CS 7280: Data Str & Alg Scalable Comp Spring 2025

Lecture 17 — March 17, 2025

Prof. Prashant Pandey *Scribe: Irakli Gurgenidze*

1 Overview

In the last lecture, we began our study of Nearest Neighbors, and ϵ-Nearest Neighbors as techniques to determine and approximate elements of a set similar to an arbitrary key. Deterministic queries with keys of very large dimensions can be extremely costly, and by compromising on search accuracy we can drastically reduce computational overhead. Finally, we reviewed fundamental properties of hashing, and discussed the merits of a good hash function vs. a poor one.

2 Locality Sensitive Hashing

2.1 Hashing Recap

To begin, let's quickly recap classical hashing.

Classical Hashing

- If $x = y$, then $h(x) = h(y)$
- If $x \neq y$, then $h(x) \neq h(y)$
- U : universe
- T : hashtable
- $|T| <<< U$
- $h : U \to [0, |T| - 1]$

A **Universal Family of Hash Functions** H is a group of hash functions such that

- Collisions are as rare as possible
- $\forall x, y \in U, x \neq y,\ Pr_{h \in H}[h(x) = (y)] = \frac{1}{|T|}$

Hashing is useful because it allows us to map items drawn from a large universe to a much smaller one.

Key Takeaway: hashed values are still ordered. You can think of a hash function as a random permutation from an input space to an output space.

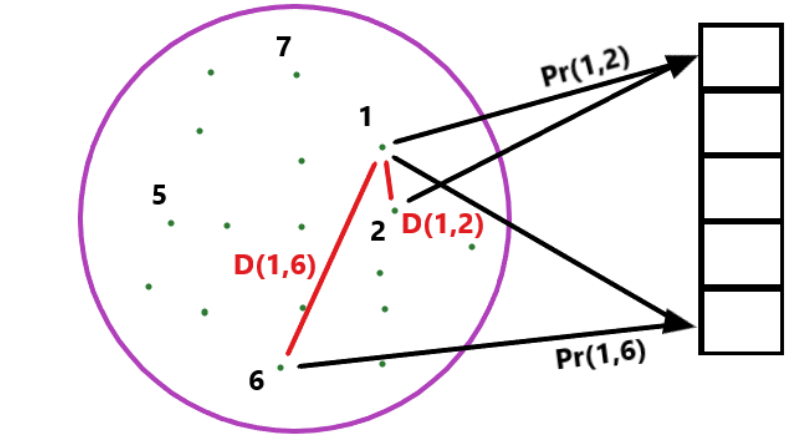
2.2 Locality Sensitive Hashing (LSH)

In classical hashing, we want to avoid collisions at all costs. With LSH, our goal is the opposite - we want similar items to hash to the same value.

LSH

- High probability of collision between similar items
- $\forall x, y \in U$, we have two guarantees:
  $E_d(x, y) \leq d_1 \rightarrow Pr_{h \in H}[h(x) = h(y)] \geq P_1$
  $E_d(x, y) \geq d_2 \rightarrow Pr_{h \in H}[h(x) = h(y)] \leq P_2$

For a visual representation:



Since $D(1, 2) < D(1, 6)$, $Pr(1, 2) >> Pr(1, 6)$. In other words, Points 1 and 2 are far more likely to be hashed together than points 1 and 6. When $d_1 = d_2$, we use the term **ungapped LSH**. When there is a gap between d_1 and d_2, we call this **gapped LSH**.

A family of hash functions where similar elements are more likely to have the same hash value than distinct elements share the following characteristics:

- Low distance $\longleftrightarrow$ high chance of collision
- High distance $\longleftrightarrow$ low chance of collision
- For distances D that fall between d_1, d_2, we hold no guarantees

2.3 Notion of Similarity

For reference, for two sets S_1 and S_2, we define **Jaccard Similarity** as

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

The distance between two sets is defined as $D(S_1, S_2) = 1 - J(S_1, S_2)$.

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

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

We will use the model of Jaccard Similarity as a distance function between two sets to frame further discussion.

