MTH 995-003: Intro to CS and Big Data Spring 2014

Lecture 8 – January 30, 2014

Inst. Mark Iwen *Scribe: Kishavan Bhola*

1 Overview

In this lecture, we begin a probablistic method for approximating the Nearest Neighbor problem by way of a locality sensitive hash function. We compute the two probabilities for the relevant hash function.

2 Problem

Given $r \in \mathbb{R}^+$, $c > 1$, and $\mathbb{X} := \{\vec{x}_1, \ldots, \vec{x}_P\} \subset \mathbb{R}^D$, compute

$$f : [P] \to [P] \cup \{-1\}$$

such that

1. $d(\vec{x}_j, \vec{x}_{f(j)}) \leq c \cdot r$ for all $j \in [P]$ such that $\exists j \neq i \in [P]$ with $d(\vec{x}_j, \vec{x}_i) \leq r$; and
2. $f(j) = -1$ if there does not exist $j \neq i \in [P]$ with $d(\vec{x}_j, \vec{x}_i) \leq c \cdot r$.

Remark 1 The above can easily be generalized to arbitrary metric spaces, where d is the metric, but in the following, we will focus on $\mathbb{R}^D$ with d being the Euclidean 2-norm.

Remark 2 This is known as the (c, r) - Nearest Neighbor Problem. Note that (1) and (2) do not uniquely determine a function, and moreover, it is possible that $\vec{x}_{f(j)}$ is not the nearest neighbor to $\vec{x}_j$ (see Figure 1). Hence, such an f is an approximation to the standard Nearest Neighbor problem.

3 Naive Solution

1. Compute every pairwise distance $\|\vec{x}_j - \vec{x}_i\|_2$, $i \neq j$. This takes $\mathcal{O}(P^2 D)$-time.
2. Output the index of the closest point to each $\vec{x}_j$ as $f(j)$.

Clearly such an f gives an exact solution to the Nearest-Neighbor problem, and thus also satisfies (1) and (2) in the Problem statement. We wish to approximate this Naive solution with better runtime than $\mathcal{O}(P^2 D)$.

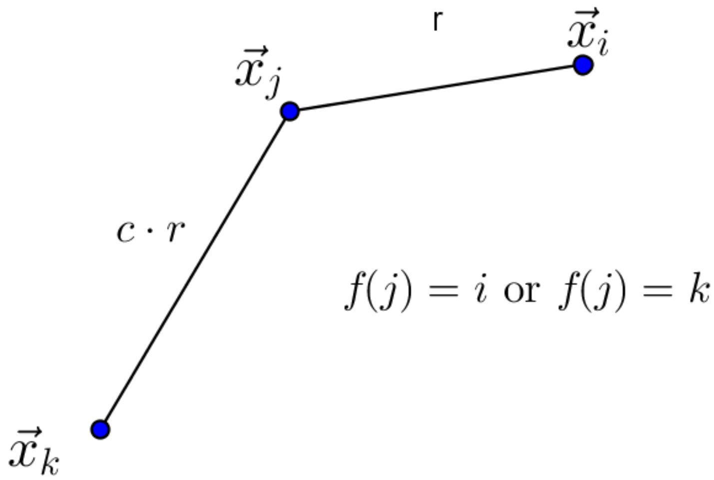

Figure 1: Ambiguity of f

4 Idea

Project $\vec{x}_1, \ldots, \vec{x}_p$ onto a 'random vector' or one dimensional subspace and then see how far their projections are from one another (see Figure 2):

Runtime of Idea

1. Projecting all times is $\mathcal{O}(PD)$-time (just inner products).
2. Finding close projected points is equivalent to sorting a list and thus has time-complexity $\mathcal{O}(P \log(P))$ (using, for example, merge-sort).

Therefore, the total time-complexity is $\mathcal{O}(P(D + \log(P)))$. This is an approximation even to our approximated problem, and so we would like error guarantees.

5 Solution

Definition Call a random function $h : \mathbb{R}^D \to \mathbb{Z}$ a **locality sensitive hash function** if there is $p_1, p_2 \in (0,1)$ with $p_1 > p_2$ and such that the following holds for arbitrary $\vec{x}, \vec{y} \in \mathbb{R}^D$,

(i) $\|\vec{x} - \vec{y}\| < r$ implies $h(\vec{x}) = h(\vec{y})$ with probability at least p_1.

(ii) if $\|\vec{x} - \vec{y}\|_2 > c \cdot r$, then $h(\vec{x}) = h(\vec{y})$ with probability at most p_2.

