0368-3248-01-Algorithms in Data Mining **Fall 2012**

Lecture 9: Nearest neighbor search

Lecturer: Edo Liberty

Warning: *This note may contain typos and other inaccuracies which are usually discussed during class. Please do not cite this note as a reliable source. If you find mistakes, please inform me.*

---

1 Introduction

The problem most commonly known as "nearest neighbor search" is a fundamental computational problem in computer vision, graphics, data mining, machine learning, and many other fields. The problem is defined as such:

Definition 1.1. Nearest Neighbor Search: *Given a set of points* $\{x_1, \ldots, x_n\} \in \mathbb{R}^d$ *preprocess them into a data structure* X *of size* $\text{poly}(n, d)$ *in time* $\text{poly}(n, d)$ *such that nearest neighbor queries can be performed in logarithmic time. In other words, given a search point* q *a radius* r *and* X *one can return an* x_i *such the* $||q - x_i|| \leq r$ *or nothing if no such point exists. The search for* x_i *should require at most* $\text{poly}(d, \log(n))$ *time.*

Just as an example, consider a simple k-nearest neighbor classifier which, for each point decides it's class by the a majority vote over its neighbors classes. As simplistic as this classifier sounds, it actually performs very well in many scenarios.

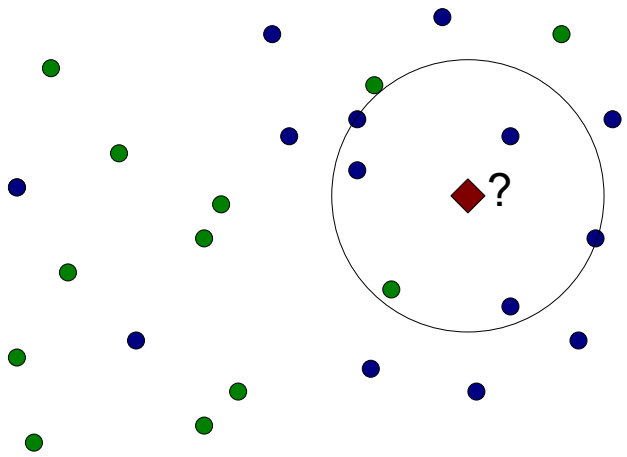

Figure 1: Illustration of the k-nearest neighbor classifier. The two classes are depicted by colors blue and grenn. The red rhombus will be tagged as being blue since most its neighbors are blue rather than green.

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 holding the points is a tree. Each subtree contains all the points in an axis aligned bounding box (maybe infinite). In each depth in the tree the bounding boxes are split along an axis (in a round robin order) at the median value of the points in the box. Splitting stops when the number of points in the box is sufficiently small, say one.

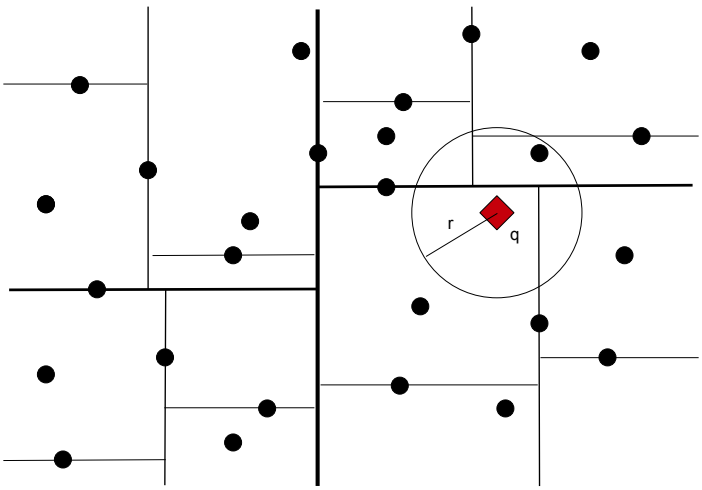

Figure 2: Illustration of space partition generated by the kd-tree algorithm in $\mathbb{R}^2$. The width of the separating lines indicate the depth in the tree of the corresponding split.

It is quite simple to prove that inserting and deleting points requires $O(log(n))$ time. Thus, the entire construction time is bounded by $O(n \log(n))$.

Searching however is quite a different matter. Let us assume w.o.l.g. that all points are inside the unit cube. A harmless assumption because we can scale the entire data set by a constant factor. Also, notice that each point in the date whose bounding box intersects the query sphere must be examined. Moreover, examining each data point can generate at most $O(log(n))$ computations. We will therefore only ask ourselves, how many point boxes have a non zero intersection with the query sphere.

