CS265, Fall 2023

Class 8: Agenda and Questions

1 Announcements

- HW3 due tomorrow!
- HW4 out now!

2 Recap and Questions

We'll do a quick recap of the JL lemma and the (approximate) nearest neighbors problem.

3 A better scheme for approximate nearest neighbors, and locality sensitive hashing

[A bit of lecture with setup. Summary below. This is also covered in the lecture notes.]

Recall the setup for c-approximate-nearest neighbors. We have a set S of size n, and **for today $S \subset \mathbb{S}^d$ lives on the surface of the d-dimensional sphere.** That is, $S = \{x_1, \ldots, x_n\}$, so that $x_i \in \mathbb{R}^{d+1}$ and $\|x_i\|_2 = 1$ for all $i \in [n]$.

Our goal is to come up with some data structure to store the x_i's, so that:

- We don't use too much space (ideally, use space poly(n), where the exponent in the polynomial doesn't depend on d).
- Given $y \in \mathbb{S}^d$, we can find $x_i \in S$ so that
$$\|x_i - y\|_2 \leq c \cdot \min_j \|x_j - y\|_2$$
in time sublinear in n.

3.1 Nearest-Neighbors vs. Near Neighbors

[A bit of lecture, summary below and also in the lecture notes]

Consider the following problem, called (r, c)**-near-neighbors.** We have a set $S \subset \mathbb{S}^d$ of size n as before, and our goal is to come up with some data structure (that doesn't use too much space) to store the x_i's, so that the following holds.

> Given $y \in \mathbb{S}^d$ so that $\min_j \|x_j - y\|_2 \leq r$, we can find $x_i \in S$, in sublinear time, so that $\|x_i - y\|_2 \leq cr$.

It turns out that if we can solve (r, c)-near-neighbors (for a decent range of r's) then we can solve c-nearest-neighbors.

3.2 A solution to (r, c)-near-neighbors

[A bit of lecture for setup; summary below and also in the lecture notes.]

Let s, k be parameters, chosen as follows:

$$s = \sqrt{n}, \qquad k = \frac{\pi \log n}{2r}$$

For $i = 1, \ldots, s$, let $A_i \in \mathbb{R}^{k \times d+1}$ have i.i.d. $\mathcal{N}(0, 1)$ entries. For $y \in \mathbb{S}^d$, define

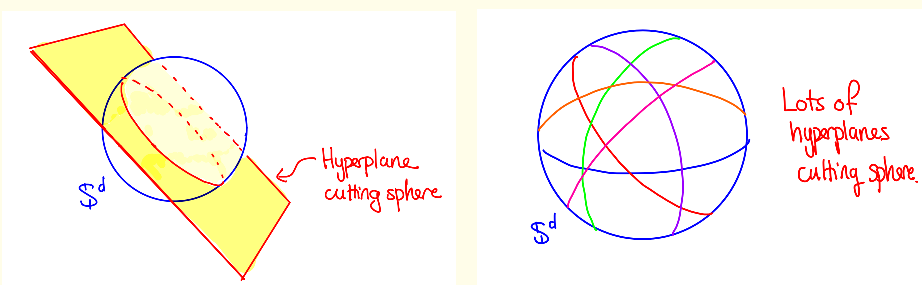
$$h_i(y) = \mathrm{sign}(A_i y),$$

where for a vector $a \in \mathbb{R}^k$, $\mathrm{sign}(a) \in \{\pm 1\}^k$ is just the vector whose i'th entry is $+1$ if $a_i > 0$ and -1 if $a_i \leq 0$.

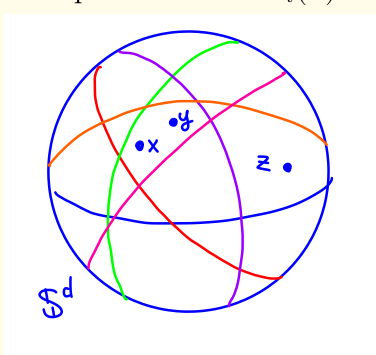
Group Work

1. Consider a hash function $h_i : \mathbb{S}^d \to \{\pm 1\}^k$ as defined above. Explain why "$h_i(x) = h_i(y)$" has the following geometric meaning:

   Imagine choosing k uniformly random hyperplanes in $\mathbb{R}^d$, and using them to slice up the sphere $\mathbb{S}^d$ like this:

   Then $h_i(x) = h_i(y)$ if and only if x and y are in the same "cell" of this slicing. For example, in the picture below $h_i(x) = h_i(y) \neq h_i(z)$.

