COMS W6998-5: Algorithms through Geometric Lens (Fall'17) Oct 4, 2017

Lecture 9 – Locality Sensitive Hashing (LSH)

Instructors: *Alex Andoni, Ilya Razenshteyn* Scribes: *Rex Lei*

1 Introduction

We start with the ANN problems and consider some data structures to solve them. We formalize the notion of locality sensitive hashing (LSH) and see an instance of it on the unit sphere.

1.1 Review

Problem 1. *(Approximate Nearest Neighbor Search)*
Given a dataset of n points in some space X (ex: $\mathbb{R}^d, \{0,1\}^d, S^{d-1}$, etc.), take queries of the form $q \in X$. The goal is to find a $\hat{p} \in X$ such that $\|q - \hat{p}\| \leq c \cdot \min_{p^* \in X} \|q - p^*\|$.

Problem 2. *(Approximate Near Neighbor Search) Given the same dataset as above, take queries of the form q such that $\exists p^* \in X$ s.t. $\|q - p^*\| \leq r$. The goal is to find a $\hat{p} \in X$ such that $\|q - \hat{p}\| \leq cr$.*

Note that the first problem takes one parameter (c), while the second problem takes two parameters (c, r). Furthermore, note that the norm is arbitrary; it can be ($\|.\|_1, \|.\|_2, \|.\|_3$, etc).

In practice, finding a solution to the second problem is usually good enough to find a solution of the first.

2 Approximate Near/Nearest Neighbor Search in $(\{0,1\}^d, \|.\|_1)$

Consider the space of vectors $\{0,1\}^d$ under the norm $\|.\|_1$ (Hamming distance). In this setting,

Observation 3. *Problem 2 implies Problem 1.*

Proof. (Sketch)
Say we know how to construct data structures to solve Problem 2 (Approximate Near Neighbor Search). We will show how to use these data structures to solve Problem 1 (Approximate Nearest Neighbor Search).

1. Build $d + 1$ data structures for Problem 2 with $r = 0, 1, 2, \ldots, d$ (and the same value for c). Call them $D_0, D_1, \ldots, D_d$.[1]
2. Upon receiving query q, query $D_{r'}(q)$. If it returns/does not return a point, decrease/increase r'. [2]

□

[1] Without loss of generality, we can assume that the data structures return either the correct answer or "I don't know".
[2] As long as we have some kind of promise of success for each data structure D_r, such as a $1 - \frac{1}{10(d+1)}$ probability of success

This reduction shows us we can focus on Problem 2 (for this space/norm). To see a general reduction, check out `https://theoryofcomputing.org/articles/v008a014/`. For more information, see the sources on `https://ilyaraz.org/static/class/materials.html`.

Theorem 4. *For norm $\|.\|_1$ in space $\{0,1\}^d$, there exists a (c,r)-ANN data structure using*

1. *Space: $O(n^{1+1/c} + nd)$*
2. *Query time: $O(n^{1/c}d)$*
3. *Probability of success[3]: .9.*

2.1 Data Structure

Now, we show the existence of one such data structure.

We'll consider the following instance: Our data structure works by restricting each data point to a subset S of the K coordinates. That is, let K (to be determined later) be a number of distinct coordinates $u_1, \ldots, u_K \in [d]$, and let $S = \{u_1, \ldots, u_K\}$, with $|S| = K$. We index data points and queries with respect to S as follows: For a vector $p \in \{0,1\}^d$, restrict p to S (denoted as $p|_S \in \{0,1\}^K$.) On query q, find all the data points that match $q|_S$ exactly and check distances from q to these points. [4]

2.2 Analysis

Let's analyze this data structure.

- space: $O(n(d+1)) = O(nd+n)$ [5]
- query time: $O(d)$ time to locate one bucket, so the total is $O(d \cdot$ number of retrieved 'far' points$)$. [6]

Claim 5. *Let E be the event that a given far point ends up in the query's bucket. Then*

$$\Pr_S[E] \leq \left(1 - \frac{cr}{d}\right)^K$$

Proof. This event occurs when each of the u_i coordinates for the far points is the same as the query. Since the far point differs from the query point on at least cr of the d coordinates, the probability that any individual u_i is not one of the different coordinates is at most $\left(1 - \frac{cr}{d}\right)$. Therefore, the probability that a far point ends up in the query's bucket is at most $\left(1 - \frac{cr}{d}\right)^K$.

□

Corollary 6. $\mathbb{E}[$*number of far points which fall in the query's bin*$] \leq n\left(1 - \frac{cr}{d}\right)^K$. *So, we should choose K to be the minimum number such that $n\left(1 - \frac{cr}{d}\right)^K \leq 1$.*

Corollary 7. *For this K, the query time is $O(d)$.*

[3] One can boost the probability by running multiple times and checking the point returned by the data structure

[4] One can think of this structure like a hash map.

[5] Slightly less than what is promised in the theorem.

[6] Here we define a "far" distance as a distance $> cr$.

Claim 8. *Let F be the event that q and p^* land in the same bucket. Then*

$$\Pr_S[F] \geq \left(1 - \frac{r}{d}\right)^K$$

Proof. Similarly as before, this event only happens if each of the K selected coordinates is a coordinate on which q and p^* do not differ. Since q and p^* differ on at most r coordinates, the probability F occurs is $\frac{d-r}{d}$.

