Long Term Memory in AI - Vector Search and Databases **COS 597A Fall 2023**

Class 4 - Low Dimensional Vector Search

Lectures: Lectures: Edo Liberty and Matthijs Douze

Warning: *Please do not cite this note as a peer reviewed source. Please submit requests and corrections as issues or pull requests at github.com/edoliberty/vector-search-class-notes*

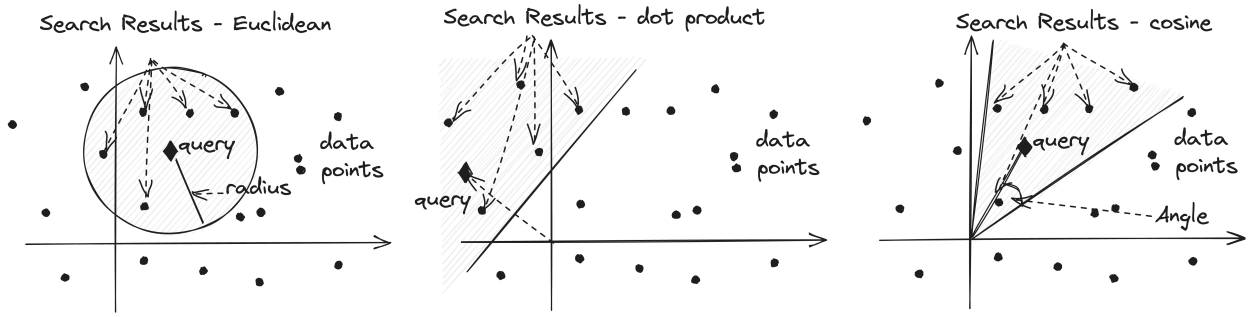
---

1 Nearest Neighbor Search Definition

As we have seen, nearest neighbor search is a fundamental computational building block in computer vision, semantic search, data mining, machine learning, and many other applications.

Definition 1.1. Nearest Neighbor Search: *Given a set of points* $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$ *preprocess them into a data structure of size* $\text{poly}(n, d)$ *in time* $\text{poly}(n, d)$ *such that nearest neighbor queries can be performed. Given a search query point* q *a radius* r *and the data structure should return return* $X_{q,r} = \{i \mid ||q - x_i|| \leq r\}$.

We will later extent this definition in the obvious way to max-inner-product search (MIPS) and cosine similarity search. Improving the practical and theoretical asymptotic time complexity for computing $X_{q,r}$ is the topic of this class.



One option we have is to do nothing at the preprocessing stage. Then, at query time, scan the data points and find those which minimize $\|q - x_i\|$. This would give a query time of $\Omega(nd)$. But, of course, it would be significantly better if we could reduce the dependence on n, or in other words, avoid scanning the data for every query point.