To make this exercise simpler, consider a simpler case and a variation of kd-trees. First, all n data points are distributed uniformly and i.i.d. over $[0,1]^d$. Also, the space partition splits every box exactly in the middle when the cut is made along the axes in a round robin order. Note that for such random data, our simple algorithm produces a very simple partition to the one kd-tree would have generated since the median point is approximately also in the middle of the box. (this holds as long as the number of points in a box is large enough)

Let's assume each box is split t times (the depth of the tree is t). That means that each axis was split in half t/d (assuming t/d is an integer) times. Therefore, the dimension of our box are $[2^{-t/d}, 2^{-t/d}, \ldots, 2^{-t/d}]$. What is the probability that a (randomly located) ball of radius r will hit this box?

This probability that the box is "hit" by a random query sphere is at most the volume of a sphere of radius $\sqrt{d}2^{-t/d-1} + r$ since if the query falls outside this ball it will not intersect the box. Assume, for now, that $\sqrt{d}2^{-t/d-1} \leq r$ and so the probability of this event is bounded by the volume of a ball of radius $2r$. The volume of such a ball is $\pi^{d/2}(2r)^d/(d/2)!$ (assuming, for convenience, that d is even). The dependance on the radius is therefore $O(r^d)$. Thus, we can expect to inspect only a $O(r^d)$ fraction of the boxes (and thus points). Since $r \leq 1$ this can be a significant speedup.

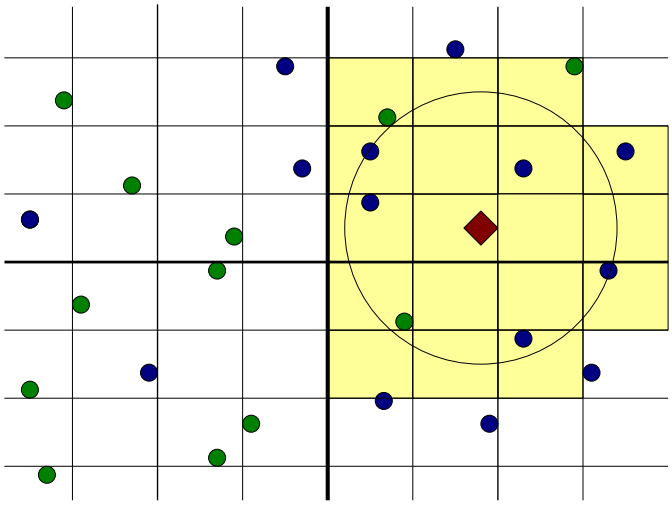

Figure 3: Illustration of space partition generated by the kd-tree algorithm in $\mathbb{R}^2$. The width of the separating lines indicate the depth in the tree of the corresponding split.

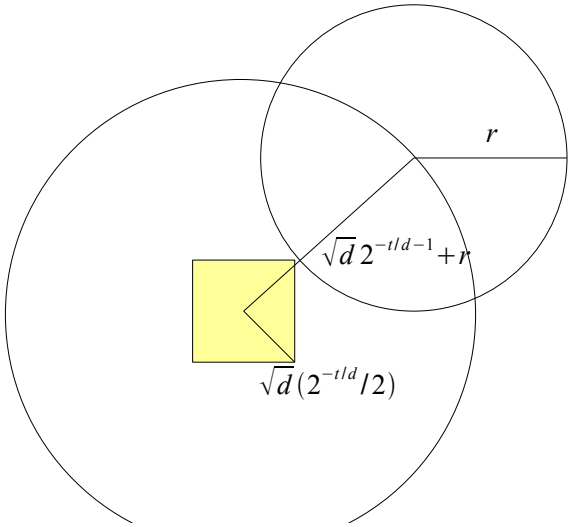

Figure 4: If the query point falls more than $\sqrt{d}2^{-t/d-1} + r$ away from the center of a box it cannot be 'hit'.

We saw that separating the space into boxes enabled us to search in it more efficiently. We, however, made some heavy assumptions. One such is that $\sqrt{d}2^{-t/d-1} \leq r$. This assumption fails when the dimension grows. To see this, consider that $t \approx \log(n)$ to give boxes of volume roughly $1/n$. Requiring that $\sqrt{d}2^{-\log(n)/d} \leq 1$ we get that n must be exponential in d, $n \geq 2^d$. Therefore, we only gain if the number of points in exponential in the dimension. This is a result of the so called "curse of dimensionality" which is discussed next.

3 Curse of dimensionality

3.1 kd-trees fail in high dimensions

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.

3.2 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".

3.3 Orthogonallity 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.