Geometric Approximation Algorithms

Sariel Har-Peled[①]

January 16, 2008

[②]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@uiuc.edu; http://www.uiuc.edu/~sariel/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

Contents

Preface

This manuscript is a collection of class notes on geometric approximation algorithms. It represents the book I wish I could have read when I started doing research.

There are without doubt errors and mistakes in the text and I would like to know about them. Please email me about any of them you find.

On optimality. Since this text is intended to explain key ideas and algorithms, I had consistently avoided the trap of optimality, when the optimal known algorithm is way more complicated and less insightful than some other algorithm. A reference to the optimal known algorithm would be usually described in the bibliographical notes.

A note on style. Injected into the text are random comments (usually as footnotes) that have nothing directly to do with the text. I hope these comments make the text more enjoyable to read, and I added them, on the spur of the moment, to amuse myself. Some readers might find these comments irritating and vain, and I humbly ask these readers to ignore them.

Acknowledgements

I had the benefit of interacting with numerous people on the work in this book. Sometime directly or indirectly. There is something mundane and predictable in enumerating a long list of people that helped and contributed to this work, but this in no way diminish their contribution and their help.

As such, I would like to thank the students in the class for their input, which helped in discovering numerous typos and errors in the manuscript. Furthermore, the content was greatly effected by numerous insightful discussions with Jeff Erickson and Edgar Ramos. Other people qthat provided comments or insights, answered nagging emails from me, for which I am thankful for include Bernard Chazelle, John Fischer, Samuel Hornus, Piotr Indyk, Mira Lee, Jirka Matoušek, and Manor Mendel.

I am sure that other people had contributed to this work, and I had forgot to mention them – they have my thanks and apologies.

Getting the source for this work

This work was written using LaTeX. Figures were drawn using ipe. You can get the source code of these class notes from `http://valis.cs.uiuc.edu/~sariel/teach/notes/aprx`. See below for detailed copyright notice.

In any case, if you are using these class notes and find them useful, it would be nice if you send me an email.

Copyright

— Sariel Har-Peled
April 2007, Daejeon, Korea

Chapter 1

The Power of Grids - Computing the Minimum Disk Containing k Points

> The Peace of Olivia. How sweat and peaceful it sounds! There the great powers noticed for the first time that the land of the Poles lends itself admirably to partition.
>
> – The tin drum, Gunter Grass

In this chapter, we are going to discuss two basic geometric algorithms. The first one, computes the closest pair among a set of n points in linear time. This is a beautiful and surprising result that exposes the computational power of using grids for geometric computation. Next, we discuss a simple algorithm for approximating the smallest enclosing ball that contains k points of the input. This at first looks like a bizarre problem, but turns out to be a key ingredient to our later discussion.

1.1 Preliminaries

For a real positive number r and a point $p = (x, y)$ in $\mathbb{R}^2$, define $\mathsf{G}_r(p)$ to be the grid point $(\lfloor x/r \rfloor r, \lfloor y/r \rfloor r)$. We call r the ***width*** of the ***grid*** G_r. Observe that G_r partitions the plane into square regions, which we call grid ***cells***. Formally, for any $i, j \in \mathbb{Z}$, the intersection of the half-planes $x \geq ri$, $x < r(i+1)$, $y \geq rj$ and $y < r(j+1)$ is said to be a grid ***cell***. Further we define a ***grid cluster*** as a block of 3×3 contiguous grid cells.

Note, that every grid cell C of G_r, has a unique ID; indeed, let $p = (x, y)$ be any point in C, and consider the pair of integer numbers $\mathrm{id}_C = \mathrm{id}(p) = (\lfloor x/r \rfloor, \lfloor y/r \rfloor)$. Clearly, only points inside C are going to be mapped to id_C. This is very useful, since we store a set P of points inside a grid efficiently. Indeed, given a point p, compute its $\mathrm{id}(p)$. We associate with each unique id a data-structure that stores all the points falling into this grid cell (of course, we do not maintain such data-structures for grid cells which are empty). So, once we computed $\mathrm{id}(p)$, we fetch the data structure associated with this cell, by using hashing. Namely, we store pointers to all those data-structures in a hash table, where each such data-structure is indexed by its unique id. Since the ids are integer numbers, we can do the hashing in constant time.

Assumption 1.1.1 *Throughout the discourse, we assume that every hashing operation takes (worst case) constant time. This is quite a reasonable assumption when true randomness is available (using for example perfect hashing [CLRS01]).*

For a point set P, and parameter r, the partition of P into subsets by the grid G_r, is denoted by $\mathsf{G}_r(\mathsf{P})$. More formally, two points $p, q \in \mathsf{P}$ belong to the same set in the partition $\mathsf{G}_r(\mathsf{P})$, if both points are being mapped to the same grid point or equivalently belong to the same grid cell.

1.2 Closest Pair

We are interested in solving the following problem:

Problem 1.2.1 Given a set P of n points in the plane, find the pair of points closest to each other. Formally, return the pair of points realizing $\mathcal{CP}(\mathsf{P}) = \min_{p,q \in P} \|p - q\|$.

Lemma 1.2.2 *Given a set P of n points in the plane, and a distance r, one can verify in linear time, whether $\mathcal{CP}(\mathsf{P}) < r$, $\mathcal{CP}(\mathsf{P}) = r$, or $\mathcal{CP}(\mathsf{P}) > r$.*

Proof: Indeed, store the points of P in the grid G_r. For every non-empty grid cell, we maintain a linked list of the points inside it. Thus, adding a new point p takes constant time. Indeed, compute $\mathrm{id}(p)$, check if $\mathrm{id}(p)$ already appears in the hash table, if not, create a new linked list for the cell with this ID number, and store p in it. If a data-structure already exist for $\mathrm{id}(p)$, just add p to it.

This takes $O(n)$ time. Now, if any grid cell in $\mathsf{G}_r(\mathsf{P})$ contains more than, say, 9 points of p, then it must be that the $\mathcal{CP}(\mathsf{P}) < r$. Indeed, consider a cell C containing more than nine points of P, and partition C into 3×3 equal squares. Clearly, one of those squares must contain two points of P, and let C' be this square. Clearly, the diameter of $C' = \mathrm{diam}(C)/3 = \sqrt{r^2 + r^2}/3 < r$. Thus, the two (or more) points of P in C' are at distance smaller than r from each other.

Thus, when we insert a point p, we can fetch all the points of P that were already inserted, in the cell of p, and the 8 adjacent cells. All those cells, must contain at most 9 points of P (otherwise, we would already have stopped since the $\mathcal{CP}(\cdot)$ of inserted points, is smaller than r). Let S be the set of all those points, and observe that $|S| \leq 9 \cdot 9 = O(1)$. Thus, we can compute by brute force the closest point to p in S. This takes $O(1)$ time. If $\mathbf{d}(p, S) < r$, we stop, otherwise, we continue to the next point.

Overall, this takes $O(n)$ time. As for correctness, first observe that if $\mathcal{CP}(\mathsf{P}) > r$ then the algorithm would never make a mistake, since it returns '$\mathcal{CP}(\mathsf{P}) < r$' only after finding a pair of points of P with distance smaller than r. Thus, assume that p, q are the pair of points of P realizing the closest pair, and $\|pq\| = \mathcal{CP}(\mathsf{P}) < r$. Clearly, when the later of them, say p, is being inserted, the set S would contain q, and as such the algorithm would stop and return '$\mathcal{CP}(\mathsf{P}) < r$'. ∎

Lemma 1.2.2 provides a natural way of computing $\mathcal{CP}(\mathsf{P})$. Indeed, permute the points of P in arbitrary fashion, and let $\mathsf{P} = \langle p_1, \ldots, p_n \rangle$. Next, let $r_i = \mathcal{CP}(\{p_1, \ldots, p_i\})$. We can check if $r_{i+1} < r_i$, by just calling the algorithm for Lemma 1.2.2 on P_{i+1} and r_i. In fact, if $r_{i+1} < r_i$, the algorithm of Lemma 1.2.2, would give us back the distance r_{i+1} (with the other point realizing this distance).

So, consider the "good" case, where $r_i = r_{i-1}$; that is, the length of the shortest pair does not change when p_i is inserted. In this case, we do not need to rebuild the data structure of Lemma 1.2.2 for the ith point. We can just reuse it from the previous iteration by inserting p_i into it. Thus, inserting a single point takes constant time, as long as the closest pair does not change.

Things become problematic when $r_i < r_{i-1}$, because then we need to rebuild the grid data structure, and reinsert all the points of $\mathsf{P}_i = \langle p_1, \ldots, p_{i+1} \rangle$ into the new grid $\mathsf{G}_{r_i}(\mathsf{P}_i)$. This takes $O(i)$ time.

Specifically, if the closest pair distance, in the sequence $r_1, \ldots, r_n$, changes only k times, then the running time of our algorithm would be $O(nk)$. In fact, we can do even better.

Theorem 1.2.3 *For set P of n points in the plane, one can compute the closest pair of P in expected linear time.*

Proof: Pick a random permutation of the points of P, let $\langle p_1, \ldots, p_n \rangle$ be this permutation. Let $r_2 = \|p_1 p_2\|$, and start inserting the points into the data structure of Lemma 1.2.2. In the ith iteration, if $r_i = r_{i-1}$, then this insertion takes constant time. If $r_i < r_{i-1}$, then we rebuild the grid and reinsert the points. Namely, we recompute $\mathsf{G}_{r_i}(\mathsf{P}_i)$.

To analyze the running time of this algorithm, let X_i be the indicator variable which is 1 if $r_i \neq r_{i-1}$, and 0 otherwise. Clearly, the running time is proportional to

$$R = 1 + \sum_{i=2}^{n} (1 + X_i \cdot i).$$

Thus, the expected running time is

$$\mathbf{E}[R] = \mathbf{E}\left[1 + \sum_{i=2}^{n} (1 + X_i \cdot i)\right] = n + \sum_{i=2}^{n} (\mathbf{E}[X_i] \cdot i) = n + \sum_{i=2}^{n} i \cdot \mathbf{Pr}[X_i = 1],$$

by linearity of expectation and since for indicator variable X_i, we have $\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1]$.

Thus, we need to bound $\mathbf{Pr}[X_i = 1] = \mathbf{Pr}[r_i < r_{i-1}]$. To bound this quantity, fix the points of P_i, and randomly permute them. A point $q \in \mathsf{P}_i$ is called ***critical***, if $\mathcal{CP}(\mathsf{P}_i \setminus \{q\}) > \mathcal{CP}(\mathsf{P}_i)$. If there are no critical points, then $r_{i-1} = r_i$ and then $\mathbf{Pr}[X_i = 1] = 0$. If there is one critical point, then $\mathbf{Pr}[X_i = 1] = 1/i$, as this is the probability that this critical point, would be the last point in the random permutation of P_i.

If there are two critical points, and let p, q be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$. The quantity r_i is smaller than r_{i-1}, one if either p or q are p_i. But the probability for that is $2/i$ (i.e., the probability in a random permutation of i objects, that one of two marked objects would be the last element in the permutation).

Observe, that there can not be more than two critical points. Indeed, if p and q are two points that realizing the closest distance, than if there is a third critical point r, then $\mathcal{CP}(\mathsf{P}_i \setminus \{r\}) = \|pq\|$, and r is not critical.

We conclude that

$$\mathbf{E}[R] = n + \sum_{i=2}^{n} i \cdot \mathbf{Pr}[X_i = 1] \leq n + \sum_{i=2}^{n} i \cdot \frac{2}{i} \leq 3n,$$

and the expected running time is $O(\mathbf{E}[R]) = O(n)$. ■

Theorem 1.2.3 is a surprising result, since it implies that uniqueness (i.e., deciding if n real numbers are all distinct) can be solved in linear time. However, there is a lower bound of $\Omega(n \log n)$ on uniqueness, using the comparison model. This reality dysfunction can be easily explained once one realizes that the computation of Theorem 1.2.3 is considerably stronger, using hashing, randomization, and the floor function.

1.3 A Slow 2-Approximation Algorithm for the k-Enclosing Disk

For a circle D, we denote by radius(D) the ***radius*** of D.

Let $D_{\text{opt}}(\mathsf{P}, k)$ be a disk of minimum radius which contains k points of P, and let $r_{\text{opt}}(\mathsf{P}, k)$ denote the radius of $D_{\text{opt}}(\mathsf{P}, k)$.

Let P be a set of n points in the plane. Compute a set of $m = O(n/k)$ horizontal lines $h_1, \ldots, h_m$ such that between two consecutive horizontal lines, there are at most $k/4$ points of P in the strip they define. This can be easily done in $O(n \log(n/k))$ time using deterministic median selection together with recursion.[2] Similarly, compute a set of vertical lines $v_1, \ldots, v_m$, such that between two consecutive lines, there are at most $k/4$ points of P.

[2] Indeed, compute the median in the x-order of the points of P, split P into two sets, and recurse on each set, till the number of points in a subproblem is of size $\leq k/4$. We have $T(n) = O(n) + 2T(n/2)$, and the recursion stops for $n \leq k/4$. Thus, the recursion tree has depth $O(\log(n/k))$, which implies running time $O(n \log(n/k))$.

Consider the (non-uniform) grid G induced by the lines $h_1, \ldots, h_m$ and $v_1, \ldots, v_m$. Let X be the set of all intersection points of G. We claim that $D_{\text{opt}}(\mathsf{P}, k)$ contains at least one point of X. Indeed, consider the center u of $D_{\text{opt}}(\mathsf{P}, k)$, and let c be the cell of G that contains u. Clearly, if $D_{\text{opt}}(\mathsf{P}, k)$ does not cover any of the four vertices of c, then it can cover only points in the vertical strip of G that contains c, and only points in the horizontal strip of G that contains c. See Figure 1.1. However, each such strip contains at most $k/4$ points. It follows that $D_{\text{opt}}(\mathsf{P}, k)$ contains at most $k/2$ points of P, a contradiction. Thus, $D_{\text{opt}}(\mathsf{P}, k)$ must contain a point of X. For every point $p \in X$, compute the smallest circle centered at p that contains k points of P. Clearly, for a point $q \in X \cap D_{\text{opt}}(\mathsf{P}, k)$, this yields the required 2-approximation. Indeed, the disk of radius $2D_{\text{opt}}(\mathsf{P}, k)$ centered at q contains at least k points of P since it also covers $D_{\text{opt}}(\mathsf{P}, k)$. We summarize as follows:

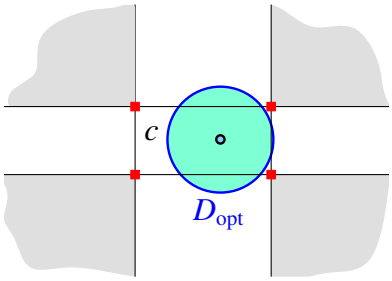


Figure 1.1: If the disk $D_{\text{opt}}(\mathsf{P}, k)$ does not contain any vertex of the cell c, then it does not cover any shaded area. As such, it can contain at most $k/2$ points, since the vertical and horizontal strips containing c, each has at most $k/4$ points of P inside it.

Lemma 1.3.1 *Given a set P of n points in the plane, and parameter k, one can compute in $O(n(n/k)^2)$ deterministic time, a circle D that contains k points of P, and* $\text{radius}(D) \leq 2r_{\text{opt}}(\mathsf{P}, k)$.

Corollary 1.3.2 *Given a set of P of n points and a parameter $k = \Omega(n)$, one can compute in linear time, a circle D that contains k points of P and* $\text{radius}(D) \leq 2r_{\text{opt}}(\mathsf{P}, k)$.

1.4 A Linear Time 2-Approximation for the k-Enclosing Disk

Stuff Warning In the following, we present a linear time algorithm for approximating the minimum enclosing disk. While interesting on their own right, the results here are not used later and can be skipped on first, second, third (or any other) reading.

1.4.1 The algorithm

We refer to the algorithm of Lemma 1.3.1 as **ApproxHeavy**(P, k).

Remark 1.4.1 For a point set P of n points, the radius r returned by the algorithm of Lemma 1.3.1 is the distance between a vertex of the non-uniform grid and a point of P. As such, a grid G_r computed using this distance is one of $O(n^3)$ possible grids. Indeed, a circle is defined by the distance between a vertex of the non-uniform grid of Lemma 1.3.1, and a point of P. A vertex of such a grid is determined by two points of P, through which the vertical and horizontal line passes. Thus, there are $O(n^3)$ such triples.

Let $\mathtt{gd}_r(\mathsf{P})$ denote the maximum number of points of P mapped to a single point by the mapping G_r. Define $\mathtt{depth}(\mathsf{P}, r)$ to be the maximum number of points of P that a circle of radius r can contain.

Lemma 1.4.2 *For any point set P, and $r > 0$, we have: (i) for any real number $A > 0$, it holds*

(i) $\mathtt{depth}(\mathsf{P}, Ar) \leq (A+1)^2\, \mathtt{depth}(\mathsf{P}, r)$,

(ii) $\mathtt{gd}_r(\mathsf{P}) \leq \mathtt{depth}(\mathsf{P}, r) \leq 9\mathtt{gd}_r(\mathsf{P})$,

(iii) if $r \leq 2r_{\text{opt}}(\mathsf{P}, k)$ *then* $\mathtt{gd}_r(P) \leq 5k$, *and*

(iv) any circle of radius r is covered by at least one grid cluster in G_r.

Proof: (i) Consider the disk D of radius Ar realizing $\mathtt{depth}(\mathsf{P}, Ar)$, and let D' be the disk of radius $(A+1)r$ having the same center as D. For every point $p \in V = \mathsf{P} \cap D$, place a disk around it of radius r, and let S denote the resulting set of disks. Since every disk of S has area πr^2, and they are all contained in D', which is of area $\pi(A+1)^2 r^2$, it follows that there must be a point p inside D' which is contained in $\mu = \left\lceil \frac{|V|\pi r^2}{\pi(A+1)^2 r^2} \right\rceil = \left\lceil \frac{\mathtt{depth}(\mathsf{P}, Ar)}{(A+1)^2} \right\rceil$ disks. This means, that the disk $\mathcal{D}$ of radius r centered at p contains at least μ points of P. Now, $\mu \leq |\mathcal{D} \cap \mathsf{P}| \leq \mathtt{depth}(\mathsf{P}, r)$. Thus, $\frac{\mathtt{depth}(\mathsf{P}, Ar)}{(A+1)^2} \leq \mu \leq \mathtt{depth}(\mathsf{P}, r)$, as claimed.

(iv) Consider a (closed) disk D of radius r, and let c be its center. If c is in the interior of a grid cell C, then the claim easily holds, since D can intersect only C or the cells adjacent to C. Namely, D is contained in the cluster centered at C. The problematic case, is when c is on the grid boundaries. Since grid cells are closed on one side, and open on the other, it is easy to verify that the claim holds again by careful and easy case analysis, which we will skip here.

(ii) Consider the grid cell C of $\mathsf{G}_r(\mathsf{P})$ that realizes $\mathtt{gd}_r(\mathsf{P})$, and let c be a point placed in the center of C. Clearly, a disk D of radius $\sqrt{r^2 + r^2}/2 = r/\sqrt{2}$ centered at c, would cover completely the cell C. Thus, $\mathtt{gd}_r(\mathsf{P}) \leq |D \cap \mathsf{P}| \leq \mathtt{depth}(\mathsf{P}, r)$. As for the other inequality, observe that the disk $\mathcal{D}$ realizing $\mathtt{depth}(\mathsf{P}, r)$, can intersect at most 9 cells of the grid G_r, by (iv). Thus, $\mathtt{depth}(\mathsf{P}, r) = |\mathcal{D} \cap \mathsf{P}| \leq 9\mathtt{gd}_r(\mathsf{P})$.

(iii) Let C be the grid cell of G_r realizing $\mathtt{gd}_r(\mathsf{P})$. Place 4 points, at the corners of C, and one point in the center of C. Placing a disk of radius $r_{\mathrm{opt}}(\mathsf{P}, k)$ at each of those points, completely covers C, as can be easily verified (since the side length of C is at most $2r_{\mathrm{opt}}(\mathsf{P}, k)$). Thus, $|\mathsf{P} \cap C| = \mathtt{gd}_r(\mathsf{P}) \leq 5\,\mathtt{depth}(\mathsf{P}, r_{\mathrm{opt}}(\mathsf{P}, k)) = 5k$. ∎

1.4.1.1 Description

As in the previous sections, we construct a grid which partitions the points into small ($O(k)$ sized) groups. The key idea behind speeding up the grid computation is to construct the appropriate grid over several rounds. Specifically, we start with a small set of points as seed and construct a suitable grid for this subset. Next, we incrementally insert the remaining points, while adjusting the grid width appropriately at each step.

Definition 1.4.3 (Gradation) Given a set P of n points, a ***k*-gradation** $(S_1, \ldots, S_m)$ of P is a sequence of subsets of P, such that (i) $S_m = \mathsf{P}$, (ii) S_i is formed by picking each point of S_{i+1} with probability $1/2$, and (iii) $|S_1| \leq k$, and $|S_2| > k$.

Lemma 1.4.4 *Given* P*, a* **gradation** *can be computed in expected linear time.*

Proof: Observe that the sampling time is $O\left(\sum_{i=1}^{m} |S_i|\right)$, where m is the length of the sequence. Also observe that $\mathbf{E}[|S_m|] = |S_m| = n$ and

$$\mathbf{E}[|S_i|] = \mathbf{E}\left[\mathbf{E}\left[|S_i| \;\middle|\; |S_{i+1}|\right]\right] = \mathbf{E}\left[\frac{|S_{i+1}|}{2}\right] = \frac{1}{2}\,\mathbf{E}[|S_i|]\,.$$

Now by induction, we get

$$\mathbf{E}[|S_{m-i}|] = \frac{n}{2^{i-1}}.$$

Thus, the running time is $O\left(\mathbf{E}\left[\sum_{i=1}^{m} |S_i|\right]\right) = O(n)$. ∎

Let $\mathcal{P} = (\mathsf{P}_1, \ldots, \mathsf{P}_m)$ be a u-gradation of P (see Definition 1.4.3), where $u = \max(k, n/\log n)$. The sequence $\mathcal{P}$ can be computed in expected linear time as shown in Lemma 1.4.4.

Since $|\mathsf{P}_1| = O(k)$, we can compute r_1, in $O(|\mathsf{P}_1|\,(|\mathsf{P}_1|/k)^2) = O(k) = O(n)$ time, using Lemma 1.3.1.

```
Grow(P_i, r_{i-1}, k)
        Output: r_i
    begin
        G_{i-1} ← G_{r_{i-1}}(i)
        for every grid cluster c ∈ G_{i-1} with |c ∩ P_i| ≥ k do
            P_c ← c ∩ P_i
            r_c ← ApproxHeavy(c, k)
            // ApproxHeavy is the algorithm of Lemma 1.3.1
            We have r_opt(c, k) ≤ r_c ≤ 2 r_opt(c, k),

        return minimum r_c computed.
    end
```

Figure 1.2: Algorithm for the ith round.

The algorithm now works in m rounds, where m is the length of the sequence $\mathcal{P}$. At the end of the ith round, we have a distance r_i such that $\mathsf{gd}_{r_i}(\mathsf{P}_i) \leq 5k$, and there exists a grid cluster in G_{r_i} containing more than k points of P_i and $r_{\mathrm{opt}}(\mathsf{P}_i, k) \leq r_i$.

At the ith round, we first construct a grid G_{i-1} for points in P_i using r_{i-1} as grid width. We know that there is no grid cell containing more than $5k$ points of P_{i-1}. As such, intuitively, we expect every cell of G_{i-1} to contain at most $10k$ points of P_i, since $\mathsf{P}_{i-1} \subseteq \mathsf{P}_i$ was formed by choosing each point of P_i into P_{i-1} with probability $1/2$. (This is of course too good to be true, but something slightly weaker does hold.) Thus allowing us to use the slow algorithm of Lemma 1.3.1 on those grid clusters. Note that, for $k = \Omega(n)$, the algorithm of Lemma 1.3.1 runs in linear time, and thus the overall running time is linear.

The algorithm used in the ith round is more concisely stated in Figure 1.2. At the end of the m rounds we have r_m, which is a 2-approximation to the radius of the optimal k enclosing circle of $\mathsf{P}_m = \mathsf{P}$. The overall algorithm is summarized in Figure 1.3.

1.4.1.2 Analysis

Lemma 1.4.5 *For $i = 1, \ldots, m$, we have $r_{\mathrm{opt}}(\mathsf{P}_i, k) \leq r_i \leq 2r_{\mathrm{opt}}(\mathsf{P}_i, k)$, and the heaviest cell in $\mathsf{G}_{r_i}(\mathsf{P}_i)$ contains at most $5k$ points of P_i.*

Proof: Consider the optimal circle D_i that realizes $r_{\mathrm{opt}}(\mathsf{P}_i, k)$. Observe that there is a cluster c of $\mathsf{G}_{r_{i-1}}$ that contains D_i, as $r_{i-1} \geq r_i$. Thus, when **Grow** handles the cluster c, we have $D_i \cap \mathsf{P}_i \subseteq c$. The first part of the lemma then follows from the correctness of the algorithm of Lemma 1.3.1.

As for the second part, observe that any grid cell of width r_i can be covered with 5 circles of radius $r_i/2$, and $r_i/2 \leq r_{\mathrm{opt}}(\mathsf{P}_i, k)$. It follows that each grid cell of $\mathsf{G}_{r_i}(i)$ contains at most $5k$ points. ■

Now we proceed to upper-bound the number of cells of $\mathsf{G}_{r_{i-1}}$ that contains "too many" points of P_i. Since each point of P_{i-1} was chosen from P_i with probability $1/2$, we can express this bound as a sum of independent random variables, and bound this using tail-bounds.

Definition 1.4.6 For a point set P, and parameters k and r, the ***excess*** of $\mathsf{G}_r(\mathsf{P})$ is

$$\mathcal{E}(\mathsf{P}, r) = \sum_{c \in \mathrm{Cells}(\mathsf{G}_r)} \left\lfloor \frac{|c \cap \mathsf{P}|}{50k} \right\rfloor,$$

where $\mathrm{Cells}(\mathsf{G}_r)$ is the set of cells of the grid G_r.

```
LinearApprox(P, k)
    Output: r - a 2-approximation to r_opt(P, k)
  begin
    Compute a gradation {P_1, ..., P_m} of P as in Lemma 1.4.4
    r_1 ← ApproxHeavy(P_1, k)
        // ApproxHeavy is the algorithm of Lemma 1.3.1
        // which outputs a 2-approximation

    for i ← 2 to m do
        r_j ← Grow(P_i, r_{i-1}, k)

    for every grid cluster c ∈ G_{r_m} with |c ∩ P| ≥ k do
        r_c ← ApproxHeavy(c ∩ P, k)

    return minimum r_c computed over all clusters
  end
```

Figure 1.3: 2-Approximation Algorithm.

Remark 1.4.7 The quantity $100k \cdot \mathcal{E}(\mathsf{P}, r)$ is an upper bound on the number of points of P in an heavy cell of $\mathsf{G}_r(\mathsf{P})$, where a cell of $\mathsf{G}_r(\mathsf{P})$ is ***heavy*** if it contains at least $50k$ points.

Lemma 1.4.8 *For any positive real t, the probability that $\mathsf{G}_{r_{i-1}}(i)$ has excess $\mathcal{E}(\mathsf{P}_i, r_{i-1}) \geq \alpha = t + 3\lceil \lg n \rceil$, is at most 2^{-t}.*

Proof: Let $\mathfrak{G}$ be the set of $O(n^3)$ possible grids that might be considered by the algorithm (see Remark 1.4.1), and fix a grid $\mathsf{G} \in \mathfrak{G}$ with excess $M = \mathcal{E}(\mathsf{P}_i, \varkappa(\mathsf{G})) \geq \alpha$, where $\varkappa(\mathsf{G})$ is the sidelength of a cell of G.

Let $U = \left\{ \{\mathsf{P}_i \cap c\} \;\middle|\; c \in \mathsf{G}, |\mathsf{P}_i \cap c| \geq 50k \right\}$ be all the heavy cells in $\mathsf{G}(i)$. Furthermore, let

$$V = \bigcup_{X \in U} P(X, 50k),$$

where $P(X, \nu)$ denotes an arbitrary partition of the set X into disjoint subsets such that each one of them contains ν points, except maybe the last subset that might contain between ν and $2\nu - 1$ points.

It is clear that $|V| = \mathcal{E}(\mathsf{P}_i, \ell)$. From the Chernoff inequality, for any $S \in V$, we have $\mu = \mathbf{E}[|S \cap \mathsf{P}_{i-1}|] \geq 25k$, and setting $\delta = 4/5$ we have

$$\mathbf{Pr}[|S \cap \mathsf{P}_{i-1}| \leq 5k] \leq \mathbf{Pr}[|S \cap \mathsf{P}_{i-1}| \leq (1-\delta)\mu] < \exp\left(-\mu\frac{\delta^2}{2}\right) \leq \exp\left(-\frac{25k(4/5)^2}{2}\right) < \frac{1}{2}.$$

Furthermore, since $\mathsf{G} = \mathsf{G}_{r_{i-1}}$ implying that each cell of $\mathsf{G}(i-1)$ contains at most $5k$ points. Thus we have

$$\mathbf{Pr}[\mathsf{G}_{r_{i-1}} = \mathsf{G}] \leq \prod_{S \in V} \mathbf{Pr}[|S \cap \mathsf{P}_{i-1}| \leq 5k] \leq \frac{1}{2^{|V|}} = \frac{1}{2^M} \leq \frac{1}{2^\alpha}.$$

Since there are n^3 different grids in $\mathfrak{G}$, we have

$$\begin{aligned}\mathbf{Pr}[\mathcal{E}(\mathsf{P}_i, r_{i-1}) \geq \alpha] &= \mathbf{Pr}\left[\bigcup_{\substack{\mathsf{G}\in\mathfrak{G},\\ \mathcal{E}(\mathsf{P}_i,\varkappa(\mathsf{G}))\geq\alpha}} (\mathsf{G} = \mathsf{G}_{r_{i-1}})\right] \\ &\leq \sum_{\substack{\mathsf{G}\in\mathfrak{G},\\ \mathcal{E}(\mathsf{P}_i,\varkappa(\mathsf{G}))\geq\alpha}} \mathbf{Pr}[\mathsf{G} = \mathsf{G}_{r_{i-1}}] \leq n^3 \frac{1}{2^\alpha} \leq \frac{1}{2^t}.\end{aligned}$$

■

We next bound the expected running time of the algorithm **LinearApprox** by bounding the expected time spent in the ith iteration. In particular, let Y be the random variable which is the excess of $\mathsf{G}_{r_{i-1}}(i)$. In this case, there are at most Y cells which are heavy in $\mathsf{G}_{r_{i-1}}(i)$, and each such cell contains at most $O(Yk)$ points. Thus, invoking the algorithm of **ApproxHeavy** on such a heavy cell takes $O(Yk\cdot((Yk)/k)^2) = O(Y^3k)$ time. Overall, the running time of **Grow**, in the ith iteration, is $T(Y) = O\Big(|\mathsf{P}_i| + Y\cdot Y^3k\Big) = O\Big(|\mathsf{P}_i| + Y^4k\Big)$.

For technical reasons, we need to consider the light and heavy cases separately to bound Y. So set

$$\Lambda = \lceil 3\lg n\rceil.$$

The Light Case: $k < \Lambda$. We have that the expected running time is proportional to

$$\begin{aligned}O(|\mathsf{P}_i|) + \sum_{t=0}^{\lceil n/k\rceil} \mathbf{Pr}[Y = t]\, T(t) &= O(|\mathsf{P}_i|) + \mathbf{Pr}\Big[0 \leq Y \leq \Lambda\Big]\, T(\Lambda) + \sum_{t=\Lambda+1}^{n/k} \mathbf{Pr}[Y = t]\, T(t) \\ &\leq O(|\mathsf{P}_i|) + T(\Lambda) + \sum_{t=1}^{n/k} \frac{1}{2^t} T(t+\Lambda) \\ &= O(|\mathsf{P}_i|) + O(k\log^4 n) + \sum_{t=1}^{n/k} \frac{(t+\Lambda)^4\, k}{2^t} \\ &= O\Big(|\mathsf{P}_i| + k\log^4 n\Big) = O(|\mathsf{P}_i|),\end{aligned}$$

by Lemma 1.4.8 and since $T(\cdot)$ is a monotone increasing function.

The Heavy Case: $k \geq \Lambda$.

Lemma 1.4.9 *The probability that $\mathsf{G}_{r_{i-1}}(i)$ has* **excess** *larger than t, is at most 2^{-t}, for $k \geq \Lambda$.*

Proof: We use the same technique as in Lemma 1.4.8. By the Chernoff inequality, the probability that any $50k$ size subset of P_i would contain at most $5k$ points of P_{i-1}, is less than

$$\leq \exp\left(-25k\cdot\frac{16}{25}\cdot\frac{1}{2}\right) \leq \exp(-5k) \leq \frac{1}{n^4}.$$

In particular, arguing as in Lemma 1.4.8, it follows that the probability that $\mathcal{E}(\mathsf{P}_i, k, r_{i-1})$ exceeds t, is smaller than $n^3/n^{4t} \leq 2^{-t}$. ■

Thus, if $k \geq \Lambda$, the expected running time of **Grow**, in the ith iteration, is at most

$$O\left(\sum_{c\in\mathsf{G}_{r_{i-1}}} |c\cap\mathsf{P}_i|\left(\frac{|c\cap\mathsf{P}_i|}{k}\right)^2\right) = O\left(|\mathsf{P}_i| + \sum_{t=1}^{\infty} t\cdot\left(tk\cdot\left(\frac{tk}{k}\right)^2\right)\cdot\frac{1}{2^t}\right) = O(|\mathsf{P}_i| + k) = O(|\mathsf{P}_i|),$$

by Lemma 1.4.9.

Overall Running Time Analysis Thus, by the above analysis and by Lemma 1.4.4, the total expected running time of **LinearApprox** inside the inner loop is $O(\sum_i |\mathsf{P}_i|) = O(n)$. As for the last step, of computing a 2-approximation, consider the grid $\mathsf{G}_{r_m}(\mathsf{P})$. Each grid cell contains at most $5k$ points, and hence each grid cluster contains at most $45k$ points. Also the smallest k enclosing circle is contained in some grid cluster. In each cluster that contain more than k points, we use the algorithm of Corollary 1.3.2 and finally output the minimum over all the clusters. The overall running time is $O((n/k)k) = O(n)$ for this step, since each point belongs to at most 9 clusters.

Theorem 1.4.10 *Given a set P of n points in the plane, and a parameter k, one can compute, in expected linear time, a radius r, such that $r_{\text{opt}}(\mathsf{P}, k) \le r \le 2r_{\text{opt}}(\mathsf{P}, k)$.*

Once we compute r such that $r_{\text{opt}}(\mathsf{P}, k) \le r \le 2r_{\text{opt}}(\mathsf{P}, k)$, using the algorithm of Theorem 1.4.10, we apply an exact algorithm to each cluster of the grid $\mathsf{G}_r(\mathsf{P})$ which contains more than k points.

Matoušek presented such an exact algorithm [Mat95a], and it has running time of $O(n \log n + nk)$ and space complexity $O(nk)$. Since r is a 2 approximation to $r_{\text{opt}}(\mathsf{P}, k)$, each cluster has $O(k)$ points. Thus the running time of the exact algorithm in each cluster is $O(k^2)$ and requires $O(k^2)$ space. The number of clusters which contain more than k points is $O(n/k)$. Hence the overall running time is $O(nk)$, and the space used is $O(n + k^2)$.

Theorem 1.4.11 *Given a set P of n points in the plane and a parameter k, one can compute, in expected $O(nk)$ time, using $O(n + k^2)$ space, the radius $r_{\text{opt}}(\mathsf{P}, k)$, and a circle $D_{\text{opt}}(\mathsf{P}, k)$ that covers k points of P.*

1.5 Bibliographical notes

Our closest-pair algorithm follows Golin *et al.* [GRSS95]. This is in turn a simplification of a result of Rabin [Rab76]. Smid provides a survey of such algorithms [Smi00]).

The proof of Lemma 1.4.2 is from [Mat95a]. The min-disk approximation algorithm follows roughly the work of Har-Peled and Mazumdar [HM03]. Exercise 1.6.2 is also taken from there.

1.6 Exercises

Exercise 1.6.1 (Compute clustering radius.) [10 Points]

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the ***covering radius*** of P by C (i.e., if we place a disk of radius r around each point of C all those disks covers the points of P).

(A) Give a $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \le \alpha \le 10r$.

(B) For $\varepsilon > 0$ a prescribed parameter, give a $O(n + k\varepsilon^{-2} \log n)$ expected time algorithm that outputs a number α, such that $\alpha \le r \le (1 + \varepsilon)\alpha$.

Exercise 1.6.2 (Randomized k-enclosing disk.) [5 Points]

Given a set P of n points in the plane, and parameter k, present a (simple) randomized algorithm that computes, in expected $O(n(n/k))$ time, a circle D that contains k points of P, and radius$(D) \le 2r_{\text{opt}}(\mathsf{P}, k)$.

(This is a faster and simpler algorithm than the one presented in Lemma 1.3.1.)

Chapter 2

Quadtrees - Hierarchical Grids

In this chapter, we discuss quadtrees which is arguably one of the simplest and most powerful geometric data-structure. We begin in Section 2.1 by giving a simple application of quadtrees and describe a clever way for performing point-location queries quickly in such a quadtree. In Section 2.2, we describe how such quadtrees can be compressed and how can they be quickly constructed and used for point-location queries. In Section 2.3 we describe a randomized extension of this data-structure, known as *skip-quadtree*, which enables us to maintain the compressed quadtree efficiently under insertions and deletions. In Section 2.5, we turn our attention to applications of compressed quadtrees, showing how quadtrees can be used to compute good triangulations of an input point set.

2.1 Quadtrees - a simple point-location data-structure

Let P_{map} be a planar map. To be more concrete, let P_{map} be a partition of the unit square into triangles (i.e., a mesh). The partition P_{map} can represent any planar map, where a region in the map might be composed of several triangles. For the sake of simplicity, assume that every vertex in P_{map} shares at most, say, nine triangles.

Let us assume that we want to preprocess P_{map} for point-location queries. Of course, there are data-structures that can do it with $O(n \log n)$ preprocessing time, linear space, and logarithmic query time. Instead, let us consider the following simple solution (which in the worst case, can be much worse).

Build a tree $\mathcal{T}$, where the root corresponds to the unit square. Every node $v \in T$ corresponds to a cell $\square_v$ (i.e., a square), and it has four children. The four children correspond to the four squares formed by splitting $\square_v$ into four equal size squares, by horizontal and vertical cuts. The construction is recursive, and we start from $v = \text{root}_T$. As long as the current node intersects more than, say, nine triangles, we create its children nodes, and we call recursively on each child, with the list of input triangles that intersect its square. We stop at a leaf, if its "conflict-list" (i.e., list of triangles it intersects) is of size at most nine. We store this conflict-list in the leaf.

Given a query point q, in the unit square, we can compute the triangle of P_{map} containing q, by traversing down $\mathcal{T}$ from the root, repeatedly going into the child of the current node, whose square contains q. We stop at soon as we reach a leaf, and then we scan the leaf conflict-list, and check which of the triangles contains q.

Of course, in the worst case, if the triangles are long and skinny, this quadtree might have unbounded complexity. However, for reasonable inputs (say, the triangles are fat), then the quadtree would have linear complexity in the input size (see Exercise 2.7.1) .The big advantage of quadtrees of course, is their simplicity. In a lot of cases, quadtree would be a sufficient solution, and seeing how to solve a problem using a quadtree might be a first insight into a problem.

2.1.1 Fast point-location in a quadtree

One possible interpretation of quadtrees is that they are a multi-grid representation of a point-set. In particular, given a node v, with a square S_v, which is of depth i (the root has depth zero), then the side length of S_v is 2^{-i}, and it is a square in the grid $\mathsf{G}_{2^{-i}}$. In fact, we will refer to $\ell(v) = -i$ as the ***level*** of v. However, a cell in a grid has a unique ID made out of two integer numbers. Thus, a node v of a quadtree is uniquely defined by the triple $\mathrm{id}(v) = (\ell(v), \lfloor x/r \rfloor, \lfloor y/r \rfloor)$, where (x, y) is any point in $\square_v$, and $r = 2^{\ell(v)}$.

Furthermore, given a query point q, and a desired level ℓ, we can compute the ID of the quadtree cell of this level that contains q in constant time. Thus, this suggests a very natural algorithm for doing a point-location in a quadtree: Store all the IDs of nodes in the quadtree in a hash-table, and also compute the maximal depth h of the quadtree. Given a query point q, we now have access to any node along the point-location path of q in $\mathcal{T}$, in constant time. In particular, we want to find the point in $\mathcal{T}$ where the point-location path "falls off" the quadtree. This we can find by performing a binary search for the dropping off point. Let **QTGetNode**(T, q, d) denote the procedure that, in constant time, returns the node v of depth d in the quadtree $\mathcal{T}$ such that $\square_v$ contains the point q. Given a query point q, we can perform point-location in $\mathcal{T}$ by calling **QTFastPntLocInner**$(T, q, 0, \mathrm{height}(T))$. See Figure 2.1 for the pseudo-code for **QTFastPntLocInner**.

```
QTFastPntLocInner(T, q, lo, hi).
    mid ← ⌊(lo + hi)/2⌋
    v ← QTGetNode(T, q, mid)
    if v = null then
       return QTFastPntLocInner(T, q, lo, mid − 1).
    w ← Child(v, q)
         //w is child of v containing the point q.
    If w = null then
       return v
    return QTFastPntLocInner(T, q, mid + 1, hi)
```

Figure 2.1: One can perform point-location in a quadtree $\mathcal{T}$ by calling **QTFastPntLocInner** $(T, q, 0, \mathrm{height}(T))$.

Lemma 2.1.1 *Given a quadtree $\mathcal{T}$ of size n and of height h, one can preprocess it in linear time, such that one can perform a point-location query in $\mathcal{T}$ in $O(\log h)$ time. In particular, if the quadtree has height $O(\log n)$ (i.e., it is "balanced"), then one can perform a point-location query in $\mathcal{T}$ in $O(\log \log n)$ time.*

2.2 Compressed Quadtrees: Range Searching Made Easy

Definition 2.2.1 (Spread.) For a set P of n points in a metric space, let

$$\Phi(\mathsf{P}) = \frac{\max_{p,q \in \mathsf{P}} \|p - q\|}{\min_{p,q \in P, p \neq q} \|p - q\|}$$

be the ***spread*** of P. In words, the spread of P is the ratio between the diameter of P and the distance between the two closest points. Intuitively, the spread tells us the range of distances that P posses.

One can build a quadtree $\mathcal{T}$ for P, storing the points of P in the leaves of $\mathcal{T}$, where one keep splitting a node as long as it contains more than one point of P. During this recursive construction, if a leaf contains no points of P, we save space by not creating this leaf, and instead creating a null pointer in the parent node for this child.

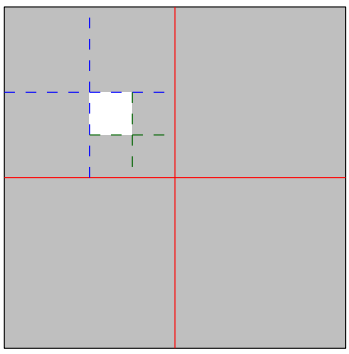
Lemma 2.2.2 *Let P be a set of n points in the unit square, such that $\mathrm{diam}(\mathsf{P}) = \max_{p,q \in \mathsf{P}} \|p - q\| \geq 1/2$. Let $\mathcal{T}$ be a quadtree of P constructed over the unit square. Then, the depth of $\mathcal{T}$ is bounded by $O(\log \Phi(\mathsf{P}))$, it can be constructed in $O(n \log \Phi(\mathsf{P}))$ time, and the total size of $\mathcal{T}$ is $O(n \log \Phi(\mathsf{P}))$.*

Proof: The construction is done by a straightforward recursive algorithm as described above.

Let us bound the depth of $\mathcal{T}$. Consider any two points $p, q \in P$, and observe that a node v of $\mathcal{T}$ of level $u = \lfloor \lg \|p - q\| \rfloor - 1$ containing p must not contain q (we remind the reader that $\lg n = \log_2 n$). Indeed, the diameter of $\square_v$ is smaller than $\sqrt{2}2^u < \sqrt{2}\,\|pq\|/2 < \|p - q\|$. Thus, $\square_v$ can not contain both p and q. In particular, any node of $\mathcal{T}$ of level $r = -\lfloor \lg \Phi \rfloor - 1$ can contain at most one point of P, where $\Phi = \Phi(P)$. Thus, all the nodes of $\mathcal{T}$ are of depth $O(\log \Phi)$.

Since the construction algorithm spends $O(n)$ time at each level of $\mathcal{T}$, it follows that the construction time is $O(n \log \Phi)$, and this also bounds the size of the quadtree $\mathcal{T}$. ■

The bounds of Lemma 2.2.2 are tight, as one can easily verify, see Exercise 2.7.2. But in fact, if you inspect a quadtree generated by Lemma 2.2.2, you would realize that there are a lot of nodes of $\mathcal{T}$ which are of degree one (the degree of a node is the number of children it has). Indeed, a node v of $\mathcal{T}$ has degree larger than one, only if it has two children, and let P_v be the subset of points of P stored in the subtree of v. Such a node v splits P_v into, at least, two subsets and globally there can be only $n - 1$ such splitting nodes.



Thus, a quadtree $\mathcal{T}$ contains a lot of "useless" nodes. We can replace such a sequence of edges by a single edge. To this end, we will store inside each quadtree node v, its square $\square_v$, and its level $\ell(v)$. Given a path of vertices in the quadtree that are all of degree one, we will replace them with a single vertex that corresponds to the first vertex in this path, and its only child would be the last vertex in this path (this is the first node of degree larger than one). This ***compressed node*** has a single child, and the region rg_v that it is in "charge" of is an annulus, see the figure on the right. Otherwise, the ***region*** that a node is in charge of is a $\mathsf{rg}_v = \square_v$. The child corresponds to the inner square. We call the resulting tree a ***compressed quadtree***. Since any node that has only a single child is compressed, we can charge it to its parent, which has two children. Since there are at most $n - 1$ internal nodes in the new compressed quadtree that have degree larger than one, it follows that it has linear size (however, it still can have linear depth in the worst case).

As an application for such a compressed quadtree, consider the problem of counting how many points are inside a query rectangle r. We can start from the root of the quadtree, and recursively traverse it, going down a node only if its region intersects the query rectangle. Clearly, we will report all the points contained inside r. Of course, we have no guarantee about the query time, but in practice, this might be fast enough.

2.2.1 Efficient construction of compressed quadtrees

Let P be a set of n points in the unit square, with unbounded spread. We are interested in computing the compressed quadtree of P. The regular algorithm for computing a quadtree when applied to P might required unbounded time. Modifying it so it requires only quadratic time is an easy exercise.

Instead, compute in linear time a disk D of radius r, which contains at least $n/10$ of the points of P, such that $r \leq 2r_{\mathrm{opt}}(\mathsf{P}, n/10)$, where $r_{\mathrm{opt}}(\mathsf{P}, n/10)$ denotes the radius of the smallest disk containing $n/10$ points. Computing D can be done in linear time, by a rather simple algorithm (Lemma 1.3.1).

Let $l = 2^{\lfloor \lg r \rfloor}$. Consider the grid G_l. It has a cell that contains $(n/10)/25$ points (since D is covered by $5 \times 5 = 25$ grid cells of G_l, since $l \geq r/2$), and no grid cell contains more than $5(n/10)$ points, by Lemma 1.4.2 (iii). Thus, compute $\mathsf{G}_l(P)$, and find the cell c containing the largest number of points. Let P_{in} be the points inside this cell c, and P_{out} the points outside this cell. We know that $|P_{\mathrm{in}}| \geq n/250$, and $|P_{\mathrm{out}}| \geq n/2$. Next, compute the compressed quadtrees for P_{in} and P_{out}, respectively, and let $\mathcal{T}_{\mathrm{in}}$ and $\mathcal{T}_{\mathrm{out}}$ denote the respective quadtrees. Since the cell of the root of $\mathcal{T}_{\mathrm{in}}$ has side length which is a power of two, and it belongs to the grid G_l, it follows that c represents a valid region, which can be a node in $\mathcal{T}_{\mathrm{out}}$ (note that if it is a node in $\mathcal{T}_{\mathrm{out}}$, then it is empty). Thus, we can do a point-location query in $\mathcal{T}_{\mathrm{out}}$, and hang the root of $\mathcal{T}_{\mathrm{in}}$

in the appropriate node of $\mathcal{T}_{\text{out}}$. This takes linear time (ignoring the time to construct $\mathcal{T}_{\text{in}}$ and $\mathcal{T}_{\text{out}}$). Thus, the overall construction time is $O(n \log n)$.

Theorem 2.2.3 *Given a set P of n points in the plane, one can compute a compressed quadtree of P in $O(n \log n)$ deterministic time.*

Definition 2.2.4 (Canonical square and grid.) A square is a ***canonical square***, if it is contained inside the unit square, it is a cell in a grid G_r, and r is a power of two (i.e., it might correspond to a node in a quadtree). We will refer to such a grid G_r, as a ***canonical grid***.

For reasons that would become clear later, we want to construct the quadtree out of a list of quadtree nodes that must appear in the quadtree. Namely, we get a list of canonical grid cells that must appear in the quadtree (i.e., the level of the node, together with its grid ID).

Lemma 2.2.5 *Given a list C of n canonical squares, all lying inside the unit square, one can construct a compressed quadtree $\mathcal{T}$ such that for any square $c \in C$, there exists a node $v \in T$, such that $\square_v = c$. The construction time is $O(n \log n)$.*

Proof: The construction is similar to Theorem 2.2.3. Let P be a set of n points, where $p_c \in \mathsf{P}$, if $c \in C$, and p_c is the center of c. Next, find, in linear time, a canonical square C that contains at least $n/250$ points of P, and at most $n/2$ points of P. Let U be the list of all squares of C that contain c, let $\mathcal{C}_{\text{in}}$ be the list of squares contained inside c, and let $\mathcal{C}_{\text{out}}$ be the list of squares of C that do not intersect the interior of c. Recursively, build a compressed quadtree for $\mathcal{C}_{\text{in}}$‘ and $\mathcal{C}_{\text{out}}$, denoted by $\mathcal{T}_{\text{in}}$ and $\mathcal{T}_{\text{out}}$, respectively.

Next, sort the nodes of U in decreasing order of their level. Also, let π be the point-location path of c in $\mathcal{T}_{\text{out}}$. Clearly, adding all the nodes of U to $\mathcal{T}_{\text{out}}$ is no more than performing a merge of π together with the sorted nodes of U. Whenever we encounter a square of U that does not have a corresponding node at π, we create this node, and insert it into π. Let $\mathcal{T}'_{\text{out}}$ denote the resulting tree. Next, we just hang $\mathcal{T}_{\text{in}}$ in the right place in $\mathcal{T}'_{\text{out}}$. Clearly, the resulting quadtree has all the squares of C as nodes.

As for the running time, we have $T(C) = T(\mathcal{C}_{\text{in}}) + T(\mathcal{C}_{\text{out}}) + O(n) + O(|U| \log |U|) = O(n \log n)$, since $|\mathcal{C}_{\text{out}}| + |\mathcal{C}_{\text{in}}| + |U| = n$ and $|\mathcal{C}_{\text{in}}|, |\mathcal{C}_{\text{out}}| \leq (249/250)n$. ■

2.2.2 Fingering a Compressed Quadtree - Fast Point Location

Let $\mathcal{T}$ be a compressed quadtree of size n. We would like to preprocess it so that given a query point, we can find the lowest node of $\mathcal{T}$ whose cell contains a query point q. As before, we can perform this by traversing down the quadtree, but this might require $\Omega(n)$ time. Since the range of levels of the quadtree nodes is unbounded, we can no longer use binary search on the levels of $\mathcal{T}$ to answer the query.

Instead, we are going to use a rebalancing technique on $\mathcal{T}$. Namely, we are going to build a balanced tree $\mathcal{T}'$, which would have cross pointers (i.e., fingers) into $\mathcal{T}$. The search would be performed on $\mathcal{T}'$ instead of on $\mathcal{T}$. In the literature, the tree $\mathcal{T}$ is known as a ***finger tree***.

Definition 2.2.6 Let $\mathcal{T}$ be a tree with n nodes. A ***separator*** in $\mathcal{T}$ is a node v, such that if we remove v from $\mathcal{T}$, we remain with a forest, such that every tree in the forest has at most $\lceil n/2 \rceil$ vertices.

Lemma 2.2.7 *Every tree has a separator, and it can be computed in linear time.*

Proof: Consider $\mathcal{T}$ to be a rooted tree, and initialize v to be the root of $\mathcal{T}$. We perform a walk on $\mathcal{T}$. If v is not a separator, then one of the children of v in $\mathcal{T}$ must have a subtree of $\mathcal{T}$ of size $\geq \lceil n/2 \rceil$ nodes. Set v to be this node. Continue in this walk, till we get stuck. The claim is that v is the required node. Indeed, since we always go down, and the size of the subtree shrinks, we must get stuck. Thus, consider w as the node

we got stuck at. Clearly, the subtree of w contains at least $\lceil n/2 \rceil$ nodes (otherwise, we would not set $v = w$). Also, all the subtrees of w have size $\leq \lceil n/2 \rceil$, and the connected component of $T \setminus \{w\}$ containing the root contains at most $n - \lceil n/2 \rceil \leq \lfloor n/2 \rfloor$ nodes. Thus, w is the required separator. ■

This suggests a natural way for processing a compressed quadtree for point-location queries. Find a separator $v \in T$, and create a root node f_v for $\mathcal{T}'$ which has a pointer to v; now recursively build finger trees to each tree of $T \setminus \{v\}$, and hang them on w. Given a query point q, we traverse $\mathcal{T}'$, where at node $f_v \in \mathcal{T}'$, we check whether the query point $q \in \square_v$, where v is the corresponding node of $\mathcal{T}$. If $q \notin \square_v$, we continue the search into the child of f_v, which corresponds to the connected component outside $\square_v$ that was hung on f_v. Otherwise, we continue into the child that contains q. This takes constant time per node. As for the depth for the finger tree $\mathcal{T}'$, observe $D(n) \leq 1 + D(\lceil n/2 \rceil) = O(\log n)$. Thus, a point-location query in $\mathcal{T}'$ takes logarithmic time.

Theorem 2.2.8 *Given a compressed quadtree $\mathcal{T}$ of size n, one can preprocess it in $O(n \log n)$ time, such that given a query point q, one can return the lowest node in $\mathcal{T}$ whose region contains q in $O(\log n)$ time.*

2.3 Dynamic Quadtrees

What if we want to maintain the compressed quadtree under insertions and deletions? There is an elegant way of doing this by using randomization. The resulting structure has similar behavior to skip-list where instead of linked list we use quadtrees.

We remind the reader the concept of gradation:

Definition 2.3.1 (*Gradation.*) Given a set P of n points, a *sampling sequence* $(S_m, \ldots, S_1)$ of P is a sequence of subsets of P, such that (i) $S_1 = \mathsf{P}$, (ii) S_i is formed by picking each point of S_{i-1} with probability $1/2$, and (iii) $|S_m| \leq 2k$, and $|S_{m-1}| > 2k$, where k is some prespecified constant. The sequence $(S_m, S_{m-1}, \ldots, S_1)$ is called a ***gradation*** of P.

Let $\mathcal{T}_1, \ldots, \mathcal{T}_m$ be the quadtrees of the sets $\mathsf{P} = S_1, \ldots, S_m$, respectively. Note, that the nodes of $\mathcal{T}_i$ are a subset of the nodes appear in $\mathcal{T}_{i-1}$. As such, every node in $\mathcal{T}_i$ would have pointers to its own copy in $\mathcal{T}_{i-1}$ and a pointer to its copy in $\mathcal{T}_{i+1}$ if it exists there. We will refer to this data-structure as ***skip-quadtree***.

Point-location queries. Given a query point q we want to find the leaf of $\mathcal{T}_1$ that contains it. The search algorithm is quite simple, starting at $\mathcal{T}_m$ you find the leaf in $\mathcal{T}_i$ that contains the query point, and then move to the corresponding node in $\mathcal{T}_{i-1}$, and continue the search from there.

2.3.1 Inserting a point into the skip-quadtree.

Let p be the point to be inserted into the skip-quadtree. We perform a point-location query and find the lowest node v in $\mathcal{T}_1$ that contains p. Next, we split v and establish a new node (hanging it from v) that contains p. Now, we flip an unbiased coin, if the coin comes up tail, we are done. Otherwise, we add p to $\mathcal{T}_2$. We continue in this fashion, adding p to the quadtrees in the relevant levels, till the coin comes up tail.

Note, that the amount of work and space needed at each level is a constant (ignoring the initial point-location query), and by implementing this operation carefully, the time to perform it would be proportional to the point-location query time.

Deleting a point from the skip-quadtree is done in a similar fashion to the insertion described above.

Given a point-set P, **constructing** the skip-quadtree can be done by inserting the points of P one by one into the skip-quadtree.

2.3.2 Running time analysis

We analyze the time needed to perform a point-location query. In particular, we claim that the expected query time $O(\log n)$. To see that we will use the standard backward analysis. Consider the leaf v of $\mathcal{T}_i$ that contains q, and consider the path $v = v_1, v_2, \ldots, v_r$ from v to the root v_r of the compressed quadtree $\mathcal{T}_i$. Let π denote this path. Clearly, the amount of time spent in the search in the tree $\mathcal{T}_i$, is proportional to how far we have to go on this list, till we hit a node that appears in $\mathcal{T}_{i+1}$. (During the point-location we traverse this snippet of the path in the other direction - we are in a leaf of $\mathcal{T}_{i+1}$, we jump into the corresponding node of $\mathcal{T}_i$, and traverse down till we reach a leaf.) Note, that the node v_j stores (at least) j points of S_i, and if any pair of them appears in S_{i+1} then at least one of the nodes on π below the node v_j would appear in $\mathcal{T}_{i+1}$ (since the quadtree $\mathcal{T}_{i+1}$ needs to "separate" these two points and this is done by a node of the path). Let denote this set of points by U_j, and let X_j be an indicator variable which is one if v_j does not appear in $\mathcal{T}_{i+1}$. Clearly,

$$\mathbf{E}\Big[X_j\Big] = \mathbf{Pr}\Big[\text{no pair of points of } U_j \text{ is in } T_{i+1}\Big] \leq \frac{j+1}{2^j},$$

since the points of S_i are randomly and independently chosen to be in S_{i+1} and the event happens only if zero or one points of U_j are in S_{i+1}.

