffix of t matches the $L_s - p_s$ long suffix of string s)
  - It is useless to compare s and t given that $J > \frac{L_s - p_s}{L_s}$ holds
- For some string s of length L_s let $p_s > \lfloor (1-J)L_s \rfloor$ $(p_s = \lfloor (1-J)L_s \rfloor + 1)$
- It might be a good idea to use some ordering of representations other than lexicographic ordering. But how?
  - Order characters according to their increasing frequency

Prefix indexing – example

- Assume $J = 0.9$ and
  $s = bcdefghij \Rightarrow L_s = 9 \Rightarrow p_s = \lfloor 0, 9 \rfloor + 1 \Rightarrow$ 'b' gets indexed
- Assume that the representation of t does not start with character b and
  - $sim(s,t) \geq 0.9 \Rightarrow t = abcdefghij \Rightarrow L_t = 10 \Rightarrow$
    $p_t = \lfloor (1 - 0.9) * 10 \rfloor + 1 = 2 \Rightarrow$ 'a', 'b' are indexed
    $\Rightarrow p_s \cap p_t \neq \emptyset$
  - starts with a character "larger" than 'b'
    - Then $p_s \cap p_t = \emptyset \Rightarrow sim(s,t) < J$
    - Indeed, even if $t = cdefghij$, we have $sim(s,t) = \frac{8}{9} < 0.9$

Using position information

- Motivation: prefix indexing can be too permissive (not strict enough)
  - Suppose $s = acdefghijk$, $t = bcdefghijk$, $J = 0.9$
    - $\Rightarrow p_s = ac$ and $p_t = bc$ (because of $L_s = L_t = 10$)
    - Then $p_s \cap p_t \neq \emptyset$, yet $sim(s,t) = 9/11 < J$
- Idea: perform indexing not only by characters but according to (character, position) tuples

Bloom filters

- Probabilistic data structure (wrt. `contains(key)` operation)
- Implemented using hash functions and a bit vector
  - If some object x gets hashed to a **non-empty** bucket, it is **possible** that x is in the set
  - If some object x gets hashed to an **empty** bucket, it is **sure** that x is **not** in the set
  - False negative rate is 0, however, we can return false positive answers
- Characterized by the length of the bit vector (n), the number of hash functions (k)
- Number of elements stored (m) also affects false positive rate
- Links Bloom filter demo and Guava API

Analyzing bloom filters

- Darts analogy: we throw a dart r times into q targets
  - $P(\text{miss a target}) = 1 - \frac{1}{q}$
  - $P(\text{miss a target r times}) = \left(1 - \frac{1}{q}\right)^r = \left(1 - \frac{1}{q}\right)^{q\frac{r}{q}} \approx e^{-\frac{r}{q}}$
  - $P(\text{some target gets a hit}) = 1 - e^{-\frac{r}{q}}$
- What are q and r? $q = n, r = m$
- What if we are allowed to throw with a dart multiple times? $q = n, r = k * m$
- $P(\text{false positive}) = (1 - e^{-\frac{k*m}{n}})^k$

Bloom filters – example

- Suppose we have 10^9 objects and a bit vector of length $8 * 10^9$
- $P(\text{false positive}) = (1 - e^{-\frac{1}{8}}) \approx 0.1175$
- Suppose we apply two hash functions per objects
- $P(\text{false positive}) = (1 - e^{-\frac{2}{8}})^2 \approx 0.0493$