CS 388R: Randomized Algorithms, Fall 2021 November 9, 2021

Lecture 21: Locally Sensitive Hashing

Prof. Eric Price *Scribe: Akash Doshi, Jiaxun Cui*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In the last lecture we discussed low dimensional computational geometry, which included constructing the 2D Convex Hull of n points and Planar point location - given n lines in a plane, find which region a point lies in.

In this lecture we will move onto high dimensional computational geometry, by studying randomized algorithms for locally sensitive hashing, first on $\{0,1\}^d$ and in Hamming distance, and then on $\mathbb{R}^d$ with both l_1 and l_2 norm.

2 Introduction

Suppose we have points $p_1, \ldots, p_n \in \mathbb{R}^d$ in high dimensional space. We want to construct a data structure such that for any query $q \in \mathbb{R}^d$, we want to find the nearest p_i to q, i.e. we want to find $\arg\min_i ||p_i - q||$. This is analogous to reverse image search, where one could have a 32 MP camera image and wants to identify the image on the web which is closest to that.

In 2D, this is easy, since we can use the approach for Closest Pair outlined in Lecture 19, by hashing to the grid. In higher dimensions, if I have a query point q in the centre of a ball, with all except one of the p_i on the surface of the ball, it can be hard to find $\arg\min_i ||p_i - q||$.

Hence a simplified goal could be to solve the **approximate nearest neighbour** problem: Find i s.t.

$$||p_i - q|| \leq C \min_j ||p_j - q|| \tag{1}$$

We will instead be solving the **approximate near neighbour** problem: Given r such that $\min_j ||p_j - q|| \leq r$, find i such that

$$||p_i - q|| \leq Cr \tag{2}$$

Note that if I can solve (2), I can solve (1) by finding the smallest r that works in an exponential fashion. Before we describe Locally Sensitive Hashing (LSH), let us consider a few deterministic algorithms and the trade-off they afford between space and query-time

- **Brute force** i.e. Scan at query time (read point, measure distance and repeat). This requires $O(nd)$ space and $O(nd)$ time to complete a query

- $(cr/\sqrt{d})$**-sided cubes** – For every query point, it has to be either in the cube associated with the point p_i or in one of the adjacent $(\sqrt{d}/c)^d$ cubes. This requires $O(n)$ space and $(\sqrt{d}/c)^d$ time to complete a query.
- If all the points are in $\{0,1\}^d$, we can precompute all distances in advance. This would take $2^d \log n$ space (where we have 2^d possible queries and $\log n$ bits to store each answer) and d time to complete a query.

Note: By Johnson-Lindenstrauss lemma [DG03], you can embed $\mathbb{R}^d$ into $\mathbb{R}^m$ for $m = O(\frac{1}{\epsilon^2} \log n)$ and preserve l_2 distances to $(1 \pm \epsilon)$. In other words, the norm of the projection is preserved w.h.p. Hence, WLOG, we can set $d = O(\frac{1}{\epsilon^2} \log n)$ i.e. assume the reduction specified by the Johnson-Lindenstrauss lemma has already been preformed before we attempt to perform LSH.

LSH will get us a space requirement of $n^{1+\rho}$ and a query time requirement of n^ρ for $\rho = 1/c$ under the l_1 metric and $\rho = 1/c^2$ under the l_2 metric. Note that by performing adaptive LSH, we can obtain $\rho = 1/2c$ and $\rho = 1/(2c^2 - 1)$ respectively.

3 Locally Sensitive Hash Function

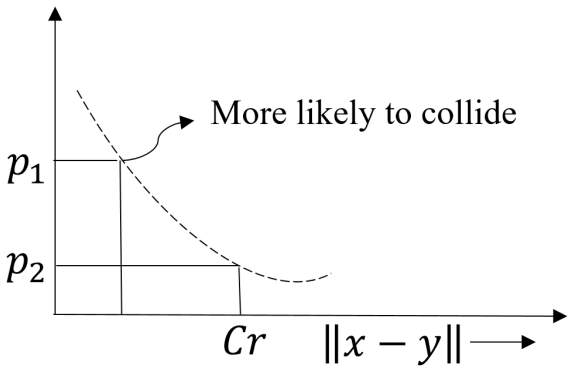


Figure 1: Probability of collision plotted as a function of $||x - y||$

This can be defined as a family of hash functions $h : \mathbb{R}^d \to U$ such that (see Fig. 1)

$$||x - y|| \leq r \implies \Pr[h(x) = h(y)] \geq p_1$$
$$||x - y|| \geq Cr \implies \Pr[h(x) = h(y)] \leq p_2$$

The characteristic efficiency of h is denoted by $\rho = \log_{p_2} p_1$. The intuition behind LSH functions is as follows: we want the probability of collision to be small not for every pair of different things, but only for things that are far apart. If they are close, we want them to collide.