Thus, the expected search time in T_i is $\mathbf{E}\Big[\sum_j X_j\Big] = \sum_j \mathbf{E}\Big[X_j\Big] = \sum_j (j+1)/2^j = O(1)$.

Thus, the overall expected search time in the skip-quadtree is proportional to the number of levels in the gradation.

Let $Z_i = |S_i|$ be the number elements stored in the ith level of the gradation. We know that $Z_1 = n$, and $\mathbf{E}[Z_i] =$. In particular, $\mathbf{E}[Z_i] = \mathbf{E}[\mathbf{E}[Z_i]] = \mathbf{E}[Z_{i-1}/2] = \cdots = n/2^{i-1}$. Thus, $\mathbf{E}[Z_\alpha] \leq 1/n^{10}$, where $\alpha = \lceil 11 \lg n \rceil$. Thus, by Markov's inequality, we have that

$$\mathbf{Pr}[m > \alpha] = \mathbf{Pr}[Z_\alpha \geq 1] \leq \frac{\mathbf{E}[Z_\alpha]}{1} = \frac{1}{n^{10}}.$$

We summarize:

Lemma 2.3.2 *A gradation defined over n elements has $O(\log n)$ levels both in expectation and with high probability.*

This implies that a point-location query in the skip quadtree takes, in expectation, $O(\log n)$ time.

Since, with high probability, there are only $O(\log n)$ levels in the gradation, it follows that the expected search time is $O(\log n)$.

High Probability. In fact, one can show that this point-location query time bound holds with high probability, and we sketch (informally) the argument why this is true. Consider the variable Y_i that is the number of nodes of T_i being visited during the point-location query. We have that $\mathbf{Pr}[Y_i \geq k] \leq \sum_{j=k} (k+1)/2^k = O(k/2^k) = O(1/c^k)$, for some constant $c > 1$. Thus, we have a sum of logarithmic number of independent random variables, each one of them behaves like a variable with geometric distribution. As such, we can apply a Chernoff-type inequality (see Exercise 25.4.3) to get an upper bound on the probably that the sum of these variables exceeds $O(\log n)$. This probability is bounded by $1/n^{O(1)}$.

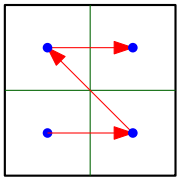
Note, the longest query time is realized by one of the points stored in the quadtree. Since there are n points stored in the quadtree, this implies that with high probability *all* point-location queries takes $O(\log n)$ time. Also, observe that the structure of the skip-quadtree is uniquely determined by the gradation. Since the gradation is oblivious to the history of the data-structure (i.e., what points where inserted and deleted). As such, these bounds on the performance hold at any point in time during the usage of the skip-quadtree. We summarize:

Theorem 2.3.3 *Let $\mathcal{T}$ be an empty skip-quadtree used for a sequence of n operations (i.e., insertions, deletions and point-location queries). Then, with high probability (and thus also in expectation), the time to perform each such operation takes $O(\log n)$ time.*

2.4 Even More on Dynamic Quadtrees

The previous section reveals that quadtrees can be maintained dynamically using ideas similar to skip-lists. Here we will take this idea even further, showing how to facilitate quadtrees using any data-structure for ordered sets.

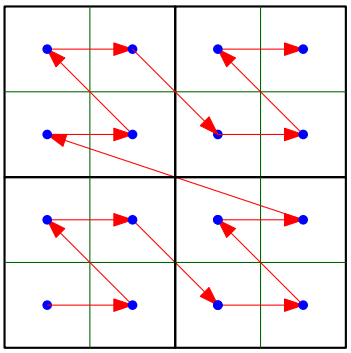
2.4.1 Ordering of nodes and points



So consider a quadtree $\mathcal{T}$, and a **DFS** traversal of $\mathcal{T}$, where the **DFS** always traverse the children of a node in the same relative order (i.e., say, first the bottom-left child, then the bottom-right child, top-left child, and top-right child).

Consider any two canonical squares $\square$ and $\widehat{\square}$, and imagine a quadtree $\mathcal{T}$ that contains both squares (i.e., there are nodes in $\mathcal{T}$ with these squares as their cells). Notice, that the above **DFS** would always visit these two nodes in a specific order, independent of the structure of the rest of the quadtree. Thus, if $\square$ gets visited before $\widehat{\square}$, we denote this fact by $\square \prec \widehat{\square}$. This defines a total ordering over all canonical squares. It would be in fact useful to extend this ordering to also includes points. Thus, consider a point p and a canonical square $\square$. If $p \in \square$ then we will say that $\square \prec p$. Otherwise, if $\square \in \mathsf{G}_i$, let $\widehat{\square}$ be the cell in G_i that contains p. We have that $\square \prec p$ if and only if $\square \prec \widehat{\square}$. Next, consider two points p and q, and let G_i be a grid fine enough such that p and q lie in two different cells, say, $\square_p$ and $\square_q$, respectively. Then $p \prec q$ if and only if $\square_p \prec \square_q$.

We will refer to the ordering induced by $\prec$ as the ***$\mathcal{Q}$-order***.



The ordering $\prec$ when restricted only to points, is the ordering along a space filling mapping that is induced by the quadtree **DFS**. This ordering is know as the ***$\mathcal{Z}$-order*** . Note, however, that since we allow comparing cells to cells, and cells to points, the $\mathcal{Q}$-order no longer has this exact interpretation. Furthermore, unlike the Peano or Hilbert curve, our mapping is not continuous. Our mapping has the advantage of being easy to define. Indeed, given a real number $\alpha \in [0,1)$, with the binary expansion $\alpha = 0.x_1x_2x_3\ldots$ (i.e., $\alpha = \sum_{i=1}^{\infty} x_i 2^{-i}$), our mapping will map it to the point $(0.x_2x_4x_6\ldots, 0.x_1x_3x_5\ldots)$.

2.4.1.1 Computing the $\mathcal{Q}$-order quickly

For our algorithmic applications, we need to be able to find the ordering according to $\prec$ between any two given cells/points quickly. To this end, let $\mathcal{LCA}(p,q)$ of two points $p, q \in [0,1]^2$ denote the smallest canonical square that contains both p and q. To compute this, let $\mathrm{bit}_\Delta(\alpha,\beta)$ of two real numbers $\alpha,\beta \in [0,1)$ be the index of the first bit after the period in which they differ. Thus, $\mathrm{bit}_\Delta(1/4, 3/4) = 1$ and $\mathrm{bit}_\Delta(7/8, 3/4) = 3$. Clearly, the level ℓ of $\square = \mathcal{LCA}(p,q)$ is equal to

$$\ell = \min(\mathrm{bit}_\Delta(x_p, x_q), \mathrm{bit}_\Delta(y_p, y_q)) - 1,$$

where x_p and y_p denote the x and y coordinates of p, respectively. Thus, the side length of $\square = \mathcal{LCA}(p,q)$ is $\Delta = 2^{-\ell}$. Let $x' = \lfloor x/\Delta \rfloor$ and $y' = \lfloor y/\Delta \rfloor$. Thus,

$$\mathcal{LCA}(p,q) = [x', x'+\Delta) \times [y', y'+\Delta).$$

The $\mathcal{LCA}$ of two cells is just the $\mathcal{LCA}$ of their centers.

Now, given two cells $\square$ and $\widehat{\square}$, we would like to determine their $\mathcal{Q}$-order. If $\square \subseteq \widehat{\square}$ then $\widehat{\square} \prec \square$. If $\widehat{\square} \subseteq \square$ then $\square \prec \widehat{\square}$. Otherwise, let $\widetilde{\square} = \mathcal{LCA}(\square, \widehat{\square})$. We can now determine which children of $\widetilde{\square}$ contains these two cells, and since we know the traversal ordering among children of a node in a quadtree we can now resolve this query in constant time.

Corollary 2.4.1 *Assuming that the* bit_Δ *operation and the* $\lfloor \cdot \rfloor$ *operation can be performed in constant time, then one can compute* $\mathcal{LCA}$ *of two points (or cells) in constant time. Similarly, the* $\mathcal{Q}$*-order can be resolved in constant time.*

Computing bit_Δ **efficiently.** It seems somewhat suspicious that one assumes that the bit_Δ operations can be done in constant time on a classical RAM machine. However it is a reasonable assumption on a real world computer. Indeed, in floating point representation, once you are given a number it is easy to access its mantissa and exponent in constant time. If the exponents are different then bit_Δ can be computed in constant time. Otherwise, we can easily *xor* the mantissas of both numbers, and compute the most significant bit that is on. This can be done in constant time by converting the xored mantissa into floating point number, and computing its $\log_2$ (some CPUs have this command built in). Observe, that all these operations are implemented in hardware in the CPU and require only constant time.

2.4.2 Performing a point-location in a quadtree

Let $\mathcal{T}$ be a given quadtree and a query point $\mathsf{q} \in [0,1]^2$. We would like to find the leaf v of $\mathcal{T}$ such that its cell contains q. We assume that $\mathcal{T}$ is given to us as a list of cells stored in an ordered-set data-structure, using the $\mathcal{Q}$-order over the cells.

To answer the query, we first find the two consecutive cells in this list such that $\square \prec \mathsf{q} \prec \widehat{\square}$. It is now easy to verify that $\square$ must be the quadtree leaf containing q. Indeed, let $\square_{\mathsf{q}}$ be the leaf of $\mathcal{T}$ that its cell contains q. By definition, we have that $\square_{\mathsf{q}} \prec \mathsf{q}$. Thus, the only bad scenario is that $\square_{\mathsf{q}} \prec \square \prec \mathsf{q}$. But this implies, by the definition of $\mathcal{Q}$-order, that $\square$ must be contained inside $\square_{\mathsf{q}}$ contradicting our assumption that $\square_{\mathsf{q}}$ is a leaf of the quadtree.

Lemma 2.4.2 *Given a quadtree* $\mathcal{T}$ *of size n, with its leaves stored in an ordered-set data-structure* $\mathcal{D}$ *according to the* $\mathcal{Q}$*-order, then one can perform point-location query in* $O(Q(n))$ *time, where* $Q(n)$ *is the time to perform a search query in* $\mathcal{D}$*.*

2.4.3 Overlaying two quadtrees

Given two quadtrees $\mathcal{T}'$ and $\mathcal{T}''$ we would like to overlay them to compute their combined quadtree. This is the minimal quadtree such that every cell of either $\mathcal{T}'$ or $\mathcal{T}''$ appears in it. Observe that if the two quadtrees are given as sorted lists of their cells (ordered by the $\mathcal{Q}$-order) then their overlay is just the merged list, with replication removed.

Lemma 2.4.3 *Given two quadtrees* $\mathcal{T}'$ *and* $\mathcal{T}''$ *given as sorted lists of their nodes, one can compute the merged quadtree in linear time (in their size) by merging the two sorted lists and removing duplicates.*

2.4.4 Point location in a compressed quadtree

so let $\mathcal{T}$ be a compressed quadtree, that its nodes are stored in an ordered-set data-structure. Let q be the query point. The required node v has $\mathsf{q} \in \mathsf{rg}_v$ and is either a leaf of the quadtree or a compressed node.

If the binary search using the $\mathcal{Q}$-order return a node v, such that $\square_v \prec \mathsf{q}$ then if $\mathsf{q} \in \square_v$ then we are done, as v is the required answer. So, if $\mathsf{q} \notin \square_v$ then it must be that the node u such that $\mathsf{q} \in \mathsf{rg}_u$ is a compressed node. As such, consider the cell $\square = \mathcal{LCA}(\square_v, \mathsf{q})$. Clearly, the compressed node w that its region contains q have the property that $\square \subseteq \square_w$. Furthermore, let z be the only child of w. We have that $\square_z \subseteq \square \subseteq \square_w$. In particular, in the ordering of nodes by the $\mathcal{Q}$-order we have that $\square_w \prec \square \prec \square_z$, where $\square_w$ and $\square_z$ are consecutive in the ordering of the nodes of the compressed quadtree. It follows, that we can find $\square_w$ by doing an additional binary search for $\square$ in the ordered set of the nodes of the compressed quadtree. We summarize:

Lemma 2.4.4 *Given a compressed quadtree $\mathcal{T}$ of size n, with its leaves stored in an ordered-set data-structure $\mathcal{D}$ according to the $\mathcal{Q}$-order, then one can perform point-location query in $\mathcal{T}$ in $O(Q(n))$ time, where $Q(n)$ is the time to perform a search query in $\mathcal{D}$.*

2.4.5 Inserting/deleting a point into/from a compressed quadtree

Let q be a point to be inserted into the quadtree, and let w be the node of the compressed quadtree such that $\mathsf{q} \in \mathsf{rg}_w$. There are several possibilities:

- The node w is a leaf, and there is no point associated with it. Then we just stored p at w, and we are done.
- The node w is a leaf, and there is a point p already stored in w. In this case, let $\square = \mathcal{LCA}(\mathsf{p}, \mathsf{q})$, and insert $\square$ into the compressed quadtree. Furthermore, split $\square$ into its children, and also insert the children into the compressed quadtree. Finally, associate p with the new leaf that contains it, and associate q with the leaf that contains it. Note, that because of the insertion w becomes a compressed node if $\square_w \neq \square$, and it becomes a regular internal node otherwise.
- The node w is a compressed node. Let z be the child of w, and consider $\square = \mathcal{LCA}(\square_z, \mathsf{q})$. Insert $\square$ into the compressed quadtree if $\square \neq \square_w$ (note that in this case w would still be a compressed node, but with a larger "hole"). Also insert all the children of $\square$ into the quadtree, and store p in the appropriate child. Hang $\square_z$ from the appropriate child, and turn this child into a compressed node.

In all three cases, the insertion requires a constant number of search/insert operations on the ordered-set data-structure.

Deletion is done in a similar fashion. We delete the point from the node that contains it, and then we trim away nodes that are no longer necessary.

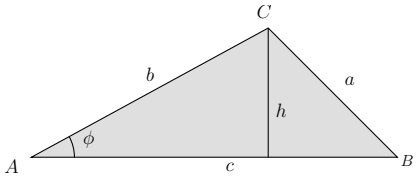
Theorem 2.4.5 *Assuming one can compute the $\mathcal{Q}$-order in constant time, then one can maintain a compressed quadtree of point-set in $O(\log n)$ time per operation, where insertion, deletion and point-location queries are supported. Furthermore, this can implemented using an ordered-set data-structure.*

2.5 Balanced quadtrees, and good triangulations

The ***aspect ratio*** of a convex body is the ratio between its longest dimension and its shortest dimension. For a triangle $\triangle = abc$, the aspect ratio $\mathcal{A}_{\text{ratio}}(\triangle)$ is the length of the longest side divided by the height of the triangle on the longest edge.

Lemma 2.5.1 *Let ϕ be the smallest angle for a triangle. We have that $1/\sin\phi \leq \mathcal{A}_{\text{ratio}}(\triangle) \leq 2/\sin\phi$.*

Proof: Consider the triangle $\triangle = \triangle abc$.



We have $\mathcal{A}_{\text{ratio}}(\triangle) = c/h$. However, $h = b \sin\phi$, and since a is the shortest edge in the triangle (since it is facing the smallest angle), it must be that b is the middle length edge. As such, $2b \geq a + b \geq c$. Thus, $\mathcal{A}_{\text{ratio}}(\triangle) \geq b/h = b/(b \sin\phi) = 1/\sin\phi$. And similarly, $\mathcal{A}_{\text{ratio}}(\triangle) \leq 2b/h = 2b/(b \sin\phi) = 2/\sin\phi$. ■

Another natural measure of sharpness is the ***edge ratio*** $E_{\text{ratio}}(\triangle)$, which is the ratio between a triangle's longest and shortest edges. Clearly, $\mathcal{A}_{\text{ratio}}(\triangle) > E_{\text{ratio}}(\triangle)$, for any triangle $\triangle$. For a triangulation $\mathcal{M}$, we denote by $\mathcal{A}_{\text{ratio}}(\mathcal{M})$ the maximum aspect ratio of a triangle in $\mathcal{M}$. Similarly, $E_{\text{ratio}}(\mathcal{M})$ denotes the maximum edge ratio of a triangle in $\mathcal{M}$.

Definition 2.5.2 A ***corner*** of a quadtree cell is one of the four vertices of its square. The *corners* of the quadtree are the points that are corners of its cells. We say that the side of a cell is *split* if either of the neighboring boxes sharing it is split. A quadtree is *balanced* if any side of an unsplit cell may contain only one quadtree corner in its interior. Namely, adjacent leaves are either of the same level, or of adjacent levels.

Lemma 2.5.3 *Let* P *be a set of points in the plane, such that* $\text{diam}(P) = \Omega(1)$ *and* $\Phi = \Phi(P)$*. Then, one can compute a (minimal size) balanced quadtree* $\mathcal{T}$ *of* P*, in time* $O(n \log n + m)$ *time, where* m *is the size of the output quadtree.*

Proof: Compute a compressed quadtree $\mathcal{T}$ of P in $O(n \log n)$ time. Next, we traverse $\mathcal{T}$, and replace every compressed edge of $\mathcal{T}$ by the sequence of quadtree nodes that defines it. To guarantee the balance condition, we create a queue of the nodes of $\mathcal{T}$, and store the nodes of $\mathcal{T}$ in a hash table, with their IDs.

We handle the nodes in the queue, one by one. For a node v, we check whether the current adjacent nodes to $\square_v$ are balanced. Specifically, let c be one of $\square_v$'s neighboring cells in the grid of $\square_v$, and let c_p be the square containing c in a grid one level up. We compute $\text{id}(c), \text{id}(c_p)$, and check if there is a node in $\mathcal{T}$ with those IDs. If not, we create a node w with region c_p and $\text{id}(c_p)$, and recursively retrieve its parent (i.e., if it exists we retrieve it, otherwise, we create it), and hang w from the parent node. We credit the work involved in creating w to the output size. We add all the new nodes to the queue. We repeat the process till the queue is empty.

Since the algorithm never creates nodes smaller than the smallest cell in the original compressed quadtree, it follows that this algorithm terminates. It is also easy to argue by induction that any balanced quadtree of P must contain all the nodes we created. Overall, the running time of the algorithm is $O(n \log n + m)$, since the work associated with any newly created quadtree node is constant. ■

Definition 2.5.4 The ***extended cluster*** of a cell c in a quadtree $\mathcal{T}$ is the set of 5×5 neighboring cells of c in the grid containing c, which are all the cells in distance $< 2l$ from c, where l is the sidelength of c.

A quadtree $\mathcal{T}$ over a point set P is ***well-balanced***, if it is balanced, and for every leaf node v that contains a (single) point of P, we have the property that all the nodes of the extended cluster of v are *leaves* in $\mathcal{T}$ (i.e., none of them is split and has children), and they do not contain any other point of P. In fact, we will also require that for every non-empty node v, all the nodes of the extended cluster of v are nodes in the quadtree.

Lemma 2.5.5 *Given a point set* P *of* n *points in the plane, one can compute a well-balanced quadtree of* P *in* $O(n \log n + m)$ *time, where* m *is the size of the output quadtree.*

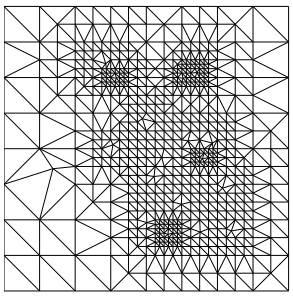


Figure 2.2: A well balanced triangulation.

Proof: We compute a balanced quadtree $\mathcal{T}$ of P. Next, for every leaf node v of $\mathcal{T}$ which contains a point of P, we verify that all its extended cluster are leaves of $\mathcal{T}$. If any other of the nodes of the extended cluster of v contains a point of P, we split v. If any of the extended cluster nodes is missing as a leaf, we insert it into the quadtree (with its ancestors if necessary). We repeat this process till we stop. Of course, during this process, we keep the balanced property valid, by adding necessary nodes. Clearly, all this work can be charged to newly created nodes, and as such takes linear time in the output size once the compressed quadtree is computed. ■

A well-balanced quadtree $\mathcal{T}$ of P provides for every point, a region (i.e., extended cluster) where it is well protected from other points. It is now possible to turn the partition of the plane induced by the leaves of $\mathcal{T}$ into a triangulation of P.

We "warp" the quadtree framework as follows. Let y be the corner nearest x of the leaf of $\mathcal{T}$ containing x; we replace y by x as a corner of the quadtree. Finally, we triangulate the resulting planar subdivision. Unwarped boxes are triangulated with isosceles right triangles by adding a point in the center. Only boxes with unsplit sides have warped corners; for these we choose the diagonal that gives better aspect ratio. Figure 2.2 shows a triangulation resulting from a variant of this method.

Lemma 2.5.6 *The method above gives a triangulation $\mathcal{QT}(P)$ with $\mathcal{A}_{\text{ratio}}(\mathcal{QT}(P)) \leq 4$.*

Proof: The right triangles used to triangulate the unwarped cells have aspect ratio 2. If a cell with side length l is warped, we have two cases.

In the first case, the input point of P is inside the square of the original cell. Then we assume that the diagonal touching the warped point is chosen; otherwise, the aspect ratio can only be better than what we prove. Consider one of the two triangles formed, with corners the input point and two other cell corners. The maximum length hypotenuse is formed when the warped point is on its original location, and has length $h = \sqrt{2}l$. The minimum area is formed when the point is in the center of the square, and has area $a = l^2/4$. Thus, the minimum height of such a triangle $\triangle$ is $\geq 2a/h$, and $\mathcal{A}_{\text{ratio}}(\triangle) \leq h/(2a/h) = h^2/2a = 4$.

In the second case, the input point is outside the original square. Since the quadtree is well balanced, the new point y is somewhere inside a square of sidelength l centered at x (since we always move the closest leaf corner to the new point). In this case, we assume that the diagonal not touching the warped point is chosen. This divides the cell into an isosceles right triangle and another triangle. If the chosen diagonal is the longest edge of the other triangle, then one can argue as before, and the aspect ratio is bounded by 4. Otherwise, the

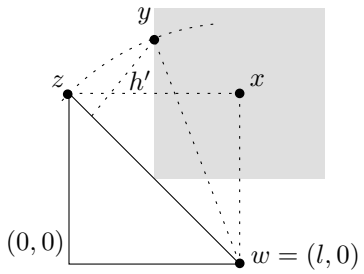


Figure 2.3: Illustration of the proof of Lemma 2.5.6.

longest edge touches the input point. The altitude is minimized when the triangle is isosceles with as sharp an angle as possible; see Figure 2.3. Using the notation of Figure 2.3, we have $y = (l/2, \sqrt{7}l/2)$. Thus,

$$\mu = \text{area}(\triangle wyz) = \frac{1}{2}\begin{vmatrix} 1 & 0 & l \\ 1 & l & 0 \\ 1 & l/2 & (\sqrt{7}/2)l \end{vmatrix} = \frac{1}{2}\begin{vmatrix} l & -l \\ l/2 & (\sqrt{7}/2-1)l \end{vmatrix} = \frac{\sqrt{7}-1}{4}l^2.$$

We have $h\sqrt{2}l/2 = \mu$, and thus $h' = \sqrt{2}\mu/l = \frac{\sqrt{7}-1}{2\sqrt{2}}l$. The longest distance y can be from w is $\alpha = \sqrt{(1/2)^2 + (3/2)^2}l = (\sqrt{10}/2)l$. Thus, the aspect ratio of the new triangle is bounded by $\alpha/h' = \left(\sqrt{10}/2\right) / \frac{\sqrt{7}-1}{2\sqrt{2}} \approx 2.717 \leq 4$. ■

For a triangulation $\mathcal{M}$, let $|\mathcal{M}|$ denote the number of triangles of $\mathcal{M}$. The ***Delaunay triangulation*** of a point set is the triangulation formed by all triangles defined by the points such that their circumscribing triangles are empty (the fact that this collection of triangles forms a triangulation requires a proof). Delaunay triangulations are extremely useful, and have a lot of useful properties. We denote by $\mathcal{DT}(P)$ the Delaunay triangulation of P.

Lemma 2.5.7 *There is a constant c', independent of P, such that $|\mathcal{QT}(P)| \leq c' \sum_{\triangle \in \mathcal{DT}(P)} \log E_{\text{ratio}}(\triangle)$.*

Proof: For this lemma, we modify the description of our algorithm for computing $\mathcal{QT}(P)$. We compute the compressed quadtree $\mathcal{T}''$ of P, and we uncompress the edges by inserting missing cells. Next, we split a leaf of $\mathcal{T}''$ if it has side length $\varkappa$, it is not empty (i.e., it contains a point of P), and there is another point of P of distance $\leq 2\varkappa$ from it. We refer to such a node as being ***crowded***. We repeat this, till there are no crowded leaves. Let $\mathcal{T}'$ denote the resulting quadtree. We now iterate over all the nodes v of $\mathcal{T}'$, and insert all the nodes of the extended cluster of v into $\mathcal{T}'$. Let $\mathcal{T}$ denote the resulting quadtree. It is easy to verify that $\mathcal{T}$ is well-balanced, and identical to the quadtree generated by the algorithm of Lemma 2.5.5 (although it is unclear how to implement the algorithm described here efficiently).

Now, all the nodes of $\mathcal{T}$ that were created when adding the extended cluster nodes can be charged to nodes of $\mathcal{T}'$. Therefore we need only count the total number of crowded cells in $\mathcal{T}'$.

Linearly many crowded cells have more than one child with points in them. It can happen at most linearly many times that a non-empty cell c has a point of P outside it of distance $2\varkappa$ from it, which in the next level is in a cell non-adjacent to the children of c, where $\varkappa$ is the side length of the cell, as this point becomes further away due to the shrinking sizes of cells as they split.

If a cell b containing a point is split because an extended neighbor was split, but no extended neighbor contains any point, then, when either b or b's parent was split, a nearby point became farther away than $2\varkappa$. Again, this can only happen linearly many times.

Finally a cell may contain two points, or several extended neighbor cells may contain points, and this situation may persist when the cells split. If splitting the children of the cell or of its neighbors separates the

points, we can charge linear total work. Otherwise, let Y be a maximal set of points in the union of cell b and its neighbors, such that splitting b, its neighbors, or the children of b and its neighbors does not further divide Y . Then some triangle of $\mathcal{DT}(P)$ connects two points y_1 and y_2 in Y with a point z outside Y.[2]

Each split not yet accounted for occurs between the step when Y is separated from z, and the step when y_1 and y_2 become more than $2\varkappa$ units apart. These steps are at most $O(\log E_{\text{ratio}}(\triangle y_1 y_2 z))$ quadtree levels apart, so we can charge all the crowded cells caused by Y to $\triangle y_1 y_2 z$. This triangle will not be charged by any other cells, because once we perform the splits charged to it all three points become far away from each other in the quadtree.

Therefore the number of crowded cells can be counted as a linear term, plus terms of the form $O(\log E_{\text{ratio}}(\triangle abc))$ for some Delaunay triangles $\triangle abc$. ■

Theorem 2.5.8 *Given any point set* P*, we can find a triangulation* $\mathcal{QT}(P)$ *such that each point of* P *is a vertex of* $\mathcal{QT}(P)$ *and* $\mathcal{A}_{\text{ratio}}(\mathcal{QT}(P)) \leq 4$*. There is a constant* c''*, independent of* P*, such that if* $\mathcal{M}$ *is any triangulation containing the points of* P *as vertices,* $|\mathcal{QT}(P)| \leq c'' |\mathcal{M}| \log \mathcal{A}_{\text{ratio}}(\mathcal{M})$.

In particular, any triangulation with constant aspect ratio containing P *is of size* $\Omega(\mathcal{QT}(P))$*. Thus, up to a constant,* $\mathcal{QT}(P)$ *is an optimal triangulation.*

Proof: Let Y be the set of vertices of $\mathcal{M}$. Lemma 2.5.7 states that there is a constant c such that $|\mathcal{QT}(Y)| \leq c \sum_{\triangle \in \mathcal{DT}(Y)} \log E_{\text{ratio}}(\triangle)$. The Delaunay triangulation has the property that it maximizes the minimum angle of the triangulation, among all triangulations of the point set [For97].

If $Y = P$, then using this maxminŋangle property, we have $\mathcal{A}_{\text{ratio}}(\mathcal{M}) \geq \frac{1}{2}\mathcal{A}_{\text{ratio}}(\mathcal{DT}(P)) \geq \frac{1}{2}E_{\text{ratio}}(\mathcal{DT}(P))$, by Lemma 2.5.1. Hence

$$|\mathcal{QT}(P)| \leq c \sum_{\triangle \in \mathcal{DT}(P)} \log E_{\text{ratio}}(\mathcal{DT}(P)) = c\,|\mathcal{M}|\, E_{\text{ratio}}(\mathcal{DT}(P)) \leq 2c\,|\mathcal{M}|\, \mathcal{A}_{\text{ratio}}(\mathcal{M}).$$

Otherwise, $P \subset Y$. Imagine running our algorithm on point set Y, and observe that $|\mathcal{QT}(P)| \leq |\mathcal{QT}(Y)|$. By the same argument as above, $|\mathcal{QT}(Y)| \leq c\,|\mathcal{M}| \log \mathcal{A}_{\text{ratio}}(\mathcal{M})$. ■

Corollary 2.5.9 $|\mathcal{QT}(P)| = O(n \log \mathcal{A}_{\text{ratio}}(\mathcal{DT}(P)))$.

Corollary 2.5.9 is tight, as can be easily verified.

2.6 Bibliographical notes

The authoritative text on quadtrees is the book by Samet [Sam89]. The idea of using hashing in quadtrees in a variant of an idea due to Van Emde Boas, and is also used in performing fast lookup in IP routing (using PATRICIA tries which are one dimensional quadtrees [WVTP97]), among a lot of other applications.

The algorithm described, in Section 2.2.1, for the efficient construction of compressed quadtrees is new, as far as I know. The classical algorithms for computing compressed quadtrees efficiently achieve the same running time, but require considerably more careful implementation, and paying careful attention to details [CK95, AMN+98]. The idea of fingering a quadtree is from [AMN+98] (although their presentation is different than ours).

The elegant skip-quadtree is from the recent work of Eppstein *et al.* [EGS05].

[2] To see that, observe that there must be an edge connecting a point $y_1 \in Y$ with a point $z \in P \setminus Y$ (since the triangulation is connected). Next, by going around y_1 and the points it is connected to, it is easy to observe that since Y diameter is (considerably) smaller then the distances between y_1 and z, there must be an edge between y_1 and another point y_2 of Y (for example, take y_2 to be the closest point in Y to y_1). This edge, together with the edge before it in the ordering around y_1, form the required triangle.

The idea of storing a quadtree in an ordered set by using the $\mathcal{Q}$-order on the nodes (or even only on the leaves) is due to Gargantini [Gar82], and it is referred to as ***linear quadtrees*** in the literature. The idea was used repeatedly for getting good performance in practice from quadtrees.

It is maybe beneficial to emphasize that if one does not require the internal nodes of the compressed quadtree for the application, then one can avoid storing them in the data-structure. In fact, if one is only interested in the point themselves, then can even skip storing the leaves themselves, and then the compressed quadtree just becomes a data-structure that stores the points according to their $\mathcal{Z}$-order. This approach can be used for example to construct a data-structure for approximate nearest neighbor [Cha02] (however, this data-structure is still inferior, in practice, to the more optimized but more complicated data-structure of Arya *et al.* [AMN+98]). The author finds that thinking about such data-structures as compressed quadtrees (with the whole additional unnecessary information) more intuitive, but the reader might disagree[②].

$\mathcal{Z}$-order and space filling curves. The idea of using $\mathcal{Z}$-order for speeding up spatial data-structures can be traced back to the above work of Gargantini [Gar82], and it is widely used in databases and seems to improve performance in practice [KF93]. The $\mathcal{Z}$-order can be viewed as a mapping from the unit interval to the unit-square, by splitting the odd bits, of a real number $\alpha \in [0,1)$, to be the x-coordinate and the even bits of α to encode the y-coordinate of the mapped point. While this mapping is simple to define it is not continuous. Somewhat surprisingly one can find a continuous mapping that maps the unit interval to the unit-square, see Exercise 2.7.4. A large family of such mappings is known by now, see Sagan [Sag94] for an accessible book on the topic.

But is it really practical? Quadtrees seems to be widely used in practice and perform quite well. Compressed quadtrees seems to be less widely used, but they have the benefit of being much simpler than their relatives which seems to be more practical but theoretically equivalent.

Good triangulations. Balanced quadtree and good triangulations are due to Bern *et al.* [BEG94], and our presentation closely follows theirs. The problem of generating good triangulations had received considerable attention recently, as it is central to the problem of generating good meshes, which in turn are important for efficient numerical simulations of physical processes. The main technique used in generating good triangulations is the method of Delaunay refinement. Here, one computes the Delaunay triangulation of the point set, and inserts circumscribed centers as new points, for "bad" triangles. Proving that this method converges and generates optimal triangulations is a non-trivial undertaking, and is due to Ruppert [Rup93]. Extending it to higher dimensions, and handling boundary conditions make it even more challenging. However, in practice, the Delaunay refinement method outperforms the (more elegant and simpler to analyze) method of Bern *et al.* [BEG94], which easily extends to higher dimensions. Namely, the Delaunay refinement method generates good meshes with fewer triangles.

Furthermore, Delaunay refinement methods are slower in theory. Getting an algorithm to perform Delaunay refinement in the same time as the algorithm of Bern *et al.* is still open, although Miller [Mil04] got an algorithm with only slightly slower running time.

Very recently, Alper Üngör came up with a "Delaunay-refinement type" algorithm, which outputs better meshes than the classical Delaunay refinement algorithm [Üng04]. Furthermore, by merging the quadtree approach with Üngör technique, one can get an optimal running time algorithm [HÜ05].

[②] The author reserves the right to disagree with himself on this topic in the future if the need arise.

2.7 Exercises

Exercise 2.7.1 (Quadtree for fat quadtrees.) [5 Points]

A triangle $\triangle$ is called α-fat if each one of its angles is at least α, where $\alpha > 0$ is a prespecified constant (for example, α is 5 degrees). Let P be a triangular planar map of the unit square (i.e., each face is a triangle), where all the triangles are fat, and the total number of triangles is n. Prove that the complexity of the quadtree constructed for P is $O(n)$.

Exercise 2.7.2 (Quadtree construction is tight.) [5 Points]

Prove that the bounds of Lemma 2.2.2 are tight. Namely, show that for any $r > 2$ and any positive integer $n > 2$, there exists a set of n points with diameter $\Omega(1)$ and spread $\Phi(P) = \Theta(r)$, and such that its quadtree has size $\Omega(n \log \Phi(P))$.

Exercise 2.7.3 (Cell queries.) [10 Points]

Let $\widehat{\square}$ be a canonical grid cell. Given a compressed quadtree $\widehat{T}$, we would like to find the *single* node $v \in \widehat{T}$, such that $P \cap \widehat{\square} = \mathsf{P}_v$. We will refer to such query as a ***cell query***. Show how to support cell queries in compressed quadtree in logarithmic time per query.

Exercise 2.7.4 (Space filling curve.) [10 Points]

The ***Peano curve*** $\sigma : [0, 1) \to [0, 1)^2$, maps a number $\alpha = 0.t_1t_2t_3 \ldots$ (the expansion is in base 3) to the point $\sigma(\alpha) = (0.x_1x_2x_3 \ldots, 0.y_1y_2 \ldots)$, where $x_1 = t_1$, $x_i = \phi(t_{2i-1}, t_2 + t_4 + \ldots + t_{2i-2})$, for $i \geq 1$. Here, $\phi(a, b) = a$ if b is even and $\phi(a, b) = 2 - a$ if b is odd. Similarly, $y_i = \phi(t_{2i}, t_1 + t_3 + \cdots + t_{2i-1})$, for $i \geq 1$.

(A) **[2 Points]** Prove that the mapping σ covers all the points in the open square $[0, 1)^2$, and it is one to one.

(B) **[8 Points]** Prove that σ is continuous.

Acknowledgments

The author wishes to thank John Fischer for his detailed comments on the manuscript.

Chapter 3

Well Separated Pairs Decomposition

In this chapter, we will investigate of how to represent distances between points efficiently. Naturally, an explicit description of the distances between n points requires listing all the $\binom{n}{2}$ distances. Here we will show that there is a considerably more compact representation which is sufficient if all we care about are approximate distances. This representation would have many nice applications.

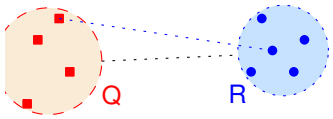
3.1 Well-separated pairs decomposition

Let P be a set of n points in $\mathbb{R}^d$, and $1/4 > \varepsilon > 0$ a parameter. One can represent the all distances between points of P by explicitly listing the $\binom{n}{2}$ pairwise distances. Of course, the listing of the coordinates of each point gives us an alternative more compact representation (of size dn), but its not a very informative representation. We interested in a representation that would capture the structure of the distances between the points.

As a concrete example, consider the three points on the right. We would like to have a representation that captures that p has similar distance to q and r, and furthermore, the q and r are close together as far as p is concerned. As such, if we are interested in the closest pair among the three points, we will only check the distance between q and r, since they are the only pair (among the three) that might realize the closest pair.

Figure 3.1

Denote by $A \otimes B = \left\{\{x, y\} \middle| x \in A,\ y \in B\right\}$ all the (unordered) pair of points formed by the sets A and B.. A pair of sets of points Q and R is $(1/\varepsilon)$***-separated*** if



$$\max(\operatorname{diam}(\mathsf{Q}), \operatorname{diam}(\mathsf{R})) \leq \varepsilon \cdot \mathbf{d}(\mathsf{Q}, \mathsf{R}),$$

where $\mathbf{d}(\mathsf{Q}, \mathsf{R}) = \min_{\mathsf{q}\in\mathsf{Q}, \mathsf{r}\in\mathsf{R}} \|\mathsf{q} - \mathsf{r}\|$. Intuitively, the pair $\mathsf{Q} \otimes \mathsf{R}$ is $(1/\varepsilon)$***-separated*** if all the points of Q have roughly the same distance to the points of R. Alternatively, imagine covering the two point sets with two balls of minimum size, and now we require that the distance between the two balls is at least $2/\varepsilon$ the radius of the larger of the two.

Thus, for the three points of Figure 3.1, the pairs $\{\mathsf{p}\} \otimes \{\mathsf{q}, \mathsf{r}\}$ and $\{\mathsf{q}\} \otimes \{\mathsf{r}\}$ are (say) 2-separated and described all the distances among these three points. (The gain here is quite marginal, as we replaced the distance description, made out of three pairs of points, by distance between two pairs of sets. But stay tuned – exciting things are about to unfold.)

Motivated by the above example, a well-separated pair decomposition is a way to describe a metric by such "well separated" pairs of sets.

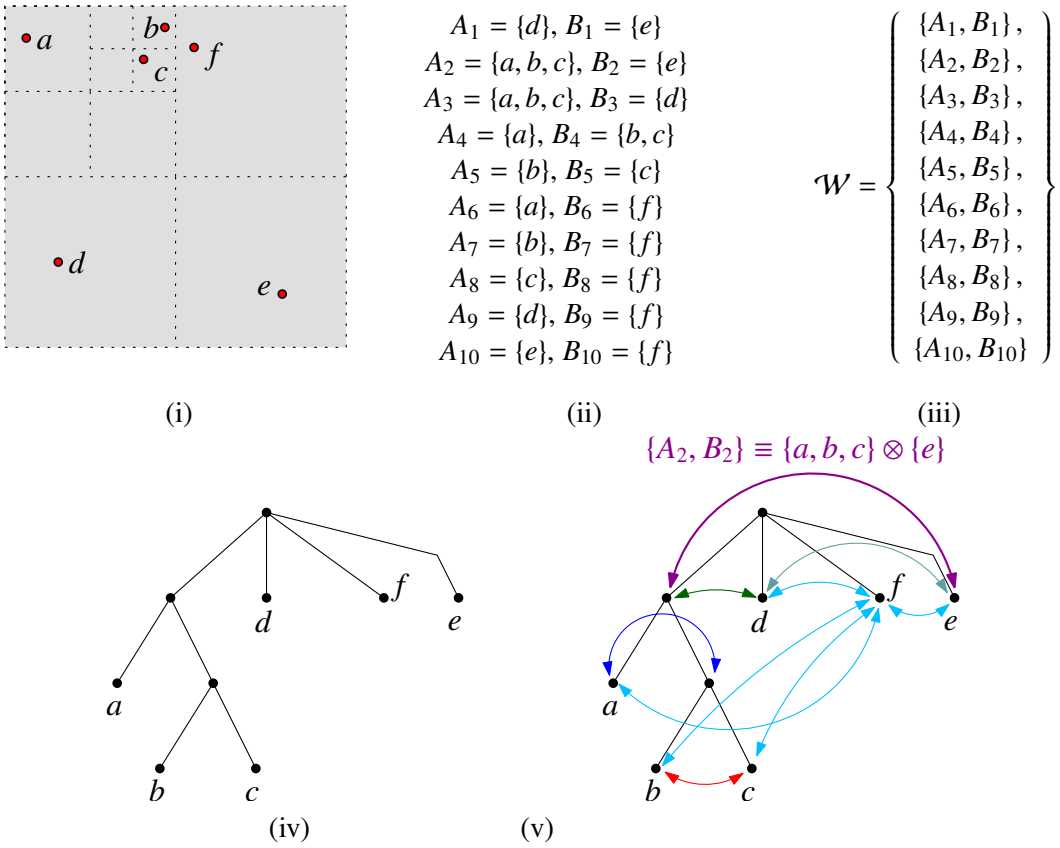


Figure 3.2: (i) A point set $\mathsf{P} = \{a, b, c, d, e\}$, (i) its decomposition into pairs, and (iii) its respective (1/2)-WSPD. For example, the pair of points b and e (and their distance) is represented by $\{A_2, B_2\}$ as $b \in A_2$ and $e \in B_2$. (iv) The quadtree $\mathcal{T}$ representing the point set P. (v) The WSPD as defined by pairs of vertices of $\mathcal{T}$.

Definition 3.1.1 (WSPD) A*well-separated pair decomposition* (***WSPD***) with parameter $1/\varepsilon$ of P is a set of pairs

$$\mathcal{W} = \left\{ \{A_1, B_1\}, \ldots, \{A_s, B_s\} \right\},$$

such that (A) $A_i, B_i \subset \mathsf{P}$ for every i.

(B) $A_i \cap B_i = \emptyset$ for every i.

(C) $\cup_{i=1}^{s} A_i \otimes B_i = \mathsf{P} \otimes \mathsf{P}$.

(D) The sets A_i and B_i are ε^{-1}-separated.

Translation: For any pair of points $\mathsf{p}, \mathsf{q} \in \mathsf{P}$, there is exactly one pair $\{A_i, B_i\} \in \mathcal{W}$ such that $\mathsf{p} \in A_i$ and $\mathsf{q} \in B_i$.

For a concrete example of a WSPD, see Figure 3.2.

Instead of maintaining such a decomposition explicitly, it is convenient to construct a tree $\mathcal{T}$ having the points of P as leaves, and every pair, (A_i, B_i) is just a pair of nodes (v_i, u_i) of $\mathcal{T}$, such that $A_i = \mathsf{P}_{v_i}$ and $B_i = \mathsf{P}_{u_i}$, where P_v denote the points of P stored in the subtree of v, where v is a node of $\mathcal{T}$. Naturally, in our case, the tree we would use is a compressed quadtree of P, but any tree that decomposes the points such that the diameter of a point set stored in a node drops quickly as we go down the tree might work.

This WSPD representation using a tree gives us a compact representation of the distances of the point set.

Corollary 3.1.2 *For a ε^{-1}-WSPD $\mathcal{W}$, it holds, for any pair $\{u, v\} \in \mathcal{W}$, that*

$$\forall \mathsf{q} \in \mathsf{P}_u, \mathsf{r} \in \mathsf{P}_v \qquad \max(\operatorname{diam}(\mathsf{P}_u), \operatorname{diam}(\mathsf{P}_v)) \leq \varepsilon \|\mathsf{q} - \mathsf{r}\| .$$

It would usually be convenient to associate with each set P_u in the WSPD, an arbitrary representative point $\mathrm{rep}_u \in \mathsf{P}$. Selecting and assigning these representative points can always be done by a simple DFS traversal of the $\mathcal{T}$ used to represent the WSPD.

3.1.1 The construction algorithm

The algorithm works by being greedy. It tries to put into the WSPD pairs of nodes in the tree that are as high as possible. In particular, if a pair $\{u, v\}$ would be generated than the pair formed by the parents of this pair of nodes will not be well separated. As such, the algorithm starts from the root, and try to separate it from itself. If the current pair is not well separated, then we replace the bigger node of the pair by its children (i.e., thus replacing a single pair by several pairs). Clearly, sooner or later this refinement process would reach well-separated pairs, which it would output. Since it considers all possible distances up front (i.e., trying to separate the root from itself), it would generate a WSPD covering all pairs of points.

Let $\Delta(v)$ denote the diameter of a cell associated with a node v of the quadtree $\mathcal{T}$. Formally, $\Delta(v) = 0$ if P_v is either empty or a single point. Otherwise, it is the diameter of the region associated with v; that is $\Delta(v) = \operatorname{diam}(\square_v)$, where $\square_v$ (we remind the reader) is the quadtree cube associated with the node v. Note, that since $\mathcal{T}$ is a compressed quadtree, we can always decide if $|\mathsf{P}_v| > 1$ by just checking if the subtree rooted at v has more than one node (since then this subtree must store more than one point).

We define the ***geometric distance*** between two nodes u and v of $\mathcal{T}$ to be

$$\mathbf{d}(u, v) = \mathbf{d}(\square_u, \square_v) = \min_{\mathsf{p} \in \square_u, \mathsf{q} \in \square_v} \|\mathsf{p} - \mathsf{q}\| .$$

AlgWSPD(u, v, T)
  if $\Delta(u) < \Delta(v)$ **then**
    Exchange u and v
  If $\Delta(u) \leq \varepsilon \cdot \mathbf{d}(u, v)$ **then**
    return $\{\{u, v\}\}$

  // $u_1, \ldots, u_r$ - the children of u
  return $\bigcup_{i=1}^{r}$ **AlgWSPD**(u_i, v, T).

Figure 3.3: The algorithm **AlgWSPD** for computing well-separated pairs decomposition. The nodes u and v belong to a compressed quadtree $\mathcal{T}$ of P.

We compute the compressed quadtree $\mathcal{T}$ of P in $O(n \log n)$ time. Next, we compute the WSPD by calling **AlgWSPD**(u_0, u_0), where u_0 is the root of $\mathcal{T}$ and **AlgWSPD** is depicted in Figure 3.3.

The following lemma is implied by an easy packing argument.

Lemma 3.1.3 *Let $\square$ be a cell of a grid G of $\mathbb{R}^d$ with cell diameter x. For $y \geq x$, the number of cells in G at distance at most y from $\square$ is $O\left((y/x)^d\right)$.*[3]

Lemma 3.1.4 *The WSPD generated by **AlgWSPD** is valid. Namely, for any pair $\{u, v\}$ in the WSPD, we have*

$$\max(\operatorname{diam}(\mathsf{P}_u), \operatorname{diam}(\mathsf{P}_v)) \leq \varepsilon \cdot \mathbf{d}(u, v) \quad \text{and} \quad \mathbf{d}(u, v) \leq \|\mathsf{q} - \mathsf{r}\| ,$$

for any $\mathsf{q} \in \mathsf{P}_u$ and $\mathsf{r} \in \mathsf{P}_v$.

[3] The $O(\cdot)$ notation here (and the rest of the chapter) hides a constant that depends on d.

Proof: For every output pair $\{u, v\}$, we have

$$\max\{\mathrm{diam}(\mathsf{P}_u), \mathrm{diam}(\mathsf{P}_v)\} \quad \leq \quad \max\{\Delta(u)\,, \Delta(v)\} \leq \frac{\varepsilon}{8}\mathbf{d}(u, v) \leq \varepsilon \cdot \mathbf{d}(u, v).$$

Also, for any $\mathsf{q} \in \mathsf{P}_u$ and $\mathsf{r} \in \mathsf{P}_v$, we have

$$\mathbf{d}(u, v) = \mathbf{d}(\square_u, \square_v) \leq \mathbf{d}(\mathsf{P}_u, \mathsf{P}_v) \leq \|\mathsf{q} - \mathsf{r}\|\,,$$

since $\mathsf{P}_u \subseteq \square_u$ and $\mathsf{P}_v \subseteq \square_v$.

Finally, by induction, it follows that every pair of points of P is covered by a pair of subsets $\{\mathsf{P}_u, \mathsf{P}_v\}$ output by the **AlgWSPD** algorithm. Note, that **AlgWSPD** always stops if both u and v are leafs, which implies that **AlgWSPD** always terminates. ■

Lemma 3.1.5 *For a pair $\{u, v\} \in \mathcal{W}$ computed by* **AlgWSPD***, we have that*

$$\max\Big(\Delta(u)\,,\ \Delta(v)\Big) \leq \min\Big(\Delta(\overline{\mathrm{p}}(u))\,,\ \Delta(\overline{\mathrm{p}}(v))\Big).$$

Proof: We trivially have that $\Delta(u) < \Delta(\overline{\mathrm{p}}(u))$ and $\Delta(v) < \Delta(\overline{\mathrm{p}}(v))$.

The pair $\{u, v\}$ was generated because of a sequence of recursive calls **AlgWSPD**(u_0, u_0), **AlgWSPD**(u_1, v_1), …, **AlgWSPD**(u_s, v_s), where $u_s = u$, $v_s = v$, and u_0 is the root of $\mathcal{T}$. Assume that $u_{s-1} = u$ and $v_{s-1} = \overline{\mathrm{p}}(v)$. Then $\Delta(u) \leq \Delta(\overline{\mathrm{p}}(v))$, since the algorithm always refine the larger cell (i.e., $v_{s-1} = \overline{\mathrm{p}}(v)$ in the pair $\{u_{s-1}, v_{s-1}\}$).

Similarly, let t be the last index such that $u_{t-1} = \overline{\mathrm{p}}(u)$ (namely, $u_t = u$ and $v_{t-1} = v_t$). Then, since v is an descendant of v_{t-1}, it holds that

$$\Delta(v) \leq \Delta(v_t) = \Delta(v_{t-1}) \leq \Delta(u_{t-1}) = \Delta(\overline{\mathrm{p}}(u))\,,$$

since (again) the algorithm always refines the larger cell. ■

Lemma 3.1.6 *The number of pairs in the computed WSPD is $O(n/\varepsilon^d)$.*

Proof: Let $\{u, v\}$ be an output pair. Consider the sequence (i.e., stack) of recursive calls that led to this output. In particular, assume that the last recursive call to **AlgWSPD**(u, v) was issued by **AlgWSPD**(u, v'), where $v' = \overline{\mathrm{p}}(v)$ is the parent of v in $\mathcal{T}$. Then

$$\Delta(\overline{\mathrm{p}}(u)) \geq \Delta(v') \geq \Delta(u)\,,$$

by Lemma 3.1.5.

We charge the pair $\{u, v\}$ to the node v', and claim that each node of $\mathcal{T}$ is charged at most $O(\varepsilon^{-d})$ times. To this end, fix a node $v' \in V(T)$, where $V(T)$ is the set of vertices of $\mathcal{T}$. Since the pair $\{u, v'\}$ was not output by **AlgWSPD** (despite being considered) we conclude that $8\Delta(v') > \varepsilon \cdot \mathbf{d}(u, v')$ and as such $\mathbf{d}(u, v') < r = 8\Delta(v')/\varepsilon$. Now, there are several possibilities:

(i) $\Delta(v') = \Delta(u)$. But there are at most $O\Big((r/\Delta(v'))^d\Big) = O(1/\varepsilon^d)$ nodes that have the same level (i.e., diameter) as v' and their cells are in distance at most r from it, by Lemma 3.1.3. Thus, this type of charge can happened at most $O(2^d \cdot (1/\varepsilon^d))$ times, since v' has at most 2^d children.

(ii) $\Delta(\overline{\mathrm{p}}(u)) = \Delta(v')$. By the same argumentation as above $\mathbf{d}(\overline{\mathrm{p}}(u), v') \leq \mathbf{d}(u, v') < r$. There are at most $O(1/\varepsilon^d)$ such nodes $\overline{\mathrm{p}}(u)$. Since the node $\overline{\mathrm{p}}(u)$ has at most 2^d children, it follows that the number of such charges is at most $O\Big(2^d \cdot 2^d \cdot (1/\varepsilon^d)\Big)$.

(iii) $\Delta(\overline{\mathrm{p}}(u)) > \Delta(v') > \Delta(u)$. Consider the canonical grid G having $\square_{v'}$ as one of its cells (see Definition 2.2.4). Let $\widehat{\square}$ be the cell in G containing $\square_u$. Observe that $\square_u \subsetneq \widehat{\square} \subsetneq \square_{\overline{\mathrm{p}}(u)}$. In addition, $\mathbf{d}(\widehat{\square}, \square_{v'}) \leq \mathbf{d}(\square_u, \square_{v'}) = \mathbf{d}(u, v') < r$. It follows that are at most $O(1/\varepsilon^d)$ cells like $\widehat{\square}$ that might participate in charging v', and as such, the total number of charges is $O(2^d/\varepsilon^d)$, as claimed.

As such, v' can be charged at most $O\left(2^{2d}/\varepsilon^d\right) = O\left(1/\varepsilon^d\right)$ times. This implies that the total number of pairs generated by the algorithm is $O(n\varepsilon^{-d})$, since the number of nodes in $\mathcal{T}$ is $O(n)$. ■

Since the running time of **AlgWSPD** is clearly linear in the output size, we have the following result.

Theorem 3.1.7 *For $1 \geq \varepsilon > 0$, one can construct a ε^{-1}-WSPD of size $n\varepsilon^{-d}$, and the construction time is $O\left(n \log n + n\varepsilon^{-d}\right)$. Furthermore, for any pair $\{u, v\}$ in the WSPD, we have*

$$\max(\operatorname{diam}(\mathsf{P}_u), \operatorname{diam}(\mathsf{P}_v)) \leq \varepsilon \cdot \mathbf{d}(u, v).$$

3.2 Applications of WSPD

3.2.1 Spanners

A ***t-spanner*** of a set of points $\mathsf{P} \subset \mathbb{R}^d$ is a weighted graph G whose vertices are the points of P, and for any $\mathsf{q}, \mathsf{r} \in \mathsf{P}$, we have

$$\|\mathsf{q} - \mathsf{r}\| \leq \mathsf{d}_{\mathsf{G}}(\mathsf{q}, \mathsf{r}) \leq t \, \|\mathsf{q} - \mathsf{r}\|,$$

where $\mathsf{d}_{\mathsf{G}}(\mathsf{q}, \mathsf{r})$ is the length of the shortest path in G between q and r (naturally, d_{G} is a metric). The ratio $\mathsf{d}_{\mathsf{G}}(\mathsf{q}, \mathsf{r}) / \|\mathsf{q} - \mathsf{r}\|$ is the ***stretch*** of q and r in G. The ***stretch*** of G is the maximum stretch of any pair of points of P.

Theorem 3.2.1 *Given a n-point set $P \subseteq \mathbb{R}^d$, and parameter $1 \geq \varepsilon > 0$, one can compute a $(1+\varepsilon)$-spanner of P with $O(n\varepsilon^{-d})$ edges, in $O\left(n \log n + n\varepsilon^{-d}\right)$ time.*

Proof: Let $c \geq 16$ be an arbitrary constant, and set $\delta = \varepsilon/c$. Compute a δ^{-1}-WSPD decomposition using the algorithm of Theorem 3.1.7. For any vertex u in the quadtree $\mathcal{T}$ (used in computing the WSPD), let rep_u be an arbitrary point of P_u. For every pair $\{u, v\} \in \mathcal{W}$, add an edge between $\{\mathrm{rep}_u, \mathrm{rep}_v\}$ with weight $\left\|\mathrm{rep}_u - \mathrm{rep}_v\right\|$, and let G be the resulting graph. Observe, that by the triangle inequality, we have that $\mathsf{d}_{\mathsf{G}}(\mathsf{q}, \mathsf{r}) \geq \|\mathsf{q} - \mathsf{r}\|$, for any $\mathsf{q}, \mathsf{r} \in \mathsf{P}$.

The upper bound on the stretch is proved by induction on the length of pairs in the WSPD. So, fix a pair $x, y \in \mathsf{P}$, and assume that by the induction hypothesis, that for any pair $z, w \in \mathsf{P}$ such that $\|z - w\| < \|x - y\|$, it holds $\mathbf{d}_{\mathsf{G}}(z, w) \leq (1 + \varepsilon) \|z - w\|$.

