The previous lectures looked at hashing as a technique to build data structures that enable quick insertion, lookup and deletion; ideally constant time. We explored variants of the basic hashing algorithm. Notice that, the design goal in hashing was to hash the items as uniformly as possible. In other words, we wanted to minimize the number of collisions, leading to the desired computational effort (with high probability).

Today we look at the nearest neighbor problem in high dimensions and see how ideas from hashing can lead to very useful practical results. The nearest neighbor problem is defined as follows, given a collection of n points in $\mathbb{R}^d$, build a data structure which, given any query point, reports the data point that is closest to the query. This problem comes up in many areas, including machine learning, pattern recognition, databases among others. The naive algorithm would be very inefficient and would take linear time (in n). Several efficient algorithms are known when the dimension d is low. However, traditional solutions suffer from either space or query time exponential in d. This is usually referred to as the "curse of dimensionality". Researchers have come up with techniques to get approximate answers. It turns out in practice, approximate nearest neighbors is as good as the exact one. The main idea behind the techniques we discuss today is the following: hash items that are "similar" to the same bucket. Use this information to then return the nearest neighbors. In particular we look at Min-hash to measure similarity and then go on to Locality Sensitive Hashing to solve the approximate nearest neighbor problem.

1 Model

Througout the lecture we will be talking about the representing a collection of n documents $\mathcal{D}$, and performing query operations on the collection. The documents are represented using the *"Bag of Words"* model. A document is represented using a bit vector corresponding to a dictionary of words. If W is the set of words in the language, a document d is represented as a bit string with each position being 1 if the corresponding word is present in the document. Note that, we lose out a lot of structural information about the document in this representation. It is remarkable that this seemingly simplistic representation is still able to help answer the queries on documents (which to a certain extent is subjective).

Formally, $d \subseteq \{0,1\}^M$, where $|W| = M$ the number of words in the dictionary. We also want to define a measure of similarity between two documents d_1 and d_2. There are multiple ways of specifying this similarity. We consider Jaccard's index of similarity and later we will consider Hamming distance.

The Jaccard's index of similarity between d_1 and d_2 is given by,

$$J(d_1, d_2) = \frac{|d_1 \cap d_2|}{|d_1 \cup d_2|}$$

The similarity index $J(d_1, d_2) \in [0, 1]$, similarity index being 0 if the documents are totally disimmilar and 1 if they are similar.

Our goal is to design a data structure,

- To store n documents, (using some preprocessing)
- Query for similarity given input document d.
  - Yes, if there exists a document $\widetilde{d}$ such that, $J(d, \widetilde{d}) > 0.9$
  - No, if for all documents $\widetilde{d} \in \mathcal{D}$, $J(d, \widetilde{d}) < 0.5$.

Note that answers to the query are guaranteed with high probability.

We will modify these conditions for query outputs appropriately when we talk about Locality Sensitive Hashing.

2 Min-hash for estimating similarity

Minhash is a technique for quickly measuring similarity between two documents (sets in general). The minhash scheme was invented by Andrei Broder [Bro97]. Consider a hash function $h : W \to [m]$. Note that this function hashes the words and not the documents. We will define the specific nature of this function h in next lemma. We define $h^{min}(d)$ to be the hash for the document d.

$$h^{min}(d) = \min_{w \in d} h(w)$$

Lemma 4.1. *Let $\mathcal{H}$ be a family of hash functions consiting the set of all permutations of W. Given documents d_1 and d_2, we have*

$$\mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{H}}[h^{min}(d_1) = h^{min}(d_2)] = J(d_1, d_2)$$

Proof. We begin the proof by defining the notion of min-wise independence.

$$\forall X \subseteq W, \forall x \in X$$

$$\mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{H}}[h(x) \text{ is the smallest among } x \in X] = \frac{1}{|X|}$$

It is easy to verify that $\mathcal{H}$ is a min-wise independent family.

Now consider, $\mathbf{Pr}_{h \leftarrow \mathcal{H}}[h^{min}(d_1) = h^{min}(d_2)]$. This collision can happen only if any word the lies in both the documents is the one that leads to h^{min}. However notice that $h^{min}(d_1), h^{min}(d_2)$ can be any one of the words from $|d_1 \cup d_2|$. Combining these arguments we have,

$$\mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{H}}[h^{min}(d_1) = h^{min}(d_2)] = J(d_1, d_2)$$

□

We will now apply this in a concrete setting.

Preprocessing

- Pick k hash functions $h_1, h_2, \ldots, h_k$ and store them.
- For all $d \in \mathcal{D}$, store $h_1^{min}(d), h_2^{min}(d), \ldots h_k^{min}(d)$.

