# SDS 385: Stat Models for Big Data <br> Lecture 7: Nearest neighbor methods 

Purnamrita Sarkar
Department of Statistics and Data Science
The University of Texas at Austin
https://psarkar.github.io/teaching

## Nearest neighbor queries

- Many applications need efficient nearest neighbor search
- It can be kernel regression
- Matching and retrieval
- Kernel density estimation


## A concrete example: Min hash

- Lets start with a simple setting.
- You have documents which can be represented by sets of words, or shingles, which are none other than moving window of words.
- If a document is 'This is Stat models for Big data', then 2-singles are \{ 'This is', 'is Stat', 'Stat models'\} etc.
- The goal is to remove duplicate documents.
- For $1 M$ documents, doing all pairs of similarity would take about 5 days.


## Jaccard similarity

- Consider two sets $S_{1}, S_{2}$
- A common similarity measure is the Jaccard index:

$$
J\left(S_{1}, S_{2}\right)=\frac{\left|S_{1} \cap S_{2}\right|}{\left|S_{1} \cup S_{2}\right|}
$$

- Consider the binary representation of two sets $S_{1}=10111$ and $S_{2}=10011$
- $\left|S_{1} \cap S_{2}\right|=3$
- $\left|S_{1} \cup S_{2}\right|=4$
- Jaccard score 3/4


## Hashing: main idea

- Goal: find a hash function $h($.$) such that$
- If $\operatorname{sim}\left(C_{1}, C_{2}\right)$ is high, then w.h.p $h\left(C_{1}\right)=h\left(C_{2}\right)$
- If $\operatorname{sim}\left(C_{1}, C_{2}\right)$ is low, then w.h.p $h\left(C_{1}\right) \neq h\left(C_{2}\right)$
- Not all similarity functions allow such a hash function
- For the Jaccard score however, such a function does exist.


## Min Hashing

- Write the document dataset as a binary matrix of shingles by documents
- Consider a permutation $\pi$ of the elements, or the words, or the shingles or the rows
- $h_{\pi}(C)$ is the index of the first (in the permuted order $\pi$ ) row in which column $C$ has value 1 .
- In other words:

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

- Use many hash functions (i.e. via random permutations) to create a signature of the columns


## Example

Permutation $\pi \quad$ Input matrix (Shingles $x$ Documents)
Signature matrix $M$

| 2 | 4 | 3 |
| :--- | :--- | :--- | :--- |
| 3 | 2 | 4 |
| 7 | 1 | 7 |
| 6 | 3 | 2 |
| 1 | 6 | 6 |
| 5 | 7 | 1 |
| 4 | 5 | 5 |


| 1 | 0 | 1 | 0 |
| :--- | :--- | :--- | :--- |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |


| $\qquad$2 1 2 1 <br> 2 1 4 1 <br>  1 2 1 2   <br> Similarities:    <br> Col/Col $1-3$ $2-4$ $1-2$ |
| :--- |
| 0.75 0.75 0 0 <br> Sig/Sig 0.67 1.00 0 0 |

## Key claim

- $P\left(h_{\pi}\left(C_{1}\right)=h_{\pi}\left(C_{2}\right)\right)=J\left(C_{1}, C_{2}\right)$
- Consider a document $X$ and let $y \in X$ be an element of it.

$$
P\left(\pi(y)=h_{\pi}(X)\right)=1 /|X|
$$

- Since it is equally likely for any element to become the smallest element under a random permutation
- For $C_{1}, C_{2}$ the probability that some element $y \in C_{1} \cup C_{2}$ is the min-hash is $1 /\left|C_{1} \cup C_{2}\right|$
- The probability that the two min-hashes are the same is the same as the probability that one of the elements in the intersection is the min-hash, i.e. the probability becomes $\left|C_{1} \cap C_{2}\right| /\left|C_{1} \cup C_{2}\right|$


## Key claim

- The hash function only returns 1 or 0 not a number in $[0,1]$
- Thats why you need multiple hash functions and take the average
- For 100 random permutations, each document is now represented as a vector in 100 dimensions, so we have compressed the original long vectors intro short signatures while not losing the signal, which is the similarity between documents in this case


## Min hashing

- Permuting rows is prohibitive.
- You can use approximate linear permutation hashing.
- $h(x ; a, b)=((a x+b) \bmod p) \bmod n$ where $a, b$ are random integers and $p$ is some prime number larger than $n$.


## Min hashing

- Permuting rows is prohibitive.
- You can use approximate linear permutation hashing.
- $h(x ; a, b)=((a x+b) \bmod p) \bmod n$ where $a, b$ are random integers and $p$ is some prime number larger than $n$.


## Efficient Min hashing algorithm

- Construct $n$ hash functions $h_{1}, \ldots, h_{n}$ Set $S(i, c)=\infty$ for $i=1: n, c=1: C$
- For each row, $r \in\{1 \ldots N\}$ of the characteristic matrix,
- For each document/column $c$,
- If column $c$ has 0 in row $r$, do nothing
- Otherwise, for each $i=1 \ldots n$, let $S(i, c) \leftarrow \min \left(S(i, c), h_{i}(r)\right)$


## Acknowledgment

Ullman's lecture notes from "Mining of Massive Datasets"