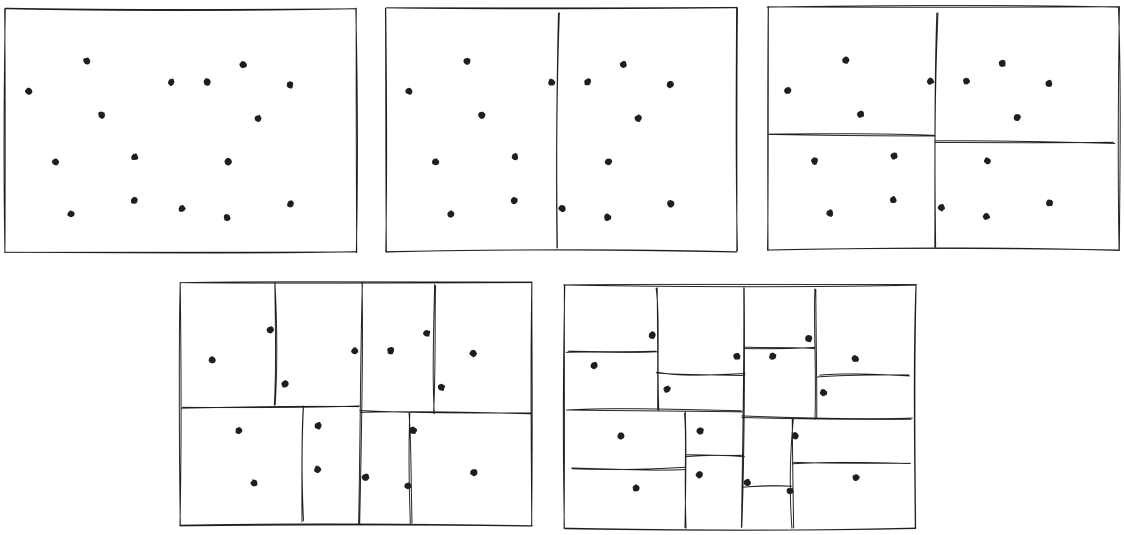
2 kd-trees

First, we shall review a well known an widely used algorithm for this problem which is called kd-trees [1]. The data structure holds the points in a geometrically partitioned tree. Each subtree contains all the points in an axis aligned bounding box, In each depth in the tree the bounding boxes are split along an axis. There are many different heuristics for splitting the boxes. For didactic reasons, let's consider a simple construction that lends itself to easy analysis.

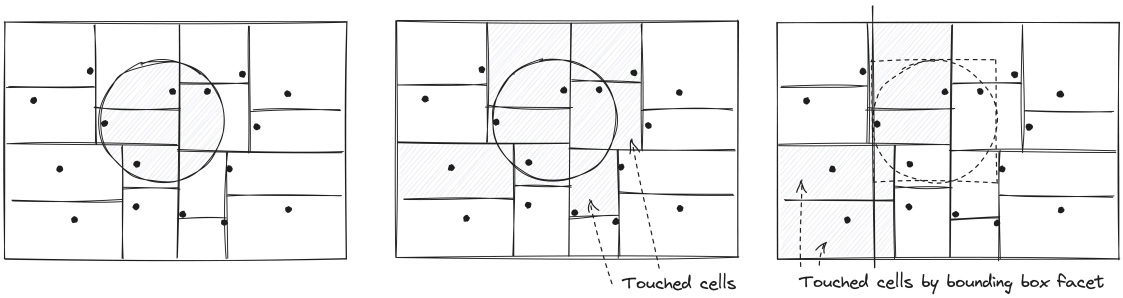
In our setting, we assuming the number of points is exactly a power of 2 and that coordinate values are distinct. Both assumptions are relatively harmless and should not change the asymptotic behavior.

We will organize the points in a perfectly balanced binary tree. Points will be associated only with leaves. The construction of the tree is simple. We start with all points in a single box (a node in the tree) We split

the box in two along the first coordinate such that exactly half the points lie on each side. Then, do the same for each of the two resulting boxes wrt to the second coordinate. Then, again, for the four resulting boxes wrt to the third coordinate and so on. Splitting stops when there is exactly one point in each leaf.

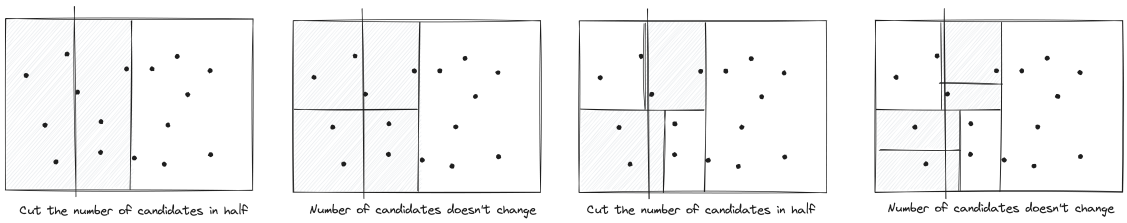


When searching the tree, we should consider only points whose bounding boxes touch the query region (ball in this case).