□

Again, we'll take K to be the smallest value such that $\left(1 - \frac{r}{d}\right)^K \leq 1$. This is

$$K = \lceil \frac{\log n}{\log\left(1 - \frac{r}{d}\right)^{-1}} \rceil$$

Therefore,

$$\begin{aligned}
\left(1 - \frac{r}{d}\right)^K &= \left(1 - \frac{r}{d}\right)^{\lceil \frac{\log n}{\log\left(1-\frac{cr}{d}\right)^{-1}} \rceil} \\
&\geq \left(1 - \frac{r}{d}\right)\left(1 - \frac{r}{d}\right)^{\frac{\log n}{\log\left(1-\frac{cr}{d}\right)^{-1}}} \\
&= \left(1 - \frac{r}{d}\right) n^{\frac{-\log\left(1-\frac{r}{d}\right)^{-1}}{\log\left(1-\frac{cr}{d}\right)^{-1}}}
\end{aligned} \tag{1}$$

By a Taylor expansion argument, the exponent of N is roughly $\frac{\log\left(1-\frac{r}{d}\right)}{\log\left(1-\frac{cr}{d}\right)} \approx \frac{r/d}{cr/d} = \frac{1}{c}$.

Therefore, $\left(1 - \frac{r}{d}\right)^K \geq \Omega_c(n^{-\rho})$. [7]

3 Locality Sensitive Hashing (LSH)

In fact, the previous section was an instance of locality sensitive hashing.

Definition 9. *(Locality-Sensitive Hashing, LSH)*

Given parameters p_1, p_2, and a $\mathcal{P}$-random partition of a space (ex: $\mathbb{R}^d, \{0,1\}^d, S^{d-1}$, etc.), a locality-sensitive hash satisfies

- *$\forall p, q$ such that $\|p - q\| \leq r$, $\Pr[\mathcal{P}(p) = \mathcal{P}(q)] \geq p_1$ and*
- *$\forall p, q$ such that $\|p - q\| > cr$, $\Pr[\mathcal{P}(p) = \mathcal{P}(q)] \leq p_2$*

This definition allows us to state the result of the previous section as follows:

Claim 10. *For $i \in [d]$-uniform, $\{x|x_i = 0\}, \{x|x_i = 1\}$, we have that $p_1 = 1 - \frac{r}{d}$, and that $p_2 = 1 - \frac{cr}{d}$.* [8]

[7] We'll define ρ in the next section.

[8] The ρ value here is $1/c$, which was proven to be the best possible in a paper.

Theorem 11. *If we have a $\mathcal{P}$-LSH with parameters p_1, p_2, and $\mathcal{P}$ is "efficient", then there exists a $[c, r]$-ANN data structure with*

- *space $O(n^{1+\rho}/p_1 + nd)$*
- *query time: $O(dn^{\rho}/p_1)$*
- *probability: 0.9*

where $\rho = \frac{\log p_1^{-1}}{\log p_2^{-1}}$.

3.1 Modified Data Structure

We can modify the way we indexed our data and queries from our previous construction. Now, we index data points and queries w.r.t a new set S defined as the keys

$$S = \mathcal{P}_1(p), \ldots, \mathcal{P}_K(p))$$

In essence, this is the same algorithm except we abstracted the sense of coordinates into the functions $\mathcal{P}_i$. The rest of the algorithm (i.e. how to handle the queries) is the same before: Find all the data points that match q on S exactly, and check the distances to these points.

3.2 Analysis

The analysis for this new data structure is mostly unchanged.

We wish to use L hash tables in total. In one table,

$$\mathbb{E}[\text{number of far points in the bucket}] \leq n \cdot p_2^K$$

Then we choose K to be the minimum number such that $n \cdot p_2^K \leq 1$, i.e. $K = \lceil \frac{\log n}{\log p_2^{-1}} \rceil$.

The query time is $O(d \cdot L)$, and the probability of success is p_1^K.

$$p_1^K = p_1^{\lceil \frac{\log n}{\log p_2^{-1}} \rceil} \geq p_1 n^{\frac{\log p_1^{-1}}{\log p_2^{-1}}}$$

Therefore, $L = O\left(\frac{n^{\rho}}{p_1}\right)$.

4 LSH for unit sphere $(S^{d-1}, \|.\|_2)$

One LSH example for the unit sphere under the l_2 norm is the "Random Hyperplane LSH."

Randomly sample a vector $u \sim S^{d-1}$. [9] Consider the map f_u that sends vector p to $\text{sgn}\langle p, q \rangle$. The vector u defines a hyperplane normal to u - vectors on one side of the hyperplane get mapped to 1 and vectors on the other side get mapped to -1. [10]

Let α be the angle between vectors p and q. By the rotational symmetry,

[9] Recall from a previous lecture we do this in d dimensions by sampling $g = (g_1, \ldots, g_d)$ where each g_i is an i.i.d. $N(0, 1)$.

[10] This algorithm is actually super useful in practice.

$$\Pr[\mathcal{P}(p) = \mathcal{P}(q)] = 1 - \frac{\alpha}{\pi}$$

The right hand side can be expressed as $1 - \frac{2}{\pi} \arcsin \frac{\|p-q\|_2}{2}$, so

$$\rho = \frac{\log(1 - \frac{2}{\pi} \arcsin \frac{r}{2})^{-1}}{\log(1 - \frac{2}{\pi} \arcsin \frac{cr}{2})^{-1}} \leq \frac{1}{c}$$

[11]

We will see more examples and analysis of LSH in the next lecture.

[11] There's a factor of p_1 in the denominator for the query time, but it's small enough that it can be thought of as part of the constant.