Given an input document d,

- Compute $h_j^{min}(d)$, for $1 \leq j \leq k$
- Go over all the documents
  - Yes, if atleast $0.9k$ of h^{min} collide with d.
  - No, otherwise

Value of k.
Case 1
Case 1 refers to the scenario that there exists a document $\widetilde{d} \in \mathcal{D}$ such that $J(d, \widetilde{d}) > 0.9$. Let us fix $\widetilde{d}$ such that $J(d, \widetilde{d}) > 0.9$.

$$\underset{h \leftarrow \mathcal{H}}{\mathbf{Pr}}[h^{min}(d) = h^{min}(\widetilde{d})] \geq 0.9$$

Define a random variable Y_j such that,

$$Y_j = \begin{cases} 1 & \text{if } h_j^{min}(d) = h_j^{min}(\widetilde{d}) \\ 0 & \text{otherwise} \end{cases}$$

We now have, $\mathbb{E}[Y_j] \geq 0.9$.

$$\underset{h \leftarrow \mathcal{H}}{\mathbf{Pr}}\left[\frac{\sum Y_j}{k} \leq 0.9 - \epsilon\right] \leq e^{-\epsilon^2(0.9)k} < 10^{-5} \text{ say}$$

$$k \geq \frac{5 \log_e 10}{0.9\epsilon^2}$$

Case 2
Fix $\widetilde{d}$ such that $J(d, \widetilde{d}) < 0.5$.

$$\underset{h \leftarrow \mathcal{H}}{\mathbf{Pr}}[h^{min}(d) = h^{min}(\widetilde{d})] \leq 0.5$$

Define a random variable Y_j such that,

$$Y_j = \begin{cases} 1 & \text{if } h_j^{min}(d) = h_j^{min}(\widetilde{d}) \\ 0 & \text{otherwise} \end{cases}$$

We now have, $\mathbb{E}[Y_j] \geq 0.9$.

$$\underset{h \leftarrow \mathcal{H}}{\mathbf{Pr}}\left[\frac{\sum Y_j}{k} \geq 0.5 + \epsilon\right] \leq e^{-\epsilon^2(0.5)k} < 10^{-5} \text{ say}$$

$$k \geq \frac{5 \log_e 10}{0.5\epsilon^2}$$

Fix a k that satisfies conditions from both Case 1 and Case 2.

Recall that, M is the size of the dictionary and n is the number of documents.
Space complexity: $n \log M + O(M \log M)$
Query time: $O(n)$
Known result There is no min-wise hash function such that size $< O(M)$.

3 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) gives up on Jaccard as a measure of similarity. We use Hamming distances instead. LSH was proposed by Piotr Indyk and Rajeev Motwani[IM98].

3.1 Model

We will be using the Bag of Words model as we did for Minhash. However, we will use Hamming distance denoted by $\Delta(d_1, d_2)$ rather than, $1 - J(d_1, d_2)$ as the distance measure. Hamming distance is defined between two bitvectors d_1 and d_2.

$$\Delta(d_1, d_2) = |\{w \in W | w \in d_1, w \notin d_2 \text{ or } , w \notin d_1, w \in d_2\}|$$

Informally, the Hamming distance measures the number of words at which the two documents differ.

Note that the Hamming distance forms a metric. We can verify that

- $\Delta(x, x) = 0$
- $\Delta(x, y) \geq 0$
- $\Delta(x, y) \leq \Delta(x, z) + \Delta(z, y)$

Consider the notion of a Hamming Ball. A Hamming Ball $B(q, r)$ centered at the query point q with radius r is given by $B(q, r) = \{p : \Delta(q, p) \leq r\}$.
We will rewrite our original goals to account for the fact that Hamming distance is a measure of distance and not similarity. On a query document d, return

- Yes, if there exists a document $\widetilde{d}$ such that $\Delta(\widetilde{d}, d) < R$
- No, if for all documents $\widetilde{d}$, $\Delta(\widetilde{d}, d) > cR, c > 1$

The following figure illustrates the idea regions of interest. Points that lie in within the inner ball are "Yes" instances and the points that lie outside the outer ball are the "No" instances.

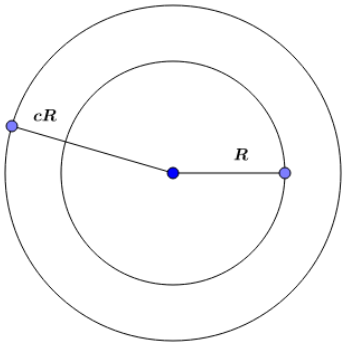

Figure 4.1: Hamming Ball

Our goal is to design a datastructure with $O(n^{1-\epsilon})$ query time and sub-quadratic storage.