The pair x, y must appear in some pair $\{u, v\} \in \mathcal{W}$, where $x \in \mathsf{P}_u$, and $y \in \mathsf{P}_v$. Thus

$$\left\|\mathrm{rep}_u - \mathrm{rep}_v\right\| \leq \mathbf{d}(u, v) + \Delta(u) + \Delta(v) \leq (1 + 2\delta) \|x - y\|$$

and

$$\begin{aligned}\max\left(\left\|\mathrm{rep}_u - x\right\|, \left\|\mathrm{rep}_v - y\right\|\right) &\leq \max(\Delta(u), \Delta(v)) \leq \delta \cdot \mathbf{d}(u, v) \leq \delta \left\|\mathrm{rep}_u - \mathrm{rep}_v\right\| \\ &\leq \delta(1 + 2\delta) \|x - y\| < \frac{1}{4} \|x - y\|,\end{aligned}$$

by Theorem 3.1.7 and since $\delta \leq 1/16$. As such, we can apply the induction hypothesis to $\mathrm{rep}_u x$ and $\mathrm{rep}_v y$, implying that

$$\mathsf{d}_{\mathsf{G}}(x, \mathrm{rep}_u) \leq (1+\varepsilon) \left\| \mathrm{rep}_u - x \right\| \quad \text{and} \quad \mathsf{d}_{\mathsf{G}}(\mathrm{rep}_v, y) \leq (1+\varepsilon) \left\| y - \mathrm{rep}_v \right\|.$$

Now, since $\mathrm{rep}_u \mathrm{rep}_v$ is an edge of G, it holds $\mathsf{d}_{\mathsf{G}}(\mathrm{rep}_u, \mathrm{rep}_v) \leq \left\| \mathrm{rep}_u - \mathrm{rep}_v \right\|$. Thus, by the inductive hypothesis and the triangle inequality, we have that

$$\begin{aligned}
\|x - y\| &\leq \mathsf{d}_{\mathsf{G}}(x, y) \leq \mathsf{d}_{\mathsf{G}}(x, \mathrm{rep}_u) + \mathsf{d}_{\mathsf{G}}(\mathrm{rep}_u, \mathrm{rep}_v) + \mathsf{d}_{\mathsf{G}}(\mathrm{rep}_v, y) \\
&\leq (1+\varepsilon) \left\| \mathrm{rep}_u - x \right\| + \left\| \mathrm{rep}_u - \mathrm{rep}_v \right\| + (1+\varepsilon) \left\| \mathrm{rep}_v - y \right\| \\
&\leq 2(1+\varepsilon) \cdot \delta \cdot \left\| \mathrm{rep}_u - \mathrm{rep}_v \right\| + \left\| \mathrm{rep}_u - \mathrm{rep}_v \right\| \\
&\leq (1 + 2\delta + 2\varepsilon\delta) \left\| \mathrm{rep}_u - \mathrm{rep}_v \right\| \\
&\leq (1 + 2\delta + 2\varepsilon\delta)(1+\delta) \|x - y\| \\
&\leq (1+\varepsilon) \|x - y\|.
\end{aligned}$$

The last step follows by an easy calculation. Indeed, since $c\delta = \varepsilon \leq 1$ and $16\delta \leq 1$ and $c \geq 11$, we have that

$$(1 + 2\delta + 2\varepsilon\delta)(1+\delta) \leq (1+4\delta)(1+\delta) = 1 + 5\delta + 4\delta^2 \leq 1 + 9\delta \leq 1 + \varepsilon,$$

as required. ∎

3.2.2 Approximating the Minimum Spanning Tree

For a graph G, let $\mathsf{G}_{\leq r}$ denote the subgraph of G resulting from removing all the edges of weight (strictly) larger than r from G.

Lemma 3.2.2 *Given a set P of n points in $\mathbb{R}^d$, one can compute a spanning tree $\mathcal{T}$ of P, such that $w(\mathcal{T}) \leq (1+\varepsilon) w(\mathcal{M})$, where $\mathcal{M}$ is the minimum spanning tree of P, and $w(\mathcal{T})$ is the total weight of the edges of $\mathcal{T}$. This takes $O\left(n \log n + n\varepsilon^{-d}\right)$ time.*

In fact, for any $r \geq 0$ and a connected component C of $\mathcal{M}_{\leq r}$, the set C is contained in a connected component of $\mathcal{T}_{\leq (1+\varepsilon) r}$.

Proof: Compute a $(1+\varepsilon)$-spanner G of P. Let $\mathcal{T}$ be the minimum spanning tree of G. Clearly, $\mathcal{T}$ is the required $(1+\varepsilon)$-approximate MST. Indeed, for any $\mathsf{q}, \mathsf{r} \in \mathsf{P}$, let π_{qr} denote the shortest path between q and r in G. Since G is a $(1+\varepsilon)$-spanner, we have that $w\left(\pi_{\mathsf{qr}}\right) \leq (1+\varepsilon) \|\mathsf{q} - \mathsf{r}\|$, where $w\left(\pi_{\mathsf{qr}}\right)$ denote the weight of π_{qr} in G.

We have that $G' = (P, E)$ is a connected subgraph of G, where

$$E = \bigcup_{(\mathsf{q},\mathsf{r}) \in \mathcal{M}} \pi_{uv}$$

and $\mathcal{M} = \mathcal{M}(\mathsf{P})$ is the minimum spanning tree of P. Furthermore,

$$w(G') = \sum_{(\mathsf{q},\mathsf{r}) \in \mathcal{M}} w\left(\pi_{\mathsf{qr}}\right) \leq \sum_{(\mathsf{q},\mathsf{r}) \in \mathcal{M}} (1+\varepsilon) \|\mathsf{q} - \mathsf{r}\| = (1+\varepsilon) w(\mathcal{M}),$$

since G is a $(1+\varepsilon)$-spanner. It thus follows that $w(\mathcal{M}(G)) \leq w(G') \leq (1+\varepsilon) w(\mathcal{M}(P))$, where $\mathcal{M}(G)$ is the minimum spanning tree of G.

The second claim follows by similar argumentation. ∎

3.2.3 Approximating the Diameter

Lemma 3.2.3 *Given a set* P *of* n *points in* $\mathbb{R}^d$*, one can compute, in* $O\left(n \log n + n\varepsilon^{-d}\right)$ *time, a pair* $u, v \in \mathsf{P}$*, such that* $\|u - v\| \geq (1 - \varepsilon)\operatorname{diam}(P)$.

Proof: Compute a $(4/\varepsilon)$-WSPD of P. As before, we assign for each node u of $\mathcal{T}$ an arbitrary representative point that belongs to P_u. Then, for every pair in the WSPD, compute the distance of the representative point of every pair. Return the pair of representatives in the WSPD farthest away from each other.

To see why it works, consider the pair $\mathsf{q}, \mathsf{r} \in \mathsf{P}$ realizing the diameter of P, and let $\{u, v\} \in \mathcal{W}$ be the pair in the WSPD that contain the two points, respectively (i.e., $\mathsf{q} \in \mathsf{P}_u$ and $\mathsf{r} \in \mathsf{P}_v$). We have that

$$\begin{aligned} \left\|\mathrm{rep}_u - \mathrm{rep}_v\right\| &\geq \mathbf{d}(u, v) \geq \|\mathsf{q} - \mathsf{r}\| - \operatorname{diam}(\mathsf{P}_u) - \operatorname{diam}(\mathsf{P}_v) \\ &\geq (1 - 2(\varepsilon/2))\,\|\mathsf{q} - \mathsf{r}\| = (1 - \varepsilon)\operatorname{diam}(\mathsf{P}), \end{aligned}$$

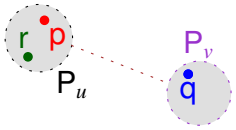
since, by Corollary 3.1.2, $\max(\operatorname{diam}(\mathsf{P}_u), \operatorname{diam}(\mathsf{P}_v)) \leq 2(\varepsilon/4)\,\|\mathsf{q} - \mathsf{r}\|$. Namely, the distance of the two points output by the algorithm is at least $(1 - \varepsilon)\operatorname{diam}(\mathsf{P})$. ■

3.2.4 Closest Pair

Let P be a set of points in $\mathbb{R}^d$. We would like to compute the ***closest pair***; namely, the two points closest to each other in P.

To this end, compute a ε^{-1}-WSPD $\mathcal{W}$ of P, for $\varepsilon = 1/2$. Next, scan all the pairs of $\mathcal{W}$, and check for all the pairs $\{u, v\}$ which connect singletons (i.e., $|\mathsf{P}_u| = |\mathsf{P}_v| = 1$), what is the distance between their representatives rep_u and rep_v. The algorithm returns the closest pair of points encountered.

Analysis. Consider the pair of closest points p and q in P, and consider the pair $\{u, v\} \in \mathcal{W}$, such that $\mathsf{p} \in \mathsf{P}_u$ and $\mathsf{q} \in \mathsf{P}_v$. If P_u contains an additional point $\mathsf{r} \in \mathsf{P}_u$, then we have that



$$\|\,\mathsf{p} - \mathsf{r}\,\| \leq \operatorname{diam}(\mathsf{P}_u) \leq \varepsilon \cdot \mathbf{d}(u, v) \leq \varepsilon\,\|\mathsf{p} - \mathsf{q}\| < \|\mathsf{p} - \mathsf{q}\|\,,$$

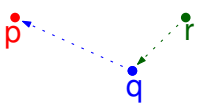
by Theorem 3.1.7 and since $\varepsilon = 1/2$. Thus, $\|\mathsf{p} - \mathsf{r}\| < \|\mathsf{p} - \mathsf{q}\|$, a contradiction to the choice of p and q as the closest pair. Thus, $|\mathsf{P}_u| = |\mathsf{P}_v| = 1$ and $\mathrm{rep}_u = \mathsf{q}$ and $\mathrm{rep}_v = \mathsf{r}$. This implies that the algorithm indeed returns the closest pair.

Theorem 3.2.4 *Given a set* P *of* n *points in* $\mathbb{R}^d$*, one can compute the closest pair of points of* P *in* $O(n \log n)$ *time.*

We remind the reader that we already saw a linear (expected) time algorithm for this problem in Section 1.2. However, this is a deterministic algorithm, and it can be applied in more abstract settings where a small WSPD still exists, while the previous algorithm would not work.

3.2.5 All Nearest Neighbors

Given a set P of n points in $\mathbb{R}^d$, we would like to compute for each point $\mathsf{q} \in \mathsf{P}$, its ***nearest neighbor*** in P (formally, this is the closet point in $\mathsf{P} \setminus \{\mathsf{q}\}$ to q). This is harder than it might seem at first, since this is *not* a symmetrical relationship. Indeed, q might be the nearest neighbor to p, but r might be the nearest neighbor to q.



3.2.5.1 The bounded spread case

Assume P is contained in the unit square, and $\mathrm{diam}(P) \geq 1/4$. Furthermore, let $\Phi = \Phi(P)$ denote the spread of P. Compute a ε^{-1}-WSPD $\mathcal{W}$ of P, for $\varepsilon = 1/4$. Arguing as in the closest pair case, we have that if the nearest neighbor to p is q, then there exists a pair $\{u, v\} \in \mathcal{W}$, such that $\mathsf{P}_u = \{\mathsf{p}\}$ and $\mathsf{q} \in \mathsf{P}_v$. Thus, scan all the pairs $\{u, v\}$ with a singleton as one of their sides (i.e., $|P_u| = 1$), and for each such singleton $\mathsf{P}_u = \{\mathsf{r}\}$, record for r the closest point to it in P_v. Maintain for each point the closet point to it that was encountered.

Analysis. The analysis of this algorithm is slightly tedious, but it reveals some additional interesting properties of WSPD.

A pair of nodes $\{x, y\}$ of $\mathcal{T}$ is a ***generator*** of a pair $\{u, v\}$ if $\{u, v\}$ was computed inside a recursive call **AlgWSPD**(x, y).

Lemma 3.2.5 *Let $\mathcal{W}$ be a ε^{-1}-WSPD of a point set P generated by* **AlgWSPD**. *Consider a pair $\{u, v\} \in \mathcal{W}$, then $\Delta(\overline{\mathrm{p}}(v)) \geq (\varepsilon/2)\mathbf{d}(u, v)$ and $\Delta(\overline{\mathrm{p}}(u)) \geq (\varepsilon/2)\mathbf{d}(u, v)$.*

Proof: Assume, for the sake of contradiction, that $\Delta(v') < (\varepsilon/2)\ell$, where $\ell = \mathbf{d}(u, v)$ and $v' = \overline{\mathrm{p}}(v)$. By Lemma 3.1.5, we have that

$$\Delta(u) \leq \Delta(v') < \varepsilon\frac{\ell}{2}.$$

But then

$$\mathbf{d}(u, v') \geq \ell - \Delta(v') \geq \ell - \varepsilon\frac{\ell}{2} \geq \frac{\ell}{2}.$$

Thus,

$$\max(\Delta(u), \Delta(v')) < \varepsilon\frac{\ell}{2} \leq \varepsilon\, \mathbf{d}(u, v').$$

Namely, u and v' are well-separated, and as such $\{u, v'\}$ can not be a generator of $\{u, v\}$. Indeed, if $\{u, v'\}$ was considered by the algorithm than it would have added it to the WSPD, and never created $\{u, v\}$.

So, the other possibility is that $\{u', v\}$ is the generator of $\{u, v\}$, where $u' = \overline{\mathrm{p}}(u)$. But then $\Delta(u') \leq \Delta(v') < \varepsilon\ell/2$, by Lemma 3.1.5. Using the same argumentation as above, we have that $\{u', v\}$ is a well-separated pair and as such it can not be a generator of $\{u, v\}$.

But this implies that $\{u, v\}$ can not be generated by **AlgWSPD**, since either $\{u, v'\}$ or $\{u', v\}$ must be a generator of $\{u, v\}$. A contradiction. ■

Claim 3.2.6 *For two pairs $\{u, v\}, \{u', v'\} \in \mathcal{W}$ such that $\square_u \subseteq \square_{u'}$, it holds that the interiors of $\square_v$ and $\square_{v'}$ are disjoint.*

Proof: If u' is ancestor of u, and v' is an ancestor of v then **AlgWSPD** returned the pair $\{u', v'\}$ and it would have never generated the pair $\{u, v\}$.

If u' is ancestor of u, and v is an ancestor of v', then

$$\Delta(u) < \Delta(u') \leq \Delta(\overline{\mathrm{p}}(v')) \leq \Delta(v) \leq \Delta(\overline{\mathrm{p}}(u)) \leq \Delta(u')$$

by Lemma 3.1.5 applied to $\{u', v'\}$ and $\{u, v\}$. Namely, $\Delta(v) = \Delta(u')$. But then, the pair $\{u', v\}$ is a generator of both $\{u, v\}$ and $\{u', v'\}$. But it is impossible that **AlgWSPD**generated both pairs when processing $\{u', v\}$ as can be easily verified.

Similar analysis applies for the case that $u = u'$. ■

Lemma 3.2.7 *Let P be a set n points in $\mathbb{R}^d$, $\mathcal{W}$ a ε^{-1}-WSPD of P, $\ell > 0$ be a distance, and W be the set of pairs $\{u, v\} \in \mathcal{W}$ such that $\ell \leq \mathbf{d}(u, v) \leq 2\ell$. Then, point any point $\mathsf{p} \in \mathsf{P}$, the number of pairs in W containing p is $O(1/\varepsilon^d)$.*

Proof: Let u be the leaf of the quadtree $\mathcal{T}$ (that is used in computing $\mathcal{W}$) storing the point p, and let π be the path between u and the root of $\mathcal{T}$. We claim that W contains at most $O(1/\varepsilon^d)$ pairs with nodes that appears along π. Let

$$T = \left\{ v \;\middle|\; u \in \pi, \{u, v\} \in W \right\}.$$

The cells of T are interior disjoint by Claim 3.2.6, and they contain all the pairs in W that covers p.

So, let r be the largest power of two which is smaller than (say) $\varepsilon\ell/(4\sqrt{d})$. Clearly, there are $O(1/\varepsilon^d)$ cells of G_r in distance at most 2ℓ from $\square_u$. We account for the nodes $v \in T$, as follows:

(i) If $\Delta(v) \geq r\sqrt{d}$ then $\square_v$ contains a cell of G_r, and there are at most $O(1/\varepsilon^d)$ such cells.

(ii) If $\Delta(v) < r\sqrt{d}$ and $\Delta(\overline{\mathsf{p}}(v)) \geq r\sqrt{d}$, then:

  (a) If $\overline{\mathsf{p}}(v)$ is a compressed node, then $\overline{\mathsf{p}}(v)$ contains a cell of G_r and it has only v as a single child. As such, there are most $O(1/\varepsilon^d)$ such charges.

  (b) Otherwise, $\overline{\mathsf{p}}(v)$ is not compressed, but then $\operatorname{diam}(\square_v) = \operatorname{diam}\left(\square_{\overline{\mathsf{p}}(v)}\right)/2$. As such $\square_v$ contains a cell of $\mathsf{G}_{r/2}$ in distance at most 2ℓ from $\square_u$, and there are $O(1/\varepsilon^d)$ such cells.

(iii) The case $\Delta(\overline{\mathsf{p}}(v)) < r\sqrt{d}$ is impossible. Indeed, by Lemma 3.1.5, we have $\Delta(\overline{\mathsf{p}}(v)) < r\sqrt{d} \leq \varepsilon\ell/4 = \frac{\varepsilon}{4}\mathbf{d}(u, v)$, a contradiction to Lemma 3.2.5.

We conclude that there are at most $O(1/\varepsilon^d)$ pairs that include p in W. ■

Lemma 3.2.8 *Let P be a set n points in the plane, then one can solve the all nearest neighbor problem, in time $O(n(\log n + \log \Phi(P)))$ time, where Φ is the spread of P.*

Proof: The algorithm is described above. We only remain with the task of analyzing the running time. For a number $i \in \{0, -1, \ldots, -\lfloor \lg \Phi \rfloor - 4\}$, consider the set of pairs W_i, such that $\{u, v\} \in W_i$, if and only if $\{u, v\} \in \mathcal{W}$, and $2^{i-1} \leq \mathbf{d}(u, v) \leq 2^i$. A point $\mathsf{p} \in \mathsf{P}$ can be scanned at most $O(1/\varepsilon^d) = O(1)$ times because of pairs in W_i by Lemma 3.2.7. As such, a point get scanned at most $O(\log \Phi)$ times overall, which implies the running time bound. ■

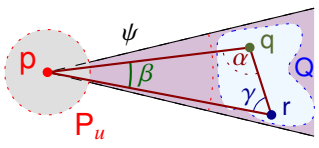
3.2.5.2 All nearest neighbor - the unbounded spread case

To handle the unbounded case, we need to use some additional geometric properties.

Lemma 3.2.9 *Let u be a node in the compressed quadtree $\mathcal{QT}$ of P, and partition the space around rep_u into cones of angle $\leq \pi/12$. Let ψ be such a cone, and let Q be the set of all points in P which are in distance $\geq 4\operatorname{diam}(\mathsf{P}_u)$ from rep_u, and they all lie inside ψ. Let q be the closest point in Q to rep_u. Then, q is the only point in Q that its nearest neighbor might be in P_u.*

Proof: Let $\mathsf{p} = \operatorname{rep}_u$ and consider any point $\mathsf{r} \in \mathsf{Q}$.

Since $\|\mathsf{r} - \mathsf{p}\| \geq \|\mathsf{q} - \mathsf{p}\|$, it follows that $\alpha = \angle\mathsf{rqp} \geq \angle\mathsf{qrp} = \gamma$. Now, $\alpha + \gamma = \pi - \beta$, where $\beta = \angle\mathsf{rpq}$. But $\beta \leq \pi/3$, and as such $\alpha \geq (\pi - \angle\mathsf{rpq})/2 \geq \pi/3 \geq \beta$. Namely, α is the largest angle in the triangle $\triangle\mathsf{pqr}$, which implies $\|\mathsf{r} - \mathsf{p}\| \geq \|\mathsf{r} - \mathsf{q}\|$. Namely, q is closer to r than p, and as such p can not serve as the nearest neighbor to r in P.



It is now straightforward (but tedious) to show that, in fact, for any $\mathsf{p} \in \mathsf{P}_u$, we have $\|\mathsf{r} - \mathsf{p}\| \geq \|\mathsf{r} - \mathsf{q}\|$, which implies the claim.[②] ■

Lemma 3.2.9 implies that we can do a top-down traversal of $\mathcal{QT}(P)$, after computing a ε^{-1}-WSPD $\mathcal{W}$ of P, for $\varepsilon = 1/16$. For every node u, we maintain a (constant size) set R_u of candidate points that P_u *might* contain their nearest neighbor.

So, assume we had computed $\mathsf{R}_{\overline{\mathsf{p}}(u)}$, and consider the set

$$X(u) = \mathsf{R}_{\overline{\mathsf{p}}(u)} \cup \bigcup_{\{u,v\} \in \mathcal{W}, |\mathsf{P}_v| = 1} \mathsf{P}_v.$$

(Note, that we do not have to consider pairs with $|\mathsf{P}_v| > 1$, since no point in P_v can have its nearest neighbor in P_u in such a scenario.) Clearly, we can compute X in linear time in the number of pairs in $\mathcal{W}$ involved with u. Now, we build a "grid" of cones around rep_u, and throw the points of $X(u)$ into this grid. For each such cone, we keep only the closest point to rep_u. Let R_u be the set of these closest points. Since the number of cones is $O(1)$, it follows that $|\mathsf{R}_u| = O(1)$.

Now, if P_u contains only a single point p, then we compute for any point $\mathsf{q} \in \mathsf{R}_u$ its distance to p, and if p is a better candidate to be a nearest neighbor, then we set p as the (current) nearest neighbor to q.

Clearly, the resulting running time (ignoring the computation of the WSPD) is linear in the number of pairs of the WSPD and the size of the compressed quadtree. The correctness follows since p is the nearest neighbor to q, then there must be a WSPD pair $\{u, v\}$ such that $\mathsf{P}_v = \{\mathsf{q}\}$ and $\mathsf{p} \in \mathsf{P}_u$. But then, the algorithm would add q to the set R_u, and it would be in R_z, for all descendants z of u in the quadtree, such that $\mathsf{p} \in \mathsf{P}_z$. In particular, if y is the leaf of the quadtree storing p, then $\mathsf{q} \in \mathsf{R}_y$, which implies that the algorithm computes correctly the nearest neighbor to q.

Theorem 3.2.10 *Given a set P of n points in $\mathbb{R}^d$, one can solve the all nearest neighbor problem in $O(n \log n)$ time.*

3.3 Bibliographical Notes

Well separated pairs decomposition was defined by Callahan and Kosaraju [CK95]. They defined a different space decomposition tree, known as the ***fair split tree***. Here, one compute the axis parallel bounding box of the point-set, and always split along the longest edge by a perpendicular plane in the middle (or near the middle). This splits the point set into two sets, which we construct fail split tree for them recursively. Implementing this in $O(n \log n)$ time requires some cleverness. See [CK95] for details.

Our presentation of WSPD (very roughly) follows [HM06]. The (easy) observation that WSPD can be generated directly from a compressed quadtree (thus avoiding the fair split tree mess) is from there.

Callahan and Kosaraju [CK95] were inspired by the work of Vaidya [Vai86] on all nearest neighbor problem (i.e., compute for each points in P, their nearest neighbor in P). He defined the fair split tree, and show how to compute the all nearest neighbors in $O(n \log n)$ time. However, the first to give an $O(n \log n)$ time algorithm for the all nearest neighbor algorithm was Clarkson [Cla83] (this was part of his PhD thesis).

[②] Here are the details for readers of little fate. By the law of sines, we have $\frac{\|\mathsf{p}-\mathsf{r}\|}{\sin\alpha} = \frac{\|\mathsf{q}-\mathsf{r}\|}{\sin\beta}$. As such, $\|\mathsf{q} - \mathsf{r}\| = \|\mathsf{p} - \mathsf{r}\| \frac{\sin\beta}{\sin\alpha}$. Now, if $\alpha \leq \pi - 3\beta$ then $\|\mathsf{q} - \mathsf{r}\| = \|\mathsf{p} - \mathsf{r}\| \frac{\sin\beta}{\sin\alpha} \leq \|\mathsf{p} - \mathsf{r}\| \frac{\sin\beta}{\sin(3\beta)} \leq \frac{\|\mathsf{p}-\mathsf{r}\|}{2} < \|\mathsf{p} - \mathsf{r}\| - \Delta(u)$, since $\|\mathsf{p} - \mathsf{r}\| \geq 4\Delta(u)$. This implies that no point of P_u can be the nearest neighbor of r.

If $\alpha \geq \pi - 3\beta$ then the maximum length of qr is achieved when $\gamma = 2\beta$. The sines law then implies that $\|\mathsf{q} - \mathsf{r}\| = \|\mathsf{p} - \mathsf{q}\| \frac{\sin\beta}{\sin(2\beta)} \leq \frac{3}{4}\|\mathsf{p} - \mathsf{q}\| \leq \frac{3}{4}\|\mathsf{p} - \mathsf{r}\| < \|\mathsf{p} - \mathsf{r}\| - \Delta(u)$, which again implies the claim.

Diameter. The algorithm for computing the diameter of Section 3.2.3 can be improved by not constructing pairs that can not improve the (current) diameter, and constructing the underlying tree on the fly together with the diameter. This yields a simple algorithm that works quite will in practice, see [Har01a].

All nearest neighbors. Section 3.2.5 is a simplification of the solution for the all k-nearest neighbor problem. Here, one can compute for every point its k-nearest neighbors in $O(n \log n + nk)$ time. See [CK95] for details.

The all nearest neighbor algorithm for bounded spread (Section 3.2.5.1) is from [HM06]. Note, that unlike the unbounded case, this algorithm only use packing arguments for its correctness. Surprisingly, the usage of the Euclidean nature of the underlying space (as done in Section 3.2.5.2) seems to be crucial in getting a faster algorithm for this problem. In particular, for the case of metric spaces of low doubling dimension (that do have a small WSPD), solving this problem requires $\Omega(n^2)$ time in the worst case.

Dynamic maintenance. WSPD can be maintained in polylogarithmic time under insertions and deletions. This is quite surprising when one considers that in the worst case, a point might participate in linear number of pairs, and in fact, a node in the quadtree might participate in linear number of pairs. This is described in detail in Callahan thesis [Cal95]. Interestingly, using randomization maintaining the WSPD can be considerably simplified, see the work by Fischer and Har-Peled [FH05].

High dimension. In high dimensions, as the uniform metric demonstrates (i.e., n points all of them in distance 1 from each other) the WSPD can have quadratic complexity. This metric is easily realizable as the vertices of a simplex in $\mathbb{R}^{n-1}$. On the other hand, doubling metrics have near linear size WSPD. Since WSPDs by themselves are so powerful, it kind of tempting to try and define dimension of a point set by the size of the WSPD it posses. This seems like an interesting direction for future research, as currently little is known about it (to the best of my knowledge).

3.4 Exercises

Exercise 3.4.1 (WSPD Structure.) [5 Points]

(A) Let $\varepsilon > 0$ be sufficiently small constant. For any n sufficiently large, show an example of a point set P of n points, such that its $(1/\varepsilon)$-WSPD (as computed by **AlgWSPD**) has the property that a single set participates in $\Omega(n)$ sets.[3]

(B) Show, that if we list explicitly the sets forming the WSPD (even if we show each set exactly once) then the total size of such a description is quadratic. (Namely, the implicit representation we use is crucial to achieve efficient representation.)

Exercise 3.4.2 (Number of resolutions that matter.) [4 Points]

Let P be a n-point set in $\mathbb{R}^d$, and consider the set $U = \left\{ i \;\middle|\; 2^i \le \|\mathsf{p} - \mathsf{q}\| \le 2^{i+1}, \text{ for } \mathsf{p}, \mathsf{q} \in \mathsf{P} \right\}$. Prove that $|U| = O(n)$ (the constant depends on d). Namely, there are only n different resolutions that "matter".

(The claim is a special case of a more general claim, see Exercise 24.7.2.)

Exercise 3.4.3 (WSPD and sum of distances.) [5 Points]

[3] Note, that there is always a WSPD construction such that each node participates in a "small" number of pairs.

Let P be a set of n points in $\mathbb{R}^d$. The ***sponginess***[4] of P is the quantity $X = \sum_{\{\mathsf{p},\mathsf{q}\}\subseteq\mathsf{P}} \|\mathsf{p} - \mathsf{q}\|$. Provide an efficient algorithm for approximating X. Namely, given P and a parameter $\varepsilon > 0$ it outputs a number Y such that $X \leq Y \leq (1 + \varepsilon)X$.

(The interested reader can also verify that computing (exactly) the sum of all ***squared*** distances (i.e., $\sum_{\{\mathsf{p},\mathsf{q}\}\subseteq\mathsf{P}} \|\mathsf{p} - \mathsf{q}\|^2$) is considerably easier.)

[4] Also known as the sum of pairwise distances in the literature, for reasons that I can not fathom.

Chapter 4

Clustering - Definitions and Basic Algorithms

> Do not read this story; turn the page quickly. The story may upset you. Anyhow, you probably know it already. It is a very disturbing story. Everyone knows it. The glory and the crime of Commander Suzdal have been told in a thousand different ways. Don't let yourself realize that the story is the truth.
>
> It isn't. not at all. There's not a bit of truth to it. There is no such planet as Arachosia, no such people as klopts, no such world as Catland. These are all just imaginary, they didn't happen, forget about it, go away and read something else.
>
> – The Crime and Glory of Commander Suzdal, Cordwainer Smith

In this chapter, we will initiate our discussion of ***clustering***. Clustering is one of the most fundamental computational tasks, but frustratingly, one of the fuzziest. It can be stated informally as: "Given data, find interesting structure in the data. Go!"

The fuzziness arise naturally from the requirement that it would be "interesting", as this is not well defined and depends on human perception which is sometime impossible to quantify clearly. Similarly, what is "structure" is also open to debate. Nevertheless, clustering is inherent to many computational tasks like learning, searching and data-mining.

Empirical study of clustering concentrates on trying various measures for the clustering, and trying out various algorithms and heuristics to compute these clusterings. See bibliographical notes for some relevant references.

Here, we will concentrate on some well defined clustering tasks, including k-center clustering, k-median clustering, and k-means clustering, and some basic algorithms for these problems.

4.1 Preliminaries

A clustering problem is usually defined by a set of items, and a distance function defined between these items. While these items might be points in $\mathbb{R}^d$ and the distance function is just the regular Euclidean distance, it is sometime beneficial to consider the more abstract setting of a general metric space.

Definition 4.1.1 A ***metric space*** is a pair $(\mathcal{X}, \mathbf{d})$ where $\mathcal{X}$ is a set and $\mathbf{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric***, satisfying the following axioms: (i) $\mathbf{d}(x, y) = 0$ if and only if $x = y$, (ii) $\mathbf{d}(x, y) = \mathbf{d}(y, x)$, and (iii) $\mathbf{d}(x, y) + \mathbf{d}(y, z) \geq \mathbf{d}(x, z)$ (triangle inequality).

For example, $\mathbb{R}^2$ with the regular Euclidean distance is a metric space. In the following, we assume that we are given a *black-box access* to $\mathbf{d}$. Namely, given two points $\mathsf{p}, \mathsf{q} \in \mathcal{X}$, we assume that $\mathbf{d}(\mathsf{p}, \mathsf{q})$ can be computed in constant time.

The input is a set of n points $\mathsf{P} \subseteq \mathcal{X}$. Given a set of centers $\mathbf{C}$, every point of P is assigned to its nearest neighbor in $\mathbf{C}$. All the points of P that are assigned to a center $\overline{\mathsf{c}}$ form the ***cluster*** of $\overline{\mathsf{c}}$, denoted by

$$\Pi(\mathbf{C}, \overline{\mathsf{c}}) = \left\{ p \in \mathsf{P} \;\middle|\; \mathbf{d}(p, \overline{\mathsf{c}}) \leq \mathbf{d}(p, \mathbf{C}) \right\}.$$

Namely, the center set $\mathbf{C}$ partition P into clusters. This specific scheme of partitioning points by assigning them to their closest center (in a given set of centers) is known as a ***Voronoi partition***.

4.2 On k-Center Clustering

In the ***k-center clustering problem***, a set $\mathsf{P} \subseteq \mathcal{X}$ of n points is provided together with a parameter k. We would like to find k points $X \subseteq \mathsf{P}$, such that the maximum distance of a point in P to the closest point in X is minimized.

As a concrete example, consider the set of points to be a set of cities. Distances between points represent the time it takes to travel between the corresponding points. We would like to build k hospitals and minimize the maximum time it takes a patient to arrive to its closest hospital.

Formally, given a set of centers $\mathbf{C}$, the k-center clustering ***price*** of P by $\mathbf{C}$ is denoted by

$$\nu_{\infty}^{\mathbf{C}}(\mathsf{P}) = \max_{\mathsf{p} \in \mathsf{P}} \mathbf{d}(\mathsf{p}, \mathbf{C}),$$

where $\mathbf{d}(\mathsf{p}, \mathbf{C}) = \min_{\overline{\mathsf{c}} \in \mathbf{C}} \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}})$ denotes the ***distance*** of p to the set $\mathbf{C}$. Note, that every point in a cluster is in distance at most $\nu_{\infty}^{\mathbf{C}}(\mathsf{P})$ from its respective center.

Formally, the ***k-center problem*** is to find a set $\mathbf{C}$ of k points, such that $\nu_{\infty}^{\mathbf{C}}(\mathsf{P})$ is minimized; namely,

$$\nu_{\infty}^{\mathrm{opt}}(\mathsf{P}, k) = \min_{\mathbf{C}, |\mathbf{C}|=k} \nu_{\infty}^{\mathbf{C}}(\mathsf{P}).$$

We will denote the set of centers realizing the optimal clustering by C_{opt}. A more explicit definition (and somewhat more confusing) of the k-center clustering is to compute the set $\mathbf{C}$ of size k realizing $\min_{\mathbf{C}} \max_{\mathsf{p}} \min_{\overline{\mathsf{c}} \in \mathbf{C}} \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}})$.

It is known that the k-center clustering is NP-Hard, and it is in fact hard to approximate within a factor of $1,86$ even in two dimensions. Surprisingly, there is a simple and elegant algorithm that achieves 2-approximation.

4.2.1 The Greedy Clustering Algorithm

The greedy algorithm **GreedyKCenter** starts by picking an arbitrary point $\overline{\mathsf{c}}_1$ into $\mathbf{C}_1$. Next, we compute for every point $\mathsf{p} \in \mathsf{P}$ its distance $d_1[\mathsf{p}]$ from $\overline{\mathsf{c}}_1$. Next, consider the point worst served by $\overline{\mathsf{c}}_1$; this is the point realizing $r_1 = \max_{\mathsf{p} \in \mathsf{P}} d_1[\mathsf{p}]$. Let $\overline{\mathsf{c}}_2$ denote this point, and add it to the set of centers $\mathbf{C}_2$.

Specifically, in the ith iteration, we compute for each point $\mathsf{p} \in \mathsf{P}$ the quantity $d_{i-1}[\mathsf{p}] = \min_{\overline{\mathsf{c}} \in \mathbf{C}_{i-1}} \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}})$. We also compute the radius of the clustering

$$r_{i-1} = \max_{\mathsf{p} \in \mathsf{P}} d_{i-1}[\mathsf{p}] = \max_{\mathsf{p} \in \mathsf{P}} \mathbf{d}(\mathsf{p}, \mathbf{C}_{i-1}), \tag{4.1}$$

and the bottleneck point $\overline{\mathsf{c}}_i$ that realizes it. Next, we add $\overline{\mathsf{c}}_i$ to $\mathbf{C}_{i-1}$ to form the new set $\mathbf{C}_i$. We repeat this process k times.

To make this algorithm slightly faster, observe that

$$d_i[\mathsf{p}] = \mathbf{d}(\mathsf{p}, \mathbf{C}_i) = \min(\mathbf{d}(\mathsf{p}, \mathbf{C}_{i-1}), \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}}_i)) = \min(d_{i-1}[\mathsf{p}], \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}}_i)).$$

In particular, if we maintain for p a single variable $d[\mathsf{p}]$ with its current distance to the closest center in the current center set, then the above formula boils down to

$$d[\mathsf{p}] \leftarrow \min(d[\mathsf{p}], \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}}_i)) .$$

Namely, the above algorithm can be implemented using $O(n)$ space, where $n = |\mathsf{P}|$. The ith iteration of choosing the ith center takes $O(n)$ time. Thus, overall this approximation algorithm takes $O(nk)$ time.

A ***ball*** of radius r around a point $\mathsf{p} \in \mathsf{P}$ is the set of points in P with distance at most r from p; namely, $\mathbf{b}(\mathsf{p}, r) = \left\{ \mathsf{q} \in \mathsf{P} \;\middle|\; \mathbf{d}(\mathsf{p}, \mathsf{q}) \le r \right\}$. Thus, the k-center problem can be interpreted as the problem of covering the points of P using k balls of minimum (maximum) radius.

Theorem 4.2.1 *Given a set of n points $\mathsf{P} \subseteq \mathcal{X}$, belonging to a metric space $(\mathcal{X}, \mathbf{d})$, the algorithm* **GreedyKCenter** *computes a set $\mathbf{K}$ of k centers, such that $\mathbf{K}$ is a 2-approximation to the optimal k-center clustering of P; namely, $\nu_\infty^{\mathbf{K}}(\mathsf{P}) \le 2\nu_\infty^{\mathrm{opt}}$, where $\nu_\infty^{\mathrm{opt}} = \nu_\infty^{\mathrm{opt}}(\mathsf{P}, k)$. The algorithm takes $O(nk)$ time.*

Proof: The running time follows by the above description, so we concern ourselves only with the approximation quality.

By definition, we have $r_k = \nu_\infty^{\mathbf{K}}(\mathsf{P})$, and let $\overline{\mathsf{c}}_{k+1}$ be the point in P realizing $r_k = \mathbf{d}(\mathbf{K}, \mathsf{P})$. Let $\mathbf{C} = \mathbf{K} \cup \{\mathsf{r}\}$. Observe that by the definition of r_i, we have that $r_1 \ge r_2 \ge \ldots \ge r_k$. Furthermore, for $i < j \le k+1$ we have that

$$\mathbf{d}(\overline{\mathsf{c}}_i, \overline{\mathsf{c}}_j) \ge \mathbf{d}(\overline{\mathsf{c}}_j, \mathbf{C}_{j-1}) = r_j \ge r_k.$$

Namely, the distance between the closest points in $\mathbf{C}$ is at least r_k. Now, assume for the sake of contradiction that $r_k > 2\nu_\infty^{\mathrm{opt}}(\mathsf{P}, k)$. Consider the optimal solution, that covers P with k balls of radius $\nu_\infty^{\mathrm{opt}}$. By the triangle inequality, any two points inside such a ball are in distance at most $2\nu_\infty^{\mathrm{opt}}$ from each other. Thus, none of these balls can cover two points of $\mathbf{C} \subseteq \mathsf{P}$, since the minimum distance in $\mathbf{C}$ is $> 2\nu_\infty^{\mathrm{opt}}$. A contradiction, since then we need $k+1$ balls of radius $\nu_\infty^{\mathrm{opt}}$ to cover P. ∎

In the spirit of never trusting a claim that has only a single proof, we provide an alternative proof.[5]

Alternative Proof: If every cluster of C_{opt} contains exactly one point of $\mathbf{K}$ then the claim follows. Indeed, consider a point $\mathsf{p} \in \mathsf{P}$, and let $\overline{\mathsf{c}}$ be the center it belongs to in C_{opt}. Also, let $\overline{\mathsf{k}}$ be the center of $\mathbf{K}$ that is in $\Pi\left(C_{\mathrm{opt}}, \overline{\mathsf{c}}\right)$. We have that $\mathbf{d}(\mathsf{p}, \overline{\mathsf{c}}) = \mathbf{d}\left(\mathsf{p}, C_{\mathrm{opt}}\right) \le \nu_\infty^{\mathrm{opt}} = \nu_\infty^{\mathrm{opt}}(\mathsf{P}, k)$. Similarly, observe that $\mathbf{d}\left(\overline{\mathsf{k}}, \overline{\mathsf{c}}\right) = \mathbf{d}\left(\overline{\mathsf{k}}, C_{\mathrm{opt}}\right) \le \nu_\infty^{\mathrm{opt}}$. As such, by the triangle inequality, we have that $\mathbf{d}\left(\mathsf{p}, \overline{\mathsf{k}}\right) \le \mathbf{d}(\mathsf{p}, \overline{\mathsf{c}}) + \mathbf{d}\left(\overline{\mathsf{c}}, \overline{\mathsf{k}}\right) \le 2\nu_\infty^{\mathrm{opt}}$.

By the pigeon hole principle, the only other possibility is that there are two centers $\overline{\mathsf{k}}$ and $\overline{\mathsf{u}}$ of $\mathbf{K}$ that are both in $\Pi\left(C_{\mathrm{opt}}, \overline{\mathsf{c}}\right)$, for some $\overline{\mathsf{c}} \in C_{\mathrm{opt}}$. Assume, without loss of generality, that $\overline{\mathsf{u}}$ was added later to the center set $\mathbf{K}$ by the algorithm **GreedyKCenter**, say in the ith iteration. But then, since **GreedyKCenter** always chooses the point furthest away from the current set of centers, we have that $\overline{\mathsf{c}} \in \mathbf{C}_{i-1}$ and

$$\nu_\infty^{\mathbf{K}}(\mathsf{P}) \le \nu_\infty^{\mathbf{C}_{i-1}}(\mathsf{P}) = \mathbf{d}(\overline{\mathsf{u}}, \mathbf{C}_{i-1}) \le \mathbf{d}\left(\overline{\mathsf{u}}, \overline{\mathsf{k}}\right) \le \mathbf{d}(\overline{\mathsf{u}}, \overline{\mathsf{c}}) + \mathbf{d}\left(\overline{\mathsf{c}}, \overline{\mathsf{k}}\right) \le 2\nu_\infty^{\mathrm{opt}}.$$

∎

4.2.2 The greedy permutation

There is an interesting phenomena associated with **GreedyKCenter**. We can ran it till it exhausts all the points of P (i.e., $k = n$). Then, this algorithm generates a permutation of P; that is $\mathsf{P} = \langle \overline{\mathsf{c}}_1, \overline{\mathsf{c}}_2, \ldots, \overline{\mathsf{c}}_n \rangle$. We will refer to P as the ***greedy permutation*** of P. There is also an associated sequence of radiuses $\langle r_1, r_2, \ldots, r_n \rangle$, where all the points of P are in distance at most r_i from the points of $\mathbf{C}_i = \langle \overline{\mathsf{c}}_1, \ldots, \overline{\mathsf{c}}_i \rangle$.

[5] Mark Twain is credited with saying that "I don't give a damn for a man that can only spell a word one way." However, there seems to be a doubt if he really said that, which brings us to the conclusion of never trusting a quote if it is credited only to a single person.

Definition 4.2.2 A set $S \subseteq \mathsf{P}$ is a ***r-net*** for P if the following two properties hold.

(i) ***Covering property***: All the points of P are in distance at most r from the points of P.

(ii) ***Separation property***: For any pair point of points $\mathsf{p}, \mathsf{q} \in S$, we have that $\mathbf{d}(\mathsf{p}, \mathsf{q}) \geq r$.

(One can relax the separation property by requiring that the points of S would be at distance $\Omega(r)$ apart.)

Intuitively, a r-net of a point-set P is a compact representation of P in the resolution r. Surprisingly, the greedy permutation of P provides us with such a representation for all resolutions.

Theorem 4.2.3 *Let P be a set of n points in a finite metric space, and let its greedy permutation be $\langle \overline{c}_1, \overline{c}_2, \ldots, \overline{c}_n \rangle$ with the associated sequence of radiuses $\langle r_1, r_2, \ldots, r_n \rangle$. For any i, we have that $\mathbf{C}_i = \langle \overline{c}_1, \ldots, \overline{c}_i \rangle$ is a r_i-net of P.*

Proof: Note, that by construction $r_k = \mathbf{d}(\overline{c}_k, \mathbf{C}_{k-1})$, for all $k = 1, \ldots, n$. As such, for $j < k < i \leq n$, we have that $\mathbf{d}(\overline{c}_j, \overline{c}_k) \geq r_k \geq r_i$, which implies the required separation property. The covering property follows by definition, see Eq. (4.1). ■

4.3 On k-median clustering

In the ***k-median clustering problem***, a set $\mathsf{P} \subseteq \mathcal{X}$ is provided together with a parameter k. We would like to find k points $\mathbf{C} \subseteq \mathsf{P}$, such that the sum of distances of points of P to their closest point in $\mathbf{C}$ is minimized.

Formally, given a set of centers $\mathbf{C}$, the k-center clustering ***price*** of clustering P by $\mathbf{C}$ is denoted by

$$\nu_1^{\mathbf{C}}(\mathsf{P}) = \sum_{\mathsf{p} \in \mathsf{P}} \mathbf{d}(\mathsf{p}, \mathbf{C}),$$

where $\mathbf{d}(\mathsf{p}, \mathbf{C}) = \min_{\overline{c} \in \mathbf{C}} \mathbf{d}(\mathsf{p}, \overline{c})$ denotes the ***distance*** of p to the set $\mathbf{C}$.

Formally, the ***k-median problem*** is to find a set $\mathbf{C}$ of k points, such that $\nu_1^{\mathbf{C}}(\mathsf{P})$ is minimized; namely,

$$\nu_1^{\mathrm{opt}}(\mathsf{P}, k) = \min_{\mathbf{C}, |\mathbf{C}| = k} \nu_1^{\mathbf{C}}(\mathsf{P}).$$

We will denote the set of centers realizing the optimal clustering by C_{opt}.

There is a simple and elegant constant factor approximation algorithm for k-median clustering using ***local search*** (its analysis however is painful).

A note on notations. Consider the set U of all k-tuples of points of P. Let p_i denote the ith point of P, for $i = 1, \ldots, n$, where $n = |\mathsf{P}|$. for $\mathbf{C} \in U$, consider the n dimensional point

$$\phi(\mathbf{C}) = (\mathbf{d}(p_1, \mathbf{C}), \mathbf{d}(p_2, \mathbf{C}), \ldots, \mathbf{d}(p_n, \mathbf{C})).$$

Clearly, we have that $\nu_\infty^{\mathbf{C}}(\mathsf{P}) = \|\phi(\mathbf{C})\|_\infty = \max_i \mathbf{d}(p_i, \mathbf{C})$. And $\nu_\infty^{\mathrm{opt}}(\mathsf{P}, k) = \min_{\mathbf{C} \in U} \|\phi(\mathbf{C})\|_\infty$.

Similarly, we have that $\nu_1^{\mathbf{C}}(\mathsf{P}) = \|\phi(\mathbf{C})\|_1 = \sum_i \mathbf{d}(p_i, \mathbf{C})$. And $\nu_1^{\mathrm{opt}}(\mathsf{P}, k) = \min_{\mathbf{C} \in U} \|\phi(\mathbf{C})\|_1$.

Namely, k-center clustering under this interpretation is just finding the point minimizing the ℓ_∞ norm in a set U of points in n dimensions. Similarly, the k-median problem is to find the point minimizing the ℓ_1 norm in the set U. Since ℓ_1 and ℓ_∞ are equal up to a factor equal to the dimension, we get the following.

Observation 4.3.1 *For any point set P of n points and a parameter k, we have that $\nu_\infty^{\mathrm{opt}}(\mathsf{P}, k) \leq \nu_1^{\mathrm{opt}}(\mathsf{P}, k) \leq n\, \nu_\infty^{\mathrm{opt}}(\mathsf{P}, k)$.*

4.3.1 Local Search

We are given a set P of n points and a parameter k. In the following, let

$$\nu_1(\mathbf{C}) = \nu_1^{\mathbf{C}}(\mathsf{P}),$$

C_{opt} denote the set of centers realizing the optimal solution, and let $\nu_1^{\mathrm{opt}} = \nu_1^{\mathrm{opt}}(\mathsf{P}, k)$

A $2n$-approximation. Observation 4.3.1 implies that if we compute a set of centers $\mathbf{C}$ using Theorem 4.2.1 then we have that

$$\nu_1(\mathbf{C})/2n \le \nu_\infty^{\mathbf{C}}(\mathsf{P})/2 \le \nu_\infty^{\mathrm{opt}}(\mathsf{P}, k) \le \nu_1^{\mathrm{opt}} \le \nu_1(\mathbf{C}) \quad \Rightarrow \quad \nu_1(\mathbf{C}) \le 2n\,\nu_1^{\mathrm{opt}}. \tag{4.2}$$

Namely, $\mathbf{C}$ is a $2n$-approximation to the optimal solution.

Improving it. Let $0 < \tau < 1$ be a parameter to be determined shortly. The local search algorithm **AlgLocalSearchKMed** initially sets the current set of centers $\mathbf{C}_{\mathrm{curr}}$ to be $\mathbf{C}$. Next, at each iteration it checks if the current solution $\mathbf{C}_{\mathrm{curr}}$ can be improved by replacing one of the centers in it by a center from the outside (we will refer to such an operation as a ***swap***. There are at most $|\mathsf{P}|\,|\mathbf{C}_{\mathrm{curr}}| = nk$ choices to consider, as we pick a center $\overline{\mathsf{c}} \in \mathbf{C}_{\mathrm{curr}}$ to throw away and a new center to replace it by $\overline{\mathsf{o}} \in (\mathsf{P} \setminus \mathbf{C}_{\mathrm{curr}})$. We consider the new candidate set of centers $\mathbf{K} \leftarrow (\mathbf{C}_{\mathrm{curr}} \setminus \{\overline{\mathsf{c}}\}) \cup \{\overline{\mathsf{o}}\}$. If $\nu_1(\mathbf{K}) \le (1-\tau)\nu_1(\mathbf{C}_{\mathrm{curr}})$ then the algorithm sets $\mathbf{C}_{\mathrm{curr}} \leftarrow \mathbf{K}$. The algorithm continue iterating in this fashion over all possible swaps.

The algorithm **AlgLocalSearchKMed** stops when there is no swap that would improve the current solution by a factor of (at least) $(1-\tau)$. The final content of the set $\mathbf{C}_{\mathrm{curr}}$ is the required constant factor approximation. Note, that the running time of the algorithm is

$$O\left((nk)^2 \log_{1/(1-\tau)} \frac{\nu_1(\mathbf{C})}{\nu_1^{\mathrm{opt}}}\right) = O\left((nk)^2 \log_{1+\tau}(2n)\right) = O\left((nk)^2 \frac{\log n}{\ln(1+\tau)}\right) = O\left((nk)^2 \frac{\log n}{\tau}\right),$$

by Eq. (4.2) and since $1/(1-\tau) \ge 1+\tau$. The final step follows since $1+\tau \le \exp(\tau) \le 1+2\tau$, for $\tau < 1/2$ as can be easily verified. Thus, if τ is polynomially small, then the running time would be polynomial.

4.3.2 Proof of correctness - a game of dictators and drifters

The proof of correctness is somewhat involved and the reader might want to skip it on a first reading.

For the sake of simplicity of exposition, let us assume (for now) that the solution returned by the algorithm can not be improved (at all) by any swap, and let $\mathbf{C}$ be this set of centers. For a center $\overline{\mathsf{c}} \in \mathbf{C}$ and a point $\overline{\mathsf{o}} \in \mathsf{P} \setminus \mathbf{C}$, let $\mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}} = (\mathbf{C} \setminus \{\overline{\mathsf{c}}\}) \cup \{\overline{\mathsf{o}}\}$ denotes the set of centers resulting from applying the swap $\overline{\mathsf{c}} \to \overline{\mathsf{o}}$ to $\mathbf{C}$. We are assuming that there is no beneficial swap; that is,

$$\forall \overline{\mathsf{c}} \in \mathbf{C}, \overline{\mathsf{o}} \in \mathsf{P} \setminus \mathbf{C} \qquad 0 \le \Delta(\overline{\mathsf{c}}, \overline{\mathsf{o}}) = \nu_1(\mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}) - \nu_1(\mathbf{C}). \tag{4.3}$$

The contribution of a point $\mathsf{p} \in \mathsf{P}$ to this quantity is

$$\delta_{\mathsf{p}} = \mathbf{d}(\mathsf{p}, \mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}) - \mathbf{d}(\mathsf{p}, \mathbf{C}).$$

Clearly, if p is served by the same center in $\mathbf{C}$ and in $\mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}$ then $\delta_{\mathsf{p}} = 0$. In particular, we have that $\forall \mathsf{p} \in \mathsf{P} \setminus \Pi(\mathbf{C}, \overline{\mathsf{c}})$ it holds $\delta_{\mathsf{p}} \le 0$. Thus,

$$\forall \overline{\mathsf{c}} \in \mathbf{C}, \overline{\mathsf{o}} \in C_{\mathrm{opt}} \qquad 0 \le \Delta(\overline{\mathsf{c}}, \overline{\mathsf{o}}) \le \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \cup \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} = \sum_{\mathsf{p} \in \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} + \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}}. \tag{4.4}$$

What Eq. (4.4) gives us is a large family of inequalities that all of them hold together. Each inequality is represented by a swap $\overline{\mathsf{c}} \to \overline{\mathsf{o}}$. We would like to pick a set of swaps such that these inequalities, when added together, would imply that $5\nu_1\left(C_{\mathrm{opt}}\right) \geq \nu_1(\mathbf{C})$; namely, that the local search algorithm provides a constant factor approximation to optimal clustering. This idea seems to be somewhat mysterious, but to see that there is indeed hope to achieve that, observe that if our set of swaps T has each center of C_{opt} appearing in it exactly once, then when we add up the first term on the right side of Eq. (4.4), we have

$$\begin{aligned}
\sum_{\overline{\mathsf{c}}\to\overline{\mathsf{o}}\in T}\left(\sum_{\mathsf{p}\in\Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)}\delta_{\mathsf{p}}\right) &= \sum_{\overline{\mathsf{c}}\to\overline{\mathsf{o}}\in T}\left(\sum_{\mathsf{p}\in\Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)}\left(\mathbf{d}(\mathsf{p},\mathbf{C}-\overline{\mathsf{c}}+\overline{\mathsf{o}})-\mathbf{d}(\mathsf{p},\mathbf{C})\right)\right) \\
&\leq \sum_{\overline{\mathsf{c}}\to\overline{\mathsf{o}}\in T}\left(\sum_{\mathsf{p}\in\Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)}\Big(\mathbf{d}(\mathsf{p},C_{\mathrm{opt}})-\mathbf{d}(\mathsf{p},\mathbf{C})\Big)\right)=\sum_{\mathsf{p}\in\mathsf{P}}\Big(\mathbf{d}(\mathsf{p},C_{\mathrm{opt}})-\mathbf{d}(\mathsf{p},\mathbf{C})\Big) \\
&= \nu_1^{\mathrm{opt}}-\nu_1(\mathbf{C}), \qquad (4.5)
\end{aligned}$$

since $\overline{\mathsf{o}}$ ranges over the elements of C_{opt} exactly once, and for $\overline{\mathsf{o}} \in C_{\mathrm{opt}}$ and $\mathsf{p} \in \Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)$ we have that $\mathbf{d}(\mathsf{p},\mathbf{C}-\overline{\mathsf{c}}+\overline{\mathsf{o}}) \leq \mathbf{d}(\mathsf{p},\overline{\mathsf{o}}) = \mathbf{d}(p,C_{\mathrm{opt}})$. Thus, we can bound the first term of the right side of Eq. (4.4) as required. We thus turn our attention to bounding the second term.

Some intuition about this term: This term is the total change in contribution by points in $\mathsf{p} \in \Pi(\mathbf{C},\overline{\mathsf{c}}) \setminus \Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)$. These are points that lose their current (beloved) center $\overline{\mathsf{c}}$. These points might be reassigned to the new center $\overline{\mathsf{o}}$ and their price might not change by a lot. However, the fact that $\mathsf{p} \notin \Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)$ is (intuitively) a sign that p and $\overline{\mathsf{o}}$ might be very far away, and p would have to look far afield for a new center in $\mathbf{C}-\overline{\mathsf{c}}+\overline{\mathsf{o}}$ to serve it. Namely, this might be too expensive overall. To minimize it, intuitively, we would like to make the overall size of the sets $\Pi(\mathbf{C},\overline{\mathsf{c}}) \setminus \Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)$ as small as possible (over our set of swaps T). Intuitively, these sets are minimized when $\Pi(\mathbf{C},\overline{\mathsf{c}})$ is as similar to $\Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)$ as possible.

Thus, we will say that $\overline{\mathsf{c}} \in \mathbf{C}$ ***dominates*** $\overline{\mathsf{o}} \in C_{\mathrm{opt}}$ if and only if

$$\left|\Pi(\mathbf{C},\overline{\mathsf{c}}) \cap \Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)\right| > \left|\Pi\left(C_{\mathrm{opt}},\overline{\mathsf{o}}\right)\right|/2.$$

Clearly, since the clusters of $\mathbf{C}$ form a partition of P, we have that a center $\overline{\mathsf{o}} \in C_{\mathrm{opt}}$ can be dominated by at most one center of $\mathbf{C}$.

In addition, there are centers in $\mathbf{C}$ that dominates more than one center in C_{opt}, and we will refer to such centers as ***dictators***. Nobody wants to deal with dictators[②] (including us) and as such we will not have any dictators involved in swaps in T. The other kind are centers of $\mathbf{C}$ that dominates no center in C_{opt}. We will refer to such centers as ***drifters***. Now, assume that there are overall D dictators in $\mathbf{C}$ dominating S centers of C_{opt}. Let F be the number of centers in C_{opt} which are not dominated by anybody ("free" centers). Then, we have that total number of drifters is exactly

$$F + S - D,$$

as can be easily verified. Note, that $S \geq 2D$, since every dictator has at least two "slaves". As such, the number of drifters is at least $F + S - D \geq F + S - (S/2) = F + S/2$.

The set of swaps T. The set of swaps T we would consider would be constructed as follows. If $\overline{\mathsf{c}} \in \mathbf{C}$ dominates exactly one center $\overline{\mathsf{o}}$ in C_{opt}, then the swap $\overline{\mathsf{c}} \to \overline{\mathsf{o}}$ will be included in T. In addition, we will also add swaps between drifters of $\mathbf{C}$ with centers of C_{opt} that are not swapped yet by swaps of T. The end

[②] Except for arm dealers, art dealers and superpowers, of course.

result is that all the centers of C_{opt} will be covered by the set of swaps T. We will do it in such a way that every drifter participates in at most two swaps of T. Note, that the resulting set of swaps T might swap-out a center of $\mathbf{C}$ at most twice, and it swap-in each center of C_{opt} exactly once. To see why this is possible, observe that we have at least $F + S/2$ drifters that we use to cover $F + S$ "free" and "slave" vertices of C_{opt}. Clearly, this can be done by using each drifter at most twice in swaps of T.[3]

With a little help from my friends. We are still left with the task of assigning the "orphaned" points in $\Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi\Big(C_{\text{opt}}, \overline{\mathsf{o}}\Big)$ to a center of $\mathbf{C} \setminus \{\overline{\mathsf{c}}\}$, for a swap $\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T$. To this end, every point p of P finds a friend $\pi(\mathsf{p}) \in \mathsf{P}$ such that p and $\pi(\mathsf{p})$ have two different centers serving them in $\mathbf{C}$. In particular, if p becomes orphaned than it can be served by going to its friend $\pi(\mathsf{p})$ and travel from there to the center of $\mathbf{C}$ that serves $\pi(\mathsf{p})$.

