CS 5968/6969: Data Str and Alg Scalable Comp	Spring 2023
Lecture 16 — May 20, 2017	
Prof. Prashant Pandey	*Scribe: Manya Bajaj*

1 Overview

In this lecture, we revisited locality sensitive hashing covered in a guest lecture.

2 Review of Hashing

2.1 Classical Hashing

- if $x = y$, then $h(x) = h(y)$
- if $x \neq y$, then $h(x) \neq h(y)$

2.2 Universal Hashing

- Collisions are as rare as possible.

- $\forall x, y \in U$ if $x \neq y$, then
$$\mathop{\mathbb{P}}_{h \in H}[h(x) = h(y)] = \frac{1}{|T|}$$
where $|T|$ is the size of the set of items to hash.

3 Locality Sensitive Hashing

- Idea: Collisions between *similar* items
- $\forall x, y \in U$
$$E_d(x, y) \leq d_1 \Rightarrow \mathop{\mathbb{P}}_{h \in H}[h(x) = h(y)] \geq P_1$$
$$E_d(x, y) \geq d_2 \Rightarrow \mathop{\mathbb{P}}_{h \in H}[h(x) = h(y)] \leq P_2$$
- This represents a family of hash functions where similar elements are more likely to have the same value as compared to dissimilar or distant elements. Hence, low distance corresponds to a high number of collisions, and a high distance corresponds to a low number of collisions.
- There is no guarantee in between d_1 and d_2.
- The above scenario is for a gapped LSH. In case of an ungapped LSH, $d_1 = d_2$.

- Note: The probability is computed over the choice of any hash function $h \in H$, and not over the elements x, y.

3.1 Notation of similarity: Jaccard Index

For any two sets S_1 and S_2, the Jaccard Index is the ratio of the cardinality of intersection of the two sets to the cardinality of union of the two sets:

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Example: $S_1 = \{3, 10, 15, 19\}$, $S_2 = \{4, 10, 15\}$

$$J(S_1, S_2) = \frac{|\{10, 15\}|}{|\{3, 4, 10, 15, 19\}|} = \frac{2}{5}$$

3.2 Minwise Hashing

Idea: From a random universal hash function: $U_i : string \rightarrow N$, take the minimum hash value. Every time we want to generate a new minhash value, we will generate a new universal hash function and compute the minimum hash value.
For any two sets, S_1 and S_2, the probability of hash collision is equal to the Jaccard Index:

$$P(minhash(S_1) = minhash(S_2)) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

- The probability of collisions increases as the similarity between the two sets increases.
- Ungapped LSH $\rightarrow$ unbiased estimator

Intuition: After hashing every element of $S_1 \cup S_2$ and finding the minimum values, the chance of having the same minimum value is equal to the number of common elements proportional to the union. If the minimum belongs to $S_1 \cap S_2$, then the minhash of both the sets will be the same.

3.3 Weighted Jaccard

$$WeightedJaccard(S_1, S_2) = \frac{\sum_i min(S_1^i, S_2^i)}{\sum_i max(S_1^i, S_2^i)}$$

3.4 Parity of Minhash

Idea: Only store the parity of minhash.

$$P(parity(m(S_1) = parity(m(S_2))) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} + 0.5(1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|})$$

3.5 Extension of Minwise Hashing

Introduce One-Permuation Hsashing to replace k-permutations hashing. This significantly reduces the pre-processing cost.
$U_i : string \rightarrow N$

- Divide the space [0, N] into k partitions and take the minimum from each partition to create the sketch.
- If the partition is empty, introduce a scheme to borrow from the next non-empty partition in an order.

References

[1] Ping Li, Art Owen, Cun-Hui Zhang. One Permutation Hashing for Efficient Search and Learning., 2012.