   Hint: *Use the spherical symmetry of the Gaussian distribution.*

2. Explain why, for $x, y \in \mathbb{S}^d$, and for any $i = 1, \ldots, s$,

$$\Pr[h_i(x) = h_i(y)] = \left(1 - \frac{\text{angle}(x, y)}{\pi}\right)^k,$$

where $\text{angle}(x, y) = \arccos(x^T y)$ is the arc-cosine of the dot product of x and y, aka, the angle between x and y.

***Hint**: Think about the geometric intuition in the plane spanned by x and y.*

3. Suppose that $x, y \in \mathbb{S}^d$. Fill in the blank, using the previous part:

$$\Pr[\forall i \in \{1, \ldots, s\}, h_i(x) \neq h_i(y)] = \_\_\_\_\_$$

(Don't worry about simplifying, you'll do that in the next part).

4. Let $x, y \in \mathbb{S}^d$ and suppose that the angle between x and y is pretty small. Using our choices of s and k above, along with extremely liberal use of the approximation that $1 - x \approx e^{-x}$ for small x, convince yourself that

$$\Pr[\forall i \in \{1, \ldots, s\}, h_i(x) \neq h_i(y)] \approx \exp\left(-n^{1/2 - \text{angle}(x,y)/(2r)}\right).$$

5. Fill in the blanks (assuming that your approximation from the previous step is valid):

   (a) If $\text{angle}(x, y) \leq r$, then

   $$\Pr[\exists i \in \{1, \ldots, s\} \text{ so that } h_i(x) = h_i(y)] \geq \_\_\_\_\_\_$$

   (b) If $\text{angle}(x, y) \geq 5r$, then

   $$\Pr[\exists i \in \{1, \ldots, s\} \text{ so that } h_i(x) = h_i(y)] \leq \_\_\_\_\_\_.$$

Suppose that $\mathcal{H}$ is a family of hash functions $h : \mathbb{S}^d \to \mathcal{D}$. We say that $\mathcal{H}$ is a *locality sensitive hash (LSH) family* (for the Euclidean metric, with some parameters R, C, p_1, p_2) if:

- If $\|x - y\|_2 \leq R$, then $h(x) = h(y)$ with probability at *least* p_1.
- If $\|x - y\|_2 \geq CR$, then $h(x) = h(y)$ with probability at *most* p_2.

Thus, if we pretend that "$\text{angle}(x, y)$" was "$\|x - y\|_2$", we have just shown that the family of random hash functions from which we chose the h_i is a locality-sensitive hash family. (Actually, formally we showed something a bit different, since we looked at the probability of *any* collision over s functions drawn from the family).

In the next two problems, you'll see how to use this LSH family to solve the approximate near-neighbors problem.

Group Work

6. Pretend that "angle(x, y)" was "$\|x - y\|_2$" everywhere.

   Come up with a data structure that uses your result from problem 5b and show that it gives a (c, r)-near-neighbors scheme for some constant c. (It's okay if each query succeeds with probability 1/2 or something like that).

   Hint: *As your data structure, consider storing s hash tables, one for each h_i. Hash each item $x \in S$ into these tables. Given a query y, in what bucket(s) should you look for a close-by $x \in S$?*

7. Explain why it's okay to pretend that "angle(x, y)" is "$\|x - y\|_2$," perhaps at the cost changing the constants around.

   Hint: *You can use the fact that*

$$\frac{2}{\pi}\text{angle}(x, y) \leq \|x - y\|_2 \leq \text{angle}(x, y)$$

   for any $x, y \in \mathbb{S}^d$.

8. **(If you have time)** What is the amount of space that your data structure uses? How much time does a query take?

Group Work

If you finish the rest, here's some bonus stuff to think about!

1. Why does a solution to (r, c)-near-neighbors give a solution to c-approximate-nearest-neighbors?
2. What happens if our data don't live on the surface of $\mathbb{S}^d$? Explain how to still use the analysis above.
3. Can you think of a way to come up with a better LSH family?
4. Can you think of a way to solve approximate near(est) neighbors *without* going through LSH? Is LSH necessary?