Consider the hash family $\mathcal{H} = \{h_i : d \to d_i\}$ for $1 \leq i \leq M$.

Now observe that,

$$\underset{h \leftarrow \mathcal{H}}{\mathbf{Pr}}[h(d_1) = h(d_2)] = 1 - \Delta(d_1, d_2)$$

Note the slight abuse of notation here $\Delta(d_1, d_2)$ is the normalized hamming distance.

3.2 Hash Table Construction

Parameters: k, l

$$G = \left\{ (h_1, h_2, \ldots, h_k) \middle| h_i \leftarrow \mathcal{H} \right\}$$

The k hash functions are picked with replacement.

Pick and store $g_1, g_2, \ldots, g_l$ from G. We maintain l hash tables and insert the documents into all tables.

On an query with input d

- Compute $g_1(d), g_2(d), \ldots, g_l(d)$
- Set count $= 0$
- Go over the list for each $g_j(d)$.
  - Collect all the points that mapped to $g_j(d)$.
  - For each of the points, compute the distance from d to it. Increment counter if distance $< cR$.
  - Break on obtaining $10l$ matching points.

Time complexity $O(kl)$.

Analysis
A family $\mathcal{H}$ is called (R, cR, p_1, p_2)-sensitive for any two points $p, q \in \mathbb{R}^d$

- if $||p - q|| \leq R$ then $\mathbf{Pr}_{\mathcal{H}}[h(q) = h(p)] \geq p_1$
- if $||p - q|| \geq cR$ then $\mathbf{Pr}_{\mathcal{H}}[h(q) = h(p)] \leq p_2$

In order for the family $\mathcal{H}$ to be useful, it should satisfy $p_1 > p_2$.

We use the hash family $\mathcal{H}$ which contains all functions that project the input to one of its coordinates.

$$\mathcal{H} = \{h_i : \{0,1\}^M \to \{0,1\} | h_i(p) = p_i\}$$

Notice that $\mathbf{Pr}_{\mathcal{H}}[h(p) = h(q)]$ is equal to the number of positions in which p and q match. Therefore, we have $p_1 = 1 - R/M$ and $p_2 = 1 - cR/M$. Since, we have $c > 1$ notice that $p_1 > p_2$.

Case 1
Document (point) $\widetilde{d}$ that is far, but mapped to the same bucket as d, with probability p_2^k. We set this to be a small number say $p_2^k \leq \frac{1}{n}$.

$$\mathbb{E}[\text{collision from outside the desired interval from } \mathcal{D} \text{ in any one of the } l \text{ hash tables}] \leq 1$$

$$\mathbb{E}[\text{Number of collisions in the } l \text{ tables}] \leq l$$

From Markov's inequality we have,

$$\mathbf{Pr}[\text{Collisions} \geq 10l] \leq \frac{1}{10}$$

Therefore, with a low probability we have too many collisions.

Case 2
If there exists a document $\widetilde{d}$ such that $\Delta(d, \widetilde{d}) \leq R$, then

$$\mathbf{Pr}[g(d) = g(\widetilde{d})] \geq p_1^k$$

From $p_2^k \leq 1/n$ we have, $k \leq \log_{\frac{1}{p_2}} n$.

We then have,

$$p_1^k \geq p_1^{\log_{\frac{1}{p_2}} n} = p_1^{\log_{\frac{1}{p_1}} n \log_{\frac{1}{p_2}} \frac{1}{p_1}} = n^{-\log_{\frac{1}{p_2}} \frac{1}{p_1}}$$

$$\log_{\frac{1}{p_2}} \frac{1}{p_1} = \frac{\log \frac{1}{p_1}}{\log \frac{1}{p_2}} = \frac{\log p_1}{\log p_2}$$

Let us assume that $c = (1 + \epsilon)$ with ϵ close to 0. From our discussion on the hash family we have $p_1 = 1 - R/M$ and $p_2 = 1 - cR/M$. Simplifying the above fraction gives us,

$$\frac{\log p_1}{\log p_2} = \frac{\log(1 - \frac{R}{M})}{\log(1 - \frac{R(1+\epsilon)}{M})} \approx \frac{\frac{R}{M}}{\frac{R(1+\epsilon)}{M}} \approx 1 - \epsilon$$

$$p_1^k \geq n^{-(1-\epsilon)}$$

We set l, the number of hash tables to $l \approx 100n^{1-\epsilon}$. This gives us the desired run time of $O(n^{1-\epsilon})$ and sub-quadratic storage.

More information on LSH can be found the Alexandr Andoni's page. [And]

References

[And] Alexandr Andoni. Locality Sensitive Hashing (LSH) Home Page. http://www.mit.edu/~andoni/LSH/. 3.2

[Bro97] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997. 2

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998. 3