Since the total distance of a point to its friend is the overhead of this reassignment policy, we would like to make these edges as short as possible. Thus, the permutation π would map the elements of a cluster $\Pi\Big(C_{\text{opt}}, \overline{\mathsf{o}}\Big)$ to itself, for any $\overline{\mathsf{o}} \in C_{\text{opt}}$. The basic idea here is that then we can charge the traveling (of a point to its friend) to the cost of the optimal clustering. In a perfect world, π would also map every cluster of $\mathbf{C}$ into a different cluster of $\mathbf{C}$. This is not quite possible, but we can prove the following weaker claim.

Lemma 4.3.2 *One can find a permutation π, such that for any $\overline{\mathsf{o}} \in C_{\text{opt}}$ and $\overline{\mathsf{c}} \in \mathbf{C}$ and $\mathsf{p} \in U = \Pi\Big(C_{\text{opt}}, \overline{\mathsf{o}}\Big) \cap \Pi(\mathbf{C}, \overline{\mathsf{c}})$ such that if $|U| < \Big|\Pi\Big(C_{\text{opt}}, \overline{\mathsf{o}}\Big)\Big|/2$ (i.e., $\overline{\mathsf{c}}$ does not dominates $\overline{\mathsf{o}}$) then $\pi(\mathsf{p}) \notin \Pi(\mathbf{C}, \overline{\mathsf{c}})$.*

Proof: Let us demystify this claim. Let $A = \Pi\Big(C_{\text{opt}}, \overline{\mathsf{o}}\Big)$, and let $B_i = A \cap \Pi(\mathbf{C}, \overline{\mathsf{c}}_i)$, for $i - 1, \ldots, k$, where $\mathbf{C} = \{\overline{\mathsf{c}}_1, \ldots, \overline{\mathsf{c}}_k\}$. In addition, let $n = |A|$ and assume that $|B_1| \geq |B_2| \geq \cdots \geq |B_k|$.

Clearly, the sets $B_1, \ldots, B_k$ form a partition of A. The claim is that we can find a permutation π of A, such that for every element $\mathsf{p} \in B_i$ we have that $\pi(\mathsf{p}) \notin B_i$, as long as $|B_i| \leq n/2$.

But coming up with such a mapping is easy. Indeed, enumerate the elements of A in such a way that the elements of B_1 are first, then the elements of B_2, and so on. Let p_i be the ith point in this ordering, and consider the mapping $\pi(\mathsf{p}_i) = \mathsf{p}_{((i+\lceil n/2 \rceil) \bmod n)+1}$. Since it shifts the elements of A by distance $n/2$, a set of size at most $\lfloor n/2 \rfloor$ is mapped by π outside its range. Thus establishing the claim. ■

Lemma 4.3.3 *We have that $\sum_{\mathsf{p} \in \mathsf{P}} \mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) \leq 2\nu_1^{\text{opt}}$.*

Proof: For any point $\mathsf{p} \in \mathsf{P}$, we have that $\pi(\mathsf{p})$ and p lie in the same cluster in the optimal clustering. As such, by the triangle inequality, we have $\mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) \leq \mathbf{d}(\mathsf{p}, C_{\text{opt}}) + \mathbf{d}(\pi(\mathsf{p}), C_{\text{opt}})$. As such, since π is a permutation, we have that

$$\sum_{\mathsf{p} \in \mathsf{P}} \mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) \leq \sum_{\mathsf{p} \in \mathsf{P}} \Big(\mathbf{d}(\mathsf{p}, C_{\text{opt}}) + \mathbf{d}(\pi(\mathsf{p}), C_{\text{opt}}) \Big) = 2\nu_1^{\text{opt}}.$$

■

Lemma 4.3.4 *For any $\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T$ and any $\overline{\mathsf{o}'} \in C_{\text{opt}}$ such that $\overline{\mathsf{o}} \neq \overline{\mathsf{o}'}$, we have that $\overline{\mathsf{c}}$ does not dominates $\overline{\mathsf{o}'}$.*

Proof: If $\overline{\mathsf{c}}$ is a drifter then the claim trivially holds. Otherwise, since $\overline{\mathsf{c}}$ can not be a dictator, it must be that it dominates only $\overline{\mathsf{o}}$, which implies that it does not dominates $\overline{\mathsf{o}'}$. ■

Lemma 4.3.5 *We have that* $\displaystyle\sum_{\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T} \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi\left(C_{\text{opt}}, \overline{\mathsf{o}}\right)} \delta_{\mathsf{p}} \leq 4\nu_1^{\text{opt}}.$

[3] Which comes to teach us that even drifters can be useful sometimes.

Proof: For a swap $\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T$ consider a point $\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi\Big(C_{\mathrm{opt}}, \overline{\mathsf{o}}\Big)$. Let $\overline{\mathsf{o}'}$ be the optimal center which its cluster contains p. By Lemma 4.3.4, we know that $\overline{\mathsf{c}}$ does not dominates $\overline{\mathsf{o}'}$, and as such, by Lemma 4.3.2 , we have $\pi(\mathsf{p}) \notin \Pi(\mathbf{C}, \overline{\mathsf{c}})$. As such, we have that

$$\mathbf{d}(\mathsf{p}, \mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}) \leq \mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) + \mathbf{d}(\pi(\mathsf{p}), \mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}) \leq \mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) + \mathbf{d}(\pi(\mathsf{p}), \mathbf{C}),$$

since the center of $\pi(\mathsf{p})$ was not swapped out. As such,

$$\delta_{\mathsf{p}} = \mathbf{d}(\mathsf{p}, \mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}) - \mathbf{d}(\mathsf{p}, \mathbf{C}) \leq \mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) + \mathbf{d}(\pi(\mathsf{p}), \mathbf{C}) - \mathbf{d}(\mathsf{p}, \mathbf{C}) = \nu_{\mathsf{p}}.$$

Note, that by the triangle inequality, ν_{p} is always non-negative. As such, we have that

$$\gamma = \sum_{\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T} \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} \leq \sum_{\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T} \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \nu_{\mathsf{p}} \leq 2 \sum_{\mathsf{p} \in \mathsf{P}} \nu_{\mathsf{p}},$$

since each center of $\overline{\mathsf{c}}$ get swapped out at most twice by T, and as such a point might contribute twice to the summation. But then

$$\gamma \leq 2 \sum_{\mathsf{p} \in \mathsf{P}} \nu_{\mathsf{p}} = 2 \sum_{\mathsf{p} \in \mathsf{P}} \Big(\mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) + \mathbf{d}(\pi(\mathsf{p}), \mathbf{C}) - \mathbf{d}(\mathsf{p}, \mathbf{C}) \Big) = 2 \sum_{\mathsf{p} \in \mathsf{P}} \mathbf{d}(\mathsf{p}, \pi(\mathsf{p})) \leq 4 \nu_1^{\mathrm{opt}},$$

since π is a permutation, and by Lemma 4.3.3. ∎

Lemma 4.3.6 *Assuming that Eq. (4.4) holds, we have that* $\nu_1(\mathbf{C}) \leq 5\nu_1^{\mathrm{opt}}$

Proof: We add Eq. (4.4) overall all the swaps in T. We have that

$$\begin{aligned} 0 &\leq \sum_{\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T} \left(\sum_{\mathsf{p} \in \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} + \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} \right) \leq \sum_{\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T} \sum_{\mathsf{p} \in \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} + \sum_{\overline{\mathsf{c}} \to \overline{\mathsf{o}} \in T} \sum_{\mathsf{p} \in \Pi(\mathbf{C}, \overline{\mathsf{c}}) \setminus \Pi(C_{\mathrm{opt}}, \overline{\mathsf{o}})} \delta_{\mathsf{p}} \\ &\leq \nu_1^{\mathrm{opt}} - \nu_1(\mathbf{C}) + 4\nu_1^{\mathrm{opt}}. \end{aligned}$$

by Eq. (4.5) and Lemma 4.3.5. Implying that $\nu_1(\mathbf{C}) \leq 5\nu_1^{\mathrm{opt}}$. ∎

Removing the strict improvement assumption. In the above proof, we assumed that the current local minimum can not be improved by a swap. Of course, this might not hold for the algorithm solution, since the algorithm allows a swap only if it makes "significant" progress. In particular, Eq. (4.3) is in fact

$$\forall \overline{\mathsf{c}} \in \mathbf{C}, \overline{\mathsf{o}} \in \mathsf{P} \setminus \mathbf{C} \qquad -\tau\, \nu_1(\mathbf{C}) \leq \nu_1(\mathbf{C} - \overline{\mathsf{c}} + \overline{\mathsf{o}}) - \nu_1(\mathbf{C})\,. \tag{4.6}$$

To adapt the proof to use this modified inequalities, observe that the proof worked by adding up k inequalities defined by Eq. (4.3) and showing that the right side is bounded by $5\nu_1\Big(C_{\mathrm{opt}}\Big) - \nu_1(\mathbf{C})$. Repeating the same argumentation on the modified inequalities, would yield

$$-\tau k\, \nu_1(\mathbf{C}) \leq 5\nu_1\Big(C_{\mathrm{opt}}\Big) - \nu_1(\mathbf{C})\,.$$

This implies $\nu_1(\mathbf{C}) \leq 5\nu_1^{\mathrm{opt}}/(1 - \tau k)$. For arbitrary $0 < \varepsilon < 1$, Setting $\tau = \varepsilon/2k$ we have that $\nu_1(\mathbf{C}) \leq 5(1 + \varepsilon k)\nu_1^{\mathrm{opt}}$, since $1/(1 - \tau k) \leq 1 + 2\tau k = 1 + \varepsilon$, for $\tau \leq 1/2k$. We summarize:

Theorem 4.3.7 *Let* P *be a set of* n *points in a metric space. For* $0 < \varepsilon < 1$*, one can compute a* $(5 + \varepsilon)$*-approximation to the optimal* k*-median clustering of* P*. The running time of the algorithm is* $O\Big(n^2 k^3 \frac{\log n}{\varepsilon}\Big)$.

4.4 On k-means clustering

In the ***k-means clustering problem***, a set $\mathsf{P} \subseteq \mathcal{X}$ is provided together with a parameter k. We would like to find a set of k points $\mathbf{C} \subseteq \mathsf{P}$, such that the sum of squared distances of all the points of P to their closest point in $\mathbf{C}$ is minimized.

Formally, given a set of centers $\mathbf{C}$, the k-center clustering ***price*** of clustering P by $\mathbf{C}$ is denoted by

$$\nu_2^{\mathbf{C}}(\mathsf{P}) = \sum_{\mathsf{p} \in \mathsf{P}} (\mathbf{d}(\mathsf{p}, \mathbf{C}))^2 ,$$

and the ***k-means problem*** is to find a set $\mathbf{C}$ of k points, such that $\nu_2^{\mathbf{C}}(\mathsf{P})$ is minimized; namely,

$$\nu_2^{\mathrm{opt}}(\mathsf{P}, k) = \min_{\mathbf{C}, |\mathbf{C}|=k} \nu_2^{\mathbf{C}}(\mathsf{P}) .$$

4.4.1 Local Search

Local search also works for k-means and yields a constant factor approximation.

Theorem 4.4.1 *Let P be a set of n points in a metric space. For $0 < \varepsilon < 1$, one can compute a $(25 + \varepsilon)$-approximation to the optimal k-means clustering of P. The running time of the algorithm is $O\left(n^2 k^3 \frac{\log n}{\varepsilon}\right)$.*

4.5 Bibliographical Notes

In this chapter we introduced the problem of clustering and showed some algorithms that achieve constant factor approximation. A lot more is known about these problems including faster and better clustering algorithms but to discuss them we need more advanced tools than what we currently have at hand.

Clustering is widely researched. Unfortunately, a large fraction of the work on this topic rely on heuristics or experimental studies. The inherent problem seems to be the lack of a universal definition of what is a good clustering. This depends on the application at hand, as is rarely clearly defined. In particular, no clustering algorithm can achieve all desired properties together, see the work by Kleinberg [Kle02] (although it is unclear if all these desired properties are indeed natural or really desired).

k-center Clustering. The algorithm **GreedyKCenter** was described by Gonzalez [Gon85], but it was probably known before, as the notion of r-net is much older. The hardness of approximating k-center clustering was shown by Feder and Greene [FG88].

k-median/means clustering. The local search algorithm is due to Arya *et al.* [AGK+01]. The extension to k-mans is due to Kanungo *et al.* [KMN+04]. The extension is not completely trivial since the triangle inequality no longer holds. However, some approximate version of the triangle inequality does hold. Instead of performing a single swap, one can decide to do p swaps simultaneously. Thus, the running time deteriorates since there are more possibilities to check. This improves the approximation constant for k-median (resp., k-means) to $(3 + 2/p)$ (resp. $(3 + 2/p)^2$). Unfortunately, this is (essentially) tight in the worst case. See [AGK+01, KMN+04] for details.

The k-median and k-mean clustering are more interesting in the Euclidean settings where there is considerably more structure, and one can compute $(1 + \varepsilon)$-approximation in polynomial time for fixed ε. We will return to this topic later.

Since k-median and k-means clustering can be easily be reduced to Dominating Set in a graph, this implies that both clustering problems are NP-Hard to solve exactly.

One can compute a similar permutation to the greedy permutation (for k-center clustering) also for k-median clustering. See the work by Mettu and Plaxton [MP03].

Handling Outliers. The problem of handling outliers is still not well understood. See the work of Charikar *et al.* [CKMN01] for some relevant results. In particular, for k-center clustering they get a constant factor approximation, and Exercise 4.6.2 is taken from there. For k-median clustering they present a constant factor approximation using linear programming relaxation, that also approximates the number of outliers. Recently, Chen [Che07] provided a constant factor approximation algorithm by extending the work of Charikar *et al.*. The problem of finding a simple algorithm with simple analysis for k-median clustering with outliers is still open, as Chen work is quite involved.

Open Problem 4.5.1 *Get a* simple *constant factor k-median clustering algorithm that runs in polynomial time and uses exactly m outliers. Alternatively, solve this problem in the case where* P *is a set of n points in the plane. (The emphasize here is that the analysis of the algorithm should be simple.)*

Bi-criteria approximation. All clustering algorithms tend to become considerably easier if one allows to trade-off in the number of clusters. In particular, one can compute a constant factor approximation to the optimal k-median/means clustering using $O(k)$ centers in $O(nk)$ time. The algorithm succeeds with constant probability. See the work by Indyk [Ind99] and Chen [Che06] and references therein.

Facility Location. All the problems mentioned here fall into the family of facility location problems. There are numerous variants. The more specific ***facility location*** problem is a variant of k-median clustering where the number of clusters is not specified, but instead one has to pay to open a facility in a certain location. Local-search also works for this variant.

Local search. As mentioned above, ***local search*** also works for k-means clustering [AGK+01]. A collection of some basic problems for which local search works is described in the book by Kleinberg and Tardos [KT06]. Local search is a widely used heuristic for attacking NP-Hard problems. The idea is usually to start from a solution and try to locally improve it. Here, one defines a neighborhood of the current solution, and one tries to move to the best solution in this neighborhood. In this sense, local search can be thought of as a hill-climbing/EM (expectation maximization) algorithm. Problem for which local search was used include Vertex Cover, Traveling Salesperson and Satisfiability, and probably many more problems.

Provable cases where local search generates a guaranteed solution are less common and include facility location, k-median clustering [AGK+01], k-means clustering [KMN+04], weighted max cut, metric labeling problem with the truncated linear metric [GT00], and image segmentation [BVZ01]. See [KT06] for more references and a nice discussion of the connection of local search to the ***Metropolis algorithm*** and ***simulated annealing***.

4.6 Exercises

Exercise 4.6.1 (Handling outliers.) [10 Points]

Given a point set P, we would like to perform k-median clustering of it, when we are allowed to ignore m of the points. These m points are ***outliers*** which we would like to ignore since they represent irrelevant data. Unfortunately, we do not know the m outliers in advance. It is natural to conjecture that one can perform a local search for the optimal solution. Here ones maintain a set of k centers and a set of m outliers. At every point in time the algorithm moves one of the centers or the outliers if it improves the solution.

Show that local-search does not work for this problem; namely, the approximation factor is not a constant.

Exercise 4.6.2 (Handling outliers for k-center clustering.) [10 Points]

Given P, k and m, present an a polynomial time algorithm that computes a constant factor approximation to the optimal k-center clustering of P with m outliers. (Hint: Assume first that you know what is the radius of the optimal solution.)

Chapter 5

On Complexity, Sampling, and ε-Nets and ε-Samples

> "I've never touched the hard stuff, only smoked grass a few times with the boys to be polite, and that's all, though ten is the age when the big guys come around teaching you all sorts to things. But happiness doesn't mean much to me, I still think life is better. Happiness is a mean son of a bitch and needs to be put in his place. Him and me aren't on the same team, and I'm cutting him dead. I've never gone in for politics, because somebody always stand to gain by it, but happiness is an even crummier racket, and their ought to be laws to put it out of business."
>
> – – Momo, Emile Ajar

In this chapter we will try to quantify the notion of geometric complexity. It is intuitively clear that a ● (i.e., disk) is a simpler shape than an ⬬ (i.e., ellipse), which is in turn simpler than a ☻ (i.e., smiley). This becomes even more important when we consider several such shapes and how they interact with each other. As these examples might demonstrate, this notion of complexity is somewhat elusive.

Next, we show that one can capture the structure of a distribution/point set by a small subset. The size here would depend on the complexity of the shapes/ranges we care about, but it would be independent of the size of the point set.

5.1 VC Dimension

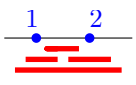
Definition 5.1.1 A ***range space*** S is a pair $(\mathsf{X}, \mathcal{R})$, where X is a ***ground set*** (finite or infinite) set and $\mathcal{R}$ is a (finite or infinite) family of subsets of X. The elements of X are ***points*** and the elements of $\mathcal{R}$ are ***ranges***. For $A \subseteq \mathsf{X}$, let $\mathcal{R}_{|A} = \left\{ \mathbf{r} \cap A \;\middle|\; \mathbf{r} \in R \right\}$ denote the ***projection*** of $\mathcal{R}$ on A.

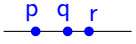
If $\mathcal{R}_{|A}$ contains all subsets of A (i.e., if A is finite, we have $\left|\mathcal{R}_{|A}\right| = 2^{|A|}$) then A is ***shattered*** by $\mathcal{R}$.

The ***Vapnik-Chervonenkis*** dimension (or VC ***dimension***) of S, denoted by $\dim_{\mathsf{VC}}(S)$, is the maximum cardinality of a shattered subset of X. If there are arbitrarily large shattered subsets then $\dim_{\mathsf{VC}}(S) = \infty$.

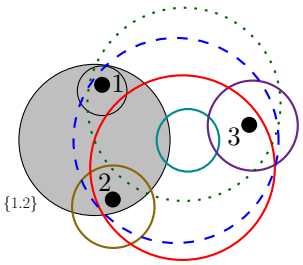
5.1.1 Examples

Intervals. Consider the set X to be the real line, and $\mathcal{R}$ be the set of all intervals on the real line. Clearly, for two points in the real line, say, $A = \{1, 2\}$ one can find 4 intervals that contain all possible subsets of A. However, this is false for a set of three points $B = \{p, q, r\}$, since there is no interval that can contain the two extreme points p and r without also containing q. Namely, the subset $\{p, r\}$ is not realizable for intervals. Implying that the largest shattered set by the range space (real line, intervals) if of size two. We conclude that the VC-dimension of this space is two.

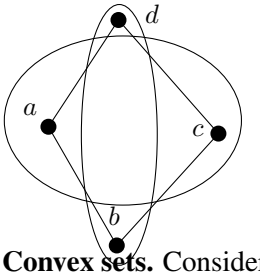




Disks. Let $\mathsf{X} = \mathbb{R}^2$, and let $\mathcal{R}$ be the set of disks in the plane. Clearly, for three points in the plane $1, 2, 3$, one can find 8 disks that realize all possible 2^3 different subsets. See figure on the right.

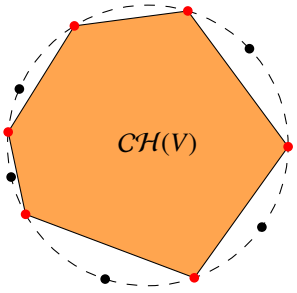


But can disks shatter a set with four points? Consider such a set P of four points, and there are two possible options. Either the convex-hull of P has three points on its boundary, and in this case, the subset having those vertices in the subset but not including the middle point is impossible, by convexity.



Alternatively, if all four points are vertices of the convex hull, and they are a, b, c, d along the boundary of the convex hull, either the set $\{a, c\}$ or the set $\{b, d\}$ is not realizable. Indeed, if both options are realizable, then consider the two disks D_1 and D_2 the realizes those assignments. Clearly, D_1 and D_2 must intersect in four points, but this is not possible, since two disks have at most two intersection points. See figure on the left.

Convex sets. Consider the range space $\mathsf{S} = (\mathbb{R}^2, \mathcal{R})$, where $\mathcal{R}$ is the set of all (closed) convex sets in the plane. We claim that the $\dim_{\mathsf{VC}}(\mathsf{S}) = \infty$. Indeed, consider a set U of n points $p_1, \ldots, p_n$ all lying on the boundary of the unit circle in the plane. Let V be any subset of U, and consider the convex-hull $\mathcal{CH}(V)$. Clearly, $\mathcal{CH}(V) \in \mathcal{R}$, and furthermore, $\mathcal{CH}(V) \cap U = V$. Namely, any subset of U is realizable by S. Thus, S can shatter sets of arbitrary size, and its VC-dimension is unbounded.



Complement. Consider the range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with $\mathsf{d} = \dim_{\mathsf{VC}}(\mathsf{S})$. Next, consider the complement space, $\overline{\mathsf{S}} = (\mathsf{X}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} = \left\{ \mathsf{X} \setminus \mathbf{r} \;\middle|\; \mathbf{r} \in \mathcal{R} \right\};$$

namely, the ranges of $\overline{\mathsf{S}}$ are the complement of the ranges in S. What is the VC-dimension of $\overline{\mathsf{S}}$? Well, clearly a set $B \subseteq \mathsf{X}$ is shattered by $\overline{\mathsf{S}}$, if and only if it is shattered by S. Thus, $\dim_{\mathsf{VC}}\left(\overline{\mathsf{S}}\right) = \dim_{\mathsf{VC}}(\mathsf{S})$.

Lemma 5.1.2 *For a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ we have that for the complement range space $\overline{\mathsf{S}}$ it holds $\dim_{\mathsf{VC}}(\mathsf{S}) = \dim_{\mathsf{VC}}\left(\overline{\mathsf{S}}\right)$.*

5.1.1.1 Half spaces

Let $S = (\mathsf{X}, \mathcal{R})$, where $\mathsf{X} = \mathbb{R}^d$ and $\mathcal{R}$ is the set of all (closed) halfspaces in $\mathbb{R}^d$. To see what is the VC-dimension of S, we need the following result of Radon.

Theorem 5.1.3 (Radon's Lemma) *Let $A = \{p_1, \ldots, p_{d+2}\}$ be a set of $d+2$ points in $\mathbb{R}^d$. Then, there exists two disjoint subsets C and D of A, such that $\mathcal{CH}(C) \cap \mathcal{CH}(D) \neq \emptyset$.*

Proof: We claim that there exists $\beta_1, \ldots, \beta_{d+2}$, non all of them zero, such that $\sum_i \beta_i p_i = 0$ and $\sum_i \beta_i = 0$.

Proof: If $p_{d+1} = p_{d+2}$ then we are done. Otherwise, without loss of generality, assume that $p_1, \ldots, p_d$ spans $\mathbb{R}^d$. Then, there are two non-zero combinations of $p_1, \ldots, p_d$, such that $p_{d+1} = \sum_{i=1}^{d} \alpha_i p_i$ and $p_{d+2} = \sum_{i=1}^{d} \gamma_i p_i$. Let $\alpha = \sum_{i=1}^{d} \alpha_i - 1$ and $\gamma = \sum_{i=1}^{d} \gamma_i - 1$. If $\alpha = 0$ then $\sum_{i=1}^{d} \alpha_i p_i - p_{d+1}$ (which is the origin) is the required combination. Similarly, we are done if $\gamma = 0$. Otherwise, consider the point $\sum_{i=1}^{d} (\alpha_i/\alpha) p_i - p_{d+1}/\alpha - \left(\sum_{i=1}^{d} (\gamma_i/\gamma) p_i - p_{d+2}/\gamma\right)$. Clearly, this is the required point. ■

Assume, for the sake of simplicity of exposition, that the $\beta_1, \ldots \beta_k \geq 0$ and $\beta_{k+1}, \ldots, \beta_{d+2} < 0$. Furthermore, let $\mu = \sum_{i=1}^{k} \beta_i$. We have that

$$\sum_{i=0}^{k} \beta_i p_i = -\sum_{i=k+1}^{d+2} \beta_i p_i.$$

In particular, $v = \sum_{i=0}^{k} (\beta_i/\mu) p_i$ is a point in the $\mathcal{CH}(\{p_1, \ldots, p_k\})$ and $\sum_{i=k+1}^{d+2} -(\beta_i/\mu) p_i \in \mathcal{CH}(\{p_{k+1}, \ldots, p_{d+2}\})$. We conclude that v is in the intersection of the two convex hulls, as required. ■

In particular, this implies that if a set Q of $d+2$ points is being shattered by S, we can partition this set Q into two disjoint sets A and B such that $\mathcal{CH}(A) \cap \mathcal{CH}(B) \neq \emptyset$. It should now be clear that any halfspace h^+ containing all the points of A, must also contain a point of the $\mathcal{CH}(B)$. But this implies that a point of B must be in h^+. Namely, the subset A can not be realized by a halfspace, which implies that Q can not be shattered. Thus $\dim_{\mathsf{VC}}(S) < d+2$. It is also easy to verify that the regular simplex with $d+1$ vertices is being shattered by S. Thus, $\dim_{\mathsf{VC}}(S) = d+1$.

5.2 Shattering Dimension and the Dual Shattering Dimension

The main property of a range space with bounded VC-dimension is that the number of ranges that one has to consider for a set of n elements, grows polynomially in n (with the power being the dimension), instead of exponentially. Formally, let

$$\mathcal{G}_d(n) = \sum_{i=0}^{d} \binom{n}{i} \leq \sum_{i=0}^{d} \frac{n^i}{i!} \leq n^d, \tag{5.1}$$

for $d > 1$ (the cases where $d = 0$ or $d = 1$ are not interesting and we will just ignore them with contempt). Note that for all $n, d \geq 1$, we have $\mathcal{G}_d(n) = \mathcal{G}_d(n-1) + \mathcal{G}_{d-1}(n-1)$[④].

Lemma 5.2.1 (Sauer's Lemma) *If $(X, \mathcal{R})$ is a range space of VC-dimension d with $|X| = n$ then $|\mathcal{R}| \leq \mathcal{G}_d(n)$.*

Proof: The claim trivially holds for $d = 0$ or $n = 0$.

Let x be any element of X, and consider the sets

$$\mathcal{R}_x = \left\{ \mathbf{r} \setminus \{x\} \;\middle|\; \mathbf{r} \cup \{x\} \in \mathcal{R} \text{ and } \mathbf{r} \setminus \{x\} \in \mathcal{R} \right\} \quad \text{and} \quad \mathcal{R} \setminus x = \left\{ \mathbf{r} \setminus \{x\} \;\middle|\; \mathbf{r} \in \mathcal{R} \right\}.$$

Observe that $|\mathcal{R}| = |\mathcal{R}_x| + |\mathcal{R} \setminus x|$. Indeed, we charge the elements of $\mathcal{R}$ to their corresponding element in $\mathcal{R} \setminus x$. The only bad case is when there is a range $\mathbf{r}$ such that both $\mathbf{r} \cup \{x\} \in \mathcal{R}$ and $\mathbf{r} \setminus \{x\} \in \mathcal{R}$, because then these two distinct ranges get mapped to the same range in $\mathcal{R} \setminus x$. But such ranges contribute exactly one element to $\mathcal{R}_x$.

Observe that $(X \setminus \{x\}, \mathcal{R}_x)$ has VC dimension $d-1$, as the largest set that can be shattered is of size $d-1$. Indeed, any set $B \subset X \setminus \{x\}$ shattered by $\mathcal{R}_x$, implies that $B \cup \{x\}$ is shattered in $\mathcal{R}$.

Thus,

$$|\mathcal{R}| = |\mathcal{R}_x| + |\mathcal{R} \setminus x| \leq \mathcal{G}_{d-1}(n-1) + \mathcal{G}_d(n-1) = \mathcal{G}_d(n),$$

by induction. ■

Definition 5.2.2 (Shatter function.) Given a range space $S = (X, \mathcal{R})$, its ***shatter function*** $\pi_S(m)$ is the maximum number of sets that might be created by S when restricted to subsets of size m. Formally,

$$\pi_S(m) = \max_{\substack{B \subset X \\ |B| = m}} \left| \mathcal{R}_{|B} \right|$$

The ***shattering dimension*** of S is the smallest d such that $\pi_S = O(m^d)$, for all m.

[④] Here is a cute (and standard) counting argument: $\mathcal{G}_d(n)$ is just the number of different subsets of size at most d out of n elements. Now, we either decide to not include the first element in these subsets (i.e., $\mathcal{G}_d(n-1)$) or, alternatively, we include the first element in these subsets, but then there are only $d-1$ elements left to pick (i.e., $\mathcal{G}_{d-1}(n-1)$).

By applying Lemma 5.2.1, to a finite subset of X, we get:

Corollary 5.2.3 *If $\mathsf{S} = (X, \mathcal{R})$ is a range space of VC-dimension d then for every finite subset B of X, we have $\left|\mathcal{R}_{|B}\right| \leq \pi_{\mathsf{S}}(|B|) \leq \mathcal{G}_d(|B|)$.*

Namely, the VC-dimension of a range space always bound the shattering dimension of this range space.

Proof: For the second part, let $n = |B|$, and observe that $\left|\mathcal{R}_{|B}\right| \leq \mathcal{G}_d(n) \leq n^d$, by Eq. (5.1). As such, $\left|\mathcal{R}_{|B}\right| \leq n^d$, and, by definition, the shattering dimension of S is at most d; namely, the shattering dimension is bounded by the VC-dimension. ■

Disks revisited. To see why the shattering dimension is more convenient to work with than VC-dimension, consider the range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, where $\mathsf{X} = \mathbb{R}^2$, and $\mathcal{R}$ is the set of disks in the plane. We know that the VC-dimension of S is 3 (see Section 5.1.1).

Now, consider the shattering dimension of this range space. Let P be a set of n points in the plane, and observe that given a disk in the plane, we can continuously deform it till it passes through three points of P on its boundary, and no point outside the disk is now inside its interior, and vice versa. As such, the number of subsets of P that one can realize by intersecting it with a disk is bounded by $n^3 2^3$ (we pick the three vertices that determine the disk, and for each of the three vertices we determine whether we consider it to be inside the disk or not). As such, the shatter dimension of S is 3.

That might not seem like a great simplification over the same bound we got by arguing about the VC-dimension. However, the above argumentation give us a very powerful tool – the shattering dimension of a range space defined by a family of shapes is always bounded by the number of points that determine a shape in the family. Thus, the shattering dimension of, say, arbitrarily oriented rectangles in the plane is five, since such a rectangle is uniquely determined by five points.

5.2.1 The Dual Shattering Dimension

Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, consider a point $\mathsf{p} \in \mathsf{X}$. There is a set of ranges of $\mathcal{R}$ associated with p; namely, the set of all ranges of $\mathcal{R}$ that contains p denoted by

$$\mathcal{R}_{\mathsf{p}} = \left\{ \mathbf{r} \;\middle|\; \mathbf{r} \in \mathcal{R},\ \text{the range } \mathbf{r} \text{ contains } \mathsf{p} \right\}.$$

This gives rise to a natural dual range space to S. Formally, the ***dual range space*** to a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ is the space $\mathsf{S}^\star = (\mathcal{R}, \mathsf{X}^\star)$, where $\mathsf{X}^\star = \left\{ \mathcal{R}_{\mathsf{p}} \;\middle|\; \mathsf{p} \in \mathsf{X} \right\}$.

To understand what the dual space is, consider X to be the plane, and $\mathcal{R}$ to be a set of m disks. Then, in the dual range space $\mathsf{S}^\star = (\mathcal{R}, \mathsf{X}^\star)$, every point p in the plane has a set associated with it in $\mathsf{X}^\star$, which is the sets of disks of $\mathcal{R}$ that contains p. In particular, if we consider the arrangement formed by the m disks of $\mathcal{R}$, then all the points lying inside a single face of this arrangement correspond to the same set of $\mathsf{X}^\star$. Namely, the number of ranges in $\mathsf{X}^\star$ is bounded by the complexity of the arrangement of these disks, which is $O(m^2)$.

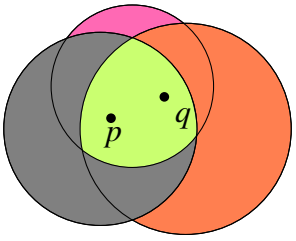


Figure 5.1: $\mathcal{R}_p = \mathcal{R}_q$.

Thus, let the ***dual shatter function*** of the range space S be $\pi^\star_{\mathsf{S}}(m) = \pi_{\mathsf{S}^\star}(m)$, where $\mathsf{S}^\star$ is the dual range space to S. The ***dual shattering dimension*** of S is just the shattering dimension of $\mathsf{S}^\star$.

Note, that the dual shattering dimension might be smaller than the shattering dimension or the VC-dimension of the range space. Indeed, in the case of disks in the plane, the dual shattering dimension is just 2, while the VC-dimension and the shattering dimension of this range space is 3. Note, also, that in

geometric settings bounding the dual shattering dimension is relatively easy, as all you have to do is to bound the complexity of the arrangement of m ranges of this space.

The following lemma shows a connection between the VC-dimension of a space and its dual. The proof is left as an exercise for the interested reader[②] (see Exercise 5.7.1).

Lemma 5.2.4 *Consider a range space $\mathsf{S} = (X, \mathcal{R})$ with VC-dimension d. The dual range space $\mathsf{S}^\star = (\mathcal{R}, X^\star)$ has VC dimension bounded by 2^d.*

5.2.1.1 Mixing range spaces

Lemma 5.2.5 *Let $\mathsf{S} = (X, \mathcal{R})$ and $\mathsf{T} = (X, \mathcal{R}')$ be two range spaces of dimension d and d', respectively, where $d, d' > 1$. Let $\widehat{\mathcal{R}} = \left\{ \mathbf{r} \cup \mathbf{r}' \;\middle|\; \mathbf{r} \in \mathcal{R}, \mathbf{r}' \in \mathcal{R}' \right\}$. Then, for the range space $\widehat{\mathsf{S}} = (X, \widehat{\mathcal{R}})$, we have that $\mathsf{VC}\left(\widehat{\mathsf{S}}\right) = O((d + d') \log(d + d'))$.*

Proof: Let B be a set of n points in X that are being shattered by $\widehat{\mathsf{S}}$. There are $\mathcal{G}_d(n)$ and $\mathcal{G}_{d'}(n)$ different assignments for the elements of B by ranges of $\mathcal{R}$ and $\mathcal{R}'$, respectively. Every subset C of B realized by $\widehat{r} \in \widehat{\mathcal{R}}$, is a union of two subsets $B \cap \mathbf{r}$ and $B \cap \mathbf{r}'$ where $\mathbf{r} \in R$ and $\mathbf{r}' \in \mathcal{R}'$. Thus, the number of different subsets of B realized by $\widehat{\mathsf{S}}$ is bounded by $\mathcal{G}_d(n)\mathcal{G}_{d'}(n)$. Thus, $2^n \leq n^d n^{d'}$, for $d, d' > 1$. We conclude $n \leq (d + d') \lg n$, which implies that $n \leq c(d + d') \log(d + d')$, for some constant c. ■

Corollary 5.2.6 *Let $\mathsf{S} = (X, \mathcal{R})$ and $\mathsf{T} = (X, \mathcal{R}')$ be two range spaces of VC-dimension d and d', respectively, where $d, d' > 1$. Let $\widehat{\mathcal{R}} = \left\{ \mathbf{r} \cap \mathbf{r}' \;\middle|\; \mathbf{r} \in \mathcal{R}, \mathbf{r}' \in \mathcal{R}' \right\}$. Then, for the range space $\widehat{\mathsf{S}} = (X, \widehat{\mathcal{R}})$, we have that $\dim_{\mathsf{VC}}(\widehat{\mathsf{S}}) = O((d + d') \log(d + d'))$*

Proof: Observe that $\mathbf{r} \cap \mathbf{r}' = \overline{\overline{\mathbf{r}} \cup \overline{\mathbf{r}'}}$, and thus the claim follows by Lemma 5.1.2 and Lemma 5.2.5. ■

In fact, we can summarize the above observations, as follows.

Corollary 5.2.7 *Any finite sequence of combining range spaces with finite VC-dimension results in a range space with a finite VC-dimension.*

5.3 On ε-nets and ε-sampling

5.3.1 ε-nets and ε-samples

Definition 5.3.1 (ε-sample) Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space, and let B be a finite subset of X. For $0 \leq \varepsilon \leq 1$, a subset $C \subseteq B$, is an ***ε-sample*** for B if for any range $\mathbf{r} \in \mathcal{R}$, we have

$$\left| \frac{|B \cap \mathbf{r}|}{|B|} - \frac{|C \cap \mathbf{r}|}{|C|} \right| \leq \varepsilon.$$

Namely, ε-sample is a subset of the ground set that "captures" the range space up to an error of ε. Namely, to estimate (approximately) the fraction of the ground set covered by a range $\mathbf{r}$, it is sufficient to count the points of C that falls inside $\mathbf{r}$.

If $B = \mathsf{X}$ and B is a finite set we will abuse notations slightly, and refers to C as a ***ε-sample*** for S.

[②] The author is quite aware that the interest of the reader in this issue might not be the result of free choice. Nevertheless, one might draw some comfort from the realization that the existence of the interested reader is as much of an illusion as the existence of free choice. Both are convenient to assume, and both are probably false. Or maybe not.

To see the usage of such a sample, consider $C = \mathsf{X}$ to be, say, the population of a country (i.e., an element of X is a citizen). A range in $\mathcal{R}$ is the set of all people in the country that answer yes to a question (i.e., would you vote for party Y? would you buy a bridge from me? stuff like that). An ε-sample of this range space enable us to estimate reliably (up to an error of ε) the answer for all these questions, by just asking the people in the sample.

The natural question of course is to find such a subset of small (or minimal) size.

Theorem 5.3.2 (ε-sample theorem, [VC71]) *There is a positive constant c such that if $(\mathsf{X}, \mathcal{R})$ is any range space with VC-dimension at most d, $B \subseteq \mathsf{X}$ is a finite subset and $\varepsilon, \delta > 0$, then a random subset C of cardinality s of B where s is at least the minimum between $|B|$ and*

$$\frac{c}{\varepsilon^2}\left(d \log \frac{d}{\varepsilon} + \log \frac{1}{\delta}\right)$$

is an ε-sample for B with probability at least $1 - \delta$.

Sometimes it is sufficient to have (hopefully smaller) samples with a weaker property – if a range is "heavy" then there is an element in our sample that is in this range.

Definition 5.3.3 (ε-net) A set $N \subseteq B$ is an ***ε-net*** for B, if for any range $\mathbf{r} \in \mathcal{R}$, if $|\mathbf{r} \cap B| \geq \varepsilon |B|$ implies that $\mathbf{r}$ contains at least one point of N (i.e., $\mathbf{r} \cap N \neq \emptyset$).

Theorem 5.3.4 (ε-net theorem, [HW87]) *Let $(\mathsf{X}, \mathcal{R})$ be a range space of VC-dimension d, let B be a finite subset of X and suppose $0 < \varepsilon, \delta < 1$. Let N be a set obtained by m random independent draws from B, where*

$$m \geq \max\left(\frac{4}{\varepsilon} \log \frac{2}{\delta}, \frac{8d}{\varepsilon} \log \frac{8d}{\varepsilon}\right). \tag{5.2}$$

Then N is an ε-net for B with probability at least $1 - \delta$.

The above two theorems also holds for spaces with shattering dimension at most d. The constant in the sample size deteriorates a bit.

5.3.2 Some Applications

We mention two easy applications of these theorems, demonstrating (hopefully) the power of these theorems.

5.3.2.1 Range searching

So, consider a (very large) set of points P in the plane, and we would like to be able to quickly decide how many points are included inside a query rectangle, and let assume that we allow ourselves 1% error (of the whole "population"). What Theorem 5.3.2 tells us, is that there is a subset of *constant size* (that depends only on ε) that can be used to perform this estimation, and it works for *all* query rectangles (we used here the fact that rectangles in the plane have finite VC-dimension). In fact, a random sample of this size works with constant probability.

5.3.2.2 Learning a concept

Assume that we have a function f defined in the plane that returns '1' inside an (unknown) disk, and '0' outside it. There is some distribution $\mathcal{D}$ defined over the plane, and we pick points from this distribution. Furthermore, we can compute the function for these labels (i.e., we can compute f for certain values, but it is expensive).

Theorem 5.3.2 tells us that if we pick (roughly) $O((1/\varepsilon)\log(1/\varepsilon))$ random points in a sample R from this distribution, compute the labels for the samples, and find the smallest disk D that contains the sampled labeled by '1' and does not contain any of the '0' points, then the function g that returns '1' inside the disk and '0' otherwise, correctly classifies all but ε-fraction of the points (i.e., the probability of misclassifying a point picked according to the given distribution is smaller than ε).

To see that, consider the range space S having the plane as the ground set, and the symmetric difference between two disks as the ranges. By Corollary 5.2.7, this range space has finite VC dimension. Now, consider the (unknown) disk D' that induces f and the region $\mathbf{r} = D \oplus D'$. Clearly, the learned classifier g returned incorrect answer only for points picked inside $\mathbf{r}$.

So the probability for a mistake in the classification, is the measure of $\mathbf{r}$ under the distribution $\mathcal{D}$. So, if $\mathbf{Pr}_{\mathcal{D}}\Big[\mathbf{r}\Big] > \varepsilon$ then by the ε-net Theorem (i.e., Theorem 5.3.4) the set R is an ε-net for S (ignore for the time being the possibility that the random sample fails to be an ε-net) and as such, R contains a point q inside $\mathbf{r}$. But then, it is not possible that g (which classifies correctly all the sampled points of R) make a mistake on q. A contradiction, because by construction, the range $\mathbf{r}$ is where g misclassifies points. We conclude that $\mathbf{Pr}_{\mathcal{D}}\Big[\mathbf{r}\Big] \leq \varepsilon$, as desired.

Tell me lies, tell me sweet little lies. The careful reader might be tearing her hair out because of the above description. First, Theorem 5.3.4 might fail, and the above conclusion might not hold. This is of course true, and in real applications one might use much larger sample to guarantee that the probability of failure is so small that it can be practically ignored. A more serious issue is that Theorem 5.3.4 is defined only for finite sets. Nowhere does it speak about a continuous distribution. Intuitively, one can approximate a continuous distribution to an arbitrary precision using a huge sample, and apply the theorem to this sample as our ground set. A formal proof is more tedious and requires extending the proof of Theorem 5.3.4 to continuous distributions. This is straightforward and we will ignore this topic altogether.

A Naive Proof of the ε-Sample Theorem. To demonstrate why the ε-sample/net Theorems are interesting, let us try to prove the ε-sample Theorem in the natural naive way. Thus, consider a finite range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with shattering dimension d. And consider a range $\mathbf{r}$ that contains, say, a p fraction of the points of X, where $p \geq \varepsilon$. Consider a random sample R of r points from X, picked with replacement.

Let p_i be the ith sample point, and let X_i be an indicator variable which is one if and only if $\mathsf{p}_i \in \mathbf{r}$. Clearly, $(\sum_i X_i)/r$ is an estimate for $p = |\mathbf{r} \cap \mathsf{X}| / |\mathsf{X}|$. We would like this estimate to be within $\pm\varepsilon$ of p, and with confidence $\geq 1 - \delta$.

As such, the sample failed if $\left|\sum_{i=1}^{r} X_i - pr\right| \geq \Delta = \varepsilon r = (\varepsilon/p)pr$. Set $\phi = \varepsilon/p$ and $\mu = \mathbf{E}[\sum_i X_i] = pr$. Using the Chernoff inequality (Theorem 25.2.6 and Theorem 25.2.9) we have

$$\begin{aligned}\mathbf{Pr}\left[\left|\sum_{i=1}^{r} X_i - pr\right| \geq (\varepsilon/p)pr\right] &= \mathbf{Pr}\left[\left|\sum_{i=1}^{r} X_i - \mu\right| \geq \phi\mu\right] \leq \exp\Big(-\mu\phi^2/2\Big) + \exp\Big(-\mu\phi^2/4\Big) \\ &\leq 2\exp\Big(-\mu\phi^2/4\Big) = 2\exp\left(-\frac{\varepsilon^2}{4p}r\right) \leq \delta,\end{aligned}$$

for $r \geq \left\lceil \frac{4}{\varepsilon^2}\ln\frac{1}{\delta}\right\rceil \geq \left\lceil \frac{4p}{\varepsilon^2}\ln\frac{1}{\delta}\right\rceil$.

Viola! We had proved the ε-sample Theorem. Well, not quite. We proved that the sample works correctly for a single range. The problem is that the number of ranges that we need to prove the theorem for is $\pi_{\mathsf{S}}(|\mathsf{X}|)$ (see Definition 5.2.2). In particular, if we plug confidence $\delta/\pi_{\mathsf{S}}(|\mathsf{X}|)$ to the above analysis, and use the union bound, we get that for

$$r \geq \left\lceil \frac{4}{\varepsilon^2}\ln\frac{\pi_{\mathsf{S}}(|\mathsf{X}|)}{\delta}\right\rceil$$

the sample estimates correctly (up to $\pm\varepsilon$) the size of all ranges with confidence $\geq 1-\delta$. Bounding $\pi_{\mathsf{S}}(|\mathsf{X}|)$ by $O(|\mathsf{X}|^d)$ (using Eq. (5.1) for a space with VC-dimension d), we can bound the required size of r by $O\Big(d\varepsilon^{-2}\log(|\mathsf{X}|/\delta)\Big)$.

Namely, the "naive" argumentation gives us a sample bound which depends on the underlying size of the ground set. However, the sample size in the ε-sample Theorem (Theorem 5.3.2) is independent of the size of the ground set. This is the magical property of the ε-sample Theorem[3].

5.3.3 A quicky proof of Theorem 5.3.4

Here we provide a sketchy proof of Theorem 5.3.4, which conveys the main ideas. The full proof in all its glory and details is provided in Section 5.5.

Let $N = (x_1, \ldots, x_m)$ be the sample obtained by m independent samples from A. Let E_1 be the probability that N fails to be an ε-net. Namely,

$$E_1 = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; |\mathbf{r} \cap A| \geq \varepsilon n \quad \text{and} \quad \mathbf{r} \cap N = \emptyset \right\}.$$

To complete the proof, we must show that $\mathbf{Pr}[E_1] \leq \delta$. Let $T = (y_1, \ldots, y_m)$ be another random sample generated in a similar fashion to N. Let E_2 be the event that N fails, but T "works", formally

$$E_2 = \left\{ \exists \mathbf{r} \in R \;\middle|\; |\mathbf{r} \cap A| \geq \varepsilon n,\ \mathbf{r} \cap N = \emptyset \text{ and } |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\}.$$

Intuitively, since $E_T[|\mathbf{r} \cap T|] \geq \varepsilon m$, then for the range $\mathbf{r}$ that N fails for, we have with "good" probability that $|\mathbf{r} \cap T| \geq \varepsilon n/2$. Namely, $\mathbf{Pr}[E_1] \approx \mathbf{Pr}[E_2]$.

Next, let

$$E_2' = \left\{ \exists \mathbf{r} \in R \;\middle|\; \mathbf{r} \cap N = \emptyset, |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\},$$

Clearly, $E_2 \subseteq E_2'$ and as such $\mathbf{Pr}[E_2] \leq \mathbf{Pr}\left[E_2'\right]$. Now, fix $Z = N \cup T$, and observe that $|Z| = 2m$. Next, fix a range $\mathbf{r}$, and observe that the bad probability of E_2' is maximized if $|\mathbf{r} \cap Z| = \varepsilon m/2$. Now, the probability that all the elements of $\mathbf{r} \cap Z$ falls only into the second half the sample is at most $2^{-\varepsilon m/2}$ as a careful calculation shows. Now, there are at most $\left|Z_{|R}\right| \leq \mathcal{G}_d(2m)$ different ranges that one has to consider. As such, $\mathbf{Pr}[E_1] \approx \mathbf{Pr}[E_2] \leq \mathbf{Pr}\left[E_2'\right] \leq \mathcal{G}_d(2m)2^{-\varepsilon m/2}$ and this is smaller than δ, as a careful calculation shows, by just plugging the value of m into the right hand side. ■

5.4 Discrepancy

The proof of the ε-sample/net Theorem is somewhat complicated. It turns out that one can get a somewhat similar result by attacking the problem from the other direction; namely, let us assume that we would like to take a truly large sample of a finite range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ defined over n elements with m ranges. We would like this sample to be as representative as possible as far as S is concerned. In fact, let us decide that we would like to pick exactly half of the points of X to our sample (assume that $n = |\mathsf{X}|$ is even).

To this end, let us color half of the points of X by -1 (i.e., black) and the other half by 1 (i.e., white). If for every range, $\mathbf{r} \in \mathcal{R}$, the number of black points inside it is equal to the number of white points, then doubling the number of black points inside a range, gives us the exact number of points inside the range. Of course, such a perfect coloring is unachievable in almost all situations. To see this, consider the clique graph K_4 – clearly, in any coloring of the vertices, there must be an edge with two endpoints having the same color.

Formally, let $\chi : \mathsf{X} \to \{-1, 1\}$ be the given coloring. The ***discrepancy*** of χ over a range $\mathbf{r}$ is the amount of imbalance in the coloring inside χ. Namely,

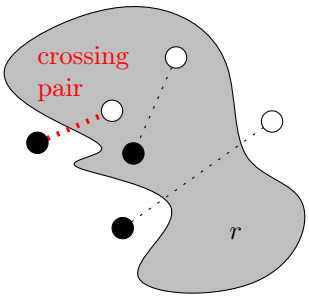
$$|\chi(\mathbf{r})| = \left| \sum_{\mathsf{p} \in \mathbf{r}} \chi(\mathsf{p}) \right|.$$

The overall ***discrepancy*** of χ is $\operatorname{disc}(\chi) = \max_{\mathbf{r} \in \mathcal{R}} |\chi(\mathbf{r})|$. The ***discrepancy*** of a (finite) range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ is the discrepancy of the best possible coloring; namely,

$$\operatorname{disc}(\mathsf{S}) = \min_{\chi : \mathsf{X} \to \{-1, +1\}} \operatorname{disc}(\chi).$$

The natural question is, of course, how to compute the coloring χ of minimum discrepancy. This seems like a very challenging question, but when you do not know what to do, you might as well do something random. So, let us pick a random coloring χ of X. To this end, let P be an arbitrary partition of X into pairs (i.e., a perfect matching). For a pair $\{\mathsf{p}, \mathsf{q}\} \in P$, we will either color $\chi(\mathsf{p}) = -1$ and $\chi(\mathsf{q}) = 1$, or the other way around; namely, $\chi(\mathsf{p}) = 1$ and $\chi(\mathsf{q}) = -1$. We will decide how to color this pair using a single coin flip. Thus, our coloring would be induced by making such a decision for every pair of P, and let χ be the resulting coloring. We will refer to χ as ***compatible*** with the partition P if $\chi(\{\mathsf{p}, \mathsf{q}\}) = 0$, for all $\{\mathsf{p}, \mathsf{q}\} \in P$.

Consider a range $\mathbf{r}$. If a pair $\{\mathsf{p},\mathsf{q}\} \in P$ falls completely inside $\mathbf{r}$ or completely outside $\mathbf{r}$ than it does not contribute anything to the discrepancy of $\mathbf{r}$. Thus, the only pairs that contribute to the discrepancy of $\mathbf{r}$ are the ones that ***cross*** it. Namely, $\{\mathsf{p},\mathsf{q}\} \cap \mathbf{r} \neq \emptyset$ and $\{\mathsf{p},\mathsf{q}\} \cap (\mathsf{X} \setminus \mathbf{r}) \neq \emptyset$. In particular, let $\#_{\mathbf{r}}$ denote the number of crossing pairs for $\mathbf{r}$, and let $X_i \in \{-1,+1\}$ be the indicator variable which is the contribution of the ith crossing pair to the discrepancy of $\mathbf{r}$. For $\Delta_{\mathbf{r}} = \sqrt{2\#_{\mathbf{r}} \ln(4m)}$, we have by the Chernoff inequality (Theorem 25.2.1), that



$$\mathbf{Pr}[|\chi(\mathbf{r})| \geq \Delta_{\mathbf{r}}] \leq 2\,\mathbf{Pr}\left[\sum_i X_i \geq \Delta_{\mathbf{r}}\right] \leq 2\exp\left(-\frac{\Delta_{\mathbf{r}}^2}{2\#_{\mathbf{r}}}\right) = \frac{1}{2m}.$$

Since there are m ranges in $\mathcal{R}$, it follows that with good probability (i.e., at least half) for all $\mathbf{r} \in \mathcal{R}$ the discrepancy of $\mathbf{r}$ is at most $\Delta_{\mathbf{r}}$.

Theorem 5.4.1 *Let $\mathsf{S} = (\mathsf{X},\mathcal{R})$ be a range space defined over $n = |\mathsf{X}|$ elements with $m = |\mathcal{R}|$ ranges. Consider any partition P of the elements of X into pairs. Then, with probability $\geq 1/2$, for any range $\mathbf{r} \in \mathcal{R}$, a random coloring $\chi : \mathsf{X} \to \{-1,+1\}$ that is compatible with the partition P, has discrepancy at most*

$$|\chi(\mathbf{r})| \leq \Delta_{\mathbf{r}} = \sqrt{2\#_{\mathbf{r}} \ln(4m)},$$

where $\#_{\mathbf{r}}$ denote the number of pairs of P that crosses $\mathbf{r}$. In particular, since $\#_{\mathbf{r}} \leq |\mathbf{r}|$, we have $|\chi(\mathbf{r})| \leq \sqrt{2\,|\mathbf{r}| \ln(4m)}$.

Observe that for every range $\mathbf{r}$ it holds $\#_{\mathbf{r}} \leq n/2$, since $2\#_{\mathbf{r}} \leq |\mathsf{X}|$. As such, we have:

Corollary 5.4.2 *Let $\mathsf{S} = (\mathsf{X},\mathcal{R})$ be a range space defined over n elements with m ranges. Let P be an arbitrary partition of X into pairs. Then a random coloring which is compatible with P has $\operatorname{disc}(\chi) \leq \sqrt{n \ln(4m)}$, with probability $\geq 1/2$.*

One can easily amplify the probability of success of the coloring, by increasing the threshold. In particular, for any constant $c \geq 1$, one has that

$$\forall \mathbf{r} \in \mathcal{R} \quad |\chi(\mathbf{r})| \leq \sqrt{2c\,\#_{\mathbf{r}} \ln(4m)},$$

with probability $\geq 1 - \dfrac{2}{(4m)^c}$.

5.4.1 Building ε-sample via Discrepancy

Let $\mathsf{S} = (\mathsf{X},\mathcal{R})$ be a range space with shattering dimension d. Let $\mathsf{P} \subseteq \mathsf{X}$ be a set of n points, and consider the induced range space $\mathsf{S}_{|\mathsf{P}} = (\mathsf{P},\mathcal{R}_{|\mathsf{P}})$, see Definition 5.1.1. Without loss of generality, we assume that n is a power of 2. Consider a coloring χ of P with discrepancy bounded by Corollary 5.4.2. In particular, let Q be the points of P colored by, say, -1. We know that $|\mathsf{Q}| = n/2$, and for any range $\mathbf{r} \in \mathcal{R}$, we have that

$$\chi(\mathbf{r}) = \Big|\,|(\mathsf{P} \setminus \mathsf{Q}) \cap \mathbf{r}| - |\mathsf{Q} \cap \mathbf{r}|\,\Big| \leq c\sqrt{n \ln(n^d)},$$

for some absolute constant c. Observe that $|(\mathsf{P} \setminus \mathsf{Q}) \cap \mathbf{r}| = |\mathsf{P} \cap \mathbf{r}| - |\mathsf{Q} \cap \mathbf{r}|$. In particular, we have that for any range $\mathbf{r}$, it holds

$$\Big|\,|\mathsf{P} \cap \mathbf{r}| - 2\,|\mathsf{Q} \cap \mathbf{r}|\,\Big| \leq c\sqrt{n \ln(n^d)}. \tag{5.3}$$

Dividing both sides by $n = |\mathsf{X}| = 2\,|\mathsf{Q}|$, we have that

$$\left|\frac{|\mathsf{P} \cap \mathbf{r}|}{|\mathsf{P}|} - \frac{|\mathsf{Q} \cap \mathbf{r}|}{|\mathsf{Q}|}\right| \leq \tau(n) = c\sqrt{\frac{d \ln n}{n}}.$$

Namely, a coloring with low discrepancy yields a $\tau(n)$-sample. Intuitively, if n is very large, then Q provides a good approximation to P. However, we are happy with a ε-sample for a prespecified $\varepsilon > 0$. To this end, we can "chain" together several approximations.

Lemma 5.4.3 *Let $\mathsf{Q} \subseteq \mathsf{P}$ be a δ-sample for P (in some underlying range space S), and let $\mathsf{R} \subseteq \mathsf{Q}$ be a ρ-sample for Q. Then R is a $(\delta+\rho)$-sample for P.*

Proof: By definition, we have that, for every $\mathbf{r} \in \mathcal{R}$, it holds

$$\left|\frac{|\mathbf{r} \cap \mathsf{P}|}{|\mathsf{P}|} - \frac{|\mathbf{r} \cap \mathsf{Q}|}{|\mathsf{Q}|}\right| \leq \delta \quad \text{and} \quad \left|\frac{|\mathbf{r} \cap \mathsf{Q}|}{|\mathsf{Q}|} - \frac{|\mathbf{r} \cap \mathsf{R}|}{|\mathsf{R}|}\right| \leq \rho.$$

By adding the two inequalities together, we get $\left|\frac{|\mathbf{r}\cap\mathsf{P}|}{|\mathsf{P}|} - \frac{|\mathbf{r}\cap\mathsf{R}|}{|\mathsf{R}|}\right| \leq \delta + \rho$. ■

Thus, let $\mathsf{P}_0 = \mathsf{P}$ and $\mathsf{P}_1 = \mathsf{Q}$. Now, in the ith iteration, we will compute a coloring χ_{i-1} of P_{i-1} with low discrepancy, as guaranteed by Corollary 5.4.2, and let P_i be the points of P_{i-1} colored white by χ_{i-1}. Let $\delta_i = \tau(n_{i-1})$, where $n_{i-1} = |\mathsf{P}_{i-1}| = n/2^{i-1}$.