3.1 Constructing LSH for $X = \{0,1\}^d$ and Hamming distance $||.||_H$

Step 1: Pick any h with good ρ. Here we consider picking one random bit.

$$h_i(x) = X_i \qquad i \in [d] \text{ uniform}$$

Then we have

$$\Pr[h(x) = h(y)] = 1 - \frac{||x-y||}{d}$$

This implies $p_1 = 1 - \frac{r}{d}$ and $p_2 = 1 - \frac{Cr}{d}$, which gives $\rho = \log_{p_2} p_1 \geq 1/c$ if r/d is small

The issue here is $r << d$ implies p_1 and p_2 are very high, i.e. almost always collide and we want to drive p_2 down.

Step 2: Repeat to decrease p_1, p_2 for some ρ. $g(x) = (h^1(x), h^2(x), ..., h^k(x))$ for independent hashing functions $h^1(x), h^2(x), ..., h^k(x)$. Then

$$\Pr[g(x) = g(y)] = \prod_i \Pr[h^i(x) = h^i(y)]$$

The above probability has the following properties

$$\begin{cases} \geq p_1^k & \text{if } \|x-y\| \leq r \\ \leq p_2^k & \text{if } \|x-y\| \geq Cr \end{cases}$$

while the ρ keeps the same.

Example

If we set $k = \Theta((\log n)\frac{d}{Cr})$, then g has $p_2 = \frac{1}{n}, p_1 = \frac{1}{n^\rho}$.

Then

$$\begin{aligned} \mathbb{E}[\#\text{ False Positive }] &= \mathbb{E}[\#P_i \text{ s.t } \|P_i - q\| \geq Cr \cap g(P_i) - g(q) = 0] \\ &= p_2 n = 1 \\ \Pr[\#\text{ Find a good point }] &= p_1 = \frac{1}{n^\rho} \end{aligned} \tag{3}$$

For this example, we have $\mathbb{E}[time] = n^\rho$, Space=$n^{1+\rho}$, and a constant success rate.

3.2 ℓ_1 Example

If $X \in [\Delta]^d$ which means it is no longer in $\{0,1\}^d$ space. The ℓ_1 distance between two points are defined as $\|x-y\|_1 = \sum_i |x_i - y_i|$.

Consider the following three points, and clearly y is closer to x in ℓ_1, but it has more different dimensions from x than z:

$$\begin{aligned} x &= (0,0,0,...,0) \\ y &= (\underbrace{1,1,1}_{r},...,0) \\ z &= (Cr,0,0,...,0) \end{aligned}$$

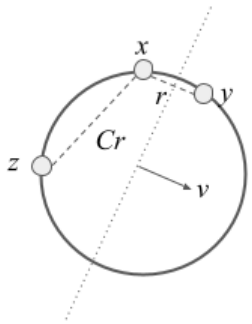
A reasonable construction of LSH is

$$h^i(x) = \left\lfloor \frac{x_i - s_i}{w} \right\rfloor$$

where i is the dimension and s_i is random in $[w]$. If $w \gg Cr$, then $\rho \approx \frac{1}{C}$

3.3 ℓ_2 Example

If $X \in \mathbb{R}^d$ and $\|X\|_2 = 1$.

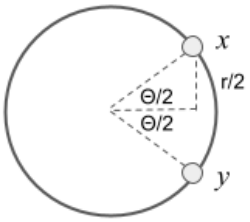


Then an example of construction of LSH is

$$h_v(x) = \text{sign}(< v, x >)$$

where v is a normal vector. The probability that two points collide is

$$\begin{aligned}
\Pr[h(x) = h(y)] &= 1 - \frac{\theta}{\pi} \\
&= 1 - \frac{2\sin^{-1}(\frac{1}{2}\|x-y\|)}{\pi} \\
&\approx 1 - \frac{\|x-y\|}{\pi} \\
&\implies \rho \approx \frac{1}{C}
\end{aligned}$$



3.4 Better Algorithm

Pick $u_1, u_2, ..., u_T \in S^{d-1} = \mathbb{R}^d \cap \|\cdot\|_2 = 1$. Let $h(x) = \arg\min_i \|u_i - x\|$.

- If $T = 2$, then it is the same as ℓ_2 case, $\rho = \frac{1}{C}$
- If $T \gg 2$, then $\rho = \frac{1}{C^2} + o_T(1)$

where $o_T(1) \to 0$, as $T \to \infty$

References

[DG03] S. Dasgupta & A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.