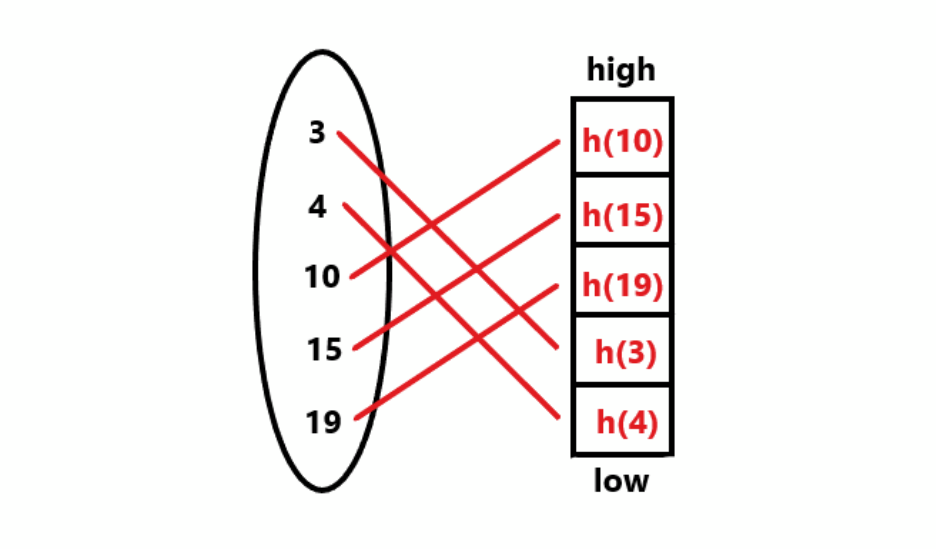
3 Building a LSH

3.1 Minwise Hashing [Andrei Broder 1997]

Let us take a random universal hash of a set of elements S. The **Minhash** of S, $M(S)$, is the value of the smallest hash of each element in S.

- We take the minimum hash value
- For any sets of objects S_1 and S_2, the probability of hash collision is exactly equal to the Jaccard Similarity
- That is, $Pr[\text{Minhash}(S_1) = \text{Minhash}(S_2)] = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$

Example Using sets S_1 and S_2 from earlier:



We're using a universal hash function. Therefore,

$Pr(\text{any item in } S_1 \cup S_2 \text{ is the smallest hash value}) = \frac{1}{|S_1 \cup S_2|} = \frac{1}{5}$
$Pr(\text{Minhash}(S_1) = \text{Minhash}(S_2)) = \sum_{|S_1 \cap S_2|} Pr(\text{any item is the smallest hash value})$

Intuitively, this makes sense. Any item in $S_1 \cup S_2$ can hash to the smallest value in the set. However, for the Minhash of both sets to be identical, the minimum hash must also be an element from $S_1 \cap S_2$.

To increase our confidence in this technique? We can repeat it an arbitrary number of times. This reduces the variance in our random hashes.

- Every time we want to generate a new Minhash value, we will generate a new hash function and compute the minimum hash
- If the similarity of 2 sets increases, then the probability of collision also increases

Example k = 50 We take a sketch of minhash values, and observe that $Pr(\text{Minhash}(S_1) = \text{Minhash}(S_2))$ follows a Bernoulli distribution.

A Bernoulli Random variable can be modeled as

$$X \begin{cases} 1 & Pr(J(S_1, S_2)) \\ 0 & Pr(1 - J(S_1, S_2)) \end{cases}$$

with variance$(X) = J(1 - J)$

3.2 Parity of Minhash

We define the parity of a Minhash value to be 0 if the value is even, and 1 if otherwise. To save space on small systems where storage is a premium, we might adopt this approach.

$Pr(\text{parity}(\text{Minhash}(S_1)) = \text{parity}(\text{Minhash}(S_2))) = J + (1 - J) \cdot \frac{1}{2} = \frac{1}{2}(1 + J)$

References

[1] Andrei Broder. AOn the Resemblance and Containment of Documents. *Proceedings. Compression and Complexity of Sequences 1997*, DOI 10.1109/6669000, 1997.