By Lemma 5.4.3, we have that P_k is a $(\sum_{i=1}^{k} \delta_i)$-sample for P. Since we would like the smallest set in the sequence $\mathsf{P}_1, \mathsf{P}_2, \ldots$ that is still an ε-sample, we would like to find the maximal k, such that $(\sum_{i=1}^{k} \delta_i) \leq \varepsilon$. We thus require that

$$\sum_{i=1}^{k} \delta_i = \sum_{i=1}^{k} \tau(n_{i-1}) = \sum_{i=1}^{k} c \sqrt{\frac{d \ln(n/2^{i-1})}{n/2^{i-1}}} \leq c_1 \sqrt{\frac{d \ln(n/2^{k-1})}{n/2^{k-1}}} = c_1 \sqrt{\frac{d \ln n_{k-1}}{n_{k-1}}} \leq \varepsilon,$$

where c_1 is a sufficiently large constant. This holds for $n_{k-1} \geq (4c_1^2 d/\varepsilon^2) \ln(c_1 d/\varepsilon)$ as can be verified by an easy calculation. In particular, taking the largest k for which this holds, results in a set P_k of size $O((d/\varepsilon^2) \ln(d/\varepsilon))$ which is an ε-sample for P.

Theorem 5.4.4 (ε-sample via discrepancy.) *There is a positive constant c such that if $(\mathsf{X}, \mathcal{R})$ is any range space with shattering dimension at most d, $B \subseteq \mathsf{X}$ is a finite subset and $\varepsilon, \delta > 0$, then there exists a subset $C \subseteq B$, of cardinality $O\Big((d/\varepsilon^2) \ln(d/\varepsilon)\Big)$, such that C is an ε-sample for B.*

5.4.2 Building ε-net via Discrepancy

We need to be slightly more careful if we want to use discrepancy to build ε-nets, and we will use Theorem 5.4.1 instead of Corollary 5.4.2 in the analysis. In particular, let $\mathbf{r}$ be a range in given space, and let

$$\nu_i = |\mathsf{P}_i \cap \mathbf{r}|$$

denote the size of the range $\mathbf{r}$ in the ith set P_i, for $i \geq 0$. We have, by Theorem 5.4.1, that

$$\Big| |\mathsf{P}_{i-1} \cap \mathbf{r}| - 2\,|\mathsf{P}_i \cap \mathbf{r}| \Big| \leq c \sqrt{d\,|\mathsf{P}_{i-1} \cap \mathbf{r}| \ln(n_{i-1})},$$

for some constant c, since the crossing number of a range $\mathbf{r} \cap \mathsf{P}_{i-1}$ is always bounded by its size. This is equivalent to

$$\left|2^{i-1}\nu_{i-1} - 2^i \nu_i\right| \leq c 2^{i-1} \sqrt{d\nu_{i-1} \ln n_{i-1}}. \tag{5.4}$$

We need the following technical claim that states the size of ν_k behaves as we expect: As long as the set P_k is large enough, the size of ν_k is roughly $\nu_0/2^k$.

Claim 5.4.5 *There is a constant c_4 (independent of d), such that for all k with $\nu_0/2^k \geq c_4 d \ln n_k$, it holds $(\nu_0/2^k)/2 \leq \nu_k \leq 2(\nu_0/2^k)$.*

Proof: The proof is by induction. For $k = 0$ the claim trivially holds. Assume that it holds for $i < k$. Adding up the inequalities of Eq. (5.4), for $i = 1, \ldots, k$, we have that

$$\left|\nu_0 - 2^k \nu_k\right| \leq \sum_{i=1}^{k} c 2^{i-1} \sqrt{d\nu_{i-1} \ln n_{i-1}} \leq \sum_{i=1}^{k} c 2^{i-1} \sqrt{2d \frac{\nu_0}{2^{i-1}} \ln n_{i-1}} \leq c_3\, 2^k \sqrt{d \frac{\nu_0}{2^k} \ln n_k},$$

for some constant c_3. Thus,

$$\frac{\nu_0}{2^k} - c_3 \sqrt{d \frac{\nu_0}{2^k} \ln n_k} \;\leq\; \nu_k \;\leq\; \frac{\nu_0}{2^k} + c_3 \sqrt{d \frac{\nu_0}{2^k} \ln n_k}.$$

By assumption, we have that $\dfrac{\nu_0}{c_4 2^k} \geq \sqrt{d \ln n_k}$. Selecting $c_4 \geq 2c_3^2$, implies that $\nu_k \leq \left(1 + \dfrac{c_3}{c_3}\right) \dfrac{\nu_0}{2^k} = 2\dfrac{\nu_0}{2^k}$. Similarly, we have

$$\nu_k \geq \frac{\nu_0}{2^k}\left(1 - \frac{c_3 \sqrt{d \ln n_k}}{\sqrt{\nu_0/2^k}}\right) \geq \frac{\nu_0}{2^k}/2,$$

since by assumption $\nu_0/2^k$ is sufficiently large. ■

So consider a "heavy" range $\mathbf{r}$ that contains at least $\nu_0 \geq \varepsilon n$ points of P. To apply Claim 5.4.5 we need a k such that $\varepsilon n/2^k \geq c_4 d \ln n_{k-1}$, or equivalently, that

$$\frac{n_k}{\ln(2n_k)} \geq \frac{c_4 d}{\varepsilon},$$

which holds for $n_k = \Omega\Big(\frac{d}{\varepsilon} \ln \frac{d}{\varepsilon}\Big)$. But then, Claim 5.4.5, we have that

$$\nu_k = |\mathsf{P}_k \cap \mathbf{r}| \geq \frac{|\mathsf{P} \cap \mathbf{r}|}{2 \cdot 2^k} \geq \frac{1}{2} \cdot \frac{\varepsilon n}{2^k} = \frac{\varepsilon}{2} n_k = \Omega\left(d \ln \frac{d}{\varepsilon}\right) > 0.$$

We conclude that the set P_k, which is of size $\Omega\Big(\frac{d}{\varepsilon} \ln \frac{d}{\varepsilon}\Big)$, is an ε-net.

Theorem 5.4.6 (ε-net via discrepancy.) *There is a positive constant c such that if $(\mathsf{X}, \mathcal{R})$ is any range space with shattering dimension at most d, $B \subseteq \mathsf{X}$ is a finite subset and $\varepsilon, \delta > 0$, then there exists a subset $C \subseteq B$, of cardinality $O((d/\varepsilon) \ln(d/\varepsilon))$, such that C is an ε-net for B.*

5.5 Proof of the ε-net Theorem

In this section, we finally prove Theorem 5.3.4.

Let $(\mathsf{X}, \mathcal{R})$ be a range space of VC-dimension d, and let A be a subset of X of cardinality n. Suppose that m satisfies Eq. $(5.2)_{\text{p68}}$. Let $N = (x_1, \ldots, x_m)$ be the sample obtained by m independent samples from A (the elements of N are not necessarily distinct, and this is why we treat N as an ordered set). Let E_1 be the probability that N fails to be an ε-net. Namely,

$$E_1 = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; |\mathbf{r} \cap A| \geq \varepsilon n \quad \text{and} \quad \mathbf{r} \cap N = \emptyset \right\}.$$

(Namely, there exists a "heavy" range $\mathbf{r}$ that does not contain any point of N.) To complete the proof, we must show that $\mathbf{Pr}[E_1] \leq \delta$. Let $T = (y_1, \ldots, y_m)$ be another random sample generated in a similar fashion to N. Let E_2 be the event that N fails, but T "works", formally

$$E_2 = \left\{ \exists \mathbf{r} \in R \;\middle|\; |\mathbf{r} \cap A| \geq \varepsilon n,\ \mathbf{r} \cap N = \emptyset \text{ and } |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\}.$$

Intuitively, since $E_T[|\mathbf{r} \cap T|] \geq \varepsilon m$, then for the range $\mathbf{r}$ that N fails for, we have with "good" probability that $|\mathbf{r} \cap T| \geq \varepsilon n/2$. Namely, E_1 and E_2 have more or less the same probability.

Claim 5.5.1 $\mathbf{Pr}[E_2] \leq \mathbf{Pr}[E_1] \leq 2\,\mathbf{Pr}[E_2]$.

Proof: Clearly, $E_2 \subseteq E_1$, and thus $\mathbf{Pr}[E_2] \leq \mathbf{Pr}[E_1]$. As for the other part, note that by the definition of conditional probability, we have

$$\mathbf{Pr}\left[E_2 \;\middle|\; E_1 \right] = \mathbf{Pr}[E_2 \cap E_1] \,/\, \mathbf{Pr}[E_1] = \mathbf{Pr}[E_2] \,/\, \mathbf{Pr}[E_1].$$

It is thus enough to show that $\mathbf{Pr}\left[E_2 \;\middle|\; E_1 \right] \geq 1/2$.

Assume that E_1 occur. There is $\mathbf{r} \in R$, such that $|r \cap A| > \varepsilon n$ and $\mathbf{r} \cap N = \emptyset$. The required probability is at least the probability that for this specific $\mathbf{r}$, we have $|\mathbf{r} \cap T| \geq \frac{\varepsilon n}{2}$. However, $|\mathbf{r} \cap T|$ is a binomial variable with expectation εm, and variance $\varepsilon(1-\varepsilon)m \leq \varepsilon m$. Thus, by Chebychev inequality (Theorem 25.1.2), it holds

$$\mathbf{Pr}\left[|\mathbf{r} \cap T| < \frac{\varepsilon m}{2} \right] \leq \mathbf{Pr}\left[\left| |\mathbf{r} \cap T| - \varepsilon m \right| > \frac{\varepsilon m}{2} \right] = \mathbf{Pr}\left[\left| |\mathbf{r} \cap T| - \varepsilon m \right| > \frac{\sqrt{\varepsilon m}}{2} \sqrt{\varepsilon m} \right] \leq \left(\frac{2}{\sqrt{\varepsilon m}} \right)^2 \leq \frac{1}{2},$$

by Eq. $(5.2)_{\text{p68}}$. Thus, for $\mathbf{r} \in E_1$, we have

$$\frac{\mathbf{Pr}[E_2]}{\mathbf{Pr}[E_1]} \geq \mathbf{Pr}\left[|\mathbf{r} \cap T| \geq \tfrac{\varepsilon m}{2} \right] = 1 - \mathbf{Pr}\left[|\mathbf{r} \cap T| < \frac{\varepsilon m}{2} \right] \geq \frac{1}{2}.$$

■

Thus, it is enough to bound the probability of E_2. Let

$$E_2' = \left\{ \exists \mathbf{r} \in R \;\middle|\; \mathbf{r} \cap N = \emptyset, |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\},$$

Clearly, $E_2 \subseteq E_2'$. Thus, bounding the probability of E_2' is enough to prove the theorem. Note however, that a shocking thing happened! We no longer have A as participating in our event. Namely, we turned bounding an event that depends on a global quantity, into bounding a quantity that depends only on local quantity/experiment. This is the crucial idea in this proof.

Claim 5.5.2 $\mathbf{Pr}[E_2] \leq \mathbf{Pr}\left[E_2'\right] \leq \mathcal{G}_d(2m) 2^{-\varepsilon m/2}$.

Proof: We imagine that we sample the elements of $N \cup T$ together, by picking $Z = (z_1, \ldots, z_{2m})$ independently from A. Next, we randomly decide the m elements of Z that go into N, and the remaining elements go into T. Clearly,

$$\mathbf{Pr}[E_2'] = \sum_Z \mathbf{Pr}\left[E_2' \;\middle|\; Z \right] \mathbf{Pr}[Z].$$

Thus, from this point on, we fix the set Z, and we bound $\mathbf{Pr}[E_2'|Z]$ (note, that $\mathbf{Pr}[E_2']$ can be interpreted as averaging $\mathbf{Pr}[E_2'|Z]$, thus a bound on this quantity would imply the same bound on $\mathbf{Pr}[E_2']$).

It is now enough to consider the ranges in the projection space $(Z, \mathcal{R}_{|Z})$. By Lemma 5.2.1, we have $\left|\mathcal{R}_{|Z}\right| \leq \mathcal{G}_d(2m)$.

Let us fix any $\mathbf{r} \in \mathcal{R}_{|Z}$, and consider the event

$$E_r = \left\{ \mathbf{r} \cap N = \emptyset \text{ and } |\mathbf{r} \cap T| > \frac{\varepsilon m}{2} \right\}.$$

For $k = |\mathbf{r} \cap (N \cup T)|$, we have

$$\begin{aligned}\mathbf{Pr}[E_r] &\leq \mathbf{Pr}\Big[\mathbf{r} \cap N = \emptyset \;\Big|\; |\mathbf{r} \cap (N \cup T)| > \frac{\varepsilon m}{2}\Big] = \frac{\binom{2m-k}{m}}{\binom{2m}{m}} \\ &= \frac{(2m-k)(2m-k-1)\cdots(m-k+1)}{2m(2m-1)\cdots(m+1)} \\ &= \frac{m(m-1)\cdots(m-k+1)}{2m(2m-1)\cdots(2m-k+1)} \leq 2^{-k} \leq 2^{-\varepsilon m/2}.\end{aligned}$$

Thus,

$$\mathbf{Pr}\Big[E_2' \;\Big|\; Z\Big] \leq \sum_{\mathbf{r} \in Z_{|R}} \mathbf{Pr}[E_r] \leq \big|Z_{|R}\big| 2^{-\varepsilon m/2} \leq \mathcal{G}_d(2m) 2^{-\varepsilon m/2},$$

implying that $\mathbf{Pr}\big[E_2'\big] \leq \mathcal{G}_d(2m)2^{-\varepsilon m/2}$. ■

Proof of Theorem 5.3.4. By Lemma 5.5.1 and Lemma 5.5.2, we have $\mathbf{Pr}[E_1] \leq 2\mathcal{G}_d(2m)2^{-\varepsilon m/2}$. It is thus remains to verify that if m satisfies Eq. (5.2), then $2\mathcal{G}_d(2m)2^{-\varepsilon m/2} \leq \delta$.

Indeed, we know that $2m \geq 8d$ and as such $\mathcal{G}_d(2m) = \sum_{i=0}^{d} \binom{2m}{i} \leq \sum_{i=0}^{d} \frac{(2m)^i}{i!} \leq (2m)^d$, for $d > 1$. Thus, it is sufficient to show that the inequality $2(2m)^d 2^{-\varepsilon m/2} \leq \delta$ holds. By taking lg of both sides and rearranging, we have that this is equivalent to

$$\frac{\varepsilon m}{2} \geq d \lg(2m) + \lg \frac{2}{\delta}.$$

By our choice of m (see Eq. (5.2)), we have that $\varepsilon m/4 \geq \lg(2/\delta)$. Thus, we need to show that

$$\frac{\varepsilon m}{4} \geq d \lg(2m).$$

We verify this inequality for $m = \frac{8d}{\varepsilon} \lg \frac{8d}{\varepsilon}$. Indeed

$$2d \lg \frac{8d}{\varepsilon} \geq d \lg\left(\frac{16d}{\varepsilon} \lg \frac{8d}{\varepsilon}\right).$$

This is equivalent to $\left(\frac{8d}{\varepsilon}\right)^2 \geq \frac{16d}{\varepsilon} \lg \frac{8d}{\varepsilon}$. Which is equivalent to $\frac{4d}{\varepsilon} \geq \lg \frac{8d}{\varepsilon}$, which is certainly true for $0 \leq \varepsilon \leq 1$ and $d > 1$.

This completes the proof of the theorem. ■

5.6 Bibliographical notes

The exposition of the ε-net and ε-sample theorems is based on [AS00]. The proof of the ε-net theorem is due Haussler and Welzl [HW87]. The proof of the ε-sample theorem is due to Vapnik and Chervonenkis [VC71]. The bound in Theorem 5.3.2 can be improved to $O\big(\frac{d}{\varepsilon^2} + \frac{1}{\varepsilon^2} \log \frac{1}{\delta}\big)$ [AB99].

The beautiful alternative proof of both theorems via the usage of discrepancy is due to Chazelle and Matoušek [CM96]. The discrepancy method is a beautiful topic which is quite deep mathematically, and we had just skimmed the thin layer of melting water on top of the tip of the iceberg[④]. Two nice books on the topic are the books by Chazelle [Cha01] and Matoušek [Mat99]. The book [Cha01] is currently available online for free from Chazelle webpage.

We will revisit discrepancy since in some geometric cases it yields better results than the ε-sample theorem. In particular, the random coloring of Theorem 5.4.1 can be derandomized using conditional probabilities. One can then use it to get ε-sample/net by applying it repeatedly. A faster algorithm results from a careful implementation of the sketch-and-merge approach that would be described when discussing streaming. The disappointing feature of all the deterministic constructions of ε-samples/nets is that their running time is exponential in the dimension d, since the number of ranges is usually exponential in d.

A similar result to the one derived by Haussler and Welzl [HW87], using a more geometric approach, was done independently by Clarkson in the same time [Cla87]. Exposing the fact that VC dimension is not necessary if we are interested only in geometric applications. This was later refined by Clarkson [Cla88] leading to a general technique that, in geometric settings, yields stronger results than the ε-net theorem. (The journal version of this paper is [CS89] – an unfortunate merger of two important papers.) This technique has numerous applications in discrete and computational geometry and leads to several "proofs from the book" for several results in discrete geometry.

Exercise 5.7.2 is from Anthony and Bartlett [AB99].

[④] The iceberg is melting because of global warming, so sorry, climate change.

5.6.1 Variants and extensions

A natural application of the ε-sample theorem is to use it to estimate the weights of ranges. In particular, given a finite range space $(\mathsf{X}, \mathcal{R})$ we would like to build a data-structure such that we can decide quickly, given a query range $\mathbf{r}$, what is the number of points of X inside $\mathbf{r}$. We could always use a sample of size (roughly) $O(\varepsilon^{-2})$ to get an estimate of the weight of a range, using the ε-sample theorem. The error of the estimate is εn, where $n = |\mathsf{X}|$; namely, the error is additive. The natural question is whether one can get an additive estimate ρ, such that $pn \le \rho \le (1+\varepsilon)pn$, where $|\mathbf{r} \cap \mathsf{X}| = pn$.

In particular, a subset $A \subset \mathsf{X}$ is a (relative) (ε, p)-***sample***, if for each $\mathbf{r} \in \mathcal{R}$ of weight exceeding pn, it holds

$$\left| \frac{|\mathbf{r} \cap A|}{|A|} - \frac{|\mathbf{r} \cap \mathsf{X}|}{|\mathsf{X}|} \right| \le \varepsilon \frac{|\mathbf{r} \cap \mathsf{X}|}{|\mathsf{X}|}.$$

Of course, one can simply generate a εp-sample of size (roughly) $O(1/(\varepsilon p)^2)$ by the ε-sample theorem. This is not very interesting when $p = 1/\sqrt{n}$. Interestingly, the dependency on p can be improved.

Theorem 5.6.1 ([LLS01]) *Let $(\mathsf{X}, \mathcal{R})$ be a range space with shattering dimension d, where $|\mathsf{X}| = n$, and let $0 < \varepsilon < 1$ and $0 < p < 1$ be given parameters. Then, consider a random sample $A \subseteq \mathsf{X}$ of size $\frac{c}{\varepsilon^2 p}\left(d \log \frac{1}{p} + \log \frac{1}{\delta}\right)$, where c is a constant. Then, it holds that for each range $\mathbf{r} \in \mathcal{R}$ of at least pn points, we have*

$$\left| \frac{|\mathbf{r} \cap A|}{|A|} - \frac{|\mathbf{r} \cap \mathsf{X}|}{|\mathsf{X}|} \right| \le \varepsilon \frac{|\mathbf{r} \cap \mathsf{X}|}{|\mathsf{X}|}.$$

In other words, A is a (p, ε)-sample for $(\mathsf{X}, \mathcal{R})$. The probability of success is $\ge 1 - \delta$.

A similar result is achievable by using discrepancy, see Exercise 5.7.3.

5.7 Exercises

Exercise 5.7.1 (On the VC-dimension of the dual range space.) [5 Points]

Prove Lemma 5.2.4. Namely, given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ of VC-dimension d, prove that the dual range space $\mathsf{S}^\star = (\mathcal{R}, \mathsf{X}^\star)$ has VC-dimension bounded by 2^d.

[**Hint:** Represent a finite range space as a matrix, where the elements are the columns, and the sets are the rows. Interpret the fact that a set of size d can be shattered in this setting. What is the dual range space in this settings?]

Exercise 5.7.2 (Flip and Flop) [20 Points]

(a) **[5 Points]** Let $b_1, \ldots, b_{2m}$ be m binary bits. Let Ψ be the set of all permutations of $1, \ldots, 2m$, such that for any $\sigma \in \Psi$, we have $\sigma(i) = i$ or $\sigma(i) = m + i$, for $1 \le i \le m$, and similarly, $\sigma(m+i) = i$ or $\sigma(m+i) = m+i$. Namely, $\sigma \in \Psi$ either leave the pair $i, i+m$ in their positions, or it exchange them, for $1 \le i \le m$. As such $|\Psi| = 2^m$.

Prove that for a random $\sigma \in \Psi$, we have

$$\mathbf{Pr}\left[\left| \frac{\sum_{i=1}^m b_{\sigma(i)}}{m} - \frac{\sum_{i=1}^m b_{\sigma(i+m)}}{m} \right| \ge \varepsilon\right] \le 2e^{-\varepsilon^2 m/2}.$$

(b) **[5 Points]** Let Ψ' be the set of all permutations of $1, \ldots, 2m$. Prove that for a random $\sigma \in \Psi'$, we have

$$\mathbf{Pr}\left[\left| \frac{\sum_{i=1}^m b_{\sigma(i)}}{m} - \frac{\sum_{i=1}^m b_{\sigma(i+m)}}{m} \right| \ge \varepsilon\right] \le 2e^{-C\varepsilon^2 m/2},$$

where C is an appropriate constant. [**Hint:** Use (a), but be careful.]

(c) **[10 Points]** Prove Theorem 5.3.2 using (b).

Exercise 5.7.3 Prove the following theorem using discrepancy.

> **Theorem 5.7.4** *Let $(\mathsf{X}, \mathcal{R})$ be a range space with shattering dimension d, where $|\mathsf{X}| = n$, and let $0 < \varepsilon < 1$ and $0 < p < 1$ be given parameters. Then one can construct a set $A \subseteq \mathsf{X}$ of size $O(\frac{d}{\varepsilon^2 p} \ln \frac{d}{\varepsilon p})$, such that, for each range $\mathbf{r} \in \mathcal{R}$ of at least pn points, we have*
>
> $$\left| \frac{|\mathbf{r} \cap A|}{|A|} - \frac{|\mathbf{r} \cap \mathsf{X}|}{|\mathsf{X}|} \right| \le \varepsilon \frac{|\mathbf{r} \cap \mathsf{X}|}{|\mathsf{X}|}.$$
>
> *In other words, A is a relative (p, ε)-sample for $(X, \mathcal{R})$.*

.

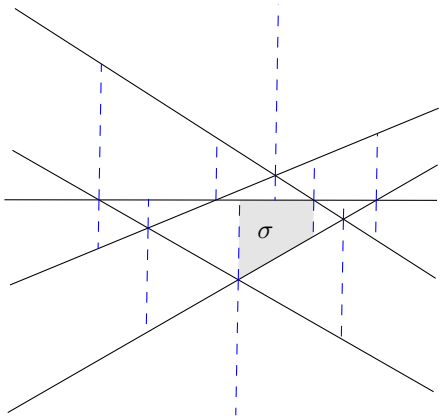
Chapter 6

Sampling and the Moments Technique

> "I've never touched the hard stuff, only smoked grass a few times with the boys to be polite, and that's all, though ten is the age when the big guys come around teaching you all sorts to things. But happiness doesn't mean much to me, I still think life is better. Happiness is a mean son of a bitch and needs to be put in his place. Him and me aren't on the same team, and I'm cutting him dead. I've never gone in for politics, because somebody always stand to gain by it, but happiness is an even crummier racket, and their ought to be laws to put it out of business."
>
> – – Momo, Emile Ajar

6.1 Vertical Decomposition

Given a set S of n segments in the plane, and a subset $\mathsf{R} \subseteq \mathsf{S}$, let $\mathcal{A}^{|}(\mathsf{R})$ denote the ***vertical decomposition*** of the plane, formed by the arrangement $\mathcal{A}(\mathsf{R})$ of the segments of R. This is the partition of the plane into interior disjoint vertical trapezoids formed by erecting vertical walls through each vertex of $\mathcal{A}^{|}(\mathsf{R})$. Formally, a ***vertex*** of $\mathcal{A}^{|}(\mathsf{R})$ is either an endpoint of a segment of R, or an intersection point of two of its segments. From each such vertex we shoot up (similarly, down) a vertical ray till it hits a segment of R, or it continues all the way to infinity. See figure on the right.



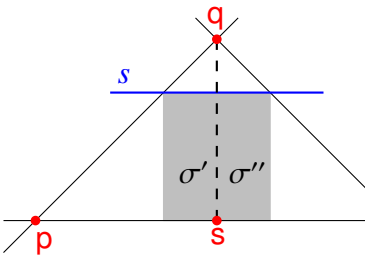
Note, that a vertical trapezoid is defined by at most 4 segments: two segments defining its ceiling and floor, and two segments defining the two intersection points that induce the two vertical walls on its boundary. Of course, a vertical trapezoid might be degenerate, and thus defined by less segments (i.e., an unbounded vertical trapezoid, or a triangle).

Vertical decomposition breaks the faces of the arrangement, that might be arbitrarily complicated, into entities (i.e., vertical trapezoids) of constant complexity. This make handling arrangements much easier computationally.

In the following, we assume that the segments of S have k intersection points overall, and we want to compute the arrangement $\mathcal{A} = \mathcal{A}(\mathsf{S})$; namely, compute the edges, vertices and faces of $\mathcal{A}(\mathsf{S})$. One possible way of doing it, is the following: Compute a random permutation of the segments of S: $\mathsf{S} = \langle s_1, \ldots, s_n \rangle$. Let $\mathsf{S}_i = \langle s_1, \ldots, s_i \rangle$ be the prefix of length i of S. Compute $\mathcal{A}^{|}(\mathsf{S}_i)$ from $\mathcal{A}^{|}(\mathsf{S}_{i-1})$, for $i = 1, \ldots, n$. Clearly, $\mathcal{A}^{|}(\mathsf{S}) = \mathcal{A}^{|}(\mathsf{S}_n)$, and we can extract $\mathcal{A}(\mathsf{S})$ from it.

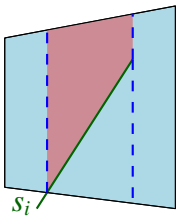
Randomized Incremental Construction (RIC). Imagine that we had computed the arrangement $\mathcal{B}_{i-1} = \mathcal{A}^{|}(\mathsf{S}_{i-1})$. In the ith iteration we compute $\mathcal{B}_i$ by inserting s_i into the arrangement $\mathcal{B}_{i-1}$. This involves splitting some trapezoids (and merging some others),

As a concrete example, consider the example on the right. Here we insert s in the arrangement. To this end we split the "vertical trapezoids" $\triangle$pqs and $\triangle$rqs, each into three trapezoids. The two trapezoids σ' and σ'' needs to be now merged together to form the new trapezoid which appears in the vertical decomposition of the new arrangement. (Note, that the figure does not show all the trapezoids in the vertical decomposition.)



To facilitate this, we need to compute the trapezoids of $\mathcal{B}_{i-1}$ that intersects s_i. This is done by maintaining a ***conflict-graph***. Each trapezoid $\sigma \in \mathcal{A}^{\vert}(\mathsf{S}_{i-1})$ maintains a ***conflict list*** $\mathrm{cl}(\sigma)$ of the segments of S that intersects its interior. We also maintain a similar structure for each segment, listing all the trapezoids of $\mathcal{A}^{\vert}(\mathsf{S}_{i-1})$ that it currently intersects (in its interior). We maintain those lists with cross-pointers, so that given an entry (σ, s) in the conflict-list of σ, we can find the entry (s, σ) in the conflict list of s in constant time.

Thus, given s_i, we know what are the trapezoids that needs to be split (i.e., all the trapezoids in $\mathrm{cl}(s_i)$). Splitting a trapezoid σ by a segment s_i is the operation of computing a set of (at most) 4 trapezoids that covers σ and have s_i on their boundary. We compute those new trapezoids, and next we need to compute the conflict-lists of the new trapezoids. This can be easily done by taking the conflict-list of a trapezoid $\sigma \in \mathrm{cl}(s_i)$ and distributing its segments among the $O(1)$ new trapezoids that covers σ. Using careful implementation this requires a linear time in the size of the conflict-list of σ.



In the above description, we ignored the need to merge adjacent trapezoids if they have identical floor and ceiling - this can be done by a somewhat straightforward and tedious implementation of the vertical-decomposition data-structure, by providing pointers between adjacent vertical trapezoids, and maintaining the conflict-list sorted (or by using hashing) so that merge operations can be done quickly. This is somewhat tedious but it can be done in linear time in the input/output size involved as can be verified.

Claim 6.1.1 *The (amortized) running time of constructing $\mathcal{B}_i$ from $\mathcal{B}_{i-1}$ is proportional to the size of the conflict lists of the vertical trapezoids in $\mathcal{B}_i \setminus \mathcal{B}_{i-1}$ (and the number of such new trapezoids).*

Proof: We charge all the work involved in the ith iteration either to the conflict lists of the newly created trapezoids, or the deleted conflict lists. Clearly, the running time of the algorithm in the ith iteration is linear in the total size of these conflict lists. Observe, that every conflict get charged twice – when it is being created, and when it is being deleted. As such, the (amortized) running time in the ith iteration is proportional to the total length of the newly created conflict lists. ■

Thus, to bound the running time of the algorithm, it is enough to bound the expected size of the destroyed conflict-lists in ith iteration (and sum this bound on the n iterations carried out by the algorithm). Or alternatively, bound the expected size of the conflict-lists created in the ith iteration.

Lemma 6.1.2 *Let S be a set of n segments (in general position[5]) with k intersection points. Let S_i be the first i segments in a random permutation of S. The expected size of $\mathcal{B}_i = \mathcal{A}^{\vert}(\mathsf{S}_i)$, denoted by $\tau(i)$, (i.e., number of trapezoids in $\mathcal{B}_i$) is $O\big(i + k(i/n)^2\big)$.*

Proof:[2] Consider an intersection point $\mathsf{p} = s \cap s'$, where $s, s' \in \mathsf{S}$. The probability that p is present in $\mathcal{A}^{\vert}(\mathsf{S}_i)$ is equivalent to the probability that both s and s' are in S_i. This probability is

$$\alpha = \frac{\binom{n-2}{i-2}}{\binom{n}{i}} = \frac{(n-2)!}{(i-2)!\,(n-i)!} \cdot \frac{i!\,(n-i)!}{n!} = \frac{i(i-1)}{n(n-1)}.$$

For each intersection point p in $\mathcal{A}(\mathsf{S})$ define an indicator variable X_{p}, which is one if the two segments defining p are in the random sample S_i, and zero otherwise. We have that $\mathbf{E}\big[X_{\mathsf{p}}\big] = \alpha$, and as such, by linearity of expectation, the expected number of intersection points in the arrangement $\mathcal{A}(\mathsf{S}_i)$ is

$$\mathbf{E}\left[\sum_{\mathsf{p} \in V} X_{\mathsf{p}}\right] = \sum_{\mathsf{p} \in V} \mathbf{E}\big[X_{\mathsf{p}}\big] = \sum_{\mathsf{p} \in V} \alpha = k\alpha,$$

[5] In this case, no two intersection points are the same, no two intersection points (or vertices) have the same x-coordinate, no two segments lie on the same line, etc. Making geometric algorithm work correctly for all degenerate inputs is a huge pain that can usually be handled by tedious and careful handling. Thus, we will always assume general position of the input. In other words, in theory all geometric inputs are inherently good, while in practice they are all evil (as anybody that tried to implement geometric algorithms can testify). The reader is encouraged not to use this to draw any conclusions on the human condition.

[2] The proof is provided in excruciating detail to get the reader used to this kind of argumentation. I would apologize for this pain, but it is a minor trifle, not to be mentioned, when compared to the other crimes in this manuscript.

where V is the set of k intersection points of $\mathcal{A}(\mathsf{S})$. Thus, since every endpoint of a segment of S_i contributed its two endpoints to the arrangement $\mathcal{A}(\mathsf{S}_i)$, we have that the expected number of vertices in $\mathcal{A}(\mathsf{S}_i)$ is

$$2i + \frac{i(i-1)}{n(n-1)}k.$$

Now, the number of trapezoids in $\mathcal{A}^{|}(\mathsf{S}_i)$ is proportional to the number of vertices of $\mathcal{A}(\mathsf{S}_i)$, which implies the claim. ■

6.1.1 Backward Analysis

In the following, we would like to consider the total amount of work involved in the ith iteration of the algorithm. The idea to analyse these iteration is (conceptually) to run the algorithm for the first i iterations, and then run "backward" the last iteration.

So, imagine, that the overall size of the conflict-lists of the trapezoids of $\mathcal{B}_i$ is W_i, and total size of the conflict lists created only in the ith iteration is C_i.

We are interested in bounding the expected size of C_i, since this is (essentially) the amount of work done by the algorithm in this iteration. To this end, let $s = s_i$ and observe that the structure of $\mathcal{B}_i$ is defined independently of the permutation S_i, and depends only on the (unordered) set $\mathsf{S}_i = \{s_1, \ldots, s_i\}$. So, fix S_i. What is the probability that $s_i = s$? Clearly, this is $1/i$ – being the probability of s to be the last element in a permutation of i elements (i.e., we consider a random permutation of S_i).

Now, consider a trapezoid $\sigma \in \mathcal{B}_i$. If σ was created in the ith iteration, than clearly s_i must be one of the (at most four) segments that defines it. Since $\mathcal{B}_i$ is independent of the internal ordering of S_i, it follows that $\mathbf{Pr}[\sigma \in (\mathcal{B}_i \setminus \mathcal{B}_{i-1})] \leq 4/i$. In particular, the overall size of the conflict lists in the end of the ith iteration is

$$W_i = \sum_{\sigma \in \mathcal{B}_i} |\operatorname{cl}(\sigma)|.$$

As such, the expected overall size of the conflict-lists created in the ith iteration is

$$\mathbf{E}\Big[C_i \;\Big|\; \mathcal{B}_i\Big] \leq \sum_{\sigma \in \mathcal{B}_i} \frac{4}{i}\Big|\operatorname{cl}(\sigma)\Big| \leq \frac{4}{i} W_i.$$

By Lemma 6.1.2, the expected size of $\mathcal{B}_i$ is $O(i + ki^2/n^2)$. Let us guess (for the time being) that on average the size of the conflict list of a trapezoid of $\mathcal{B}_i$ is about $O(n/i)$. In particular, assume that we know that

$$\mathbf{E}[W_i] = O\left(\left(i + \frac{i^2}{n^2}k\right)\frac{n}{i}\right) = O\left(n + k\frac{i}{n}\right),$$

by Lemma 6.1.2. Implying

$$\mathbf{E}\Big[C_i\Big] = \mathbf{E}\Big[\mathbf{E}\Big[C_i \;\Big|\; \mathcal{B}_i\Big]\Big] = \mathbf{E}\left[\frac{4}{i}W_i\right] = \frac{4}{i}\mathbf{E}[W_i] = O\left(\frac{4}{i}\left(n + \frac{ki}{n}\right)\right) = O\left(\frac{n}{i} + \frac{k}{n}\right).$$

In particular, the expected amount of work in the ith iteration is proportional to $\mathbf{E}[C_i]$. Thus, the overall expected running time of the algorithm is

$$\mathbf{E}\left[\sum_{i=1}^{n} C_i\right] = \sum_{i=1}^{n} O\left(\frac{n}{i} + \frac{k}{n}\right) = O(n \log n + k).$$

Theorem 6.1.3 *Given a set S of n segments in the plane with k intersections, one can compute the vertical decomposition of $\mathcal{A}(\mathsf{S})$ in expected $O(n \log n + k)$ time.*

Intuition and discussion. What remains to be seen, is how we came up with the guess that the average size of a conflict-list of a trapezoid of $\mathcal{B}_i$ is about $O(n/i)$. Note, that ε-nets imply that the bound $O((n/i)\log i)$ holds with constant confidence (see Theorem 5.3.4), so this result is only slightly surprising. To prove this, we present in the next section a "strengthening" of ε-nets to geometric settings.

To get an intuition how we came up with this guess, consider a set P of n points on the line, and a random sample R of i points from P. Let $\widehat{\mathcal{I}}$ be the partition of the real line into open intervals by the endpoints of R that do not contain points of R in their interior.

Consider a interval of $\widehat{\mathcal{I}}$ (i.e., a one dimensional trapezoid) of $\widehat{\mathcal{I}}$. It is intuitively clear that this interval (in expectation) would contain $O(n/i)$ points. Indeed, fix a point x on the real line, and imagine that we pick each point with probability i/n to the random sample. The random variable which is the number of points of P we have to scan to the right of x till we "hit" a point that is in the random sample behaves like a geometric variable with probability i/n, and as such its expected value is n/i. The same argument works if we scan P to the left of x. We conclude that the number of points of P in the interval of $\widehat{\mathcal{I}}$ that contains x but does not contain any point of R is $O(n/i)$ in expectation.

Of course, the vertical decomposition case is more involved. Instead of proving the required result for this case, we will prove a more general result which can be applied in a lot of other settings.

6.2 General Settings

Let S be a set of objects. For a subset $\mathsf{R} \subseteq \mathsf{S}$, we define a collection of 'regions' called $\mathcal{F}(\mathsf{R})$. For vertical decomposition of segments (i.e., Theorem 6.1.3), the objects are segments, the regions are trapezoids, and $\mathcal{F}(\mathsf{R})$ is the set of vertical trapezoids forming $\mathcal{A}^{\downarrow}(\mathsf{R})$. Let

$$\mathcal{T} = \mathcal{T}(\mathsf{S}) = \bigcup_{\mathsf{R} \subseteq \mathsf{S}} \mathcal{F}(\mathsf{R})$$

denote the set of ***all possible regions*** defined by subsets of S.[3] We associate two subsets $D(\sigma), K(\sigma) \subseteq \mathsf{S}$ with each region $\sigma \in \mathcal{T}$.

The ***defining set*** $D(\sigma)$ of σ is a subset of S defining the region σ (the precise requirements from this set are specified in the axioms below). We assume that for every $\sigma \in \mathcal{T}$, $|D(\sigma)| \leq d$ for a (small) constant d. The constant d is sometime referred to as the ***combinatorial dimension***. In the case of Theorem 6.1.3, each trapezoid σ is defined by at most 4 segments (or lines) of S that define the region covered by the trapezoid σ, and this set of segments is $D(\sigma)$. See figure on the right.

The ***killing set*** $K(\sigma)$ of σ is the set of objects of S such that including any object of $K(\Delta)$ into R prevents σ from appearing in $\mathcal{F}(\mathsf{R})$ (i.e., the killing set is the conflict list of σ, if σ is being created by the RIC algorithm). In many applications $K(\sigma)$ is just the set of objects intersecting the cell σ; this is also the case in Theorem 6.1.3, where $K(\sigma)$ is the set of segments of S intersecting the interior of the trapezoid σ, see Figure 6.1. The ***weight*** of σ is $\omega(\sigma) = |K(\sigma)|$.

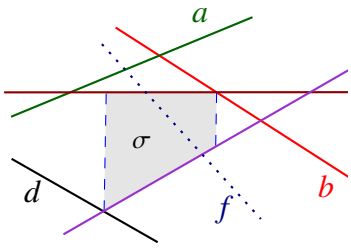


Figure 6.1: $D(\sigma) = \{b, c, d, e\}$ and $K(\sigma) = \{f\}$.

Let $\mathsf{S}, \mathcal{F}(\mathsf{R}), D(\sigma)$, and $K(\sigma)$ be such that for any subset $\mathsf{R} \subseteq \mathsf{S}$, the set $\mathcal{F}(\mathsf{R})$ satisfies the following axioms:

(i) For any $\sigma \in \mathcal{F}(\mathsf{R})$, it holds $D(\sigma) \subseteq \mathsf{R}$ and $\mathsf{R} \cap K(\sigma) = \emptyset$.

(ii) If $D(\sigma) \subseteq \mathsf{R}$ and $K(\sigma) \cap \mathsf{R} = \emptyset$, then $\sigma \in \mathcal{F}(\mathsf{R})$.

For any natural number r and a number $t > 0$, consider R to be a random sample of size r from S, and let's denote

$$\mathcal{F}_t(\mathsf{R}) = \left\{ \sigma \in \mathcal{F}(\mathsf{R}) \;\middle|\; \omega(\sigma) \geq t \cdot \frac{n}{r} \right\}.$$

This is the set of regions in $\mathcal{F}(\mathsf{R})$ with a weight that is t times larger than what we expect[4]. We intuitively expect the size of this set to drop fast as t increases. So, let

$$\#(r) = \mathbf{E}\Big[\big|\mathcal{F}(\mathsf{R})\big|\Big] \quad \text{and} \quad \#_t(r) = \mathbf{E}\Big[\big|\mathcal{F}_t(\mathsf{R})\big|\Big],$$

where the expectation is over random subsets $\mathsf{R} \subseteq \mathsf{S}$ of size r. Note, that $\#(r) = \#_0(r)$ is just the expected number of regions of a random sample of size r. In words, $\#_t(r)$ is the expected number of regions in a structure created by r random objects, such that these regions have weight which is t times larger than the "expected" weight of n/r.

Let

$$\mathcal{T}_t(r) = \bigcup_{\mathsf{R} \subseteq \mathsf{S}, |\mathsf{R}| = r} \mathcal{F}_t(\mathsf{R})$$

denote the set all t-heavy regions that might be created by a sample of size r.

In the following, S is a set of n objects complying with Axioms (i) and (ii), and

$$d = \max_{\sigma \in \mathcal{T}(\mathsf{S})} |D(\sigma)|$$

is the combinatorial dimension of the system induced by S.

Lemma 6.2.1 *Let $r \leq n$ and t be parameters, such that $1 \leq t \leq r/d$. Furthermore, let R be a sample of size r, R' be a sample of size $r' = \lfloor r/t \rfloor$, both from S. Let $\sigma \in \mathcal{T}$ be a trapezoid with weight $\omega = \omega(\sigma) \geq t(n/r)$. Then, $\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R})] = O\Big(t^d 2^{-t} \mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R}')]\Big)$.*

Intuitively, (but not quite correctly) Lemma 6.2.1 states that the probability of a t-heavy trapezoid to be created drops exponentially with t.

An Almost Proof of Lemma 6.2.1. We provide a back of the envelope argument "proving" Lemma 6.2.1. A more formal proof is provided in Section 6.5.

Let us pick R (resp., R') by picking each element of S with probability $p = r/n$ (resp. $p = r'/n$). Note, that this sampling is different than the one used by the lemma, but it provides samples having roughly the same size, and we expect the relevant

[3] Paraphrasing Voltaire, this does not imply that a member of $\mathcal{T}$ lives in the best of all possible sets.

[4] These are the regions that are t times overweight. Speak about an obesity problem.

probabilities to remain roughly the same. Let $\delta = |D(\sigma)|$ and $\omega = \omega(\sigma)$. We have that $\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R})] = p^{\delta}(1-p)^{\omega}$ (i.e., this is the probability that we pick the elements of $D(\sigma)$ to the sample, and do not pick any of the elements of $K(\sigma)$ to the sample, which is by Axiom (ii) exactly the event $\sigma \in \mathcal{F}(\mathsf{R})$). Similarly, $\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R}')] = p'^{\delta}(1-p')^{\omega}$. As such,

$$\alpha = \frac{\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R})]}{\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R}')]} = \frac{p^{\delta}(1-p)^{\omega}}{p'^{\delta}(1-p')^{\omega}} = \left(\frac{r/n}{r'/n}\right)^{\delta}\left(\frac{1-r/n}{1-r'/n}\right)^{\omega} \leq (t+1)^{d}\left(\frac{n-r}{n-r'}\right)^{\omega}.$$

Now,

$$\begin{aligned}\left(\frac{n-r}{n-r'}\right)^{\omega} &= \left(1+\frac{r'-r}{n-r'}\right)^{\omega} \leq \left(1+\frac{r/t-r}{n-r'}\right)^{\omega} \leq \left(1-\left(1-\frac{1}{t}\right)\frac{r}{n}\right)^{\omega} \leq \exp\left(-\left(1-\frac{1}{t}\right)\frac{r}{n}\omega\right)\\ &\leq \exp\left(-\left(1-\frac{1}{t}\right)\frac{r}{n}t\frac{n}{r}\right) \leq \exp(-(t-1)),\end{aligned}$$

as $\omega \geq t(n/r)$. As such, $\alpha \leq (t+1)^{d}\exp(-(t-1))$. ■

Since the formal proof is less enlightening than the above "almost proof", we delegate it to the end of the chapter, see Section 6.5. The following exponential decay lemma testifies that truly heavy regions are (exponentially) rare.

Lemma 6.2.2 *Given a set S of n objects. Let $r \leq n$ and t be parameters, such that $1 \leq t \leq r/d$, where $d = \max_{\sigma \in \mathcal{F}(\mathsf{S})} |D(\sigma)|$. Assuming that Axioms (i) and (ii) above hold for any subset of S, then we have*

$$\#_t(r) = O\left(t^{d}2^{-t}\,\#\left(\left\lfloor \frac{r}{t} \right\rfloor\right)\right) = O\left(t^{d}2^{-t}\,\#(r)\right). \tag{6.1}$$

Proof: Let R be a random sample of size r from S and R' be a random sample of size $r' = \lfloor r/t \rfloor$ from S. Let $\mathcal{T}_t = \mathcal{T}_t(r)$. We have

$$\begin{aligned}\#_t(r) &= \mathbf{E}\left[|\mathcal{F}_t(\mathsf{R})|\right] = \sum_{\sigma \in \mathcal{T}_t} \mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R})] = O\left(t^{d}2^{-t}\sum_{\sigma \in \mathcal{T}_t}\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R}')]\right)\\ &= O\left(t^{d}2^{-t}\sum_{\sigma \in \mathcal{T}}\mathbf{Pr}[\sigma \in \mathcal{F}(\mathsf{R}')]\right) = O\left(t^{d}2^{-t}\#(r')\right),\end{aligned}$$

by Lemma 6.2.1. ■

Theorem 6.2.3 *Let $\mathsf{R} \subseteq \mathsf{S}$ be a random subset of size r. Let $\#(r) = \mathbf{E}[|\mathcal{F}(\mathsf{R})|]$ and $c \geq 1$ be an arbitrary constant. Then,*

$$\mathbf{E}\left[\sum_{\sigma \in \mathcal{F}(\mathsf{R})} (\omega(\sigma))^{c}\right] = O\left(\#(r)\left(\frac{n}{r}\right)^{c}\right).$$

Proof: For a subset $\mathsf{R} \subseteq \mathsf{S}$ of size r, we have by Lemma 6.2.2, that

$$\begin{aligned}\mathbf{E}\left[\sum_{\sigma \in \mathcal{F}(\mathsf{R})} \omega(\sigma)^{c}\right] &= \mathbf{E}\left[\sum_{t \geq 1}\left(t\frac{n}{r}\right)^{c}\left(|\mathcal{F}_{t-2}(\mathsf{R})| - |\mathcal{F}_{t-1}(\mathsf{R})|\right)\right]\\ &\leq \left(\frac{n}{r}\right)^{c}\sum_{t \geq 0}(t+1)^{c}\cdot\mathbf{E}\left[|\mathcal{F}_t(\mathsf{R})|\right]\\ &= \left(\frac{n}{r}\right)^{c}\sum_{t \geq 0}(t+1)^{c}\,\#_t(r) = \left(\frac{n}{r}\right)^{c}\sum_{t \geq 0} O\left(t^{c+d}\,2^{-t}\,\#\left(\frac{r}{t}\right)\right)\\ &= O\left(\#(r)\left(\frac{n}{r}\right)^{c}\sum_{t \geq 0} t^{c+d}\,2^{-t}\right) = O\left(\#(r)\left(\frac{n}{r}\right)^{c}\right),\end{aligned}$$

since c and d are both constants. ■

6.3 Applications

6.3.1 Analyzing the RIC Algorithm for Vertical Decomposition

As shown in Lemma 6.1.2, $\#(i) = O\Big(i + k(i/n)^2\Big)$. Thus, by Theorem 6.2.3, we have that

$$\mathbf{E}\left[\sum_{\sigma\in\mathcal{B}_i}\omega(\sigma)\right] = O\Big(\tau(i)\frac{n}{i}\Big) = O(n + ki/n)\,.$$

This is the missing piece in the analysis of Section 6.1.1.

6.3.2 Cuttings

Let S be a set of n lines in the plane, and let r be an arbitrary parameter. A $(1/r)$-***cutting*** of S is a partition of the plane into constant complexity regions such that each region intersect at most n/r lines of S. It is natural to try and minimize the number of regions in the cutting, as cuttings are a natural tool for performing divide and conquer.

Consider the range space having S as its ground set, and vertical trapezoids as its ranges (i.e., given a vertical trapezoid σ, its corresponding range is the set of all lines of S that intersects the interior of σ). This range space has a VC dimension which is a constant as can be easily verified. Let $X \subseteq \mathsf{S}$ be a ε-net for this range space, for $\varepsilon = 1/r$. By Theorem 5.3.4 (ε-net theorem), there exists such an ε-net X, of this range space, of size $O(1/\varepsilon \log(1/\varepsilon)) = O(r \log r)$. Consider a vertical trapezoid σ in the arrangement $\mathcal{A}^{\downarrow}(X)$. It does not intersect any of the lines of X in its interior, and X is an ε-net for S. It follows, that σ intersects at most $\varepsilon n = n/r$ lines of S in its interior. Since the arrangement $\mathcal{A}^{\downarrow}(X)$ has complexity $O(|X|^2)$, we get the following result.

Lemma 6.3.1 *There exists $(1/r)$-cutting of a set of segments in the plane of size $O\Big((r \log r)^2\Big)$.*

Since an arrangement of n lines has at most $\binom{n}{2}$ intersects, and the number of intersections of the lines intersecting a single region in the cutting is at most $\binom{n/r}{2}$, this implies that any cutting must be of size $\Omega(r^2)$. We can get cuttings of such size easily using the moments technique.

Theorem 6.3.2 *Let S be a set of n lines in the plane, and let r be a parameter. One can compute a $(1/r)$-cutting of S of size $O(r^2)$.*

Proof: Let R be a random sample of size r, and consider its vertical decomposition $\mathcal{A}^{\downarrow}(\mathsf{R})$. If a vertical trapezoid $\sigma \in \mathcal{A}^{\downarrow}(\mathsf{R})$ intersects at most n/r lines of S, then we can add it to the output cutting. The other possibility is that a σ intersects $t(n/r)$ lines of S, for some $t > 1$, and let $\mathrm{cl}(\sigma) \subset \mathsf{S}$ be the conflict list of σ (i.e., the list of lines of S that intersect the interior of σ). Clearly, a $(1/t)$-cutting for the set $\mathrm{cl}(\sigma)$ forms a vertical decomposition (clipped inside σ) such that each trapezoid in this cutting intersects at most n/r lines of S. Thus, we compute such a cutting inside each such "heavy" trapezoid using the algorithm Lemma 6.3.1, and these subtrapezoids to the resulting cutting. Clearly, the size of the resulting cutting inside σ is $O\Big(t^2 \log^2 t\Big) = O\Big(t^4\Big)$. The resulting two-level partition is clearly the required cutting. By Theorem 6.2.3, the expected size of the cutting is

$$\begin{aligned} O\left(\#(r) + \mathbf{E}\left[\sum_{\sigma\in\mathcal{F}(\mathsf{R})}\left(2\frac{\omega(\sigma)}{n/r}\right)^4\right]\right) &= O\left(\#(r) + \left(\frac{r}{n}\right)^4 \mathbf{E}\left[\sum_{\sigma\in\mathcal{F}(\mathsf{R})}(\omega(\sigma))^4\right]\right) \\ &= O\left(\#(r) + \left(\frac{r}{n}\right)^4 \cdot \#(r)\left(\frac{n}{r}\right)^4\right) = O(\#(r)) = O\Big(r^2\Big)\,, \end{aligned}$$

since $\#(r)$ is proportional to the complexity of $\mathcal{A}(\mathsf{R})$ which is $O(r^2)$. ■

6.4 Bibliographical notes

The technique describe in this chapter is generally attributed to Clarkson-Shor [CS89], which is historically inaccurate as the technique was developed by Clarkson [Cla88]. Instead of mildly confusing the water by referring to it as the Clarkson technique, we decided to make sure to really confuse the reader, and refer to it as the ***moments technique***. The Clarkson technique [Cla88] is in fact more general and implies a connection between the number of "heavy" regions and "light" regions. The general framework can be traced back to the earlier paper [Cla87]. This implies several beautiful results, which we might cover later in the book. The interested reader is referred to the recent presentation by Sharir on this aspect [Sha03].

For the full details of the algorithm of Section 6.1, the interested reader is refereed to the following books [dBvKOS00, BY98]. Interestingly, in a some cases the merging stage can be skipped, see [Har00a].

Agarwal *et al.* [AMS94] presented a slightly stronger variant than the original version of Clarkson [Cla88], that allows a region to disappear even if none of the members of its killing set are in the random sample. This stronger settings are used in computing

the vertical decomposition of a single face in an arrangement. Here an insertion of a faraway segment the random sample might cut off a portion of the face of interest. In particular, in the settings of Agarwal *et al.* (ii) is replaced by

(ii') If $\sigma \in \mathcal{F}(\mathsf{R})$ and R' is a subset of R with $D(\sigma) \subseteq \mathsf{R}'$, then $\sigma \in \mathcal{F}(\mathsf{R}')$.

Interestingly, Clarkson [Cla88] did not prove Theorem 6.2.3 using the exponential decay lemma, but gave a direct proof. Although, his proof implicitly contains the exponential decay lemma. We chosen the current exposition since it is technically only slightly more challenging but provides a better intuition of what is really going on.

The exponential decay lemma (Lemma 6.2.2), was proved by Chazelle and Friedman [CF90]. The work of [AMS94] is a further extension of this result. Another analysis was provided by Clarkson *et al.* [CMS93].

Another way to reach similar results, is using the technique of Mulmuley [Mul94a], which relies on a direct analysis on 'stoppers' and 'triggers'. This technique is somewhat less convenient to use but is applicable to some settings where the moments technique does not apply directly. Mulmuley came up with randomized incremental construction of vertical decomposition. Also, his concept of the omega function might explain why randomized incremental algorithms perform better in practice than their worst case analysis [Mul94b].

Backwards analysis in geometric settings was first used by Chew [Che86], and formalized by Seidel [Sei93]. Its similar to the "leave one out" argument used in statistics for cross validation. The basic idea was probably known to the Greeks (or Russians or French) at some point in time.

(Naturally, our summary of the development is cursory at best and not necessarily accurate, and all possible disclaimers apply. A good summary is provided in the introduction of [Sei93].)

Sampling model. Our "almost" proof of Lemma 6.2.1 used a different sampling than the one used by the algorithm (i.e., sampling without replacement). Furthermore, Clarkson [Cla88] used random sampling with replacement. As a rule of thumb all these sampling approaches are similar and yield similar results, and its a good idea to use which ever sampling scheme is the easiest to analyse in figuring out whats going on. Of course, a formal proof requires analysing the algorithm in the sampling model its uses.

Lazy randomized incremental construction. If one wants to compute a single face that contains a marking point in an arrangement of curves, then the problem in using randomized incremental construction is that as you add curves, the region of interest shrinks, and regions that were maintained should be ignored. One option is to perform flooding in the vertical decomposition to figure out what trapezoids are still reachable from the marking point and maintaining only these trapezoids in the conflict graph. Doing it in each iteration is way too expensive, but luckily one can use a lazy strategy that performs this clean up only logarithmic number of times (i.e., you perform a clean up in an iteration if the iteration number is, say, a power of 2). This strategy complicates the analysis a bit, see [dBDS95] for more details on this ***lazy randomize incremental construction*** technique. An alternative technique was suggested by the author for the (more restricted) case of planar arrangements, see [Har00b]. The idea is to compute only what the algorithm really need to compute the output, by computing the vertical decomposition in an exploratory online fashion. The details are unfortunately overwhelming although the algorithm seems to perform quite well in practice.

Cuttings. The concept of cuttings was introduced by Clarkson. The first optimal size cutting were constructed by Chazelle and Friedman [CF90], who proved the exponential decay lemma to this end. Our elegant proof follows the presentation by de Berg and Schwarzkopf [dBS95]. The problem with this approach is that the constant involved in the cuttings size are awful[5]. Matoušek [Mat98] showed that there $(1/r)$-cuttings with $8r^2 + 6r + 4$ trapezoids, by using level approximation. A different approach, was taken by the author [Har00a], who showed how to get cuttings which seems to be quite small (i.e., constant-wise) in practice. The basic idea is to do randomized incremental construction, but at each iteration greedily add all the trapezoids with conflict list small enough to the output cutting. One can prove that this algorithm also generate $O(r^2)$ cuttings, but the details are not trivial as the framework described in this chapter is not applicable for analyzing this algorithm.

Cuttings also can be computed in higher dimensions for hyperplanes, and in the place for well behaved curves, see [SA95].

Even more on randomized algorithms in geometry. We had only scratched the surface of this fascinating topic, which is one of the corner stones of "modern" computational geometry. The interested reader should have a look in the books by Mulmuley [Mul94a], Sharir and Agarwal [SA95], Matoušek [Mat02] and Boissonnat and Yvinec [BY98].

[5] This is why all computations related to cuttings should be done on waiter's bill pad. As Douglas Adams put it: "On a waiter's bill pad, reality and unreality collide on such a fundamental level that each becomes the other and anything is possible, within certain parameters."