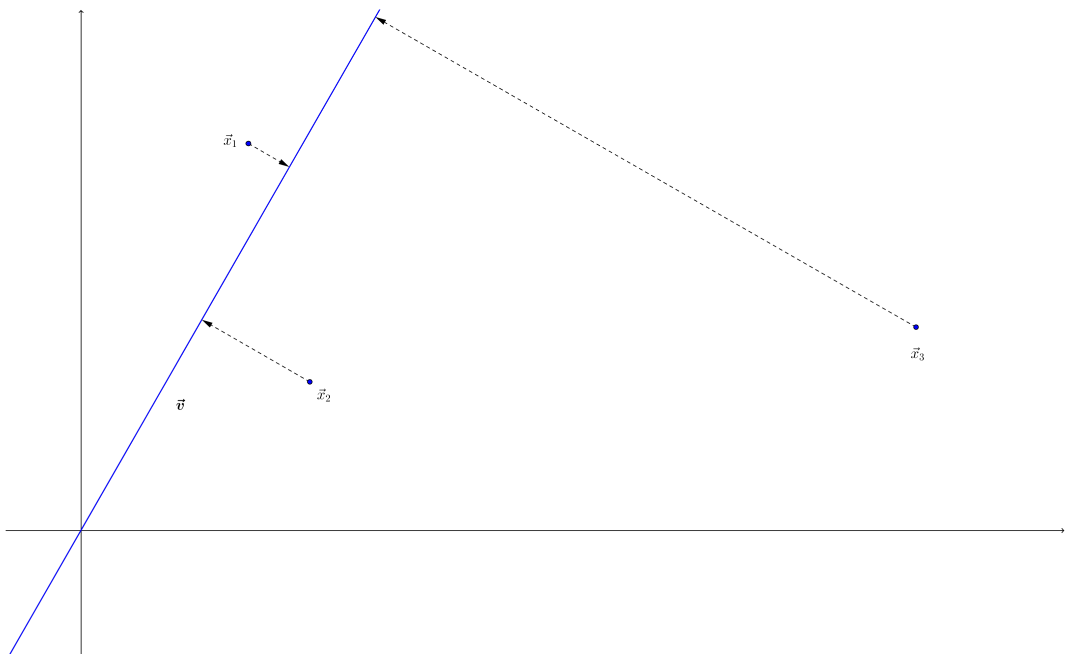

Figure 2: Projection onto 1-Dimensional Subspace

Remark A locality sensitive hash function h sends points that are close to the same integer and sends far points to different integers (this follows from (i) and (ii) and the fact that $p_1 > p_2$).

Now consider the following random function: Pick $w \in \mathbb{R}^+$. Then let $\vec{G} \sim N(\vec{0}, I_{D\times D})$ and $U \sim U([0,w])$. Finally, define $h : \mathbb{R}^{\mathbb{D}} \to \mathbb{Z}$ as

$$h(x) = \left\lfloor \frac{\langle \vec{g}, \vec{x} \rangle + u}{w} \right\rfloor \tag{*}$$

where $U = u$ and $\vec{G} = \vec{g}$ are instances of the random variable and vector defined above.

Remark The above notation means that $\vec{g}$ is a random vector with independent, identically distributed, mean 0, and variance 1, Gaussian entries. Similarly, u is a random uniform variable from the closed interval $[0, w]$.

Theorem 1. *The function h defined by (*) is a locality-sensitive hash function.*

Proof: Let $\vec{x}, \vec{y} \in \mathbb{R}^D$ be arbitrary. Define the following two events A and B,

$$A : h(\vec{x}) = h(\vec{y})$$
$$B : |\langle \vec{g}, \vec{x} - \vec{y} \rangle| < w$$

Note, by the definition of h (*), if A occurs, then B occurs; that is $\mathbb{P}\left[B|A\right] = 1$. Therefore, Bayes' Law simplifies:

$$\mathbb{P}[A] \cdot \mathbb{P}[B|A] = \mathbb{P}[B] \cdot \mathbb{P}[A|B]$$

$$\mathbb{P}[A] = \mathbb{P}[B] \cdot \mathbb{P}[A|B] \tag{1}$$

Then, by using the variable $z := |\langle \vec{g}, \vec{x} - \vec{y} \rangle|$, and considering all possible values of z for which event B is true, we may transform the right-hand side of (I) to an integral. Writing this all out, we get,

$$\mathbb{P}\left[h(\vec{x}) = h(\vec{y})\right] = \int_0^w \mathbb{P}\left[h(\vec{x}) = h(\vec{y}) | z = |\langle \vec{g}, \vec{x} - \vec{y} \rangle|\right] \cdot \mathbb{P}\left[z = |\langle \vec{g}, \vec{x} - \vec{y} \rangle|\right] \mathrm{d}z \tag{2}$$