Fact 2.1. *Any axis aligned hyperplane touches at most $n^{1-1/d}$ boxes.*

Proof. Assume the hyperplane is aligned with the i'th coordinate. That is, the points $x \in \mathbb{R}^d$ such that $x_i = c$. When splitting according to coordinate i, the split value is either larger or smaller than c. When that happens, the hyperplane only touches one of the two nodes a level lower in the tree. The result is that, a splitting according to i's coordinate cuts the number of touched leaf boxes (points) by half. Finally, since we cycle through the coordinated in a round robin fashion, each coordinate is used for splitting $log_2(n)/d$ times. Note that $log_2(n)$ is the depth of the tree. We can now calculate the number of touched leaf boxes by $n(1/2)^{log_2(n)/d} = n^{1-1/d}$. □



To complete the analysis, note that the axis aligned bounding box of the query has $2d$ facets. Therefore, the number of touched cells is $O(dn^{1-1/d})$. For each touched cell we need to decide whether the associated point is a valid search result which takes $O(d)$ operations. The resulting search complexity of the algorithm is $O(d^2 n^{1-1/d})$ which is sub-linear in n. While this analysis is rather simple it it actually cannot be made much better [3]. There are many variants of this algorithm [2] and practical observations that claim kd-trees perform well in low dimensions.

Nevertheless, this runtime is only better than the brute force solution of $O(nd)$ when $n > d^d$ which already hints that there might be a problem with high dimensions. Before introducing the source of the problem, we should cover some basic facts about random variables. Facts that will serve us for the rest of the course.

3 Probability Tools and Facts Recap

A variable X is a random variable if it assumes different values according to a probability distribution. For example, X can denote the outcome of a three sided die throw. The variable X takes the values $x = 1, 2, 3$ with equal probabilities.

The expectation of X is the sum over the possible values times the probability of the events.

$$\mathbb{E}[X] = \sum_{x=1}^{3} x \Pr(X = x) = 1\frac{1}{3} + 2\frac{1}{3} + 3\frac{1}{3} = 2 \tag{1}$$

Continuous random variable are described by their distribution function φ.

$$\Pr[Y \in [a, b]] = \int_a^b \varphi(t)dt.$$

For a function φ to be a valid distribution we must have:

$$\forall\, t, \quad \varphi(t) \geq 0 \quad \text{(where it is defined)} \tag{2}$$

$$\int_a^b \varphi(t)dt \qquad \text{is well defined for all } a \text{ and } b \tag{3}$$

$$\int_{-\infty}^{-\infty} \varphi(t)dt = 1 \tag{4}$$

For example consider the continuous variable Y taking values in $[0, 1]$ uniformly. That means $\varphi(t) = 1$ if $t \in [0, 1]$ and zero else. This means that the probability of Y being in the interval $[t, t + dt]$ is exactly dt. And so the expectation of Y is:

$$\mathbb{E}[Y] = \int_{t=0}^{1} t\varphi(t)dt = \int_{t=0}^{1} t \cdot dt = \frac{1}{2}t^2|_0^1 = 1/2 \tag{5}$$

Remark 3.1. *Strictly speaking, distributions are not necessarily continuous or bounded functions. In fact, they can even no be a function at all. For example, the distribution of X above includes three Dirac-δ functions which are not valid functions.*

3.1 Dependence and Independence

A variable X is said to be *dependent* on Y if the distribution of X given Y is different than the distribution of X. For example. Assume the variable X takes the value 1 if Y takes a value of less than $1/3$ and the values 2 or 3 with equal probably otherwise ($1/2$ each). Clearly, the probability of X assuming each of its

values is still 1/3. however, if we know that Y is 0.7234 the probability of X assuming the value 1 is zero. Let us calculate the expectation of X given Y as an exercise.

$$\mathbb{E}(X|Y) = \sum_{x=1}^{3} x \Pr(X = x|Y \leq 1/3) = 1 \cdot 1 \tag{6}$$

$$\mathbb{E}(X|Y) = \sum_{x=1}^{3} x \Pr(X = x|Y > 1/3) = 1 \cdot 0 + 2\frac{1}{2} + 3\frac{1}{2} = 2.5 \tag{7}$$

$E(X|Y) = 1$ for $y \in [0, 1/3]$ and $E(X|Y) = 2.5$ for $y \in (1/3, 1]$.
Remember: $\mathbb{E}(X|Y)$ is a function of y!