6.5 Proof of Lemma 6.2.1

Proof of of Lemma 6.2.1: Let $\mathcal{E}_\sigma$ be the event that $D(\sigma) \subseteq \mathsf{R}$ and $K(\sigma) \cap \mathsf{R} = \emptyset$. Similarly, let $\mathcal{E}'_\sigma$ be the event that $D(\sigma) \subseteq \mathsf{R}'$ and $K(\sigma) \cap \mathsf{R}' = \emptyset$. By the axioms, we have that $\sigma \in \mathcal{F}(\mathsf{R})$ (resp., $\sigma \in \mathcal{F}(\mathsf{R}')$) if and only if $\mathcal{E}_\sigma$ (resp., $\mathcal{E}'_\sigma$) happens.

The proof of this lemma is somewhat tedious and follows by careful calculations. Let $\delta = |D(\sigma)| \leq d$, $\omega = \omega(\sigma)$, and for two non-negative integers $a \leq x$, let $x^{\underline{a}} = x(x-1)\cdots(x-a+1)$. Then

$$\begin{aligned}
\frac{\mathbf{Pr}[\mathcal{E}_\sigma]}{\mathbf{Pr}[\mathcal{E}'_\sigma]} &= \frac{\binom{n-\omega-\delta}{r-\delta}}{\binom{n}{r}} \cdot \frac{\binom{n}{r'}}{\binom{n-\omega-\delta}{r'-\delta}} = \frac{\binom{n}{r'}}{\binom{n}{r}} \cdot \frac{\binom{n-\omega-\delta}{r-\delta}}{\binom{n-\omega-\delta}{r'-\delta}} \\
&= \frac{(n-r)!\,r!}{(n-r')!\,r'!} \cdot \frac{(n-\omega-r')!\,(r'-\delta)!}{(n-\omega-r)!\,(r-\delta)!} \\
&= \frac{r!}{(r-\delta)!} \cdot \frac{(r'-\delta)!}{r'!} \cdot \frac{(n-\omega-r')!}{(n-\omega-r)!} \cdot \frac{(n-r)!}{(n-r')!} \\
&= \frac{r^{\underline{\delta}}}{r'^{\underline{\delta}}} \cdot \frac{(n-\omega-r')^{\underline{r-r'}}}{(n-r')^{\underline{r-r'}}} \leq \frac{r^{\underline{d}}}{r'^{\underline{d}}} \cdot \frac{(n-\omega-r')^{\underline{r-r'}}}{(n-r')^{\underline{r-r'}}}\,.
\end{aligned}$$

By our assumption $r' = \lfloor r/t \rfloor \geq d$, so we obtain

$$\frac{r^{\underline{\delta}}}{r'^{\underline{\delta}}} \leq \left(\frac{r-\delta+1}{r'-\delta+1}\right)^{\delta} \leq \left(\frac{r-d+1}{r'-d+1}\right)^{\delta} \leq \left(\frac{r-d+1}{r'-d+1}\right)^{d} \leq \left(\frac{r}{r'/d}\right)^{d} = O\Big(((t+1)d)^d\Big) = O\Big(t^d\Big),$$

since $r' - d + 1 \geq r'/d$ and d is a constant. To bound the second factor, we observe that, for $i = r', r'+1, \ldots, r-1$,

$$\frac{(n-\omega-r')^{\underline{r-r'}}}{(n-r')^{\underline{r-r'}}} \leq \left(\frac{n-\omega-r+1}{n-r+1}\right)^{r-r'} = \left(1 - \frac{\omega}{n-r+1}\right)^{r-r'} \leq \left(1-\frac{\omega}{n}\right)^{r-r'} \leq \exp\left(\frac{\omega(r-r')}{n}\right).$$

Since $\omega \geq t(n/r)$, we have $\omega/n \geq t/r$, and therefore

$$\begin{aligned}
\frac{\mathbf{Pr}[\mathcal{E}_\sigma]}{\mathbf{Pr}[\mathcal{E}'_\sigma]} &= O\left(t^d \exp\left(\frac{-\omega(r-r')}{n}\right)\right) = O\left(t^d \exp\left(\frac{-t(r-r')}{r}\right)\right) \\
&= O(t^d)\exp(-(t-1)) = O(2^{-t})\,,
\end{aligned}$$

as desired. ∎

Chapter 7

Depth estimation via sampling

> "Maybe the Nazis told the truth about us. Maybe the Nazis were the truth. We shouldn't forget that: perhaps they were the truth. The rest, just beautiful lies. We've sung many beautiful lies about ourselves. Perhaps that's what I'm trying to do - to sing another beautiful lie."
>
> – –The roots of heaven, Romain Gary

In this chapter, we introduce a "trivial" but yet powerful idea. Given a set S of objects, a point p that is contained in some of the objects, and let its ***weight*** be the number of objects that contains it. We can estimate the depth/weight of p by counting the number of objects that contains it in a random sample of the objects. In fact, by considering points induced by the sample, we can bound the number of "light" vertices induced by S. This idea can be extended to bounding the number of "light" configurations induced by a set of objects.

This approach leads to a sequence of short, beautiful, elegant and correct[6] proofs of several hallmark results in discrete geometry.

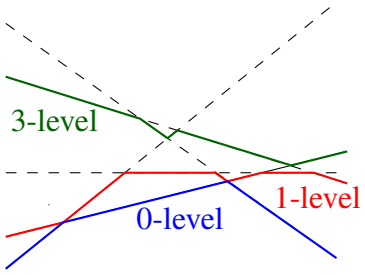
While the results in this chapter are not directly related to approximation algorithms, the insights and general approach would be useful for us later, or so one hopes.

7.1 The at most k-levels

Let L be a set of n lines in the plane. A point $\mathsf{p} \in \bigcup_{\ell \in \mathsf{L}} \ell$ is of ***level*** k, if there are k lines of L strictly below it. The ***k-level*** is the closure of set of points of level k. Namely, the k-level is an x-monotone curve along the lines of L.

The 0-level are just the boundary of the "bottom" face of the arrangement of L (i.e., the face containing the negative y-axis). It is easy to verify that 0-level has at most $n-1$ vertices, as each line might contribute at most one segment to the 0-level (which is an unbounded convex polygon).

It is natural to ask what is the number of vertices at the k-level (i.e., what is the combinatorial complexity of the polygonal chain forming the k-level). This is a surprisingly hard question, but the question of what is the complexity of the at most k level is considerably easier.



Theorem 7.1.1 *The number of vertices of level at most k in an arrangement of n lines in the plane is $O(nk)$.*

Proof: Pick a random sample R of L, by picking each line into the sample with probability $1/k$. Observe that

$$\mathbf{E}\Big[|\mathsf{R}|\Big] = \frac{n}{k}.$$

Let $V_{\le k} = V_{\le k}(\mathsf{L})$ be the set of all vertices of $\mathcal{A}(\mathsf{L})$ of level at most k, for $k > 1$. For a vertex $\mathsf{p} \in V_{\le k}$, let X_{p} be an indicator variable which is one if p is a vertex of the 0-level of $\mathcal{A}(\mathsf{R})$. The probability that p is in the 0-level of $\mathcal{A}(\mathsf{R})$ is the probability that none of the j lines below it are picked to the sample, and the two lines that define it do get selected to the sample. Namely,

$$\mathbf{Pr}\Big[X_{\mathsf{p}} = 1\Big] = \left(1 - \frac{1}{k}\right)^{j} \left(\frac{1}{k}\right)^{2} \ge \left(1 - \frac{1}{k}\right)^{k} \frac{1}{k^2} \ge \exp\left(-2\frac{k}{k}\right) \frac{1}{k}^{2} = \frac{1}{e^2 k^2}.$$

[6] The saying goes that "hard theorems have short, elegant and incorrect proofs". This chapter can maybe serve as a counterexample to this claim.

since $j \leq k$ and $1 - x \geq e^{-2x}$, for $0 < x < 1/2$.

On the other hand, the number of vertices on the 0-level of R is $|\mathsf{R}| - 1$. As such,

$$\sum_{\mathsf{p} \in V_{\leq k}} X_{\mathsf{p}} \leq |\mathsf{R}| - 1.$$

And this, of course, also holds in expectation, implying

$$\mathbf{E}\left[\sum_{\mathsf{p} \in V_{\leq k}} X_{\mathsf{p}}\right] \leq \mathbf{E}[|\mathsf{R}| - 1] \leq \frac{n}{k}.$$

On the other hand, by linearity of expectation, we have

$$\mathbf{E}\left[\sum_{\mathsf{p} \in V_{\leq k}} X_{\mathsf{p}}\right] = \sum_{\mathsf{p} \in V_{\leq k}} \mathbf{E}\left[X_{\mathsf{p}}\right] \geq \frac{|V_{\leq k}|}{e^2 k^2}.$$

Putting these two inequalities together, we get that $\frac{|V_{\leq k}|}{e^2 k^2} \leq \frac{n}{k}$. Namely, $|V_{\leq k}| \leq e^2 nk$. ■

The connection to depth is simple. Every line defines a halfplane (i.e., the region above the line). A vertex of depth at most k, is contained in at most k halfplanes. The above proof (intuitively) first observed that there are at most n/k vertices of the random sample of zero depth (i.e., 0-level), and then showing that every vertex has probability (roughly) $1/k^2$ to have depth zero in the random sample. It thus follows, that the if the number of vertices of level at most k is μ, then $\mu/k^2 \leq n/k$; namely, $\mu = O(nk)$.

7.2 The Crossing Lemma

A graph $\mathsf{G} = (V, E)$ is ***planar***, if it can be drawn in the plane so that none of its edges are crossing. We need the following result of Euler.

Theorem 7.2.1 (Euler's formula.) *For a connected planar graph G, we have $f - e + v = 2$, where f, e, v are the number of faces, edges and vertices in a planar drawing of G.*

Lemma 7.2.2 *If G is a planar graph, then $e \leq 3v - 6$*

Proof: We assume that the number of edges of G is maximal (i.e., no edges can be added without introducing a crossing). If it is not maximal, then add edges till it becomes maximal. This implies that G is a triangulation (i.e., every face is a triangle). Then, every face is adjacent to three edges, and as such $2e = 3f$. By Euler's formula, we have $f - e + v = (2/3)e - e + v = 2$. Namely, $-e + 3v = 6$. Alternatively, $e = 3v - 6$. However, if e is not maximal, this equality deteriorates to the required inequality. ■

For example, the above inequality implies that the complete graph over 5 vertices (i.e., K_5) is not planar. Indeed, it has $e = \binom{5}{2} = 10$ edges, and $v = 5$ vertices, but if it was planar, the above inequality would imply that $10 = e \leq 3v - 6 = 9$, which is of course false. (The reader can amuse herself by trying to prove that $K_{3,3}$, the bipartite complete graph with 3 vertices on each side, is not planar.)

Kuratowski's celebrated theorem states that a graph is planar if and only if it does not contain either K_5 or $K_{3,3}$ induced inside it (formally, it does not have K_5 or $K_{3,3}$ as a minor).

For a graph G, we define the crossing number of G, denoted as $c(\mathsf{G})$, as the minimal number of edge crossings in any drawing of G in the plane. For a planar graph $c(\mathsf{G})$ is zero, and it "larger" for "less planar" graphs.

Claim 7.2.3 *For a graph G, we have $c(\mathsf{G}) \geq e - 3v + 6$.*

Proof: If $e - 3v + 6 \leq 0 \leq c(\mathsf{G})$ and the claim holds trivially. Otherwise, the graph G is not planar by Lemma 7.2.2. Draw G in such a way that $c(\mathsf{G})$ is realized and assume, for the sake of contradiction, that $c(\mathsf{G}) < e - 3v + 6$. Let H be the graph resulting from G, by removing from each pair of edges of G that intersects in the drawing one of the edges. We have $e(\mathsf{H}) \geq e(\mathsf{G}) - c(\mathsf{G})$. But H is planar (since its drawing has no crossings), and by Lemma 7.2.2, we have $e(\mathsf{H}) \leq 3v(\mathsf{H}) - 6$, or equivalently, $e(\mathsf{G}) - c(\mathsf{G}) \leq 3v - 6$. Namely, $e - 3v + 6 \leq c(\mathsf{G})$. Which contradicts our assumption. ■

Lemma 7.2.4 (The crossing lemma.) *For a graph G, such that $e \geq v$, we have $c(\mathsf{G}) = \Omega(e^3/v^2)$.*

Proof: We consider a specific drawing D of G in the plane that has $c(G)$ crossings. Next, let U be a random subset of V selected by choosing each vertex of V to be in the sample with probability $p > 0$.

Let $\mathsf{H} = \mathsf{G}_U$ be inducted subgraph over U. Note, that only edges of G with both their endpoints in U "survive" in H.

Thus, the probability of a vertex v to survive in H is p. The probability of an edge of G to survive in H is p^2, and the probability of a crossing (in this specific drawing D) to survive in the induced drawing D_{H} (of H) is p^4. Let X_v and X_e denote the (random variable which is the) number of vertices and edges surviving in H, respectively. Similarly, let X_c be the number of crossing surviving in D_{H}. By Claim 7.2.3, we have

$$X_c \geq c(\mathsf{H}) \geq X_e - 3X_v + 6.$$

In particular, this holds in the expectation, and as such

$$\mathbf{E}[X_c] \geq \mathbf{E}[X_e] - 3\,\mathbf{E}[X_v] + 6.$$

By linearity of expectation, we have

$$c(\mathsf{G})p^4 \geq ep^2 - 3vp + 6,$$

where e and v are the number of edges and vertices of G, respectively. In particular, $c(\mathsf{G}) \geq e/p^2 - 3v/p^3 + 6/p^4$. In particular, setting $p = v/e$ (we assume here that $e \geq v$), we have $c(\mathsf{G}) = \Omega(e^3/v^2)$. ■

Surprisingly, despite its simplicity, Lemma 7.2.4 is a very strong tool, as the following results testify.

7.2.1 On the number of incidences

Let P be a set of n disjoint points in the plane, and let L be a set of m distinct lines in the plane (note that all the lines might pass through a common point, as we do not assume general position here). Let $I(\mathsf{P}, \mathsf{L})$ denote the number of point/line pairs (p, ℓ), where $\mathsf{p} \in \mathsf{P}, \ell \in \mathsf{L}$, such that $\mathsf{p} \in \ell$. The number $I(\mathsf{P}, \mathsf{L})$ is the number of ***incidences*** between lines of L and points of P. Let $I(n, m) = \max_{|\mathsf{P}|=n, |\mathsf{L}|=m} I(\mathsf{P}, \mathsf{L})$.

The following "easy" result has a long history and required major effort to prove before this elegant proof was discovered[②].

Lemma 7.2.5 *The maximum number of incidences between n points and m lines is $I(n, m) = O(n^{2/3}m^{2/3} + n + m)$.*

Proof: Let P and L be the set of n points and set of m lines, respectively, realizing $I(m, n)$. Let G be a graph over the points of P (we assume that P contains an additional point at infinity). We connect two points if they lie consecutively on a common line of L. Clearly, $e = e(\mathsf{G}) = I + m$ and $v = v(\mathsf{G}) = n + 1$, where $I = I(m, n)$. Since we can interpret the arrangement of lines $\mathcal{A}(\mathsf{L})$ as a drawing of G, where a crossing of two edges of G is just a vertex of $\mathcal{A}(\mathsf{L})$. As such, it follows that $c(\mathsf{G}) \leq m^2$, since m^2 is a trivial bound on the number of vertices of $\mathcal{A}(\mathsf{L})$. On the other hand, by Lemma 7.2.4, we have $c(\mathsf{G}) = \Omega(e^3/v^2)$. Thus,

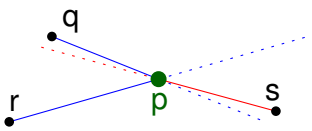
$$\frac{(I+m)^3}{(n+1)^2} = \frac{e^3}{v^2} = O(c(\mathsf{G})) = O(m^2).$$

Assuming $I \geq m$ and $I \geq n$, we have $I = O(m^{2/3}n^{2/3})$. Or alternatively, $I = O(n^{2/3}m^{2/3} + m + n)$. ■

7.2.2 On the number of k-sets

Let P be a set of n points in the plane in general position (i.e., no three points are collinear). A pair of points $\mathsf{p}, \mathsf{q} \in \mathsf{P}$ form a ***k-set*** if there are exactly k points in the (closed) halfplane below the line passing through p and q. Consider the graph $\mathsf{G} = (\mathsf{P}, E)$ that has an edge for every k-set. We will be interested in bounding the size of E as a function of n. Observe, that via duality, it is easy to observe that the number of k-sets, is exactly the complexity of the k-level in the dual arrangement $\mathcal{A}(\mathsf{P}^\star)$.

Lemma 7.2.6 (Antipodality.) *Let qp and rp be two k-set edges of G, with q and r to the left of p. Then there exists a point $\mathsf{s} \in \mathsf{P}$ to the right of p such that ps is a k-set, and $\mathrm{line}(\mathsf{p}, \mathsf{s})$ lies between $\mathrm{line}(\mathsf{q}, \mathsf{p})$ and $\mathrm{line}(\mathsf{r}, \mathsf{p})$.*



Proof: Let $f(\alpha)$ be the number of points below or on the line passing through p and having slope α, where α is a real number. Rotating this line counterclockwise around p corresponds to increasing α. In the following, let $f_+(x)$ (resp., $f_-(x)$) denote the value of $f(\cdot)$ just to the right (resp., left) of x.

Any point swept over by this line which is to the right of p increases f, and any point swept over to the left of p decreases f by one.

Let α_{q} and α_{r} be the slope of the lines containing qp and rp, respectively. Assume, for the sake of simplicity of exposition, that $\alpha_{\mathsf{q}} < \alpha_{\mathsf{r}}$. Clearly, $f\left(\alpha_{\mathsf{q}}\right) = f(\alpha_{\mathsf{r}}) = k$ and $f_+\left(\alpha_{\mathsf{q}}\right) = k - 1$. Let y be the smallest value such that $y > \alpha_{\mathsf{q}}$ and $f(y) = k$. We have

[②] Or invented – I have no dog in this argument.

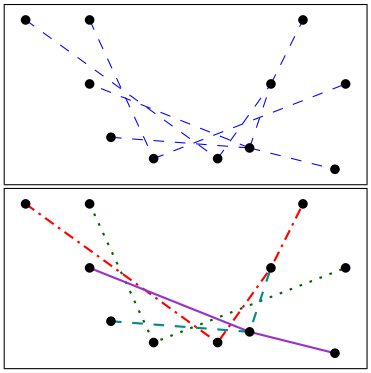
that $f_{-}(y) = k-1$, which implies that the line passing through p with slope $f(y)$ has a point $\mathsf{s} \in \mathsf{P}$, on it, and s is to the right of p. Clearly, if we continue sweeping, the line would sweep over rp, which implies the claim. ■

Lemma 7.2.6 also holds by symmetry in the other direction: Between any two edges to the right of p, there is an antipodal edge on the other side.

Lemma 7.2.7 *Let p be a point of P, and let q be a point to its left, such that $\mathsf{qp} \in \mathcal{E}(\mathsf{G})$ and it has the largest slope among all such edges. Furthermore, assume that there are $k-1$ points of P to the right of p. Then, there exists a point $\mathsf{r} \in \mathsf{P}$, such that $\mathsf{pr} \in \mathcal{E}(\mathsf{G})$ and pr has larger slope than qp.*

Proof: Let α be the slope of qp, and observe that $f(\alpha) = k$ and $f_{+}(\alpha) = k-1$, and $f(\infty) \geq k$. Namely, there exists $y > \alpha$ such that $f(y) = k$. We conclude that there is k-set adjacent to p on the right, with slope larger than α. ■

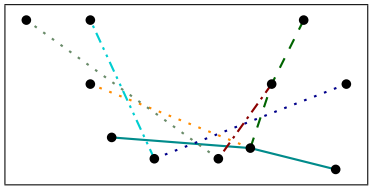
So, imagine that we are at an edge $\mathsf{e} = \mathsf{qp} \in \mathcal{E}(\mathsf{G})$, where q is to the left of p. We rotate a line around p (counterclockwise) till we encounter an edge $\mathsf{e}' = \mathsf{pr} \in E(\mathsf{G})$, where r is a point to the right of p. We can now walk from e to e', and continue walking in this way, forming a chain of edges in G. Note, that by Lemma 7.2.6, no two such chains can be "merged" into using the same edge. Furthermore, by Lemma 7.2.7, such a chain can end only in the last $k-1$ points of P (in their ordering along the x-axis). Namely, we decomposed the edges of G into $k-1$ edge disjoint convex chains (the chains are convex since we rotate clockwise as we walk along a chain). The picture on the right shows the 5-sets and their decomposition into 4 convex chains.



Lemma 7.2.8 *The edges of G can be decomposed into $k-1$ convex chains $\mathsf{C}_1, \ldots, \mathsf{C}_{k-1}$.*

Similarly, the edges of G can be decomposed into $m = n-k+1$ concave chains $\mathsf{D}_1, \ldots, \mathsf{D}_m$.

Proof: The first part of the claim is proved above. As for the second claim, rotate the plane by 180°. Every k-set is now $(n-k+2)$-set, and by the above argumentation, the edges of G can be decomposed into $n-k+1$ convex chains, which are concave in the original orientation. ■



Theorem 7.2.9 *The number of k-sets defined by a set of n points in the plane is $O\left(nk^{1/3}\right)$.*

Proof: The graph G has $n = |\mathsf{P}|$ vertices, and let $m = |\mathcal{E}(\mathsf{G})|$ be the number of k-sets. By Lemma 7.2.8, any crossing of two edges of G, is an intersection point of one convex chain of $\mathsf{C}_1, \ldots, \mathsf{C}_{k-1}$ with a concave chain of $\mathsf{D}_1, \ldots, \mathsf{D}_{n-k+1}$. Since a convex chain and a concave chain can have at most two intersections, we conclude that there are at most $2(k-1)(n-k+1)$ crossings in G. By the Crossing Lemma (Lemma 7.2.4), there are at least $\Omega\left(m^3/n^2\right)$ crossings. Putting this two inequalities together, we conclude $m^3/n^2 = O(nk)$, which implies $m = O\left(nk^{1/3}\right)$. ■

7.3 A general bound for the at most k-weight

We now extend the at most k-level technique to the general moments technique settings (see). We quickly restate the abstract settings.

Let S be a set of objects. For a subset $\mathsf{R} \subseteq \mathsf{S}$, we define a collection of 'regions' called $\mathcal{F}(\mathsf{R})$. Let $\mathcal{T} = \mathcal{T}(\mathsf{S}) = \bigcup_{\mathsf{R} \subseteq \mathsf{S}} \mathcal{F}(\mathsf{R})$ denote the set of ***all possible regions*** defined by subsets of S. We associate two subsets $D(\sigma), K(\sigma) \subseteq \mathsf{S}$ with each region $\sigma \in \mathcal{T}$. The ***defining set*** $D(\sigma)$ of σ is a subset of S defining the region σ. We assume that for every $\sigma \in \mathcal{T}$, $|D(\sigma)| \leq d$ for a (small) constant d, which is the ***combinatorial dimension***. The ***killing set*** $K(\sigma)$ of σ is the set of objects of S such that including any object of $K(\Delta)$ into R prevents σ from appearing in $\mathcal{F}(\mathsf{R})$. The ***weight*** of σ is $\omega(\sigma) = |K(\sigma)|$.

Let $\mathsf{S}, \mathcal{F}(\mathsf{R}), D(\sigma)$, and $K(\sigma)$ be such that for any subset $\mathsf{R} \subseteq \mathsf{S}$, the set $\mathcal{F}(\mathsf{R})$ satisfies the following axioms: (i) For any $\sigma \in \mathcal{F}(\mathsf{R})$, it holds $D(\sigma) \subseteq \mathsf{R}$ and $\mathsf{R} \cap K(\sigma) = \emptyset$. (ii) If $D(\sigma) \subseteq \mathsf{R}$ and $K(\sigma) \cap \mathsf{R} = \emptyset$, then $\sigma \in \mathcal{F}(\mathsf{R})$.

Let $\mathcal{T}_{\leq k}(\mathsf{S})$ be the set of regions of $\mathcal{T}$ with weight at most k. Furthermore, assume that the expected number of regions of zero weight of a sample of size r is at most $f(r)$. We have the following theorem.

Theorem 7.3.1 *Let S be a set of n objects as above, with combinatorial dimension d, and let k be a parameter. Let R be a random sample created by picking each element of S with probability $1/k$. Then, we have*

$$|\mathcal{T}_{\leq k}(\mathsf{S})| \leq c\, \mathbf{E}\left[k^d f(|\mathsf{R}|)\right],$$

for a constant c.

Proof: We reproduce the proof of Theorem 7.1.1. Every region $\sigma \in \mathcal{T}_{\leq k}$ appears in $\mathcal{F}(\mathsf{R})$ with probability $\geq 1/k^d(1-1/k)^k \geq e^{-2}/k^d$. As such, $\mathbf{E}[f(|\mathsf{R}|)] \geq \mathbf{E}[|\mathcal{F}(\mathsf{R})|] \geq |\mathcal{T}_{\leq k}|/\left(k^d e^2\right)$. ■

Lemma 7.3.2 *Let $f(\cdot)$ be a monotone increasing function which is well behaved; namely, that there exists a constant c, such that $f(xr) \leq c f(r)$, for any r and $1 \leq x \leq 2$. Let Y be the number of heads in n coin-flips where the probability for head is $1/k$. Then $\mathbf{E}[f(Y)] = O(f(n/k))$.*

Proof: This follows easily from Chernoff inequality. Indeed,

$$\begin{aligned}\mathbf{E}[f(Y)] &\leq f(10(n/k)) + \sum_{t=10}^{k} f((t+1)k)\,\mathbf{Pr}[Y \geq t(n/k)] \\ &\leq O(f(n/k)) + \sum_{t=10}^{k} c^{\lceil \lg t+1 \rceil} f(n/k) 2^{-t(n/k)} = O(f(n/k))\,,\end{aligned}$$

by the simplified form of Chernoff inequality, see Theorem 25.2.6. ■

The following is an immediate consequence of Theorem 7.3.1 and Lemma 7.3.2.

Theorem 7.3.3 *Let S be a set of n objects, with combinatorial dimension d, and let k be a parameter. Assume that the number of regions formed by a set of m objects is bounded by a function $f(m)$, and furthermore, $f(m)$ is well behaved in the sense of Lemma 7.3.2. Then, $|\mathcal{T}_{\leq k}(\mathsf{S})| = O\left(k^d f(n/k)\right)$.*

Note, that if the function $f(\cdot)$ grows polynomially then Theorem 7.3.3 applies. It fails if $f(\cdot)$ grows exponentially.
We need the following fact, which we state without proof.

Theorem 7.3.4 (The Upper Bound Theorem.) *The complexity of the convex-hull of n points in d dimensions is bounded by $O\left(n^{\lfloor d/2 \rfloor}\right)$.*

Example 7.3.5 (At most k-sets.) Let P be a set of n points in $\mathbb{R}^d$. A region here is a halfspace with d points on its boundary. The set of regions defined by P is just the faces of the convex hull of P. The complexity of the convex hull of n points in d dimensions is $f(n) = O\left(n^{\lfloor d/2 \rfloor}\right)$, by Theorem 7.3.4. Two halfspaces h, h' would be considered to be combinatorially different if $\mathsf{P} \cap h \neq \mathsf{P} \cap h'$. As such, the number of combinatorially different halfspaces containing at most k points of P is at most $O\left(k^d f(n/k)\right) = O\left(k^{\lceil d/2 \rceil} n^{\lfloor d/2 \rfloor}\right)$.

7.4 Bibliographical notes

The reader should not mistaken the simplicity of the proofs in this chapter with easiness. Almost all the results presented, in this chapter, have long and painful history with earlier results which were technically much more involved (and weaker). In some sense, these results are the limit of mathematical evolution: They are simple, (in some cases) breathtakingly elegant, and on their own (without exposure to previous work on the topic), it is inconceivable that one can come up with them.

At most k-level. The technique for bound the complexity of the at most k-level (or at most depth k) is generally attributed to Clarkson-Shor [CS89] and more precisely it is from [Cla88]. Previous work on just the two dimensional variant include [GP84, Wel86, AG86]. Our presentation in Section 7.1 and Section 7.3 follows (more or less) Sharir [Sha03]. The connection of this technique to the crossing lemma is from there.

For a proof of th Upper Bound Theorem (Theorem 7.3.4), see Matoušek [Mat02].

The crossing lemma. The crossing lemma is originally by Ajtai *et al.* [ACNS82] and Leighton [Lei84]. The current greatly simplified "proof from the book" is attributed to Sharir. The insight that this lemma has something to do with incidences and similar problems is due to Székely [Szé97]. Elekes [Ele97] used the crossing lemma to prove surprising lower bounds on sum and products problems.

The complexity of k-level and number of k-sets. This is considered to be one of the hardest problems in discrete geometry, and there is still a big gap between the best lower bound [Tót01] and best upper bound currently known [Dey98]. Our presentation in Section 7.2.2 follows suggestions by Micha Sharir, and is based on the result of Dey [Dey98] (which was in turn inspired by the work of Agarwal *et al.* [AACS98]). This problem has long history, and the reader is referred to Dey [Dey98] for its history.

Incidences. This problem again has long and painful history. The reader is referred to [Szé97] for details.

We only skimmed the surface of some problems in discrete geometry and results known in this field related to incidences and k-sets. Good starting points for learning more are the books by Brass *et al.* [BMP05] and Matoušek [Mat02].

Chapter 8

Approximating the Depth via Sampling and Emptiness

> As far as he personally was concerned there was nothing else for him to do except either shoot himself as soon as he came home or send for his greatcoat and saber from the general's apartment, take a bath in the town baths, stop at Volgruber's wine-cellar l afterwords, put his appetite in order again and book by telephone a ticket for the performance in the town theater that evening.
>
> – – The good soldier Svejk, Jaroslav Hasek

8.1 From Emptiness to Approximate Range Counting

Assume that there exists a data structure that can be constructed in $T(n)$ time for a set S of n objects such that, given a query range $\mathbf{r}$, we can check in $Q(n)$ time whether $\mathbf{r}$ intersects any of the objects in S. Let $S(n)$ be the space required to store this data structure. In this section, we show how to build a data structure that quickly returns an approximate number of objects in S intersecting $\mathbf{r}$ using emptiness testing as a subroutine.

In particular, let $\mu_{\mathbf{r}} = \mathrm{depth}(\mathbf{r}, \mathsf{S})$ denote the ***depth*** of $\mathbf{r}$; namely, its the number of objects of S intersected by $\mathbf{r}$. Below we use $\varepsilon > 0$ to denote the required approximation quality; namely, we would like the data structure to output a number $\alpha_{\mathbf{r}}$ such that $(1-\varepsilon)\mu_{\mathbf{r}} \leq \alpha_{\mathbf{r}} \leq \mu_{\mathbf{r}}$.

8.1.1 The decision procedure

Given parameters $z \in [1, n]$ and ε, with $1/2 > \varepsilon > 0$, we construct a data structure, such that given a query range $\mathbf{r}$, we can decide whether $\mu_{\mathbf{r}} < z$ or $\mu_{\mathbf{r}} \geq z$. The data structure is allowed to make a mistake if $\mu_{\mathbf{r}} \in \Big[(1-\varepsilon)z, (1+\varepsilon)z\Big]$.

The data structure. Let $\mathsf{R}_1, \ldots, \mathsf{R}_M$ be M independent random samples of S, formed by picking every element with probability $1/z$, where

$$M = \nu(\varepsilon) = \Big\lceil c_2 \varepsilon^{-2} \log n \Big\rceil,$$

and c_2 is a sufficiently large absolute constant. Build M separate emptiness-query data structures $D_1, \ldots, D_M$, for the sets $\mathsf{R}_1, \ldots, \mathsf{R}_M$, respectively, and put $\mathcal{D} = \mathcal{D}(z, \varepsilon) = \{D_1, \ldots, D_M\}$.

Answering a query. Consider a query range $\mathbf{r}$, and let $X_i = 1$ if $\mathbf{r}$ intersects any of the objects of R_i and $X_i = 0$ otherwise, for $i = 1, \ldots, N = \nu(\varepsilon)$. The value of X_i can be determined using a single emptiness query in D_i, for $i = 1, \ldots, M$. Compute $Y_{\mathbf{r}} = \sum_i X_i$.

For a range σ of depth k, the probability that σ intersects one of the objects of R_i is

$$\rho(k) = 1 - \left(1 - \frac{1}{z}\right)^k. \tag{8.1}$$

If a range σ has depth z, then $\Lambda = \mathbf{E}[Y_\sigma] = \nu(\varepsilon)\rho(z)$. Our data structure returns "$\mathrm{depth}(\mathbf{r}, \mathsf{S}) < z$" if $Y_{\mathbf{r}} < \Lambda$, and "$\mathrm{depth}(\mathbf{r}, \mathsf{S}) \geq z$" otherwise.

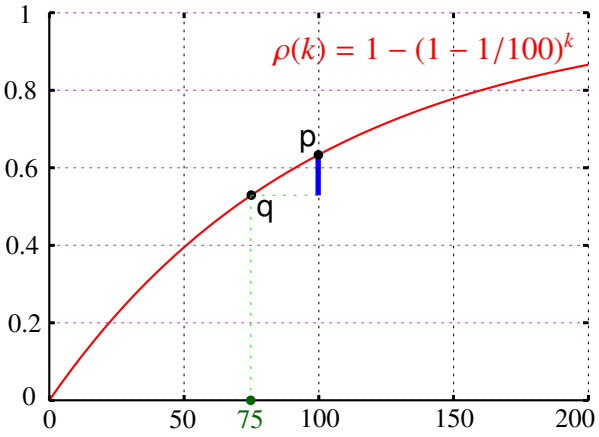
8.1.1.1 Correctness

In the following, we show that, with high probability, the data structure indeed returns the correct answer if the depth of the query range is outside the "uncertainty" range $[(1-\varepsilon)z, (1+\varepsilon)z]$. For simplicity of exposition, we assume in the following that $z \geq 10$ (the case $z < 10$ follows by similar arguments). Consider a range $\mathbf{r}$ of depth at most $(1-\varepsilon)z$. The data structure returns wrong answer if $Y_{\mathbf{r}} > \Lambda$. We will show that the probability of this event is polynomially small. The other case, where $\mathbf{r}$ has depth at least $(1+\varepsilon)z$ but $Y_{\mathbf{r}} < \Lambda$ is handled in a similar fashion.

Intuition. Before jumping into the murky proof, let us consider the situation. Every sample R_i is an experiment. The experiment succeeds if the sample contains an object that intersects the query range $\mathbf{r}$. The probability of success is $\rho(z)$, see Eq. (8.1), where z is the weight of $\mathbf{r}$. Now, if there is a big enough gap between $\rho((1-\varepsilon)z)$ and $\rho(k)$, then we could decide if the range is "heavy" (i.e., weight exceeding z) or "light" (i.e., weight smaller than $(1-\varepsilon)z$) by estimating the probability γ that $\mathbf{r}$ intersects an object in the random sample.

Indeed, if $\mathbf{r}$ is "light" then $\gamma \leq \rho((1-\varepsilon)z)$, and if it is "heavy" then $\gamma \geq \rho(z)$. We estimate γ by the quantity $Y_{\mathbf{r}}/M$; namely, repeating the experiment M times, and dividing the number of successes by the number of experiments (i.e., M). Now, we need to determine how many experiments we need to perform till we get a good estimate, which is reliable enough to carry out our nefarious task of distinguishing the light case from the heavy case. Clearly, the bigger the gap is between $\rho((1-\varepsilon)z)$ and $\rho(z)$, the fewer experiments required. Our proof would first establish that the gap between these two probabilities is $\Omega(\varepsilon)$, and next we will plug this into Chernoff inequality to figure out how large M has to be for this estimate to be reliable.

To estimate this gap, we need to understand how the function $\rho(\cdot)$ looks like. So consider the graph on the right. Here $z = 100$ and $\varepsilon = 0.2$. For the heavy case, where the weight of range exceeds z, the probability of success is $\mathsf{p} = \rho(100)$, and probability of failure is $\mathsf{q} = \rho(80)$. Since the function behaves like a line around these values, its kind of visually obvious that the required gap (the vertical blue segment in the graph) is $\approx \varepsilon$. Proving this formally is somewhat more tedious.



In the following, we need the following two easy observations.

Observation 8.1.1 *For $0 \leq x \leq y < 1$, we have $\dfrac{1-x}{1-y} = 1 + \dfrac{y-x}{1-y} \geq 1 + y - x$.*

Observation 8.1.2 *For $x \in [0, 1/2]$, it holds $\exp(-2x) \leq 1 - x$. Similarly, for $x \geq 0$, we have $1 - x \leq \exp(-x)$ and $1 + x \leq \exp(x)$.*

Lemma 8.1.3 *Let $\Lambda = \mathbf{E}[Y_\sigma] = M\rho(z)$. We have*

$$\alpha = \mathbf{Pr}\Big[Y_{\mathbf{r}} > \Lambda \;\Big|\; \mathrm{depth}(\mathbf{r}, \mathsf{S}) \leq (1-\varepsilon)z\Big] \leq \frac{1}{n^{c_4}},$$

where $c_4 = c_4(c_2) > 0$ depends only on c_2 and can be made arbitrarily large by a choice of a sufficiently large c_2.

Proof: The probability α is maximized when $\mathrm{depth}(\mathbf{r}, \mathsf{S}) = (1-\varepsilon)z$. Thus

$$\alpha \leq \mathbf{Pr}[Y_{\mathbf{r}} > \Lambda \mid \mathrm{depth}(\mathbf{r}, \mathsf{S}) = (1-\varepsilon)z].$$

Observe that $\mathbf{Pr}[X_i = 1] = \rho\Big((1-\varepsilon)z\Big)$, so

$$\mathbf{E}[Y_{\mathbf{r}}] = M\rho((1-\varepsilon)z) = M \cdot \left(1 - \left(1 - \frac{1}{z}\right)^{(1-\varepsilon)z}\right) \geq M \cdot \left(1 - e^{-(1-\varepsilon)}\right) \geq \frac{M}{3},$$

since $1 - 1/z \leq \exp(-1/z)$ and $\varepsilon \leq 1/2$. By definition, $\Lambda = M\rho(z)$, therefore, by Observation 8.1.1, we have

$$\xi = \frac{\Lambda}{\mathbf{E}[Y_{\mathbf{r}}]} = \frac{1 - \left(1 - \frac{1}{z}\right)^{z}}{1 - \left(1 - \frac{1}{z}\right)^{(1-\varepsilon)z}} \geq 1 + \left(1 - \frac{1}{z}\right)^{(1-\varepsilon)z} - \left(1 - \frac{1}{z}\right)^{z} = 1 + \left(1 - \frac{1}{z}\right)^{(1-\varepsilon)z}\left(1 - \left(1 - \frac{1}{z}\right)^{\varepsilon z}\right).$$

Now, by applying Observation 8.1.2 repeatedly, we have

$$\xi \geq 1 + \exp\left(-\frac{2}{z}(1-\varepsilon)z\right) \cdot \left(1 - \exp\left(-\frac{1}{z}\varepsilon z\right)\right) = 1 + \frac{1}{e^2}\left(1 - \exp(-\varepsilon)\right)$$
$$\geq 1 + \frac{1}{e^2}\left(1 - \left(1 - \frac{\varepsilon}{2}\right)\right) \geq 1 + \frac{\varepsilon}{15}.$$

Deploying the Chernoff inequality (Theorem 25.2.6), we have that if $\mu_{\mathbf{r}} = \mathrm{depth}(\mathbf{r}, \mathsf{S}) = (1-\varepsilon)z$ then

$$\alpha = \mathbf{Pr}[Y_{\mathbf{r}} > \Lambda] \leq \mathbf{Pr}\Big[Y_{\mathbf{r}} > \xi\,\mathbf{E}[Y_{\mathbf{r}}]\Big] \leq \mathbf{Pr}\Big[Y_{\mathbf{r}} > (1+\varepsilon/15)\,\mathbf{E}[Y_{\mathbf{r}}]\Big]$$
$$\leq \exp\left(-\mathbf{E}[Y_{\mathbf{r}}]\,\frac{1}{4}\left(\frac{\varepsilon}{15}\right)^2\right) \leq \exp\left(-\frac{M\varepsilon^2}{c_3}\right) \leq \exp\left(-\frac{\varepsilon^2\left\lceil c_2\varepsilon^{-2}\log n\right\rceil}{c_3}\right) \leq n^{-c_4},$$

where c_3 is some absolute constant, and by setting c_2 to be sufficiently large. ■

This implies the following lemma.

Lemma 8.1.4 *Given a set S of n objects, a parameter $0 < \varepsilon < 1/2$, and $z \in [0, n]$, one can construct a data structure $\mathcal{D}$ which, given a range $\mathbf{r}$, returns either* L *or* R*. If it returns* L*, then $\mu_{\mathbf{r}} \leq (1+\varepsilon)z$, and if it returns* R *then $\mu_{\mathbf{r}} \geq (1-\varepsilon)z$. The data structure might return either answer if $\mu_{\mathbf{r}} \in [(1-\varepsilon)z, (1+\varepsilon)z]$.*

The data structure $\mathcal{D}$ consists of $M = O(\varepsilon^{-2}\log n)$ emptiness data structures. The space and preprocessing time needed to build them are $O(S(2n/z)\varepsilon^{-2}\log n)$ where $S(m)$ is the space (and preprocessing time) needed for a single emptiness data structure storing m objects.

The query time is $O(Q(2n/z)\varepsilon^{-2}\log n)$, where $Q(m)$ is the time needed for a single query in such a structure, respectively. All bounds hold with high probability.

Proof: The lemma follows immediately from the above discussion. The only missing part is observing that by the Chernoff inequality we have that $|\mathsf{R}_i| \leq 2n/z$, and this holds with high probability. ■

8.1.2 Answering approximate counting query

The path to answering approximate counting query is now clear. We use Lemma 8.1.4 to perform binary search, repeatedly narrowing the range containing the answer. We stop when the size of the range is within our error tolerances. At the start of the process, this range is large, so we use large values of ε. As the range narrows ε is reduced.

8.1.3 The data structure

Lemma 8.1.4 provides us with a tool for performing a "binary" search for the count value $\mu_{\mathbf{r}}$ of a range $\mathbf{r}$. For small values of i, we just build a separate data structure of Lemma 8.1.4 for depth values $i/2$, for $i = 1, \ldots, U = O(\varepsilon^{-1})$. For depth i, we use accuracy $1/8i$ (i.e., this is the value of ε when using Lemma 8.1.4). Using these data structures, we can decide whether the query range count is at least U, or smaller than U. If it is smaller than U, then we can perform a binary search to find its exact value. The result is correct with high probability.

Next, consider the values $v_j = (U/4)(1+\varepsilon/16)^j$, for $j = U+1, \ldots, W$, where $W = c\log_{1+\varepsilon/16} n = O(\varepsilon^{-1}\log n)$, for an appropriate choice of an absolute constant $c > 0$, so that $v_W = n$. We build a data structure $\mathcal{D}(v_j)$ for each $z = v_j$, using Lemma 8.1.4.

Answering a query. Given a range query $\mathbf{r}$, each data structure in our list returns L or R. Moreover, with high probability, if we were to query all the data structures, we would get a sequence of Rs, followed by a sequence of Ls. It is easy to verify that the value associated with the last data structure returning R (rounded to the nearest integer) yields the required approximation. We can use binary search on $\mathcal{D}(v_1), \ldots, \mathcal{D}(v_W)$ to locate this changeover value using a total of $O(\log W) = O(\log(\varepsilon^{-1}\log n))$ queries in the structures of $\mathcal{D}_1, \ldots, \mathcal{D}_W$. Namely, the overall query time is $O\Big(Q(n)\varepsilon^{-2}(\log n)\log(\varepsilon^{-1}\log n)\Big)$.

Theorem 8.1.5 *Given a set of S of n objects, and assume that one can construct, using $S(n)$ space, in $T(n)$ time, a data structure that answers emptiness queries in $Q(n)$ time.*

Then, one can construct, using $O\Big(S(n)\varepsilon^{-3}\log^2 n\Big)$ space, in $O(T(n)\varepsilon^{-3}\log^2 n)$ time, a data structure that, given a range $\mathbf{r}$, outputs a number $\alpha_{\mathbf{r}}$, with $(1-\varepsilon)\mu_{\mathbf{r}} \leq \alpha_{\mathbf{r}} \leq \mu_{\mathbf{r}}$. The query time is $O\Big(\varepsilon^{-2}Q(n)(\log n)\log(\varepsilon^{-1}\log n)\Big)$. The result returned is correct with high probability for all queries and the running time bounds hold with high probability.

The bounds of Theorem 8.1.5 can be improved, see Section 8.4 for details.

8.2 Application: halfplane and halfspace range counting

Using the data structure of Dobkin and Kirkpatrick [DK85], one can answer emptiness halfspace range searching queries in logarithmic time. In this case, we have $S(n) = O(n)$, $T(n) = O(n \log n)$, and $Q(n) = O(\log n)$.

Corollary 8.2.1 *Given a set P of n points in two (resp., three) dimensions, and a parameter $\varepsilon > 0$, one can construct in $O(n\mathrm{poly}(1/\varepsilon, \log n))$ time a data structure, of size $O(n\mathrm{poly}(1/\varepsilon, \log n))$, such that given a halfplane (resp. halfspace) $\mathbf{r}$, it outputs a number α, such that $(1-\varepsilon)|\mathbf{r} \cap P| \leq \alpha \leq |\mathbf{r} \cap P|$, and the query time is $O(\mathrm{poly}(1/\varepsilon, \log n))$. The result returned is correct with high probability for all queries.*

Using the standard lifting of points in $\mathbb{R}^2$ to the paraboloid in $\mathbb{R}^3$ implies a similar result for approximate range counting for disks, as a disk range query in the plane reduces to a halfspace range query in three dimensions.

Corollary 8.2.2 *Given a set of P of n points in two dimensions, and a parameter ε, one can construct a data structure in $O(n\mathrm{poly}(1/\varepsilon, \log n))$ time, using $O(n\mathrm{poly}(1/\varepsilon, \log n))$ space, such that given a disk $\mathbf{r}$, it outputs a number α, such that $(1-\varepsilon)|\mathbf{r} \cap P| \leq \alpha \leq |\mathbf{r} \cap P|$, and the query time is $O(\mathrm{poly}(1/\varepsilon, \log n))$. The result returned is correct with high probability for all possible queries.*

Depth queries. By computing the union of a set of n pseudodisks in the plane, and preprocessing the union for point-location queries, one can perform "emptiness" queries in this case in logarithmic time. (Again, we are assuming here that we can perform the geometric primitives on the pseudodisks in constant time.) The space needed is $O(n)$ and it takes $O(n \log n)$ time to construct it. Thus, we get the following result.

Corollary 8.2.3 *Given a set of S of n pseudodisks in the plane, one can preprocess them in $O(n\varepsilon^{-2} \log^2 n)$ time, using $O(n\varepsilon^{-2} \log n)$ space, such that given a query point q, one can output a number α, such that $(1-\varepsilon)\mathrm{depth}(p, \mathsf{S}) \leq \alpha \leq \mathrm{depth}(p, \mathsf{S})$, and the query time is $O\left(\varepsilon^{-2} \log^2 n\right)$. The result returned is correct with high probability for all possible queries.*

8.3 Relative approximation via sampling

In the above discussion, we had used several random samples of a set of n objects S, and by counting in how many samples a query point lies in, we got a good estimate of the depth of the point in S. It is natural to ask what can be done if we insist on using a single sample. Intuitively, if we sample each object with probability p in a random sample R, then if the query point $\mathbf{r}$ had depth that is sufficiently large (roughly, $1/(p\varepsilon^2)$) then its depth can be estimated reliably by counting the number of objects in R containing $\mathbf{r}$, and multiplying it by $1/p$. The interesting fact is that the deeper $\mathbf{r}$ is the better this estimate is.

Lemma 8.3.1 ((Reliable sampling.)) *Let S be a set of n objects, $0 < \varepsilon < 1/2$, and let $\mathbf{r}$ be a point of depth $u \geq k$ in S. Let R be a random sample of S, such that every element is picked into the sample with probability*

$$p = \frac{8}{k\varepsilon^2} \ln \frac{1}{\delta}.$$

Let X be the depth of $\mathbf{r}$ in R. Then, with probability $\geq 1-\delta$, we have that estimated depth of $\mathbf{r}$, that is X/p, lies in the interval $[(1-\varepsilon)u, (1+\varepsilon)u]$.

In fact, this estimates succeeds with probability $\geq 1 - \delta^{u/k}$.

Proof: We have that $\mu = \mathbf{E}[X] = pu$. As such, by Chernoff inequality (Theorem 25.2.6 and Theorem 25.2.9), we have

$$\begin{aligned}
\mathbf{Pr}\Big[X \notin [(1-\varepsilon)\mu, (1+\varepsilon)\mu]\Big] &= \mathbf{Pr}[X < (1-\varepsilon)\mu] + \mathbf{Pr}[X > (1+\varepsilon)\mu] \\
&\leq \exp\Big(-pu\varepsilon^2/2\Big) + \exp\Big(-pu\varepsilon^2/4\Big) \\
&\leq \exp\left(-4\ln\frac{1}{\delta}\right) + \exp\left(-2\ln\frac{1}{\delta}\right) \leq \delta,
\end{aligned}$$

since $u \geq k$. ∎

Note, that if $\mathbf{r}$ depth in S is (say) $u \leq 10k$ then the depth of $\mathbf{r}$ in sample is (with the stated probabilities)

$$\mathrm{depth}(\mathbf{r}, \mathsf{R}) \leq (1+\varepsilon)pu = O\left(\frac{1}{\varepsilon^2} \ln \frac{1}{\delta}\right).$$

Which is (relatively) a small number. Namely, via sampling, we turned the task of estimating the depth of heavy ranges, into the task of estimating the depth of a shallow range. To see why this is true, observe that we can perform a binary (exponential) search for the depth of $\mathbf{r}$ by a sequence of coarser to finer samples.

8.4 Bibliographical notes

The presentation here follows the work by Aronov and Har-Peled [AH05]. The basic idea is folklore and predates this paper, but the formal connection between approximate counting to emptiness is from this paper. One can improve the efficiency of this reduction by being more careful, see the full version of [AH05] for details. Followups to this work include [KS06, AC07, AHS07].

Chapter 9

Linear programming in Low Dimensions

> At the sight of the still intact city, he remembered his great international precursors and set the whole place on fire with his artillery in order that those who came after him might work off their excess energies in rebuilding.
>
> – – The tin drum, Gunter Grass

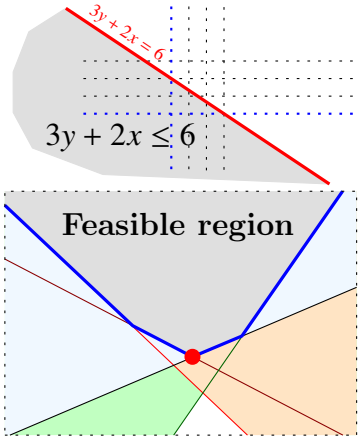
In this chapter, we shortly describe (and analyze) a simple randomized algorithm for linear programming in low dimensions. Next, we show how to extend this algorithm to solve linear programming with violations. Finally, we would show how one can efficiently approximate the number constraints one need to violate to make a linear program feasible. This serves as a fruitful ground to demonstrate some the techniques we visited already.

Our discussion is going to be somewhat intuitive. We will fill in the details, and prove correctness formally of our algorithms in the next chapter.

9.1 Linear Programming

Assume we are given a set of n linear inequalities defined of the form $a_1x_1 + \cdots + a_dx_d \leq b$, where $a_1, \ldots, a_d, b$ are constants, and $x_1, \ldots, x_d$ are the variables. In the ***linear programming*** (LP) problem, one has to find a ***feasible solution***; that is, a point $(x_1, \ldots, x_d)$ for which all the linear inequalities hold. In fact, usually we would like to find a feasible point that maximizes a linear expression (referred to as the ***target function*** of the LP) of the form $c_1x_1 + \cdots + c_dx_d$, where $c_1, \ldots, c_d$ are prespecified constants.

The set of points complying with a linear inequality $a_1x_1 + \cdots + a_dx_d \leq b$, is just a halfspace of $\mathbb{R}^d$, having the hyperplane $a_1x_1 + \cdots + a_dx_d = b$ as a boundary, see figure on the right. As such, the feasible region of the LP is the intersection of n halfspaces; that is, it is a ***polyhedron***. The linear target function is no more than specifying a direction, such that we need to find the point inside the polyhedron which is extreme in this direction. If the polyhedron is unbounded in this direction, the optimal solution is ***unbounded***.



For the sake of simplicity of exposition, it would be easiest to think on the direction that one has to optimize for as the negative x_d-axis direction. This can be easily realized by rotating space such that the required direction is pointing downward. Since the feasible region is the intersection of convex sets (i.e., halfspaces), it is convex. As such, one can imagine the boundary of the feasible region as vessel (with a convex interior). Next, we release a ball at the top of vessel, and the ball roll down (by "gravity" in the direction of the negative x_d-axis) till it reaches the lowest point in the vessel and get "stuck". This point is the optimal solution to the LP that we are interested in computing.

In the following, we will assume that the given LP is in general position. Namely, if we intersect k hyperplanes, induced by k inequalities in the given LP, then their intersection is $d - k$ dimensional affine subspace. In particular, intersection of d of them is a point (referred to as a ***vertex***). Similarly, intersection of any $d + 1$ of them is empty.

A polyhedron defined by a LP with n constraints might has $O\left(n^{\lfloor d/2 \rfloor}\right)$ vertices on its boundary (this is known as the upper-bound theorem [Grü03]). As we argue below, the optimal solution is a vertex. As such a naive algorithm would enumerate all relevant vertices (this is a non-trivial undertaking) and return the best possible vertex. Surprisingly, in low dimension, one can do much better, and get an algorithm with linear running time.

The fact we are interested in the best vertex of the feasible region, while this polyhedron is defined implicitly as the intersection of halfspaces also hints into the quandary that we are in: We are looking for an optimal vertex in a large graph that is defined implicitly. Intuitively, this is why proving the correctness of the algorithms we present here is a non-trivial undertaking (as mentioned before, we will prove correctness in the next chapter).

9.1.1 A solution, and how to verify it

Observe that an optimal solution of a LP is either a vertex or unbounded. Indeed, if the optimal solution p lies in the middle of a segment s, such that s is feasible, then either one of its endpoints provide a better solution (i.e., one of them is lower in the x_d direction than p), or both endpoints of s have the same target value. But then, we can move the solution to one of the endpoints of s. In particular, if the solution lies on a k-dimensional facet F of the boundary of the feasible polyhedron (i.e., formally F is a set with affine dimension k formed by intersection the boundary of the polyhedron by a hyperplane), we can move it so that it lies on a $(k-1)$-dimensional facet F' of the feasible polyhedron, using the proceedings argumentation. Using it repeatedly, one ends up in a vertex of the polyhedron, or in an unbounded solution.

Thus, given an instance of LP, the LP solver should output one of the following answers.

(A) **Finite.** The optimal solution is finite, and the solver would provides a vertex which realizes the optimal solution.

(B) **Unbounded.** The given LP has an unbounded solution. In this case, the LP would output a ray ζ, such that the ζ lies inside the feasible region, and it points downward the negative x_d-axis direction.

(C) **Infeasible.** The given LP does not have any point which comply with all the given inequalities. In this case the solver would output $d+1$ constraints which are infeasible on their own.

Lemma 9.1.1 *Given a set of d linear inequalities in $\mathbb{R}^d$, one can compute the vertex formed by the intersection of their boundaries in $O(d^3)$ time.*

Proof: Write down the system of equalities that the vertex must fulfil. Its a system of d equalities in d variables and it can be solved in $O(d^3)$ time using Gaussian elimination. ■

A ***cone*** is the intersection of d constraints, where its apex is the vertex associated with this set of constraints. A set of such d constraints is a ***basis***. An intersection of $d-1$ of the hyperplanes of a basis form a line and clipping this line to the cone of the basis form a ray. Clipping the same line to the feasible region would yield either a segment, referred to as an ***edge*** of the polytope, or a ray. An edge of the polyhedron connects two vertices of the polyhedron As such, one can think about the boundary of the feasible region as inducing a graph – its vertices are the vertices of the polyhedron, and the edges of the polyhedron. Since every vertex has d hyperplanes defining it (its basis), and an adjacent edge is defined by $d-1$ of these hyperplanes, it follows that each vertex has $\binom{d}{d-1} = d$ edges adjacent to it.

The following lemma tells us when we have an optimal vertex. While it is intuitively clear, its proof requires a systematic understanding of how the feasible region of a linear program looks like, and we delegate it to the next chapter.

Lemma 9.1.2 *Let $\mathcal{L}$ be a given linear program, and let $\mathcal{P}$ denote its feasible region. Let v be a vertex $\mathcal{P}$, such that all the d rays emanating from v are in the upward x_d-axis direction, then v is the lowest (in the x_d-axis direction) point in $\mathcal{P}$ and it is thus the optimal solution to $\mathcal{L}$.*

Interestingly, when we are at vertex of v of the feasible region, it is easy to find the adjacent vertices. Indeed, compute the d rays emanating from v. For such a ray, intersect it with all the constraints of the LP. The closest intersection point along this ray is the vertex u of the feasible region adjacent to v. Doing this naively takes $O(dn + d^{\text{const}})$ time.

Lemma 9.1.2 offers a simple algorithm for computing the optimal solution for an LP. Start from a feasible vertex of the LP. As long as this vertex has at least one ray that points downward, follow this ray to adjacent vertex on the feasible polytope that is lower than the current vertex (i.e., compute the d rays emanating from the current vertex, and follow one of the rays that points downward, till you hit a new vertex). Repeat this till the current vertex has all rays pointing upward, by Lemma 9.1.2 this is the optimal solution. Up to tedious (and non-trivial) details this is the ***simplex*** algorithm.

We need also the following lemma, which its proof is delegated to the next chapter.

Lemma 9.1.3 *If $\mathcal{L}$ is a LP in d dimensions which is not feasible, then there exists $d+1$ inequalities in $\mathcal{L}$ which are infeasible on their own.*

Note, that given a set of $d+1$ inequalities, its easy to verify if it feasible or not. Indeed, compute the $\binom{d+1}{d}$ vertices formed by this set of constraints, and check whether any of this vertices are feasible. If all of them are infeasible, then this set of constraints is infeasible.