We now wish to simplify $\mathbb{P}\left[h(\vec{x}) = h(\vec{y}) | z = |\langle \vec{g}, \vec{x} - \vec{y} \rangle|\right]$. One can show that for $0 \leq z \leq w$, we have,

$$\mathbb{P}\left[h(\vec{x}) = h(\vec{y}) | z = |\langle \vec{g}, \vec{x} - \vec{y} \rangle|\right] = \frac{w - z}{w}.$$

This follows from considering the different values of u in (*) that will either (i) shift the integer parts of $\langle \vec{g}, \vec{x} \rangle / w$ and $\langle \vec{g}, \vec{y} \rangle / w$ to be the same when they are different, or (ii) shift them so that they stay the same when they are already the same.

So (2) becomes

$$\begin{aligned}\int_0^w \frac{w - z}{w} \ \mathbb{P}\left[|\langle \vec{g}, \vec{x} - \vec{y} \rangle| = z\right] \mathrm{d}z &= \int_0^w \mathbb{P}\left[|\langle \vec{g}, \vec{x} - \vec{y} \rangle| = z\right] \mathrm{d}z - \int_0^w \frac{z}{w} \ \mathbb{P}\left[|\langle \vec{g}, \vec{x} - \vec{y} \rangle| = z\right] \mathrm{d}z \\ &= \frac{\sqrt{2}}{\|\vec{x} - \vec{y}\| \sqrt{\pi}} \left(\int_0^w \exp\left(\frac{-z^2}{2\|\vec{x} - \vec{y}\|^2} \right) \mathrm{d}z - \int_0^w \frac{z}{w} \exp\left(\frac{-z^2}{2\|\vec{x} - \vec{y}\|^2} \right) \mathrm{d}z \right)\end{aligned}$$

Now writing $n := \|\vec{x} - \vec{y}\|$, using the change of variables $\frac{z}{\sqrt{2}n} \longmapsto z$, and integrating the second integral, we get,

$$\mathbb{P}\left[h(\vec{x}) = h(\vec{y})\right] = \frac{2}{\sqrt{\pi}} \int_0^{\frac{w}{\sqrt{2}n}} \exp(-z^2) \mathrm{d}z + \sqrt{\frac{2}{\pi}} \frac{n}{w} \left[\exp\left(-\left(\frac{w}{\sqrt{2}n} \right)^2 \right) - 1 \right] \tag{3}$$

Define $p_w(n) = \mathbb{P}\left[h(\vec{x}) = h(\vec{y})\right]$. (3) shows that

$$p'_w(n) = \frac{\sqrt{2}}{\sqrt{\pi}} \frac{e^{-\left(\frac{w}{\sqrt{2}n}\right)^2} - 1}{w}$$

so that $p'_w(n) < 0$ for all n (since $n \geq 0$).

Now, let's consider the two cases:

(i) $n \leq r$: this implies

$$p_w(n) \geq \operatorname{erf}\left(\frac{w}{\sqrt{2}r} \right) + \sqrt{\frac{2}{w}} \frac{r}{w} \left(e^{-\left(\frac{w}{\sqrt{2}r}\right)^2} - 1 \right) =: p_1$$

where we have defined p_1 above.

(ii) $n \geq cr$: this and $p'_w(n) < 0$ imply

$$p_w(n) \leq \operatorname{erf}\left(\frac{w}{\sqrt{2}rc}\right) + \sqrt{\frac{2}{w}}\frac{rc}{w}\left(e^{-\left(\frac{w}{\sqrt{2}r}\right)^2} - 1\right) =: p_2$$

where we have defined p_2 above.

Finally, we note p_1 and p_2 satisfy the required properties in the definition of a locality sensitive hash function. In particular, $p_1 > p_2$. Therefore, h is a locality sensitive hash function for Euclidean distance. □

References

[1] M. Datar, P. Indyk, N. Immorlica, and V. Mirrokni. *Locality-Sensitive Hashing Scheme Based on p-Stable Distributions.* SCG'04 Proceedings of the twentieth annual symposium on Computational Geometry, pages 253-262, 2004.

[2] P. Indyk and R. Motwani. *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality.* STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing, pages 604-613, 1998.