Definition 3.1 (Independence)**.** *Two variables are said to be Independent if:*

$$\forall y, \ \Pr[X = x|Y = y] = \Pr[X = x].$$

They are dependent otherwise.

Fact 3.1. *If two variables X and Y are Independent the so are $f(X)$ and $g(Y)$ for any functions f and g.*

Fact 3.2 (Linearity of expectation 1)**.** *For any random variable and any constant α:*

$$\mathbb{E}[\alpha X] = \alpha \mathbb{E}[X] \tag{8}$$

Fact 3.3 (Linearity of expectation 2)**.** *For any two random variables*

$$\mathbb{E}_{X,Y}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \tag{9}$$

even if they are dependent.

Fact 3.4 (Multiplication of random variables)**.** *For any two* **independent** *random variables*

$$\mathbb{E}_{X,Y}[XY] = \mathbb{E}[X]\mathbb{E}[Y] \tag{10}$$

This does not necessarily hold if they are dependent.

Definition 3.2 (Variance)**.** *For a random variable X we have*

$$\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \tag{11}$$

The standard deviation σ of X is defined to be $\sigma(X) \equiv \sqrt{\mathrm{Var}[X]}$.

Definition 3.3 (Additivity of variances)**.** *For any two* **independent** *variables X and Y we have*

$$\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] \tag{12}$$

Fact 3.5 (Markov's inequality)**.** *For any positive random variable X:*

$$\Pr(X > t) \leq \frac{E[X]}{t} \tag{13}$$

Fact 3.6 (Chebyshev's inequality)**.** *For any random variable X*

$$\Pr[|X - E[X]| > t] \leq \frac{\sigma^2(X)}{t^2} \tag{14}$$

Lemma 3.1 (Chernoff's bound). *Let $X_1, \ldots, X_n$ be independent Bernoulli trials $\Pr[X_i = 1] = p_i$. And let $X = \sum_{i=1}^n X_i$ and $\mu = E[X] = \sum_{i=1}^n p_i$.*

$$\Pr[X > (1+\varepsilon)\mu] \leq e^{-\mu\varepsilon^2/4} \tag{15}$$

$$\Pr[X < (1-\varepsilon)\mu] \leq e^{-\mu\varepsilon^2/2} \tag{16}$$

Lemma 3.2 (The union bound). *For any set of m events $A_1, \ldots, A_m$:*

$$\Pr[\cup_{i=1}^m A_i] \leq \sum_{i=1}^m \Pr A_i.$$

In words, the probability that one or more events happen is at most the sum of the individual event probabilities.

4 Curse of Dimensionality

A prime example for the curse of dimensionality is that a random point in $[0,1]^d$ is likely to be far from any set of n points in the unit cube. Consider the distance between the query point q and an input data vector x. We want to show that $\|x_i - q\|^2 \in \Omega(d)$.

First, notice that $\Pr[|x(j) - q(j)| \geq 1/4] \geq 1/2$. The expected distance between x and q is at least $d/8$. Since $q(j)$ are independently drown, we can apply the Chernoff bound and get that for all n points in the data set $\|x_i - q\|^2 \geq d/16$ if $d \geq \text{const} \cdot \log(n)$.

Now, consider the kd-tree data structure and algorithm run on a random query. If the radius of the ball around q is less than $d/16$ the query is "uninteresting" since it is likely to return no results at all. On the other hand, if the radius is greater than $d/16$ than the ball around q will cross all the major partitions along one of the axis. That means that the algorithm will visit at least 2^d partitions.

5 Volumes of balls and cubes

Another interesting phenomenon that occurs in high dimensions in the fact that unit spheres are exponentially smaller (in volume) than their containing boxes. Let us see this without using the explicit formulas for the volume of d dimensional spheres.