9.2 Low Dimensional Linear Programming

9.2.1 An algorithm for a restricted case

There are a lot of tedious details that one has to take care of to make things work with linear programming. As such, we will first describe the algorithm for a special case, and then provide the envelope required so that one can use it to solve the general case.

We remind the reader that the input to the algorithm is the LP $\mathcal{L}$ which is defined by a set of n linear inequalities in $\mathbb{R}^d$. We are looking for the lowest point in $\mathbb{R}^d$ which is feasible for $\mathcal{L}$.

A vertex v is ***acceptable*** if all the d rays associated with it points upward (note, that the vertex itself might not be feasible). The optimal solution (if it is finite) must be located at an acceptable vertex. Assume that we are given the basis $B = \{h_1, \ldots, h_d\}$ of such an acceptable vertex. Let $h_{d+1}, \ldots, h_m$ be a random permutation of the remaining constraints of the LP $\mathcal{L}$.

Our algorithm is randomized incremental. At the ith step, for $i \geq d$, it would maintain he optimal solution for the first i constraints. As such, in the ith step, the algorithm checks whether the optimal solution v_{i-1} of the previous iteration is still feasible with the new constraint h_i (namely, the algorithm checks if v_i is inside the halfspace defined by h_i). If v_{i-1} is still feasible, then it is still the optimal solution, and we set $\mathsf{v}_i \leftarrow \mathsf{v}_{i-1}$.

The more interesting case, is when $\mathsf{v}_{i-1} \notin h_i$. First, we check if the basis of v_{i-1} together with h_i form a set of constraints which is infeasible. If so, the given LP is infeasible, and we output $B(\mathsf{v}_{i-1}) \cup \{h_i\}$ as our proof of infeasibility.

Otherwise, the new optimal solution must lie on the hyperplane associated with h_i. As such, we recursively compute the lowest vertex in the $(d-1)$-dimensional polyhedron $(\partial h_i) \cap \bigcap_{j=1}^{i-1} h_j$. This is a linear program involving $i-1$ constraints, and it involves $d-1$ variables since it lies on the $(d-1)$-dimensional hyperplane ∂h_i. The solution found v_i is defined by a basis of $d-1$ constraints, and adding h_i to it, results in an acceptable vertex that is feasible, and we continue to the next iteration.

Clearly, the vertex v_n is the required optimal solution.

9.2.1.1 Running time analysis

Checking if a set of $d+1$ constraints is infeasible takes $O\left(d^4\right)$ time. The bad case for us, is when the vertex v_i is recomputed in the ith iteration. But this happens only if h_i is one of the d constraints in the basis of v_i. Since there are most d constraints that define the base, and there are at least $i-d$ constraints that are being randomly ordered (as the first d slots are fixed), we have that the probability that $\mathsf{v}_i \neq \mathsf{v}_{i1-}$ is

$$\alpha_i \leq \min\left(\frac{d}{i-d},\ 1\right) \leq \frac{2d}{i},$$

for $i \geq d+1$, as can be easily verified.[3] So, let $T(m,d)$ be the expected time to solve an LP with n constraints in d dimensions, we have

$$T(m,d) \leq O\left(md^3\right) + \sum_{i=d+1}^{m} \alpha_i(di + T(i,d-1)) = O\left(md^3\right) + \sum_{i=d+1}^{m} \frac{2d}{i} T(i,d-1).$$

Guessing that $T(n,d) \leq c_d n$, we have that

$$T(m,d) \leq c_1 md^3 + \sum_{i=d+1}^{m} \frac{2d}{i} c_{d-1} i = \left(c_1 d^3 + 2dc_{d-1}\right) m,$$

where c_1 is some absolute constant. We need that

$$c_1 d^3 + 2c_{d-1} d \leq c_d,$$

Which holds for $c_d = O\left((3d)^d\right)$, and $T(m,d) = O\left((3d)^d m\right)$.

Lemma 9.2.1 *Given an LP with n constraints in d dimensions, and an acceptable vertex for this LP, then can compute the optimal solution in expected $O\left((3d)^d n\right)$ time.*

9.2.2 The algorithm for the general case

Let $\mathcal{L}$ be the given LP, and let $\widehat{\mathcal{L}}$ be the instance formed by translating all the constraints so that they pass through the origin. Next, let h be the hyperplane $x_d = -1$. Consider a solution to the LP $\widehat{\mathcal{L}}$ when restricted to h. This is a $(d-1)$-dimensional instance of linear programming, and it can be solved recursively.

If the recursive call on $\widehat{\mathcal{L}} \cap h$ returned no solution, then the d constraints that prove that the LP $\widehat{\mathcal{L}}$ is infeasible on h, corresponds to a basis in $\mathcal{L}$ of a vertex which is acceptable. Indeed, as we move these d constraints to the origin, their intersection is empty with h (i.e., the "quadrant" that their intersection forms is unbounded only in the upward direction). As such, we can now apply the algorithm of Lemma 9.2.1 to solve the given LP.

If there is a solution to $\widehat{\mathcal{L}} \cap h$, then it is a vertex v on h which is feasible. Thus, consider the original set of $d-1$ constraints in $\mathcal{L}$ that corresponds to the basis B of v. Let ℓ be the line formed by the intersection of the hyperplanes of B. Its now easy to verify that the intersection of the feasible region with this line is an unbounded ray, and the algorithm returns this unbounded (downward oriented) ray, as a proof that the LP is unbounded.

Theorem 9.2.2 *Given a LP with n constraints defined over d variables, it can be solved in expected $O\left((3d)^d m\right)$ time.*

[3] Indeed, $\frac{(d)+d}{(i-d)+d}$ lies between $\frac{d}{i-d}$ and $\frac{d}{d} = 1$.

Proof: The expected running time is

$$S(m,d) = O(md) + S(m,d-1) + T(m,d),$$

where $T(m,d)$ is the time to solve a LP in the restricted case of Section 9.2.1. The solution to this recurrence is $O\left((3d)^d m\right)$, see Lemma 9.2.1. ■

9.3 Linear Programming with Violations

Let $\mathcal{L}$ be a linear program with d variables, and $k > 0$ be a parameter. We are interested in the optimal solution of $\mathcal{L}$ if we are allowed to throw away k constraints. A naive solution would be to try and throw away all possible subsets of k constraints, solve each one of these instances and return the best solution found. This would require $O\left(n^{k+1}\right)$ time. Luckily, it turns out that one can do much better if the dimension is small enough.

The idea is the following: The vertex realizing the optimal k-violated solution is a vertex v defined by d constraints (let it basis be B), and is of depth k. We remind the reader that a point p has ***depth*** k (in $\mathcal{L}$), if it is outside k halfspaces of $\mathcal{L}$ (namely, we complement each constraint of $\mathcal{L}$, and p is contained inside k of these complemented hyperplanes) . As such, we can use the depth estimation technique we encountered before, see Chapter 7. Specifically, if we pick each constraint of $\mathcal{L}$ into a new instance of LP with probability $1/k$, then the probability the new instance $\widehat{\mathcal{L}}$ would have all the elements of B in its random sample, and will not contain any of the k constraints opposing v is

$$\alpha = \left(\frac{1}{k}\right)^{|B|} \left(1 - \frac{1}{k}\right)^{\mathrm{depth}(\mathsf{p})} \geq \frac{1}{k^d}\left(1 - \frac{1}{k}\right)^k \geq \frac{1}{k^d}\exp\left(-\frac{2}{k}k\right) \geq \frac{1}{8k^d},$$

since $1 - x \geq e^{-2x}$, for $0 < x < 1/2$. If this happens then the optimal solution for $\widehat{\mathcal{L}}$ is v. This can be verified by computing how many constraints of $\mathcal{L}$ the optimal solution of $\widehat{\mathcal{L}}$ violates. If it violates more than k constraints we ignore it. Otherwise, we return this as our candidate solution.

Next, we amplify the probability of success by repeating this process $M = 8k^d \ln(1/\delta)$ times, returning the best solution found. The probability that in all these (independent) iterations we had failed to generate the optimal (violated) solution is at most

$$(1-\alpha)^M \leq \left(1 - \frac{1}{8k^d}\right)^M \leq \exp\left(-\frac{M}{8k^d}\right) = \exp\left(-\ln\left(\frac{1}{\delta}\right)\right) = \delta.$$

Theorem 9.3.1 *Let $\mathcal{L}$ be a linear program with n constraints over d variables, let $k > 0$ be a parameter, and $\delta > 0$ a confidence parameter. Then one can compute the optimal solution to $\mathcal{L}$ violating at most k constraints of $\mathcal{L}$, in $O(mk^d \log(1/\delta))$ time. The solution returned is correct with probability $\geq 1 - \delta$.*

9.4 Approximate Linear Programming with Violations

The magic of Theorem 9.3.1 is that it provides us with a linear programming solver which is robust and can handle a small number of factious constraints. But what happens if the number of violated constraints k is large?[2] As a concrete example, for $k = \sqrt{n}$ and a LP with n constraints (defined over d variables) the algorithm for computing optimal solution violating k constraints has running time roughly $O(n^{1+d/2})$. In this case, if one still wants a near linear running time, one can use random sampling to get approximate solution in near linear time.

Lemma 9.4.1 *Let $\mathcal{L}$ be a linear program with n constraints over d variables, let $k > 0$ and $\varepsilon > 0$ be parameters. Then one can compute a solution to $\mathcal{L}$ violating at most $(1+\varepsilon)k$ constraints of $\mathcal{L}$ such that its value is better than the optimal solution violating k constraints of $\mathcal{L}$. The expected running time of the algorithm is*

$$O\left(n + n\min\left(\frac{\log^{d+1} n}{\varepsilon^{2d}}\ ,\ \frac{\log^{d+2} n}{k\,\varepsilon^{2d+2}}\right)\right).$$

The algorithm succeeds with high probability.

Proof: Let $\rho = O\left(\frac{d}{k\varepsilon^2}\ln n\right)$ and pick each constraints of $\mathcal{L}$ into $\widehat{\mathcal{L}}$ with probability ρ. Next, the algorithm computes optimal solution u in $\widehat{\mathcal{L}}$ violating

$$k' = (1 + \varepsilon/3)\rho k$$

'constraints, and return this as the required solution.

[2] I am sure the reader guessed correctly the consequences of such a despicable scenario: The universe collapses and is replaced by a cucumber.

We need to prove the correctness of this algorithm. To this end, the reliable sampling lemma (Lemma 8.3.1) states that for any vertex v of depth u in $\mathcal{L}$, has depth in the range

$$\left[(1-\varepsilon/3)u\rho,\ (1+\varepsilon/3)u\rho\right]$$

in $\widehat{\mathcal{L}}$, and this holds with high probability, where $u \geq k$ (here we are using the fact that there are at most n^d vertices defined by $\mathcal{L}$).

In particular, let $\mathsf{v}_{\mathrm{opt}}$ be the optimal solution for $\mathcal{L}$ of depth k. With high probability, $\mathsf{v}_{\mathrm{opt}}$ has depth $\leq (1+\varepsilon/3)pk = k'$ in $\widehat{\mathcal{L}}$, which implies that the returned solution v is better than $\mathsf{v}_{\mathrm{opt}}$, since v has depth k' in $\widehat{\mathcal{L}}$.

Next, we need to prove that v is not too deep. So, assume that v is of depth β in $\mathcal{L}$. By the reliable sampling lemma, we have that the depth of v in $\widehat{\mathcal{L}}$ is in the range $\left[(1-\varepsilon/3)\beta\rho, (1+\varepsilon/3)\beta\rho\right]$. In particular, we know that $(1-\varepsilon/3)\beta\rho \leq k' = (1+\varepsilon/3)\rho k$. That is

$$\beta \leq \frac{1+\varepsilon/3}{1-\varepsilon/3}k \leq (1+\varepsilon/3)(1+\varepsilon/2)k \leq (1+\varepsilon)k,$$

since $1/(1-\varepsilon/3) \leq 1+\varepsilon/2$ for $\varepsilon \leq 1$[3].

As for the running time, we are using the algorithm of Theorem 9.3.1. The input size is $O(n\rho)$ and the depth threshold is k'. (The bound on the input size holds with high probability. We omit the easy but the tedious proof of that using Chernoff inequality.) As such, the running time is

$$O\Big(n + n\rho(k')^d \log n\Big) = O\Big(n + n\min\Big(n, n\rho\Big)(\rho k)^d \log n\Big) = O\left(n + n\min\left(\frac{\log^{d+1} n}{\varepsilon^{2d}},\ \frac{\log^{d+2} n}{k\,\varepsilon^{2d+2}}\right)\right).$$

■

Note, that the running time of Lemma 9.4.1 is linear if k is sufficiently large and ε is fixed.

9.5 LP-type problems

Interestingly, the above algorithms for linear programming can be extended to more abstract settings. Indeed, assume we are given a set of constraints $\mathcal{H}$, and a function w, such that for any subset $G \subset \mathcal{H}$ returns the value of the optimal solution of the constraint problem when restricted to G. We denote this value by $w(G)$. Our purpose is to compute $w(\mathcal{H})$.

For example, $\mathcal{H}$ is a set of points in $\mathbb{R}^d$, and $w(G)$ is the radius of the smallest ball containing all the points of $F \subseteq \mathcal{H}$. As such, in this case, we would like to compute (the radius of) the smallest enclosing ball for $\mathcal{H}$.

We assume that the following axioms hold:

1. (**Monotonicity.**) For any $F \subseteq G \subseteq \mathcal{H}$, we have
$$w(F) \leq w(G).$$
2. (**Locality.**) For any $F \subseteq G \subseteq \mathcal{H}$, with $-\infty < w(F) = w(G)$, and any $h \in H$, if
$$w(F) < w(F \cup \{h\}) \quad \text{then} \quad w(G) < w(G \cup \{h\}).$$

If these two axioms holds, we refer to $(\mathcal{H}, w)$ as a ***LP-type*** problem. It is easy to verify that linear programming is a LP-type problem.

Definition 9.5.1 A ***basis*** is a subset $B \subseteq \mathcal{H}$ such that $w(B) > -\infty$, and $w(B') < w(B)$, for any proper subset B' of B.

As in linear programming, we have to assume that certain ***basic operations*** can be performed quickly. These operations are:

(A) (**Violation test.**) For a constraint h and a basis B, test whether h is violated by B or not. Namely, test if $w(B \cup \{h\}) > w(B)$.

(B) (**Basis computation.**) For a constraint h, and a basis B, computes the basis of $B \cup \{h\}$.

We also need to assume that we are given an initial basis B_0 from which to start our computation. The ***combinatorial dimension*** of $(\mathcal{H}, w)$ is the maximum size of s basis of $\mathcal{H}$. Its easy to verify that the algorithm we presented for linear programming (the special case of Section 9.2.1) works verbatim in this settings. Indeed, start with B_0, and randomly permute the remaining constraints. Now, add the constraints in a random order, and each step check if the new constraints violates the current solution, and if so, update the basis of the new solution. The recursive call here, corresponds to solving a subproblem where some members of the basis are fixed. We conclude:

Theorem 9.5.2 *Let $(\mathcal{H}, w)$ be a LP-type problem with n constraints with combinatorial dimension d. Assume that the basic operations takes constant time, we have that $(\mathcal{H}, w)$ can be solved using $d^{O(d)}m$ basic operations.*

[3] Indeed $(1-\varepsilon/3)(1+\varepsilon/2) \leq 1-\varepsilon/3+\varepsilon/2-\varepsilon^2/6 \geq 1$.

9.5.1 Examples for LP-type problems

Smallest enclosing ball. Given a set P of n points in $\mathbb{R}^d$, and let $r(\mathsf{P})$ denote the radius of the smallest enclosing ball in $\mathbb{R}^d$. Under general position assumptions, there are at most $d+1$ points on the boundary of this smallest enclosing ball. We claim that the problem is an LP-type problem. Indeed, the basis in this case is the set of points determining the smallest enclosing ball. The combinatorial dimension is thus $d+1$. The monotonicity property holds trivially. As for the locality property, assume that we have a set $\mathsf{Q} \subseteq \mathsf{P}$ such that $r(\mathsf{Q}) = r(\mathsf{P})$. As such, P and Q have the same enclosing ball. Now, if we add a point p to Q and the radius of its minimum enclosing ball increases, then the ball enclosing P must also change (and get bigger) when we insert p to P. Thus, this is a LP-type problem, and it can be solved in linear time.

Theorem 9.5.3 *Given a set P of n points in $\mathbb{R}^d$, one can compute its smallest enclosing ball in (expected) linear time.*

Finding time of first intersection. Let $C(t)$ be a parameterized convex shape in $\mathbb{R}^d$, such that $C(0)$ is empty, and $C(t) \subsetneq C(t')$ if $t < t'$. We are given n such shapes $C_1, \ldots, C_n$, and we would like to decide the minimal t for which they all have a common intersection. Assume, that given a point p and such a shape C, we can decide (in constant time) the minimum t for which $\mathsf{p} \in C(t)$. Similarly, given (say) $d+1$ of these shapes, we can decide in constant time the minimum t for which they intersect, and this common point of intersection. We would like to find the minimum t for which they all intersect. Let also assume that these shapes are well behaved in the sense that, for any t, we have $\lim_{\Delta \to 0} \mathrm{Vol}(C(t+\Delta) \setminus C(t)) = 0$ (namely, such a shape can not "jump" – it grows continuously). It is easy to verify that this is a LP-type problem, and as such it can be solved in linear time.

Note, that this problem is an extension of the previous problem. Indeed, if we group a ball of radius t around each point of P, then the problem of deciding the minimal t when all these growing balls have a non-empty intersection, is equivalent to finding the minimum radius ball enclosing all points.

9.6 Bibliographical notes

History. Linear programming has a rich and fascinating history. It can be traced back to the early 19th century. It started in earnest in 1939 when L. V. Kantorovich noticed the importance of certain type of Linear Programming problems. Unfortunately, for several years, Kantorovich work was unknown in the west and unnoticed in the east.

Dantzig, in 1947, invented the simplex method for solving LP problems for the US Air force planning problems. T. C. Koopmans, in 1947, showed that LP provide the right model for the analysis of classical economic theories. In 1975, both Koopmans and Kantorovich got the Nobel prize of economics. Dantzig probably did not get it because his work was too mathematical. So it goes.

The simplex algorithm was developed before computers (and computer science) really existed in wide usage, and its standard description is via a careful maintenance of a tableau of the LP, which is easy to handle by hand (this might also explain the unfortunate name "linear programming"). This makes however the usual description of the simplex algorithm pretty mysterious and counter-intuitive. Furthermore, since the universe is not in general position (as we assumed), there are numerous technical difficulties (that we glossed over) in implementing any of this algorithms, and the descriptions of the simplex algorithms usually detail how to handle these cases. See the book by Vanderbei [Van97] for an accessible and readable coverage of this topic.

Linear programming in low dimensions. The first to realize that linear programming can be solved in linear time in low dimensions was Megiddo [Meg83, Meg84]. His algorithm was deterministic but considerably more complicated than the randomized algorithm we present. Clarkson [Cla95] showed how to use randomization to get a simple algorithm for linear programming with running time $O(d^2 n + \mathit{noise})$, where the noise is a constant exponential in d. Our presentation follows the paper by Seidel [Sei91]. Surprisingly, one can achieve running time with the noise being subexponential in d. This follows by plugging in the subexponential algorithms of Kalai [Kal92] or Matoušek *et al.* [MSW96] into Clarkson algorithm [Cla95]. The resulting algorithm has expected running time $O\left(d^2 n + \exp\left(c\sqrt{d \log d}\right)\right)$, for some constant c. See the survey by Goldwasser [Gol95] for more details.

More information on Clarkson's algorithm. Clarkson's algorithm contains some interesting new ideas that are worth mentioning shortly. (Matoušek *et al.* [MSW96] algorithm is somewhat similar to the algorithm we presented.)

Observe that if the solution for a random sample R is being violated by a set X of constraints, then X must contains (at least) one constraint which is in the basis of the optimal solution. Thus, by picking R to be of size (roughly) $\sqrt{n}$, we know that it is a $1/\sqrt{n}$-net, and there would be at most $\sqrt{n}$ constraints violating the solution of R. Thus, repeating this d times, at each stage solving the problem on the collected constraints from previous iteration, together with the current random sample, results in a set of $O(d\sqrt{n})$ constraints that contains the optimal basis. Now solve recursively the linear program on this (greatly reduced) set of constraints. Namely, we spent $O(d^2 n)$ time (d times checking if the n constraints violates a given solution), called recursively d times on "small" subproblems of size (roughly) $O(\sqrt{n})$, resulting in a fast algorithm.

An alternative algorithm, uses the same observation, by using the reweighting technique. Here each constraint is sampled according to its weight (which is initially 1. By doubling the weight of the violated constraints, one can argue that after a small number of iterations, the sample would contain the required basis, while being small. See Chapter 17 for more details.

Clarkson algorithm works by combining these two algorithms together.

Linear programming with violations. The algorithm of Section 9.3 seems to be new, although it is implicit in the work of Matoušek [Mat95b], which present a slightly faster deterministic algorithm. The first paper on this problem (in two dimensions), is due to Everett *et al.* [ERvK96]. This was extended by Matoušek to higher dimensions [Mat95b]. His algorithm relies on the idea of computing all $O(k^d)$ local maximas in the "k-level" explicitly, by traveling between them. This is done by solving linear programming instances which are "similar". As such, these results can be further improved using techniques for dynamic linear programming that allows insertion and deletions of constraints, see the work by Chan [Cha96]. Chan [Cha05] showed how to further improve these algorithms for dimensions 2, 3 and 4, although these improvements disappear if k is close to linear.

The idea of approximate linear programming with violations is due to Aronov and Har-Peled [AH05], and our presentation follows their results. Using more advanced data-structures these results can be further improved (as far as the polylog noise is concerned), see the work by Afshani and Chan [AC07].

LP-type problems. The notion of LP-type algorithm is mentioned in the work of Sharir and Welzl [SW92]. They also showed up that deciding if a set of (axis parallel) rectangles can be pierced by 3-points is a LP-type problem (quite surprising as the problem has no convex programming flavor). Our example of computing first intersection of growing convex sets, is motivated by the work of Amenta [Ame94] on the connection between LP-type problems and Helly-type theorems.

Intuitively, any lower dimensional convex programming problem is a natura'l candidate to be solved using LP-type techniques.

9.7 Exercises

Chapter 10

Polyhedrons, Polytopes and Linear Programming

> I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.
>
> – – Three men in a boat, Jerome K. Jerome

In this chapter, we formally investigate how the feasible region of a linear program looks like, and establish the correctness of the algorithm we presented for linear programming. Linear programming seems to be one case where the geometric intuition is quite clear, but crystallizing it into a formal proof requires quite a bit of work. In particular, we prove in this chapter more than we strictly need, since it support (and may we dare suggesting, with due humble and respect to the most esteemed reader, that it even expands[4]) our natural intuition.

Underlining our discussion is the dichotomy between the input to LP, which is a set of halfspaces, and the entities LP works with, which are vertices. In particular, we need to establish that speaking about the feasible region of a LP in terms of (convex hull of) vertices, or alternatively, as the intersection of halfspaces is equivalent.

10.1 Preliminaries

We had already encountered Radon's theorem, which we restate.

Theorem 10.1.1 (Radon's Theorem.) *Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_{d+2}\}$ be a set of $d+2$ points in $\mathbb{R}^d$. Then, there exists two disjoint subsets Q and R of P, such that $\mathcal{CH}(\mathsf{Q}) \cap \mathcal{CH}(\mathsf{R}) \neq \emptyset$.*

Theorem 10.1.2 (Helly's theorem.) *Let $\mathcal{F}$ be a set of n convex sets in $\mathbb{R}^d$. The intersection of all the sets of $\mathcal{F}$ is non-empty if and only if any $d+1$ of them has non-empty intersection.*

Proof: This theorem is "dual" to Radon's theorem.

If the intersection of all sets in $\mathcal{F}$ is non-empty then any intersection of $d+1$ of them is non-empty. As for the other direction, assume for the sake of contradiction, that $\mathcal{F}$ is the minimal set of convex sets for which the claim fails. Namely, for $m = |\mathcal{F}| > d+1$, any subset of $m-1$ sets of $\mathcal{F}$ has non-empty intersection, and yet the intersection of all the sets of $\mathcal{F}$ is empty.

As such, for $X \in \mathcal{F}$, let p_X be a point in the intersection of all sets of $\mathcal{F}$ excluding X. Let $\mathsf{P} = \left\{ \mathsf{p}_X \;\middle|\; X \in \mathcal{F} \right\}$. Here $|\mathsf{P}| = |\mathcal{F}| > d+1$. By Radon's theorem, there is a partition of P into two disjoint sets R and Q such that $\mathcal{CH}(\mathsf{R}) \cap \mathcal{CH}(\mathsf{Q}) \neq \emptyset$. Let r be any point inside this non-empty intersection.

Let $U(\mathsf{R}) = \left\{ X \;\middle|\; \mathsf{p}_X \in \mathsf{R} \right\}$ and $U(\mathsf{Q}) = \left\{ X \;\middle|\; \mathsf{p}_X \in \mathsf{Q} \right\}$ be the two subsets of $\mathcal{F}$ corresponding to R and Q, respectively. By definition, we have that, for $X \in U(\mathsf{R})$, it holds

$$\mathsf{p}_X \in \bigcap_{Y \in \mathcal{F}, Y \neq X} Y \subseteq \bigcap_{Y \in \mathcal{F} \setminus U(\mathsf{R})} Y = \bigcap_{Y \in U(\mathsf{Q})} Y,$$

[4] Hopefully the reader is happy we are less polite to him/her in the rest of the book, since otherwise the text would be insufferably tedious.

since $U(\mathsf{Q}) \cup U(\mathsf{R}) = \mathcal{F}$. As such, $\mathsf{R} \subseteq \bigcap_{Y \in U(\mathsf{Q})} Y$ and $\mathsf{Q} \subseteq \bigcap_{Y \in U(\mathsf{R})} Y$. Now, by the convexity of the sets of $\mathcal{F}$, we have $\mathcal{CH}(\mathsf{R}) \subseteq \bigcap_{Y \in U(\mathsf{Q})} Y$ and $\mathcal{CH}(\mathsf{Q}) \subseteq \bigcap_{Y \in U(\mathsf{R})} Y$. Namely, we have

$$\mathsf{r} \in \mathcal{CH}(\mathsf{R}) \cap \mathcal{CH}(\mathsf{Q}) \subseteq \left(\bigcap_{Y \in U(\mathsf{Q})} Y \right) \cap \left(\bigcap_{Y \in U(\mathsf{R})} Y \right) = \bigcap_{Y \in \mathcal{F}} Y.$$

Namely, the intersection of all the sets of $\mathcal{F}$ is not empty. A contradiction. ■

Theorem 10.1.3 (Carathéodory theorem.) *Let X be a convex set in $\mathbb{R}^d$, and let p be some point in the interior of X. Then p is a convex combination of $d+1$ points of X.*

Proof: Suppose $\mathsf{p} = \sum_{k=1}^{m} \lambda_i x_i$ is a convex combination of $m > d+1$ points, where $\{x_1, \ldots, x_m\} \subseteq X$, $\lambda_1, \ldots, \lambda_m > 0$ and $\sum_i \lambda_i = 1$. We will show that p can be rewritten as a convex combination of $m-1$ of these points, as long as $m > d+1$.

So, consider the following system of equations

$$\sum_{i=1}^{m} \gamma_i x_i = 0 \quad \text{and} \quad \sum_{i=1}^{m} \gamma_i = 0. \tag{10.1}$$

It has $m > d+1$ variables (i.e., $\gamma_1, \ldots, \gamma_m$) but only $d+1$ equations. As such, there is a non-trivial solution to this system of equations, and denote it by $\widehat{\gamma}_1, \ldots, \widehat{\gamma}_m$. Since $\widehat{\gamma}_1 + \cdots + \widehat{\gamma}_m = 0$, some of $\widehat{\gamma}_i$s are strictly positive, and some of them are strictly negative. Let

$$\tau = \min_{j,\, \widehat{\gamma}_j > 0} \frac{\lambda_j}{\widehat{\gamma}_j} > 0.$$

And assume without loss of generality, that $\tau = \lambda_1 / \widehat{\gamma}_1$. Let

$$\widetilde{\lambda}_i = \lambda_i - \tau \widehat{\gamma}_i,$$

for $i = 1, \ldots, m$. Then $\widetilde{\lambda}_1 = \lambda_i - (\lambda_1 / \widehat{\gamma}_1)\, \widehat{\gamma}_i = 0$. Furthermore, if $\widehat{\gamma}_i < 0$, then $\widetilde{\lambda}_i = \lambda_i - \tau \widehat{\gamma}_i \geq \lambda_i > 0$. Otherwise, if $\widehat{\gamma}_i > 0$, then

$$\widetilde{\lambda}_i = \lambda_i - \left(\min_{j,\, \widehat{\gamma}_j > 0} \frac{\lambda_j}{\widehat{\gamma}_j} \right) \widehat{\gamma}_i \geq \lambda_i - \frac{\lambda_i}{\widehat{\gamma}_i} \widehat{\gamma}_i \geq 0.$$

So, $\widetilde{\lambda}_1 = 0$ and $\widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_m \geq 0$. Furthermore,

$$\sum_{i=1}^{m} \widetilde{\lambda}_i = \sum_{i=1}^{m} (\lambda_i - \tau \widehat{\gamma}_i) = \sum_{i=1}^{m} \lambda_i - \tau \sum_{i=1}^{m} \widehat{\gamma}_i = \sum_{i=1}^{m} \lambda_i = 1,$$

since $\widehat{\gamma}_1, \ldots, \widehat{\gamma}_m$ is a solution to Eq. (10.1). As such, $\mathsf{q} = \sum_{i=2}^{m} \widetilde{\lambda}_i x_i$ is a convex combination of $m-1$ points of X. Furthermore, as $\widetilde{\lambda}_1 = 0$, we have

$$\mathsf{q} = \sum_{i=2}^{m} \widetilde{\lambda}_i x_i = \sum_{i=1}^{m} \widetilde{\lambda}_i x_i = \sum_{i=1}^{m} (\lambda_i - \tau \widehat{\gamma}_i)\, x_i = \sum_{i=1}^{m} \lambda_i x_i - \tau \sum_{i=1}^{m} \widehat{\gamma}_i x_i = \sum_{i=1}^{m} \lambda_i x_i - \tau\, 0 = \sum_{i=1}^{m} \lambda_i x_i = \mathsf{p},$$

since (again) $\widehat{\gamma}_1, \ldots, \widehat{\gamma}_m$ is a solution to Eq. (10.1). As such, we found a representation of p as a convex combination of $m-1$ points, and we can continue in this process till $m = d+1$, establishing the theorem. ■

10.1.1 Properties of polyhedrons

A ***$\mathcal{H}$-polyhedron*** (or just polyhedron) is the region formed by the intersection of (finite number of) halfspaces. Namely, its the feasible region of a LP.

It would be convenient to consider the LP to be specified in a matrix form. Namely, we are given matrix $\mathsf{M} \in \mathbb{R}^{m \times d}$ with m rows and d columns, and a m dimensional vector b, and a d dimensional c. Our purpose, is to find a $x \in \mathbb{R}^d$, such that $\mathsf{M}x \leq \mathsf{b}$, while $\mathsf{c} \cdot x$ is minimized. Namely, the ith row of M corresponds to the ith inequality of the LP, that is the inequality $\mathsf{M}_i x \leq \mathsf{b}_i$, for $i = 1, \ldots, m$, where M_i denotes the ith row of M.

In this form it is easy to combine inequalities together: You multiply several rows by positive constants, and add them up (you also need to do this for the relevant constants in b), and get a new inequality. Formally, given a row vector $\mathsf{w} \in \mathbb{R}^m$, such that $\mathsf{w} \geq 0$ (i.e., all entries are non-negative), the resulting inequality is $\mathsf{w}\mathsf{M} \leq \mathsf{w} \cdot \mathsf{c}$. Note that such an inequality *must* hold inside the feasible region of the original LP.

Fourier-Motzkin elimination. Let $L = (\mathsf{M}, \mathsf{b})$ be an instance of LP (we care only about feasibility here, so we ignore the target function). Consider the ith variable x_i in the LP L. If it appears only with positive coefficient in all the inequalities (i.e., the ith column of M has only positive numbers) then we can set the ith variable to be sufficiently large negative number, and all the inequalities where it appear would be immediately feasible. Same holds if all such coefficients are negative. Thus, consider the case where the variable x_i appears with both negative coefficients and positive coefficients in the LP. Let us inspect two such inequalities, say the kth and lth inequality, and assume, for the sake of concreteness, that $\mathsf{M}_{ki} > 0$ and $\mathsf{M}_{li} < 0$. Clearly, we can multiply the lth inequality by a positive number and add it to the kth inequality, so that in the resulting inequality the coefficient of x_i is zero.

Let $L' = \mathrm{elim}(L, i)$ denote the resulting linear program, where we copy all inequalities of the original LP where x_i has zero as coefficient. In addition, we add all the inequalities that can formed by taking "positive" and "negative" appearances in the original LP of x_i and canceling them out as described above. Note, that L' might have $m^2/4$ inequalities, but since x_i is now eliminated (i.e., all of its appearances are with coefficient zero), the LP L' is defined over $d - 1$ variables.

Lemma 10.1.4 *Let L be an instance of LP with d variables and m inequalities, the linear program $L' = \mathrm{elim}(L, i)$ is feasible if and only if L is feasible, for any $i \in \{1, \ldots, d\}$.*

Proof: One direction is easy, if L is feasible then its solution (omitting the ith variable) is feasible for L'.

The other direction, becomes clear once we understand what the elimination really do. So, consider two inequalities in L, such that $\mathsf{M}_{ki} < 0$ and $\mathsf{M}_{ji} > 0$. We can rewrite these inequalities such that they become

$$a_0 + \sum_{\tau \neq i} a_\tau x_\tau \leq x_i \tag{A}$$

$$\text{and} \quad x_i \leq b_0 + \sum_{\tau \neq i} b_\tau x_\tau, \tag{B}$$

respectively. The eliminated inequality described above, is no more than the inequality we get by chaining these inequalities together; that is

$$a_0 + \sum_{\tau \neq i} a_\tau x_\tau \leq x_i \leq b_0 + \sum_{\tau \neq i} b_\tau x_\tau \quad \Rightarrow \quad a_0 + \sum_{\tau \neq i} a_\tau x_\tau \leq b_0 + \sum_{\tau \neq i} b_\tau x_\tau.$$

In particular, for a feasible solution to L', all the left sides if inequalities of type (A) must be smaller (equal) than the right side of all the inequalities of type (B), since we combined all such pairs of inequalities into an inequality in L'.

In particular, given a feasible solution sol to L', one can extend it into a solution of L by computing a value of x_i, such that all the original inequalities hold. Indeed, every pair of inequalities as above of L, when we substitute the values of $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d$ in sol into L, results in an interval $\mathcal{I}$, such that x_i must lie inside $\mathcal{I}$. Each inequality of type (A) induces a left endpoint of such an interval, and each inequality of type (B) induced a right endpoint of such an interval. We create all possible intervals of this type (using these endpoints) when creating L', and as such for sol, all these intervals must be non-empty. We conclude that the intersection of all these intervals is non-empty, implying that one can pick a value to x_i such that L is feasible. ■

Given a $\mathcal{H}$-polyhedron $\mathcal{P}$, the elimination of the ith variable $\mathrm{elim}(\mathcal{P}, i)$ can be interpreted as projecting the polyhedron $\mathcal{P}$ into the hyperplane $x_i = 0$. Furthermore, the projected polyhedron is still a $\mathcal{H}$-polyhedron. By change of variables, this implies that any projection a $\mathcal{H}$-polyhedron into a hyperplane, is a $\mathcal{H}$-polyhedron. We can repeat this process of projecting down the polyhedron several times, which implies the following lemma.

Lemma 10.1.5 *The projection of a $\mathcal{H}$-polyhedron into any affine subspace is a $\mathcal{H}$-polyhedron.*

Lemma 10.1.6 (Farakas Lemma) *Let $\mathsf{M} \in \mathbb{R}^{m \times d}$ and $\mathsf{b} \in \mathbb{R}^m$ specify a LP. Then either:*

(i) There exists a feasible solution $x \in \mathbb{R}^d$ to the LP. That is $\mathsf{M}x \leq \mathsf{b}$,

(ii) Or, there is no feasible solution, and we can prove it by combining the inequalities of the LP into an infeasible inequality. Formally, there exists a vector $\mathsf{w} \in \mathbb{R}^m$, such that $\mathsf{w} \geq 0$, $\mathsf{wM} = 0$ and $\mathsf{w} \cdot \mathsf{b} < 0$. Namely, the inequality (that must hold if the LP is feasible)

$$(\mathsf{wM})\, x \leq \mathsf{w} \cdot \mathsf{b}$$

is infeasible.

Proof: Clearly, the two options can not hold together, so all we need to show is that if an LP is infeasible than the second option holds.

Observe, that if we apply a sequence of eliminations to an LP, all the resulting inequalities in the new LP are positive combination of the original inequalities. So, let us apply this process of elimination to each one of the variables in turn. If the original LP is not feasible, then sooner or later, this elimination process would get stuck, as it will generate an infeasible inequality. Such an inequality would have all the variables with zero coefficients, and the constant would be negative. Thus establishing the claim. ■

Note, that the above elimination process provides us with a naive algorithm for solving LP. This algorithm is extremely inefficient, as in the last elimination stage, we might end up with m^{2^d} inequalities. This would lead to an algorithm which is unacceptably slow for solving LP.

We remind the reader that a linear equality of the form $\sum_i a_i x_i = c$ can be rewritten as the two inequalities $\sum_i a_i x_i \le c$ and $\sum_i a_i x_i \ge c$. The great success of the Farakas Lemma in the market place had lead to several sequels to it, and here is one of them.[2]

Lemma 10.1.7 (Farakas Lemma II) *Let $\mathsf{M} \in \mathbb{R}^{m\times d}$ and $\mathsf{b} \in \mathbb{R}^m$, and consider the following linear program $\mathsf{M}x = \mathsf{b}$, $x \ge 0$. Then, either (i) this linear program is feasible, or (ii) there is a vector $\mathsf{w} \in \mathbb{R}^m$ such that $\mathsf{wM} \ge 0$ and $\mathsf{w} \cdot \mathsf{b} < 0$.*

Proof: If (ii) holds, then consider a feasible solution x to the LP. We have that $\mathsf{M}x = \mathsf{b}$, multiplying this equality from the left by w, we have that

$$(\mathsf{wM})x = \mathsf{w} \cdot \mathsf{b}.$$

But (ii) claims that the quantity on the left is non-negative (since $x \ge 0$), and the quantity on the right is negative. A contradiction. As such these two options are mutually exclusive.

The linear program $\mathsf{M}x = \mathsf{b}$, $x \ge 0$ can be rewritten as $\mathsf{M}x \le \mathsf{b}$, $\mathsf{M}x \ge \mathsf{b}$, $x \ge 0$, which in turn is equivalent to the LP:

$$\begin{array}{c} \mathsf{M}x \le \mathsf{b} \\ -\mathsf{M}x \le -\mathsf{b} \\ -x \le 0 \end{array} \quad \Longleftrightarrow \quad \begin{pmatrix} \mathsf{M} \\ -\mathsf{M} \\ -I_d \end{pmatrix} x \le \begin{pmatrix} \mathsf{b} \\ -\mathsf{b} \\ 0 \end{pmatrix},$$

where I_d is the $d \times d$ identity matrix. Now, if the original LP does not have a feasible solution, then by the original Farakas lemma (Lemma 10.1.6), there must be a vector $(\mathsf{w}_1, \mathsf{w}_2, \mathsf{w}_3) \ge 0$ such that

$$(\mathsf{w}_1, \mathsf{w}_2, \mathsf{w}_3)\begin{pmatrix} \mathsf{M} \\ -\mathsf{M} \\ -I_d \end{pmatrix} = 0 \quad \text{and} \quad (\mathsf{w}_1, \mathsf{w}_2, \mathsf{w}_3) \cdot \begin{pmatrix} \mathsf{b} \\ -\mathsf{b} \\ 0 \end{pmatrix} < 0.$$

Namely, $(\mathsf{w}_1 - \mathsf{w}_2)\,\mathsf{M} - \mathsf{w}_3 = 0$ and $(\mathsf{w}_1 - \mathsf{w}_2) \cdot \mathsf{b} < 0$. But $\mathsf{w}_3 \ge 0$, which implies that

$$(\mathsf{w}_1 - \mathsf{w}_2)\,\mathsf{M} = \mathsf{w}_3 \ge 0.$$

Namely, the claim holds for $\mathsf{w} = \mathsf{w}_1 - \mathsf{w}_2$. ■

Cones and vertices. For a set of vectors $\mathcal{V} \subseteq \mathbb{R}^d$, let $\mathrm{cone}(\mathcal{V})$ denote the ***cone*** they generate. Formally,

$$\mathrm{cone}(\mathcal{V}) = \left\{ \sum_i t_i \overrightarrow{\mathsf{v}}_i \;\middle|\; \overrightarrow{\mathsf{v}}_i \in \mathcal{V}, t_i \ge 0 \right\}.$$

A halfspace ***passes*** through a point p, if p is contained in the hyperplane bounding this halfspace. Since 0 is the apex of $\mathrm{cone}(\mathcal{V})$, it is natural to presume that $\mathrm{cone}(\mathcal{V})$ can be generated by a finite intersection of halfspaces, all of them passing through the origin (which is indeed true).

In the following, let $\mathsf{e}(i, d)$ denote the ith orthonormal vector in $\mathbb{R}^d$; namely,

$$\mathsf{e}(i,d) = (\overbrace{0,\ldots,0}^{i-1 \text{ coords}}, 1, \overbrace{0,\ldots,0}^{d-i \text{ coords}}).$$

Lemma 10.1.8 *Let $\mathsf{M} \in \mathbb{R}^{m\times d}$ be a given matrix, and consider the $\mathcal{H}$-polyhedron $\mathcal{P}$ formed by all points $(x, w) \in \mathbb{R}^{d+m}$, such that $\mathsf{M}x \le w$. Then $\mathcal{P} = \mathrm{cone}(\mathcal{V})$, where $\mathcal{V}$ is a finite set of vectors in $\mathbb{R}^{d+m}$.*

Proof: Let $E_i = \mathsf{e}(i, d+m)$, for $i = d+1, \ldots, d+m$. Clearly, the inequality $\mathsf{M}x \le w$ trivially holds if $(x, w) = E_i$, for $i = d+1, \ldots, d+m$ (since $x = 0$ in such a case). Also, let

$$\overrightarrow{\mathsf{v}}_i = \Big(\mathsf{e}(i,d), \mathsf{M}\,\mathsf{e}(i,d)\Big),$$

for $i = 1, \ldots, d$. Clearly, the inequality holds for $\overrightarrow{\mathsf{v}}_i$, since trivially, $\mathsf{M}\,\mathsf{e}(i,d) \le \mathsf{M}\,\mathsf{e}(i,d)$, for $i = 1, \ldots, d$. Let $\mathcal{V} = \left\{\overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_d, E_{d+1}, \ldots, E_{d+m}\right\}$.

We claim that $\mathrm{cone}(\mathcal{V}) = \mathcal{P}$. One direction is easy, as the inequality holds for all the vectors in $\mathcal{V}$, it also holds for any positive combination of these vectors, implying that $\mathrm{cone}(\mathcal{V}) \subseteq \mathcal{P}$. The other direction is slightly more challenging. Consider an $(x, w) \in \mathbb{R}^{d+m}$ such that $\mathsf{M}x \le w$. Clearly, $w - \mathsf{M}x \ge 0$. As such, $(x, w) = (x, \mathsf{M}x) + (0, w - \mathsf{M}x)$. Now, $(x, \mathsf{M}x) \in \mathrm{cone}\Big(\left\{\overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_d\right\}\Big)$ and since $w - \mathsf{M}x \ge 0$, we have $(0, w - \mathsf{M}x) \in \mathrm{cone}(E_{d+1}, \ldots, E_{d+m})$. Thus, $\mathcal{P} \subseteq \mathrm{cone}\Big(\left\{\overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_d\right\}\Big) + \mathrm{cone}(E_{d+1}, \ldots, E_{d+m}) = \mathrm{cone}(\mathcal{V})$. ■

We need the following simple pairing lemma, which we leave as an exercise for the reader (see Exercise 10.5.1).

[2] As in most sequels, Farakas Lemma II is equivalent to the original Farakas Lemma. I only hope the reader does not feel cheated.

Lemma 10.1.9 *Let $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m$ be positive numbers, such that $\sum_{i=1}^{n} \alpha_i = \sum_{j=1}^{m} \beta_j$, and consider the positive combination $\overrightarrow{\mathsf{w}} = \sum_{i=1}^{n} \alpha_i \overrightarrow{\mathsf{v}}_i + \sum_{i=j}^{m} \beta_j \overrightarrow{\mathsf{u}}_j$, where $\overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_n, \overrightarrow{\mathsf{u}}_1, \ldots, \overrightarrow{\mathsf{u}}_m$ are vectors (say, in $\mathbb{R}^d$). Then, there are non-negative $\delta_{i,j}$s, such that $\overrightarrow{\mathsf{w}} = \sum_{i,j} \delta_{i,j} \left(\overrightarrow{\mathsf{v}}_i + \overrightarrow{\mathsf{u}}_j \right)$.*

Lemma 10.1.10 *Let $C = \operatorname{cone}(\mathcal{V})$ be a cone generate by a set of vectors $\mathcal{V}$ in $\mathbb{R}^d$. Consider the region $\mathcal{P} = C \cap h$, where h is a hyperplane that passes through the origin. Then, $\mathcal{P}$ is a cone; namely, there exists a set of vectors $\mathcal{V}'$ such that $\mathcal{P} = \operatorname{cone}(\mathcal{V}')$.*

Proof: By rigid rotation of the axis system, we can assume that $h \equiv x_1 = 0$. Furthermore, by scaling, we can assume that the first coordinate of all points in $\mathcal{V}$ is either $-1, 0$ or 1 (clearly, scaling of vectors generating the cone does not effect the cone itself). Let $\mathcal{V} = \mathcal{V}_{-1} \cup \mathcal{V}_0 \cup \mathcal{V}_1$, where $\mathcal{V}_0$ is the set of vectors in $\mathcal{V}$ with the first coordinate being zero, and $\mathcal{V}_{-1}$ (resp. $\mathcal{V}_1$) are the vectors in $\mathcal{V}$ with the first coordinate being -1 (resp. 1).

Let $\mathcal{V}_{-1} = \left\{ \overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_n \right\}$, $\mathcal{V}_1 = \left\{ \overrightarrow{\mathsf{u}}_1, \ldots, \overrightarrow{\mathsf{u}}_m \right\}$, and let

$$\mathcal{V}' = \mathcal{V}_0 \cup \left\{ \overrightarrow{\mathsf{v}}_i + \overrightarrow{\mathsf{u}}_j \;\middle|\; i = 1, \ldots, n, j = 1, \ldots, m \right\}.$$

Clearly, all the vectors of $\mathcal{V}'$ have zero in their first coordinate, and as such $\mathcal{V}' \subseteq h$, implying that $\operatorname{cone}(\mathcal{V}') \subseteq C \cap h$.

As for the other direction, consider a vector $\overrightarrow{\mathsf{w}} \in C \cap h$. It can be rewritten as $\overrightarrow{\mathsf{w}} = \overrightarrow{\mathsf{w}}_0 + \sum_i \alpha_i \overrightarrow{\mathsf{v}}_i + \sum_j \beta_j \overrightarrow{\mathsf{u}}_j$, where $\overrightarrow{\mathsf{w}}_0 \in \operatorname{cone}(\mathcal{V}_0)$. Since the first coordinate of $\overrightarrow{\mathsf{w}}$ is zero, we must have that $\sum_i \alpha_i = \sum_j \beta_j$. Now, by the above pairing lemma (Lemma 10.1.9), we have that there are (non-negative) $\delta_{i,j}$s, such that

$$\overrightarrow{\mathsf{w}} = \overrightarrow{\mathsf{w}}_0 + \sum_{i,j} \delta_{i,j} \left(\overrightarrow{\mathsf{v}}_i + \overrightarrow{\mathsf{u}}_j \right) \in \operatorname{cone}(\mathcal{V}'),$$

implying that $C \cap h \subseteq \operatorname{cone}(\mathcal{V}')$. ■

We next handle the general question of how the intersection of a general cone with a hyperplane looks like.

Lemma 10.1.11 *Let $C = \operatorname{cone}(\mathcal{V})$ be a cone generate by a set of vectors $\mathcal{V}$ in $\mathbb{R}^d$. Consider the region $\mathcal{P} = C \cap h$, where h is any hyperplane in $\mathbb{R}^d$. Then, there exists sets of vectors $\mathcal{U}$ and $\mathcal{W}$ such that $\mathcal{P} \cap h = \mathcal{CH}(\mathcal{U}) + \operatorname{cone}(\mathcal{W})$.*

Proof: As before, by change of variables, we can assume that $h \equiv x_1 = 1$. As before, we can normalize the vectors of $\mathcal{V}$ so that the first coordinate is either $0, -1$ or 1. Next, as before, break $\mathcal{V}$ into three disjoint sets, such that $\mathcal{V} = \mathcal{V}_{-1} \cup \mathcal{V}_0 \cup \mathcal{V}_1$, where $\mathcal{V}_i$ are the vectors in $\mathcal{V}$ with the first coordinate being i, for $i \in \{-1, 0, 1\}$.

By Lemma 10.1.10, there exists a set of vectors $\mathcal{W}_0$ that spans $\operatorname{cone}(\mathcal{W}_0) = C \cap (x_1 = 0)$. Also, let $X = \operatorname{cone}(\mathcal{V}_1) \cap (x_1 = 1)$. A point $\mathsf{p} \in X$ is a positive combination of $\mathsf{p} = \sum_i t_i \overrightarrow{\mathsf{v}}_i$, where $\overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1$, where $t_i \geq 0$ for all i. But the first coordinate of all the points of $\mathcal{V}_1$ is 1, and so is the first coordinate of p. Namely, it must be that $\sum_i t_i = 1$. Implying that $X = \mathcal{CH}(\mathcal{V}_1)$.

We claim that $\mathcal{P} = \operatorname{cone}(\mathcal{V}) \cap (x_1 = 1)$ is equal to $Y = \mathcal{CH}(\mathcal{V}_1) + \operatorname{cone}(\mathcal{W}_0)$. One direction is easy, as $\mathcal{V}_1, \mathcal{W}_0 \subseteq \operatorname{cone}(\mathcal{V})$, it follows that $Y \subseteq \operatorname{cone}(\mathcal{V})$. Now, a point of Y is the sum of two vectors, one of them has 1 in the first coordinate, and the other has zero there. As such, a point of Y lies on the hyperplane $x_1 = 1$, implying $Y \subseteq \mathcal{P}$.

As for the other direction, consider a point $\mathsf{p} \in \mathcal{P}$. It can be written as

$$\mathsf{p} = \sum_{i, \overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1} \alpha_i \overrightarrow{\mathsf{v}}_i + \sum_{j, \overrightarrow{\mathsf{u}}_j \in \mathcal{V}_0} \beta_j \overrightarrow{\mathsf{u}}_j + \sum_{k, \overrightarrow{\mathsf{w}}_k \in \mathcal{V}_{-1}} \gamma_k \overrightarrow{\mathsf{w}}_k,$$

where $\alpha_i, \beta_j, \gamma_k \geq 0$, for all i, j, k. Now, by considering only the first coordinate of this sum of vectors, we have

$$\sum_i \alpha_i - \sum_k \gamma_k = 1.$$

In particular, α_i can be rewritten as $\alpha_i = a_i + b_i$, where $a_i, b_i \geq 0$, $\sum_i b_i = \sum_k \gamma_k$ and $\sum_i a_i = 1$. As such,

$$\mathsf{p} = \sum_{i, \overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1} a_i \overrightarrow{\mathsf{v}}_i + \left(\sum_{j, \overrightarrow{\mathsf{u}}_j \in \mathcal{V}_0} \beta_j \overrightarrow{\mathsf{u}}_j + \sum_{i, \overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1} b_i \overrightarrow{\mathsf{v}}_i + \sum_{k, \overrightarrow{\mathsf{w}}_k \in \mathcal{V}_{-1}} \gamma_k \overrightarrow{\mathsf{w}}_k \right),$$

Now, since $\sum_i b_i = \sum_k \gamma_k$, we have $\left(\sum_{i, \overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1} b_i \overrightarrow{\mathsf{v}}_i + \sum_{k, \overrightarrow{\mathsf{w}}_k \in \mathcal{V}_{-1}} \gamma_k \overrightarrow{\mathsf{w}}_k \right) \in (x_1 = 0)$. As such, we have

$$\sum_{j, \overrightarrow{\mathsf{u}}_j \in \mathcal{V}_0} \beta_j \overrightarrow{\mathsf{u}}_j + \sum_{i, \overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1} b_i \overrightarrow{\mathsf{v}}_i + \sum_{k, \overrightarrow{\mathsf{w}}_k \in \mathcal{V}_{-1}} \gamma_k \overrightarrow{\mathsf{w}}_k \quad \in \quad \operatorname{cone}(\mathcal{W}_0).$$

Also, $\sum_i a_i = 1$ and, for all i, $a_i \geq 0$, implying that $\sum_{i, \overrightarrow{\mathsf{v}}_i \in \mathcal{V}_1} a_i \overrightarrow{\mathsf{v}}_i \in \mathcal{CH}(\mathcal{V}_1)$. Thus p is a sum of two points, one of them in $\operatorname{cone}(\mathcal{W}_0)$ and the other in $\mathcal{CH}(\mathcal{V}_1)$. Implying that $\mathsf{p} \in Y$ and thus $\mathcal{P} \subseteq Y$. We conclude that $\mathcal{P} = Y$. ■

Theorem 10.1.12 *A cone C is generated by a finite set $\mathcal{V} \subseteq \mathbb{R}^d$ (that is $C = \text{cone}(\mathcal{V})$), if and only if, there exists a finite set of halfspaces, all passing through the origin, such that their intersection is C.*

In linear programming lingo, a cone C is finitely generated by $\mathcal{V}$, if and only if there exists a matrix $\mathsf{M} \in \mathbb{R}^{m \times d}$, such that $x \in C$ if and only if $\mathsf{M}x \leq 0$.

Proof: Let $C = \text{cone}(\mathcal{V})$, and observe that a point $\mathsf{p} \in \text{cone}(\mathcal{V})$ can be written as (part of) a solution to a linear program, indeed let $\mathcal{V} = \left\{\overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_m\right\}$, and consider the linear program:

$$x \in \mathbb{R}^d \qquad \sum_{i=1}^{m} t_i \overrightarrow{\mathsf{v}}_i = x$$
$$\forall i \qquad t_i \geq 0.$$

Clearly, any $x \in C$, there are $t_1, \ldots, t_m$, such that $\sum_i t_i \overrightarrow{\mathsf{v}}_i = x$. Thus, the projection of this LP with $m + d$ variables into the subspace formed by the coordinates of the d variables $x = (x_1, \ldots, x_d)$ is the set C. Now, by Lemma 10.1.5, the projection of this LP is also a $\mathcal{H}$-polyhedron, implying that C is a $\mathcal{H}$-polyhedron.

As for the other direction, consider the $\mathcal{H}$-polyhedron $\mathcal{P}$ formed by the set of points $(x, w) \in \mathbb{R}^{d+m}$, such that $\mathsf{M}x \leq w$. By Lemma 10.1.8, there exists $\mathcal{V} \subseteq \mathbb{R}^{d+m}$ such that $\text{cone}(\mathcal{V}) = \mathcal{P}$. Now,

$$C = \mathcal{P} \cap (w = 0) = \text{cone}(\mathcal{V}) \cap (w_1 = 0) \cap \cdots \cap (w_m = 0)\,.$$

A repeated application of Lemma 10.1.10 implies that the above set is a cone generated by a (finite) set of vectors, since $w_i = 0$ is a hyperplane, for $i = 1, \ldots, m$. ■

Theorem 10.1.13 *A region $\mathcal{P}$ is a $\mathcal{H}$-polyhedron in $\mathbb{R}^d$, if and only if, there exist finite sets $\mathsf{P}, \mathcal{V} \subseteq \mathbb{R}^d$ such that $\mathcal{P} = \mathcal{CH}(\mathsf{P}) + \text{cone}(\mathcal{V})$.*

Proof: Consider the linear inequality $\sum_{i=1}^{d} a_i x_i \leq b$, which is one the constraints defining the polyhedron $\mathcal{P}$. We can lift this inequality into an inequality passing through the origin, by introducing an extra variable. The resulting inequality in $\mathbb{R}^{d+1}$ is $\sum_{i=1}^{d} a_i x_i - b x_{d+1} \leq 0$. Clearly, this inequality defines a halfspace that goes through the origin, and furthermore, its intersection with $x_{d+1} = 1$ is exactly $(\mathcal{P}, 1)$ (i.e., the set of points in the polyhedron $\mathcal{P}$ concatenated with 1 as the last coordinate). Thus, by doing this "lifting" for all the linear constraints defining $\mathcal{P}$, we get a cone C with an apex in the origin, defined by the intersection of halfspaces passing through the origin. As such, by Theorem 10.1.12, there exists a set of vectors $\mathcal{V} \subseteq \mathbb{R}^{d+1}$, such that $C = \text{cone}(\mathcal{V})$ and $C \cap (x_{d+1} = 1) = (\mathcal{P}, 1)$. Now, Lemma 10.1.11 implies that there exists $\mathsf{P}, \mathcal{W} \subseteq \mathbb{R}^{d+1}$, such that $C \cap (x_{d+1} = 1) = \mathcal{CH}(\mathsf{P}) + \text{cone}(\mathcal{W})$. Dropping the $d + 1$ coordinate from this points, imply that $\mathcal{P} = \mathcal{CH}(\mathsf{P}') + \text{cone}(\mathcal{W}')$, where P', and $\mathcal{W}'$ are P and $\mathcal{W}$ projected on the first d coordinates, respectively.