To compute the volume of a unit sphere, we perform a thought experiment. First, bound the sphere in a box (with sides of length 2). Then, pick a point in the box uniformly at random. What is the probability p that the point is also in the sphere? This is exactly the ratio between the volume of the ball and the box (2^d). More accurately, $V = p2^d$ where V is the volume of the sphere.

Now, we can bound p from above. A uniformly random chosen point from the cube is a vector $x \in \mathbb{R}^d$ such that each coordinate $x(i)$ is chosen uniformly from $[-1, 1]$. The event that x is in the unit sphere is the event that $\|x\|^2 = \sum_{i=1}^d x(i)^2 \leq 1$. Let $z_i = x(i)^2$, and note that $\mathbb{E}[z(i)] = \int_{-1}^1 \frac{1}{2}t^2 dt = 1/3$. Therefore, $\mathbb{E}[\|x\|^2] = d/3$. Also,

$$\text{Var}(z_i) = \int_{-1}^1 \frac{1}{2}t^4 dt - (1/3)^2 = 1/5 - 1/9 \leq 1/10$$

so by Chernoff's inequality. $p = \Pr[\sum_{i=1}^d x(i)^2 \leq 1] = \Pr[\sum_{i=1}^d (z_i - \mathbb{E}[z_i]) \leq 1 - d/3] \leq e^{-\frac{(d/3)^2}{4d/10}} \leq e^{-d/4}$. This concludes the observation that the fraction of the volume which is inside the sphere is exponentially small compared to the cube. A counter intuitive way of viewing this is that almost the entire volume of the cube is concentrated at the "corners".

6 Orthogonality of random vectors

It turns out that two random vectors are also almost orthogonal. We can see this in two ways.

First, we can see that the expected, dot product of any vector x with a random vector y is small. It is trivial that $\mathbb{E}[\langle x, y\rangle] = 0$ since the distribution of y is symmetric. Moreover, $\mathbb{E}[\langle x, y\rangle^2] = 1/d$. To see this, consider $y_1, y_2, \ldots, y_d$ where $y_1 = y$ and $y_2, \ldots, y_d$ complete y to an orthogonal basis. Clearly, the distribution of all y_i are identical (but not independent) $\mathbb{E}[\langle x, y\rangle^2] = \mathbb{E}[\langle x, y_1\rangle^2] = \mathbb{E}[\frac{1}{d}\sum_{i=1}^{d}\langle x, y_i\rangle^2] = \frac{1}{d}\|x\|^2 = \frac{1}{d}$.

It is not hard to show that in fact for any vector x, if y is chosen uniformly at random from the unit sphere then $\Pr[\langle x, y\rangle \geq \frac{t}{\sqrt{d}}] \leq e^{-t^2/2}$. First, replace that uniform distribution over the unit sphere with an i.i.d. distribution of Gaussians $y(i) \sim \mathcal{N}(0, \frac{1}{\sqrt{d}})$. Note that $E[\|y\|^2] = 1$, moreover, from the sharp concentration of the χ^2 distribution we know that $E[\|y\|^2] \approx 1$. For convenience we will assume that $E[\|y\|^2] = 1$ and will ignore the small inaccuracy. Moreover, due to the rotational invariance of the Gaussian distribution we have that any direction is equally likely and thus this new distribution approximates the uniform distribution over the sphere. Next, notice that due to the rotational invariance $\langle x, y\rangle \sim \mathcal{N}(0, \frac{\|x\|}{\sqrt{d}}) = \mathcal{N}(0, \frac{1}{\sqrt{d}})$. Therefore, letting $Z \sim \mathcal{N}(0, 1)$ we have $\Pr[\langle x, y\rangle \geq \frac{t}{\sqrt{d}}] = \Pr[Z \geq t] \leq e^{-t^2/2}$.

References

[1] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, September 1975.

[2] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 537–546, New York, NY, USA, 2008. Association for Computing Machinery.

[3] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Inf.*, 9(1):23–29, mar 1977.