As for the other direction, assume that $\mathcal{P} = \mathcal{CH}(\mathsf{P}) + \text{cone}(\mathcal{V})$. Let $\mathsf{P}' = (\mathsf{P}, 1)$ and $\mathcal{V}' = (\mathcal{V}, 1)$ be the lifting of P and $\mathcal{V}$ to $d + 1$ dimensions by padding it with an extra coordinate. Now, clearly,

$$(\mathcal{P}, 1) = \mathcal{CH}(\mathsf{P}') + \text{cone}(\mathcal{V}') \subseteq \text{cone}(\mathsf{P}' \cup \mathcal{V}') \cap (x_{d+1} = 1)\,.$$

In fact, the containment holds also in the other direction, since a point $\mathsf{p}ß n\text{cone}(\mathsf{P}' \cup \mathcal{V}') \cap (x_{d+1} = 1)$, is made out of a convex combination of the points of P' (since they have 1 in the $(d + 1)$th coordinate) and a positive combination of the points of $\mathcal{V}'$ (that have 0 in the $(d + 1)$th coordinate). Thus, we have that $(\mathcal{P}, 1) = \text{cone}(\mathsf{P}' \cup \mathcal{V}') \cap (x_{d+1} = 1)$. The cone $C' = \text{cone}(\mathsf{P}' \cup \mathcal{V}')$ can be described as a finite intersection of halfspaces (all passing through the origin), by Theorem 10.1.12. Let $\mathcal{L}$ be the equivalent linear program. Now, replace $x_d = 1$ into this linear program. This results a linear program over $\mathbb{R}^d$, such that its feasible region is $\mathcal{P}$. Namely, $\mathcal{P}$ is a $\mathcal{H}$-polyhedron. ■

A ***polytope*** is the convex hull of a finite point set. Theorem 10.1.13 implies that a polytope is also formed by the intersection of a *finite* set of halfspaces. Namely, a polytope is a bounded polyhedron.

A linear inequality $a \cdot x \leq b$ is valid for a polytope $\mathcal{P}$ if it holds for all $x \in \mathcal{P}$. A ***face*** of $\mathcal{P}$ is a set

$$F = \mathcal{P} \cap (a \cdot x = b)\,,$$

where $a \cdot x \leq b$ is a valid inequality for $\mathcal{P}$. The ***dimension*** of F is the affine dimension of the affine space it spans. As such, a ***vertex*** is 0 dimensional. Intuitively, vertices are the "corners" of the polytopes.

Lemma 10.1.14 *A vertex p of a polyhedron $\mathcal{P}$ can not be written as a convex combination of a set X of points, such that $X \subseteq \mathcal{P}$ and $\mathsf{p} \notin X$.*

Proof: Let h be the hyperplane that its intersection with $\mathcal{P}$ is (only) p. Now, all the points of X must lie on one side of h, and there is no point of X on h itself. As such, any convex combination of the points of X would lie strictly on one side of h. ■

Claim 10.1.15 *Let $\mathcal{P}$ be a polytope in $\mathbb{R}^d$. Then, every polytope is the convex hull of its vertices; namely, $\mathcal{P} = \mathcal{CH}(\text{vert}(\mathcal{P}))$. Furthermore, if for a set $V \subseteq \mathbb{R}^d$, we have $\mathcal{P} = \mathcal{CH}(V)$, then $\text{vert}(\mathcal{P}) \subseteq V$.*

Proof: The polytope $\mathcal{P}$ is a bounded intersection of halfspaces. By Theorem 10.1.13, it can represented as the sum of a convex hull of a finite point set, with a cone. But the cone here is empty, since $\mathcal{P}$ is bounded. Thus, there exists a fine set V, such that $\mathcal{P} = \mathcal{CH}(V)$. Let p be a point of V. If p can be expressed as convex combination of the points of $V \setminus \{\mathsf{p}\}$, then $\mathcal{CH}(V \setminus \{\mathsf{p}\}) = \mathcal{CH}(V)$, since any point expressed as a convex combination involving p can be rewritten to exclude p. So, let X be the resulting set after we remove all such superfluous vertices. We claim that $X = \text{vert}(\mathcal{P})$.

Indeed, if $\mathsf{p} \in X$, then the following LP does not have a solution: $\sum_i t_i = 1$ and $\sum_{i=1}^m t_i \mathsf{p}_i = \mathsf{p}$, where $X \setminus \mathsf{p} = \{\mathsf{p}_1, \ldots, \mathsf{p}_m\}$. In matrix form, this LP is

$$\underbrace{\begin{pmatrix} 1 & 1 & \ldots & 1 \\ \boxed{\mathsf{p}_1} & \boxed{\mathsf{p}_2} & \cdots & \boxed{\mathsf{p}_m} \end{pmatrix}}_{\mathsf{M}} \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{pmatrix} = \begin{pmatrix} 1 \\ \boxed{\mathsf{p}} \end{pmatrix} \quad \text{and} \quad (t_1, \ldots, t_m) \geq 0.$$

By Farakas Lemma II (Lemma 10.1.7), since this LP is not feasible, there exists a vector $\mathsf{w} \in \mathbb{R}^{d+1}$, such that $\mathsf{wM} \geq 0$ and $\mathsf{w} \cdot (1, \mathsf{p}) < 0$. Writing $\mathsf{w} = (\alpha, \mathsf{s})$, we can restate these inequalities as

$$\text{for } i = 1, \ldots, m \quad \alpha + \mathsf{s} \cdot \mathsf{p}_i \geq 0 \quad \text{and} \quad \alpha + \mathsf{s} \cdot \mathsf{p} < 0.$$

In particular, this implies that $\mathsf{s} \cdot \mathsf{p}_i > \mathsf{s} \cdot \mathsf{p}$, for $i = 1, \ldots, m$. As such, p lies on the hyperplane $\mathsf{s} \cdot x = \mathsf{s} \cdot \mathsf{p}$, and all the other points of X are strictly on one side of it. Thus, we conclude that p is a vertex of $\mathcal{P} = \mathcal{CH}(X)$. Namely, $X \subseteq \text{vert}(\mathcal{P})$.

Since $\mathcal{P} = \mathcal{CH}(X)$, and a vertex of $\mathcal{P}$ can not be written as a convex combination of other points inside $\mathcal{P}$, by Lemma 10.1.14, it follows that $\text{vert}(\mathcal{P}) \subseteq X$. Thus, $\mathcal{P} = \mathcal{CH}(\text{vert}(\mathcal{P}))$. ■

10.1.2 Vertices of a polytope

Since a vertex is a corner of the polytope, we can cut it off by a hyperplane. This introduces a new face which captures the structure of how the vertex is connected to the rest of the polytope. Formally, consider a polytope $\mathcal{P}$, with $V = \text{vert}(\mathcal{P})$. Let $\mathsf{w} \cdot x \leq c$ be a valid inequality for $\mathcal{P}$, such that the intersection of $\mathcal{P}$ with the boundary hyperplane $\mathsf{w} \cdot x = c$ is a vertex $\mathsf{v} \in V$. Furthermore, for all other $\mathsf{u} \in V$, we have $\mathsf{w} \cdot \mathsf{u} < c_1 < c$, where c_1 is some constant. (We are using here implicitly Claim 10.1.15.) The ***vertex figure*** of $\mathcal{P}$ at v is

$$\mathcal{P}/\mathsf{v} = \mathcal{P} \cap \Big(\mathsf{w} \cdot x = c_1\Big)$$

The set $\mathcal{P}/\mathsf{v}$ depends (of course) on w and c_1, but its structure is independent of these values.

For a vertex v, let $\text{cone}_{\mathsf{v},\mathcal{P}}$ denote the cone spanned locally by $\mathcal{P}$; formally,

$$\text{cone}_{\mathsf{v},\mathcal{P}} = \mathsf{v} + \text{cone}\Big(\mathcal{P}/\mathsf{v} \;-\; \mathsf{v}\Big)$$

Lemma 10.1.16 *We have $\mathcal{P} \subseteq \text{cone}_{\mathsf{v},\mathcal{P}}$.*

Proof: Consider a point $\mathsf{p} \in \mathcal{P}$. We can assume that for the hyperplane $h \equiv (\mathsf{w} \cdot x = c_1)$, defining $\mathcal{P}/\mathsf{v}$, it holds $\mathsf{w} \cdot \mathsf{p} < c_1$ and (otherwise, we can can translate h so this holds). In particular, consider the point $\mathsf{q} = \mathsf{vp} \cap (\mathcal{P}/\mathsf{v})$. By the convexity of $\mathcal{P}$, we have $\mathsf{q} \in \mathcal{P}$, and as such $\mathsf{q} \in \mathcal{P}/\mathsf{v}$. Thus, $\mathsf{q} - \mathsf{v} \in \text{cone}\Big(\mathcal{P}/\mathsf{v} \;-\; \mathsf{v}\Big)$ and thus $\mathsf{p} - \mathsf{v} \in \text{cone}\Big(\mathcal{P}/\mathsf{v} \;-\; \mathsf{v}\Big)$, which implies that $\mathsf{p} \in \mathsf{v} + \text{cone}\Big(\mathcal{P}/\mathsf{v} \;-\; \mathsf{v}\Big) = \text{cone}_{\mathsf{v},\mathcal{P}}$. ■

Lemma 10.1.17 *Let h be a halfspace defined by $\mathsf{w} \cdot x \leq c_3$, such that for a vertex v of $\mathcal{P}$, we have $\mathsf{w} \cdot \mathsf{v} = c_3$ and h is valid for $\mathcal{P}/\mathsf{v}$ (i.e., for all $x \in \mathcal{P}/\mathsf{v}$ we have $\mathsf{w} \cdot x \leq c_3$), then h is valid for $\mathcal{P}$.*

Proof: Consider the linear function $f(x) = c_3 - \mathsf{w} \cdot x$. Its zero for v and non-negative on $\mathcal{P}/\mathsf{v}$. As such, its non-negative for any ray starting at v and passing through a point of $\mathcal{P}/\mathsf{v}$. As such $f(\cdot)$ is non-negative for any point of $\text{cone}_{\mathsf{v},\mathcal{P}}$, which implies by Lemma **??** that $f(\cdot)$ is non-negative for $\mathcal{P}$. ■

Lemma 10.1.18 *There is a bijection between the k-dimensional faces of $\mathcal{P}$ that contain v and the $(k-1)$-dimensional faces of $\mathcal{P}/\mathsf{v}$. Specifically, for a face f of $\mathcal{P}$, the corresponding face is*

$$\pi(\mathsf{f}) = \mathsf{f} \cap h,$$

where $h \equiv \left(\mathsf{w} \cdot x = c_1\right)$ *is the hyperplane of* $\mathcal{P}/\mathsf{v}$*. Similarly, for a* $(k-1)$*-dimensional face* g *of* $\mathcal{P}/\mathsf{v}$*, the corresponding face of* $\mathcal{P}$ *is*

$$\sigma(\mathsf{g}) = \text{affine}(\mathsf{v}, \mathsf{g}) \cap \mathcal{P},$$

where $\text{affine}(\mathsf{v}, \mathsf{g})$ *is the affine subspace spanned by* v *and the points of* g*.*

Proof: For the sake of simplicity of exposition, assume that $h \equiv x_d = 0$ and furthermore $\mathsf{v}[d] > 0$, where $\mathsf{v}[d]$ denotes the dth coordinate of v. This can always be realized by a rigid rotation and translation of space.

A face f of $\mathcal{P}$ where $\mathsf{v} \in \mathsf{f}$ is defined as $\mathcal{P} \cap \left(\mathsf{w}_2 \cdot x = c_2\right)$, where $\mathsf{w}_2 \cdot x \leq c_2$ is a valid inequality for $\mathcal{P}$. Now,

$$\pi(\mathsf{f}) = \mathsf{f} \cap h = \mathcal{P} \cap \left(\mathsf{w}_2 \cdot x = c_2\right) \cap h = (\mathcal{P} \cap h) \cap \left(\mathsf{w}_2 \cdot x = c_2\right) = (\mathcal{P}/\mathsf{v}) \cap \left(\mathsf{w}_2 \cdot x = c_2\right),$$

where $\mathsf{w}_2 \cdot x \leq c_2$ is a valid inequality for the polytope $\mathcal{P}/\mathsf{v} \subseteq \mathcal{P}$. As such, $\pi(\mathsf{f})$ is a face of $\mathcal{P}/\mathsf{v}$. Note, that if f is k-dimensional and $k > 1$, then it contains two vertices of $\mathcal{P}$, which are on different sides of h, and as such $\pi(\mathsf{f})$ is not empty.

For g be a face of $\mathcal{P}/\mathsf{v}$, defined as $\mathsf{g} = \mathcal{P}/\mathsf{v} \cap (\mathsf{w}_3 \cdot x = c_3)$, where $\mathsf{w}_3 \cdot x \leq c_3$ is an inequality valid for $\mathcal{P}/\mathsf{v}$. Note, that by setting $\mathsf{w}_3[d]$ to be a sufficiently large negative number, we can guarantee that $\mathsf{w}_3 \cdot \mathsf{v} < c_3$.

For $\lambda > 0$, consider the convex combination of the two inequalities $\mathsf{w}_3 \cdot x \leq c_3$ and $x_d \leq 0$; that is

$$h(\lambda) \equiv \qquad (\mathsf{w}_3) \cdot x + \lambda x_d \leq c_3.$$

Geometrically, as we increase $\lambda \geq 0$, the halfspace $h(\lambda)$ is formed by a hyperplane rotating around the affine subspace s formed by the intersection of the hyperplanes $\mathsf{w}_3 \cdot x = c_3$ (for $\lambda = 0$) and $x_d = 0$ (for $\lambda = \infty$).[3] Since the two original inequalities are valid for $\mathcal{P}/\mathsf{v}$, it follows that $h(\lambda)$ is valid for $\mathcal{P}/\mathsf{v}$, for any $\lambda \geq 0$. On the other hand, $\mathsf{v} \in h(0)$ and $\mathsf{v} \notin h(\infty)$. It follows, that there is a value λ_0 of λ, such that v lies on the hyperplane bounding $h(\lambda_0)$. Since $h(\lambda_0)$ is valid for $\mathcal{P}/\mathsf{v}$, it follows, by Lemma 10.1.17, that $h(\lambda_0)$ is valid for $\mathcal{P}$. As such, $\mathsf{f} = h(\lambda_0) \cap \mathcal{P}$ is a face of $\mathcal{P}$ that contains both v and g. As such, $\text{affine}(\mathsf{v}, \mathsf{g}) \cap \mathcal{P} \subseteq \mathsf{f}$. It remain to show equality. So consider a point $\mathsf{p} \in \mathsf{f}$, and as before we can assume that $\mathsf{p}[d] < 0$. As such, $\mathsf{r} = \mathsf{pv} \cap (x_d = 0)$ is a point that is on the boundary of $\mathcal{P}/\mathsf{v}$, and furthermore $\mathsf{r} \in \mathsf{g}$ since $h(\lambda_0)$ and $\mathsf{w}_3 \cdot x \leq c_3$ have the same intersection with the hyperplane $x_d = 0$ (i.e., the boundary of this intersection is s), which implies that $\mathsf{r} \in \text{affine}(\mathsf{v}, \mathsf{g})$, and as such $\mathsf{f} = \text{affine}(\mathsf{v}, \mathsf{g}) \cap \mathcal{P}$.

As such, the maps π and σ are well defined. We remain with the task of verifying that they are the inverse of each other. Indeed, for a face g of $\mathcal{P}/\mathsf{v}$ we have

$$\begin{aligned} \pi\Big(\sigma(\mathsf{g})\Big) &= \pi(\text{affine}(\mathsf{v}, \mathsf{g}) \cap \mathcal{P}) = (\text{affine}(\mathsf{v}, \mathsf{g}) \cap \mathcal{P}) \cap h = \text{affine}(\mathsf{g}) \cap (\mathcal{P} \cap h) \\ &= \text{affine}(\mathsf{g}) \cap \mathcal{P}/\mathsf{v} = \mathsf{g}, \end{aligned}$$

since $\mathsf{v} \notin h$ and $\mathsf{g} \subseteq h$. Similarly, for a face f of $\mathcal{P}$ that contains v, we have

$$\sigma\Big(\pi(\mathsf{f})\Big) = \sigma\Big(\mathsf{f} \cap h\Big) = \text{affine}(\mathsf{v}, \mathsf{f} \cap h) \cap \mathcal{P} = \text{affine}(\mathsf{f}) \cap \mathcal{P} = \mathsf{f},$$

since $\text{affine}(\mathsf{f})$ can be written as the affine subspace of v together with a set of points in $\mathsf{f} \cap h$. ■

10.2 Linear Programming Correctness

We are now ready to prove that the algorithms we presented for linear programing do indeed work. As a first step, we prove that locally checking if we are in the optimal vertex is sufficient.

Lemma 10.2.1 *Let* v *be a vertex of a* $\mathcal{H}$*-polyhedron* $\mathcal{P}$ *in* $\mathbb{R}^d$*, and let* $f(\cdot)$ *be a linear function, such that* $f(\cdot)$ *is non-decreasing along all the edges leaving* v *(i.e.,* v *is a "local" minimum of* $f(\cdot)$ *along the edges adjacent to* v*), then* v *realizes the global minimum of* $f(\cdot)$ *on* $\mathcal{P}$*.*

Proof: Assume for the sake of contradiction that this is false, and let x be a point in $\mathcal{P}$, such that $f(x) < f(\mathsf{v})$. Let $\mathcal{P}/\mathsf{v} = \mathcal{P} \cap h$, where h is a hyperplane.

By convexity, the segment $x\mathsf{v}$ must intersect $\mathcal{P}/\mathsf{v}$, and let y be this intersection point. Since $\mathcal{P}/\mathsf{v}$ is a convex polytope in $d-1$ dimensions, y can be written as a convex combination of d of its vertices $\mathsf{u}_1, \ldots, \mathsf{u}_d$; namely, $y = \sum_i \alpha_i \mathsf{u}_i$, where $\alpha_i \geq 0$ and $\sum_i \alpha_i = 1$. By Lemma 10.1.18, each one of these vertices lie on edge of $\mathcal{P}$ adjacent to v, and by the local optimality of v, we have $f(\mathsf{u}_i) \geq f(\mathsf{v})$. Now, by the linearity of $f(\cdot)$, we have

$$f(y) = f\left(\sum_i \alpha_i \mathsf{u}_i\right) = \sum_i \alpha_i f(\mathsf{u}_i) \geq f(\mathsf{v}).$$

But y is a convex combination of x and v and $f(x) < f(\mathsf{v})$. As such, $f(y) < f(\mathsf{v})$. A contradiction. ■

[3] As such, $\mathsf{g} \subseteq \mathsf{s}$. Not that we need this fact anywhere.

10.3 Garbage

In the following, assume that the polytope (i.e., feasible region) is bounded.

Lemma 10.3.1 *Let $\mathcal{P}$ be a bounded polytope, and let V be the set of vertices of $\mathcal{P}$. Then $\mathcal{P} = C\mathcal{H}(V)$.*

Consider the intersection of $d-1$ hyperplanes of the LP with $\mathcal{P}$. Clearly, this is either empty or a segment connecting two vertices of $\mathcal{P}$. If the intersection is not empty, we will refer to it as ***edge*** connecting the two vertices forming its endpoints. Consider the graph $\mathsf{G} = \mathsf{G}(V, \mathcal{E})$ formed by this set of edges. The target function assign each vertex a value. By general position assumption, we can assume that no pair of vertices has the same target value assigned to it. A vertex is a ***sink*** if its target value is lower than all its neighbors. Assume that G contains a single sink and its connected.

Start a traversal of G in an arbitrary vertex v, and repeatedly move to any of its neighbors that has lower target value than itself. This walk would stop once we arrive to the sink, which is the required optimal solution to the LP. This is essentially the ***simplex*** algorithm for Linear Programming.

To this end, we need to prove that G is connected and has a single sink.

10.4 Bibliographical notes

10.5 Exercises

Exercise 10.5.1 (Pairing lemma.) [3 Points]

Prove Lemma 10.1.9. Specifically, let $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m$ be positive numbers, such that $\sum_{i=1}^{n} \alpha_i = \sum_{j=1}^{m} \beta_j$, and consider the positive combination $\overrightarrow{\mathsf{w}} = \sum_{i=1}^{n} \alpha_i \overrightarrow{\mathsf{v}}_i + \sum_{i=j}^{m} \beta_j \overrightarrow{\mathsf{u}}_j$, where $\overrightarrow{\mathsf{v}}_1, \ldots, \overrightarrow{\mathsf{v}}_n, \overrightarrow{\mathsf{u}}_1, \ldots, \overrightarrow{\mathsf{u}}_m$ are vectors (say, in $\mathbb{R}^d$). Then, there are non-negative $\delta_{i,j}$s, such that $\overrightarrow{\mathsf{w}} = \sum_{i,j} \delta_{i,j} \left(\overrightarrow{\mathsf{v}}_i + \overrightarrow{\mathsf{u}}_j \right)$.

In fact, at most $n + m - 1$ of the $\delta_{i,j}$s have to be non-zero.

Chapter 11

Approximate Nearest Neighbor Search in Low Dimension

> "Napoleon has not been conquered by man. He was greater than all of us. But god punished him because he relied on his own intelligence alone, until that prodigious instrument was strained to breaking point. Everything breaks in the end."
>
> – Carl XIV Johan, King of Sweden

11.1 Introduction

Let P be a set of n points in $\mathbb{R}^d$. We would like to preprocess it, such that given a query point q, one can determine the closest point in P to q quickly. Unfortunately, the exact problem seems to require prohibitive preprocessing time. (Namely, computing the Voronoi diagram of P, and preprocessing it for point-location queries. This requires (roughly) $O(n^{\lceil d/2 \rceil})$ time.)

Instead, we will specify a parameter $\varepsilon > 0$, and build a data-structure that answers ***$(1+\varepsilon)$-approximate nearest neighbor*** queries.

Definition 11.1.1 For a set $\mathsf{P} \subseteq \mathbb{R}^d$, and a point q, we denote by $\widehat{\mathsf{q}}$ the closest point in P to q. We denote by $\mathbf{d}_{\mathsf{P}}(\mathsf{q})$ the distances between q and its closest point in P; that is $\mathbf{d}_{\mathsf{P}}(\mathsf{q}) = \left\|\mathsf{q} - \widehat{\mathsf{q}}\right\|$.

For a query point q, and a set P of n points in $\mathbb{R}^d$, the point $\mathsf{r} \in \mathsf{P}$ is an ***$(1+\varepsilon)$-approximate nearest neighbor*** (or just ***$(1+\varepsilon)$-ANN***) if $\|\mathsf{q} - \mathsf{r}\| \leq (1+\varepsilon)\mathbf{d}_{\mathsf{P}}(\mathsf{q})$. Alternatively, for any $\mathsf{s} \in \mathsf{P}$, we have $\|\mathsf{q} - \mathsf{r}\| \leq (1+\varepsilon)\|\mathsf{q} - \mathsf{s}\|$.

This is yet another instance where solving the bounded spread case is relatively easy.

11.2 The bounded spread case

Let P be a set of n points contained inside the unit hypercube in $\mathbb{R}^d$, and let $\mathcal{T}$ be a quadtree of P, where $\operatorname{diam}(\mathsf{P}) = \Omega(1)$. We assume that with each (internal) node u of $\mathcal{T}$, there is an associated representative point rep_u which is one of the points of P stored in the subtree rooted at u.

Let q be a query point, such that $\left\|\mathsf{q} - \widehat{\mathsf{q}}\right\| \geq r$ and let $\varepsilon > 0$ be a parameter. We would like to find a $(1+\varepsilon)$-ANN to q.

The algorithm. Let $A_0 = \{\operatorname{root}(T)\}$, and let $r_{\text{curr}} = \left\|\mathsf{q} - \operatorname{rep}_{\operatorname{root}(T)}\right\|$. The value of r_{curr} is the distance to the closet neighbor of q that was found so far by the algorithm.

In the ith iteration, for $i > 0$, the algorithm expands the nodes of A_{i-1} to get A_i. Formally, for $v \in A_{i-1}$, let C_v be the set of children of v in $\mathcal{T}$ and $\square_v$ denote the cell (i.e., region) v corresponds to. For every node $w \in C_v$, we compute

$$r_{\text{curr}} \leftarrow \min\left(r_{\text{curr}}\,,\, \left\|\mathsf{q} - \operatorname{rep}_w\right\|\right).$$

The algorithm checks if

$$\left\|\mathsf{q} - \operatorname{rep}_w\right\| - \operatorname{diam}(\square_w) < (1 - \varepsilon/2) r_{\text{curr}}, \tag{11.1}$$

and if so, it adds w to A_i. The algorithm continues in this expansion process till all the elements of A_{i-1} were considered, and then it moves to the next iteration. The algorithm stops when the generated set A_i is empty. The algorithm returns the point realizing the value of r_{curr} as the ANN.

The set A_i is a set of nodes of depth i in the quadtree that the algorithm visits. Note, all these nodes belong to the canonical grid $\mathsf{G}_{2^{-i}}$ of level $-i$, where every canonical square has sidelength 2^{-i}. (Thus, nodes of depth i in the quadtree are of ***level*** $-i$. This is somewhat confusing but it in fact makes the presentation simpler.)

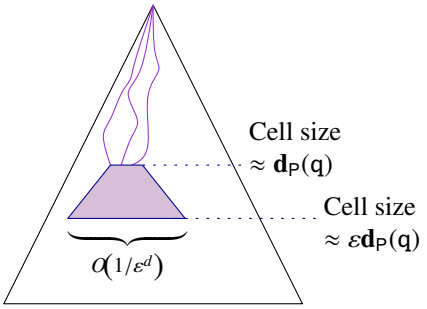
Correctness. Note that the algorithm adds a node w to A_i only if the set P_w might contain points which are closer to q than the (best) current nearest neighbor the algorithm found, where P_w is the set of points stored in the subtree of w. (More precisely, P_w might contain a point which is $(1-\varepsilon/2)$ closer to q than any point encountered so far.)

Consider the last node w inspected by the algorithm such that $\widehat{\mathsf{q}} \in \mathsf{P}_w$. Since the algorithm decided to throw this node away, we have, by the triangle inequality, that

$$\left\|\mathsf{q}-\widehat{\mathsf{q}}\right\| \geq \left\|\mathsf{q}-\mathrm{rep}_w\right\| - \mathrm{diam}(\square_w) \geq (1-\varepsilon/2) r_{\mathrm{curr}}.$$

Thus, $\left\|\mathsf{q}-\widehat{\mathsf{q}}\right\|/(1-\varepsilon/2) \geq r_{\mathrm{curr}}$. However, $1/(1-\varepsilon/2) \leq 1+\varepsilon$, for $1 \geq \varepsilon > 0$, as can be easily verified. Thus, $r_{\mathrm{curr}} \leq (1+\varepsilon)\mathbf{d}_{\mathsf{P}}(\mathsf{q})$, and the algorithm returns an $(1+\varepsilon)$-ANN to q.

Running time analysis. Before barging into a formal proof of the running time of the above search procedure, it is useful to visualize the execution of the algorithm. It visits the quadtree level by level. As long as the level grid cells are bigger than the ANN distance $r = \mathbf{d}_{\mathsf{P}}(\mathsf{q})$, the number of nodes visited is a constant (i.e., $|A_i| = O(1)$). This number "explodes" only when the cell size become smaller than r, but then the search stops when we reach grid size $O(\varepsilon r)$. In particular, since the number grid cells visited (in the second stage) grows exponentially with the level, we can use the number of nodes visited in the bottom level (i.e., $O(1/\varepsilon^d)$) to bound the query running time for this part of the query.



Lemma 11.2.1 *Let P be a set of n points contained inside the unit hypercube in $\mathbb{R}^d$, and let $\mathcal{T}$ be a quadtree of P, where* $\mathrm{diam}(\mathsf{P}) = \Omega(1)$. *Let q be a query point, and let $\varepsilon > 0$ be a parameter. An $(1+\varepsilon)$-ANN to q can be computed $O\left(\varepsilon^{-d} + \log(1/\varpi)\right)$ time, where $\varpi = \left\|\mathsf{q}-\widehat{\mathsf{q}}\right\|$.*

Proof: The algorithm is described above. We only left with the task of bounding the query time. Observe that if a node $w \in \mathcal{T}$ is considered by the algorithm, and $\mathrm{diam}(\square_w) < (\varepsilon/4)\varpi$ then

$$\left\|\mathsf{q}-\mathrm{rep}_w\right\| - \mathrm{diam}(\square_w) \geq \left\|\mathsf{q}-\mathrm{rep}_w\right\| - (\varepsilon/4)\varpi \geq r_{\mathrm{curr}} - (\varepsilon/4) r_{\mathrm{curr}} \geq (1-\varepsilon/4) r_{\mathrm{curr}},$$

which implies that neither w nor any of its children would be inserted into the sets $A_1, \ldots, A_m$, where m is the depth $\mathcal{T}$, by Eq. (11.1). Thus, no nodes of depth $\geq h = \lceil -\lg(\varpi\varepsilon/4) \rceil$ are being considered by the algorithm.

Consider the node u of $\mathcal{T}$ of depth i containing $\widehat{\mathsf{q}}$. Clearly, the distance between q and rep_u is at most $\ell_i = \varpi + \mathrm{diam}_u = \varpi + \sqrt{d}2^{-i}$. As such, in the end of the ith iteration, we have $r_{\mathrm{curr}} \leq \ell_i$, since the algorithm had inspected u. Thus, the only cells of $\mathsf{G}_{2^{-i-1}}$ that might be considered by the algorithm are the ones in distance $\leq \ell_i$ from q. The number of such cells is

$$n_i = \left(2\left\lceil \frac{\ell_i}{2^{-i-1}} \right\rceil\right)^d = O\left(\left(1 + \frac{\varpi + \sqrt{d}2^{-i}}{2^{-i-1}}\right)^d\right) = O\left(\left(1 + \frac{\varpi}{2^{-i-1}}\right)^d\right) = O\left(1 + \left(2^i\varpi\right)^d\right),$$

since for any $a, b \geq 0$ we have $(a+b)^d \leq (2\max(a,b))^d \leq 2^d\left(a^d + b^d\right)$. Thus, the total number of nodes visited is

$$\sum_{i=0}^{h} n_i = O\left(\sum_{i=0}^{\lceil -\lg(\varpi\varepsilon/4)\rceil} \left(1 + \left(2^i\varpi\right)^d\right),\right) = O\left(\lg\frac{1}{\varpi\varepsilon} + \left(\frac{\varpi}{\varpi\varepsilon/4}\right)^d\right) = O\left(\log\frac{1}{\varpi} + \frac{1}{\varepsilon^d}\right),$$

and this also bounds the overall query time. ■

One can apply Lemma 11.2.1 to the case the input has bounded spread. Indeed, if the distance between the closest pair of points of P is $\mu = \mathcal{CP}(\mathsf{P})$, then the algorithm would never search in (the children of) cells that have diameter $\leq \mu/2$, since all such nodes are leafs. As such, we can replace in the above argumentation r by μ.

Lemma 11.2.2 *Let P be a set of n points in $\mathbb{R}^d$, and let $\mathcal{T}$ be a quadtree of P, where* $\mathrm{diam}(\mathsf{P}) = \Omega(1)$. *Given a query point q and $1 \leq \varepsilon > 0$, one can return an $(1+\varepsilon)$-ANN to q in $O\left(1/\varepsilon^d + \log\Phi(\mathsf{P})\right)$ time.*

A less trivial task, is to adapt the algorithm, so that it uses compressed quadtrees. To this end, the algorithm would still handle the nodes by levels. This requires us to keep a heap of integers in the range $0, -1, \ldots, -\lfloor \lg\Phi(\mathsf{P}) \rfloor$. This can be easily done by maintaining an array of size $O(\log\Phi(\mathsf{P}))$, where each array cell, maintains a linked list of all nodes with this level. Clearly, an insertion/deletion into this heap data-structure can be handled in constant time by augmenting it with a hash table. Thus, the above algorithm would work for this case after modifying it to use this "level" heap instead of just the sets A_i.

Theorem 11.2.3 *Let P be a set of n points in $\mathbb{R}^d$. One can preprocess P in $O(n \log n)$ time, and using linear space, such that given a query point q and parameter $1 \geq \varepsilon > 0$, one can return an $(1+\varepsilon)$-ANN to q in $O\Big(1/\varepsilon^d + \log \Phi(\mathsf{P})\Big)$ time. In fact, the query time is $O(1/\varepsilon^d + \log(\operatorname{diam}(\mathsf{P})/\varpi))$, where $\varpi = \mathbf{d}_{\mathsf{P}}(\mathsf{q})$.*

11.3 ANN – the unbounded general case

The Snark and the unbounded spread case. (Or a meta-philosophical pretentious discussion that the reader might want to skip. The reader might consider this to be a footnote of a 1footnote, which finds itself inside the text because of lack of space in the bottom of the page.) We have a data-structure that supports insertions, deletions and approximate nearest neighbor reasonably quickly. The running time for such operations is roughly $O(\log \Phi(\mathsf{P}))$ (ignoring additive terms in $1/\varepsilon$). Since the spread of P in most real world applications is going to be bounded by a constant degree polynomial in n, it seems this is sufficient for our purposes, and we should stop now, while ahead in the game. But the nagging question remains: If the spread of P is not bounded by something reasonable, what can be done?

The rule of thumb is that $\Phi(\mathsf{P})$ can always be replaced by n (for this problem, but also in a lot of other problems). This usually requires some additional machinery, and sometimes this machinery is quite sophisticated and complicated. At times, the search for the ultimate algorithm that can work for such "strange" inputs, looks like the Hunting of the Snark [Car76] – a futile waste of energy looking for some imaginary top-of-the-mountain, which has no practical importance. (At times the resulting solution is so complicated, it feels like a Boojum [Car76].)

However, solving the bounded spread case can be acceptable in many situations, and it is the first stop in trying to solve the general case. Furthermore, solving the general case provide us with more insights on the problem, and in some cases leads to more efficient solutions than the bounded spread case.

With this caveat emptor warning duly given, we plunge ahead into solving the ANN for the unbounded spread case.

Plan of attack. To answer ANN query in the general case, we will first get a fast rough approximation. Next, using a compressed quadtree, we would find a constant number of relevant nodes, and apply Theorem 11.2.3 to those nodes. This would yields the required approximation. Before solving this problem, we need a minor extension of the compressed quadtree data-structure.

11.3.1 Extending a compressed quadtree to support cell queries

Let $\widehat{\square}$ be a canonical grid cell (we remind the reader that this is a cell of the grid $\mathsf{G}_{2^{-i}}$, for some integer $i \leq 0$). Given a compressed quadtree $\widehat{T}$, we would like to find the *single* node $v \in \widehat{T}$, such that $P \cap \widehat{\square} = \mathsf{P}_v$. We will refer to such query as a ***cell query***.

It is not hard to see that the quadtree data-structure can be modified to support cell queries in logarithmic time (its essentially a glorified point-location query), and we omit the easy but tedious details. See Exercise 2.7.3.

Lemma 11.3.1 *One can perform a call query in a compressed quadtree $\widehat{T}$, in $O(\log n)$ time, where n is the size of $\widehat{T}$. Namely, given a query canonical cell $\widehat{\square}$, one can find, in $O(\log n)$ time, the node $w \in \widehat{T}$ such that $\square_w \subseteq \widehat{\square}$ and $P \cap \widehat{\square} = \mathsf{P}_w$.*

11.3.2 Putting things together

Let P be a set of n points in $\mathbb{R}^d$ contained in the unit hypercube. We build the compressed quadtree $\widehat{T}$ of P, so that it supports cell queries, using Lemma 11.3.1. We will also need a data-structure that supports very rough ANN. We describe several ways to build such a data-structure in the next section, and in particular, we will use the following result (see Theorem 11.4.7).

Lemma 11.3.2 *Let P be a set of n points in $\mathbb{R}^d$. One can build a data structure $\mathcal{T}_R$, in $O(n \log n)$ time, such that given a query point $\mathsf{q} \in \mathbb{R}^d$, one can return a $(1+4n)$-ANN of q in P in $O(\log n)$ time.*

Given a query point q, using $\mathcal{T}_R$, we compute a point $u \in P$, such that $\varpi \leq \|u - q\| \leq (1+4n)\varpi$, where $\varpi = \mathbf{d}_{\mathsf{P}}(\mathsf{q})$. Let $R = \|u - q\|$ and $r = \|u - q\|/(4n+1)$. Clearly, $r \leq \varpi \leq R$. Next, compute $\ell = \lceil \lg R \rceil$, and let C be the set of cells of G_{2^ℓ} that are in distance $\leq R$ from q. Clearly, since $R \leq 2^\ell$, it follows that $\widehat{\mathsf{q}} \in \bigcup_{\square \in C} \square$, where $\widehat{\mathsf{q}}$ is the nearest neighbor to q in P. For each cell $\square \in C$, we compute the node $v \in \widehat{T}$ such that $P \cap \square = \mathsf{P}_v$, using a cell query (i.e., Lemma 11.3.1). Let V be the resulting set of nodes of $\widehat{T}$.

For each node of $v \in V$, we now apply the algorithm of Theorem 11.2.3 to the compressed quadtree rooted at v. Since $|V| = O(1)$, and $\operatorname{diam}(\mathsf{P}_v) = O(R)$, for all $v \in V$, the query time is

$$\begin{aligned}\sum_{v \in V} O\left(\frac{1}{\varepsilon^d} + \log \frac{\operatorname{diam}(\mathsf{P}_v)}{r}\right) &= O\left(\frac{1}{\varepsilon^d} + \sum_{v \in V} \log \frac{\operatorname{diam}(\mathsf{P}_v)}{r}\right) = O\left(\frac{1}{\varepsilon^d} + \sum_{v \in V} \log \frac{R}{r}\right) \\ &= O\left(\frac{1}{\varepsilon^d} + \log n\right).\end{aligned}$$

As for the correctness of the algorithm, notice that there is a node $w \in V$, such that $\widehat{\mathsf{q}} \in \mathsf{P}_w$. As such, when we apply the algorithm of Theorem 11.2.3 to w, it would return us a $(1+\varepsilon)$-ANN to q.

Theorem 11.3.3 *Let P be a set of n points in $\mathbb{R}^d$. One can construct a data-structure of linear size, in $O(n \log n)$ time, such that given a query point $q \in \mathbb{R}^d$, and a parameter $1 \geq \varepsilon > 0$, one can compute a $(1+\varepsilon)$-ANN to q in $O(1/\varepsilon^d + \log n)$ time.*

11.4 Low Quality ANN Search

To perform ANN in the unbounded spread case, all we need is a rough approximation (i.e., polynomial factor in n) to the distance to the nearest-neighbor (note that we need only the distance). We present two different ways to get this rough ANN.

11.4.1 Low Quality ANN Search - Point Location with Random Shifting

11.4.1.1 The data-structure and search procedure

Let P be a set of n points in $\mathbb{R}^d$, contained inside the square $[0,1/2]^d$. Let $\overrightarrow{\mathsf{v}}$ be a random vector in the square $[0,1/2]^d$, and consider the point set $\mathsf{Q} = \mathsf{P} + \overrightarrow{\mathsf{v}} = \left\{ \mathsf{p} + \overrightarrow{\mathsf{v}} \;\middle|\; \mathsf{p} \in \mathsf{P} \right\}$. Clearly, given a query point q, we can answer the ANN on P, by answering the ANN query $\mathsf{q} + \overrightarrow{\mathsf{v}}$ on Q. Note, that Q is contained inside the unit cube, and consider the compressed quadtree $\widehat{T}$ build for Q.

Given a query point q, let v be the node of $\widehat{T}$ that its region rg_v contains $\mathsf{q}' = \mathsf{q} + \overrightarrow{\mathsf{v}}$. If rg_v is a cube (i.e., v is a leaf) and the v stores a point $\mathsf{p} \in \mathsf{Q}$ inside it, then we return $\|\mathsf{q}' - \mathsf{p}\|$ as the distance to the ANN. If v does not store a point of Q, then we return $2\mathrm{diam}(\mathsf{rg}_v)$ as the distance to the ANN.

Things get more exciting if v is a compressed node. In this case, there is no point of Q associated with v (by construction). Let w be its only child, and return $\mathbf{d}(\mathsf{rg}_w, \mathsf{q}') + \mathrm{diam}(\mathsf{rg}_v)$ as the distance to the ANN.

11.4.1.2 Proof of correctness

Lemma 11.4.1 *Let $\mathcal{I} = [\alpha, \beta]$ be an interval on the real line, and let $r = 2^{-\ell}$ be a given real number, where $\ell \geq 1$ is an integer. Let $U = \left\{ ir \;\middle|\; i \text{ is an integer} \right\}$, and consider a random number $x \in [0,1/2]$. Then, the probability that $\mathcal{I} + x$ contains a point of U is (at most) $\|\mathcal{I}\|/r$.*

Proof: If $\|\mathcal{I}\| \geq r$, then any translation of $\mathcal{I}$ contains a point of U, and the claim trivially holds.

Otherwise, let X be the set of real numbers such that if $x \in X$, then $\mathcal{I} + x$ contains a point of U. The set X is a repetitive set of intervals. Formally, $X = \bigcup_k ([-\beta, -\alpha] + kr)$. As such, the total length of X inside the interval $[0,1/2]$ is

$$\rho = \frac{\|\mathcal{I}\|}{r} \left\| [0,1/2] \right\|,$$

since r is a power of 2. As such, the probability that a random point inside $[0,1/2]$ would fall inside X is $\rho / \left\| [0,1/2] \right\| = \|\mathcal{I}\|/r$. ■

Lemma 11.4.2 *Let s be a segment of length Δ. Consider the randomly shifted segment $s + \overrightarrow{\mathsf{v}}$. The probability that $s + v$ intersect the boundary of the canonical grid G_r is at most $d\Delta/r$, where $r = 2^{-\ell}$ and $\ell \geq 1$.*

Proof: Let $\mathcal{I}_i = \{\alpha_i, \beta_i\}$ be the projection of s into the ith axis, for $i = 1, \ldots, d$. Clearly, $s+v$ intersects the separating hyperplanes orthogonal to the ith dimension, if (the randomly shifted) interval $\mathcal{I}_i$ contains a point of $U = \left\{ ir \;\middle|\; i \text{ integer} \right\}$. By Lemma 11.4.1, the probability for that is at most δ_i/r, where $\delta_i = \|\mathcal{I}_i\|$. Thus, the probability that $s + \overrightarrow{\mathsf{v}}$ intersects G_r is bounded by

$$\sum_{i=1}^{d} \frac{\delta_i}{r} \leq \frac{d\Delta}{r},$$

as claimed. ■

Lemma 11.4.3 *For any integer i, and a query point q, the above data-structure returns a $4n^i$ approximation to the distance to the nearest neighbor of q in P, with probability $\geq 1 - 2n^{-i+1}$.*

Proof: For the time being, it would be easier to consider $\widehat{T}$ to be a regular (i.e., not compressed) quadtree of the point set Q. And let v be the leaf of $\widehat{T}$ that contains the query point $\mathsf{q}' = \mathsf{q} + \overrightarrow{\mathsf{v}}$. Let $\mathsf{p} = \widehat{\mathsf{q}'}$ be the nearest neighbor to q' in Q, and consider the segment $s = \mathsf{q}\widehat{\mathsf{q}}$. Clearly, $s' = s + \overrightarrow{\mathsf{v}} = \mathsf{q}'\mathsf{p}$.

Now, if s' is completely contained in the leaf v of $\widehat{T}$ that contains q', then p is stored in v, and we return $\|s'\|$ as the distance to the ANN.

If s' intersects the boundary of the leaf v, let r be the side length of rg_v. Observe, that $r \geq \|s\| / \sqrt{d}$. Indeed, if $r \leq \|s\| / \sqrt{d}$ then $\mathsf{rg}_v \subseteq \mathbf{b}(\mathsf{q}', \|s\|)$, but the interior of $\mathbf{b}(\mathsf{q}', \|s\|)$ is empty of any points of Q. Namely, the region of v will not be further refined by the construction algorithm. As such, the result returned by the algorithm is never too small, it can only be too large.

In particular, if the side length of the leaf that contains q' is r, then s' intersects the boundary of the grid G_r, and the probability for that to happened is at most $d\varpi/r$, by Lemma 11.4.2. Thus, let x be the smallest power of two which is larger than $n^i\varpi$. We have that

$$\mathbf{Pr}\Big[\text{dist. returned } \geq 2\sqrt{d}n^i\varpi\Big] \leq \mathbf{Pr}\Big[r \geq n^i\varpi\Big] \leq \sum_{i=0}^{\infty} \frac{d\varpi}{2^i x} \leq \sum_{i=0}^{\infty} \frac{d\varpi}{2^i n^i \varpi} \leq \frac{2}{n^{i-1}}.$$

Thus, the only case that remains, is when v is a compressed node. Let w be the only child of v in $\widehat{T}$, and observe that there are points of Q stored in rg_w, which implies that the answer returned is indeed an upper bound on the distance to the ANN. There are several cases:

- If $\mathsf{p} = \widehat{\mathsf{q}'}$ is outside the bounding cube of v. Then s' intersects the outer boundary of rg_v, and its easy to argue that the answer returned is shorter than the $2\mathrm{diam}(\mathsf{rg}_v)$, which implies by the above analysis, that the answer returned is ANN with the required bounds.
- If $\mathsf{p} \in \mathsf{Q}_w$, then there are two cases:

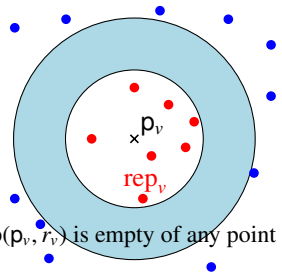
  (i) If $\mathbf{d}(\mathsf{q}', \mathsf{rg}_w) > \mathrm{diam}(\mathsf{rg}_w)/n^i$ then clearly, the answer returned is a $2n^i$-approximation to the ANN, since $\|\mathsf{q}'\mathsf{p}\| \geq \mathbf{d}(\mathsf{q}', \mathsf{rg}_w)$.

  (ii) If $\mathbf{d}(\mathsf{q}', \mathsf{rg}_w) < \mathrm{diam}(\mathsf{rg}_w)/n^i$ then $\mathsf{q}'\mathsf{p}$ intersects the boundary of rg_w. This is the same segment we would have intersected if $\widehat{T}$ was not compressed, and as such the analysis for the uncompressed quadtree implies the claim. ■

11.4.2 Low Quality ANN Search - The Ring Separator Tree

Definition 11.4.4 A binary tree $\mathcal{T}$ having the points of P as leaves, is a ***t-ring tree*** for P, if every node $v \in T$, is associated with a ring (hopefully "thick"), such that the ring separates the points into two sets (hopefully both relatively large), and the interior of the ring is empty of any point of P.

For a node v of $\mathcal{T}$, let P_v denote the subset of points of P stored in the subtree of v in $\mathcal{T}$, and let p_v be a point stored in v. We require, that for any node v of $\mathcal{T}$, there is an associated ball $\mathbf{b}_v = \mathbf{b}(\mathsf{p}_v, r_v)$, such that all the points of $P^v_{\text{in}} = \mathsf{P}_v \cap \mathbf{b}_v$ are in one child of $\mathcal{T}$. Furthermore, all the other points of P_v are outside the interior of the enlarged ball $\mathbf{b}(\mathsf{p}_v, (1+t)r_v)$, and are stored in the other child of v.



We will also store an arbitrary representative point $\mathsf{rep}_v \in P^v_{\text{in}}$ in v.

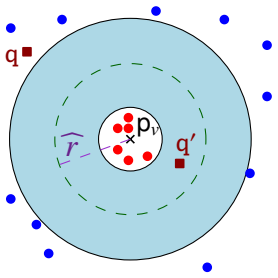
Namely, if $\mathcal{T}$ is a t-ring tree, then for any node $v \in \mathcal{T}$, the interior of the ring $\mathbf{b}(\mathsf{p}_v, (1+t)r_v) \setminus \mathbf{b}(\mathsf{p}_v, r_v)$ is empty of any point of P. Intuitively, the bigger t is, the better $\mathcal{T}$ clusters P.

The ANN search procedure. Let q denote the query point. Initially, set v to be the root of $\mathcal{T}$, and $r_{\text{curr}} \leftarrow \infty$. The algorithm answer the ANN query by traversing down $\mathcal{T}$.

During the traversal, we first compute the distance $l = \|\mathsf{q} - \mathsf{rep}_v\|$. If this is smaller than r_{curr} (the distance to the current nearest neighbor found) then we update r_{curr} (and store the point realizing the new value of r_{curr}).

If $\|\mathsf{q} - p_v\| \leq \widehat{r}$, we continue the search recursively in the child containing P^v_{in}, where $\widehat{r} = (1 + t/2)r_v$ is the "middle" radius of the ring. Otherwise, we continue the search in the subtree containing P^v_{out}. The algorithm stops when reaching a leaf of $\mathcal{T}$, and returns the point realizing r_{curr} is the approximate nearest neighbor.

Intuition. If the query point q is outside the outer ball of a node v, it is so far from the points inside inner ball (i.e., P^v_{in}), and we can treat all of them as a single point (i.e., rep_v). On the other hand, if the query point q' is inside the inner ball, then it must have a neighbor nearby (i.e., a point of P^v_{in}), and all the points of P^v_{out} are far enough away that they can be ignored. Naturally, if the query point falls inside the ring, the same argumentation works (with slightly worst constants), using the middle radius as the splitting boundary in the search. See figure on the right.



Lemma 11.4.5 *Given a t-ring tree $\mathcal{T}$, one can answer $(1 + 4/t)$-approximate nearest neighbor queries, in $O(\mathtt{depth}(\mathcal{T}))$ time.*

Proof: Clearly, the query time is $O(\mathtt{depth}(\mathcal{T}))$. As for the quality of approximation, let π denote the generated search path in $\mathcal{T}$ and $\widehat{\mathsf{q}}$ denote the nearest neighbor to q in P. Furthermore, let w denote the last node in the search path π, such that $\widehat{\mathsf{q}} \in \mathsf{P}_w$. Clearly, if $\widehat{\mathsf{q}} \in P^w_{\text{in}}$, but we continued the search in P^w_{out}, then q is outside the middle sphere, and $\|\mathsf{q} - \widehat{\mathsf{q}}\| \geq (t/2)r_w$ (since this is the distance between the middle sphere and the inner sphere). Thus,

$$\|\mathsf{q} - \text{rep}_w\| \leq \|\mathsf{q} - \widehat{\mathsf{q}}\| + \|\widehat{\mathsf{q}} - \text{rep}_w\| \leq \|\mathsf{q} - \widehat{\mathsf{q}}\| + 2r_w,$$

since $\widehat{\mathsf{q}}, \text{rep}_w \in \mathbf{b}_w = \mathbf{b}(p_w, r_w)$. In particular,

$$\frac{\|\mathsf{q} - \text{rep}_w\|}{\|\mathsf{q} - \widehat{\mathsf{q}}\|} \leq \frac{\|\mathsf{q} - \widehat{\mathsf{q}}\| + 2r_w}{\|\mathsf{q} - \widehat{\mathsf{q}}\|} \leq 1 + \frac{4}{t}.$$

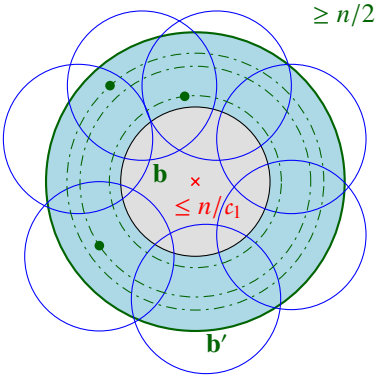
Namely, rep_w is a $(1 + 4/t)$-approximate nearest neighbor to q.

Similarly, if $\widehat{\mathsf{q}} \in P^w_{\text{out}}$, but we continued the search in P^w_{in}, then $\|\mathsf{q} - \widehat{\mathsf{q}}\| \geq (t/2)r_w$ and

$$\|\mathsf{q} - \text{rep}_w\| \leq \|\mathsf{q} - p_w\| + \|p_w - \text{rep}_w\| \leq (1 + t/2)r_w + r_w = (2 + t/2)r_w.$$

Thus, rep_w is a $\frac{(2+t/2)r_w}{(t/2)r_w}$-ANN of q. Namely, rep_w is a $(1 + 4/t)$-ANN of q. ■

In low dimensions, there is always a good separating ring. Indeed, consider the smallest ball $\mathbf{b} = \mathbf{b}(\mathsf{p}, r)$ that contains n/c_1 points of P, where c_1 is a sufficiently large constant. Let $\mathbf{b}'$ be the scaling of this ball by a factor of two. By a standard packing argument, the ring $\mathbf{b}' \setminus \mathbf{b}$ can be covered with $c = O(1)$ copies of $\mathbf{b}$, none of which can contain more than n/c_1 points of P. It follows, that by picking $c_1 = 3c$, we are guaranteed that at least half the points of P are outside $\mathbf{b}'$. Now, the ring can be split into $n/2$ empty rings (by taking a sphere that passes through each point inside the ring), and one of them would be of thickness at least r/n, and it would separate n/c points of P from the outer $n/2$ of P. Doing this efficiently requires trading off some constants, and some tedious details, as described in the following lemma.



Lemma 11.4.6 *Given a set P of n points in $\mathbb{R}^d$, one can compute a $(1/n)$-ring tree of P in $O(n \log n)$ time.*

Proof: The construction is recursive. Compute the ball $D = \mathbf{b}(p, r)$ that contains $\geq n/c$ points of P, such that $r \leq 2r_{\text{opt}}(P, n/c)$, where c is a constant to be determined shortly. We remind the reader that $r_{\text{opt}}(P, n/c)$ is the radius of the smallest ball that contains n/c points of P, and the ball D can be computed in linear time, by Lemma 1.3.1. Consider the ball D' of radius $2r$ centered at p. The ball D' can be covered by a hypercube S with side length $4r$. Furthermore, partition S into a grid such that every cell is of side length r/L, for $L = \lceil 16\sqrt{d} \rceil$. Every cell in this grid has diameter $\leq 4r\sqrt{d}/L \leq r/4 \leq r_{\text{opt}}(P, n/c)/2$. Thus, every grid cell can contain at most n/c points, since it can be covered with a ball of radius $r_{\text{opt}}(P, n/c)/4$. There are $M = \left(\frac{4r}{r/L}\right)^d = (4L)^d$ grid cells. Thus, D' contains at most $(4L)^d(n/c)$ points. Specifically, for $c = 2(4L)^d$ the ball D' contains at most $n/2$ points of P.

In particular, there must be a radius r' such that $r \leq r' \leq 2r$, and there is a $h \geq r/n$, such that the ring $\mathbf{b}(p, r' + h) \setminus \mathbf{b}(p, r')$ does not contain any points of P in its interior.

Indeed, sort the points of P inside $D' \setminus D$ by their distances from p. There are $n/2$ numbers in the range of distances $[r, 2r]$. As such, there must be an interval of length $r/(n/2 + 1)$ which is empty. And this empty range, corresponds to the empty ring.

Computing r' and h is done by computing the distance of each point from p, and partitioning the distance range $[r, 2r]$ into $2n$ equal length segments. In each segment, we register the point with minimum and maximum distance from c in this range. This can be done in linear time using the floor function. Next, scan those buckets from left to right. Observe, that the maximum length gap is realized by a maximum of one bucket together with a consecutive sequence of empty buckets, ending by the minimum of a non empty bucket. As such, the maximum length interval can be computed in linear time, and yield r' and h.

Thus, let v be the root of the new tree, set P^v_{in} to be $P \cap \mathbf{b}(p, r')$ and $P^v_{\text{out}} = P \setminus P^v_{\text{in}}$, store $\mathbf{b}_v = \mathbf{b}(p, r')$ and $p_v = p$. Continue the construction recursively on those two sets. Observe that $\left|P^v_{\text{in}}\right|, \left|P^v_{\text{out}}\right| \geq n/c$, where c is a constant. It follows that the construction time of the algorithm is $T(n) = O(n) + T\left(\left|P^v_{\text{in}}\right|\right) + T\left(\left|P^v_{\text{out}}\right|\right) = O(n \log n)$, and the depth of the resulting tree is $O(\log n)$. ■

Combining the above two lemmas, we get the following result.

Theorem 11.4.7 *Let P be a set of n points in $\mathbb{R}^d$. One can preprocess it in $O(n \log n)$ time, such that given a query point $\mathsf{q} \in \mathbb{R}^d$, one can return a $(1 + 4n)$-ANN of q in P in $O(\log n)$ time.*

11.5 Bibliographical notes

The presentation of the ring tree follows the recent work of Har-Peled and Mendel [HM06]. Ring trees are probably an old idea. A more elaborate but similar data-structure is described by Indyk and Motwani [IM98]. Of course, the property that "thick" ring separators exist, is inherently low dimensional, as the regular simplex in n dimensions demonstrates. One option is to allow the rings to contain points, and replicate the points inside the ring in both subtrees. As such, the size of the resulting tree is not necessarily linear. However, careful implementation yields linear (or small) size, see Exercise 11.6.1 for more details. This and several additional ideas are used in the construction of the cover tree of Indyk and Motwani [IM98].

Section 11.2 is a simplification of Arya *et al.* [AMN+98] work. Section 11.3 is also inspired to a certain extent by Arya *et al.* work, although it is essentially a simplification of Har-Peled and Mendel [HM06] data-structure to the case of compressed quadtrees. In particular, we believe that the data-structure presented is conceptually simpler than previously published work.

There is a huge literature on approximate nearest neighbor search, both in low and high dimensions in the theory, learning and database communities. The reason for this huge work lies in the importance of this problem, special input distributions, different computation models (i.e., I/O-efficient algorithms), search in high-dimensions, and practical efficiency.

Liner space. In the low dimensions, the seminal work of Arya *et al.* [AMN+98], mentioned above, was the first to offer linear size data-structure, with logarithmic query time, such that the approximation quality is specified with the query. The query time of Arya *et al.* is slightly worse than the running time of Theorem 11.3.3, since they maintain a heap of cells, always handling the cell closest to the query point. This results in query time $O(\varepsilon^{-d} \log n)$. It can be further improved to $O(1/\varepsilon^d \log(1/\varepsilon) + \log n)$ by observing that this heap has only very few delete-min, and many insertions. This observation is due to Duncan [Dun99].

Instead of having a separate ring-tree, Arya *et al.* rebalance the compressed quadtree directly. This results in nodes, which correspond to cells that have the shape of an annulus (i.e., the region formed by the difference between two canonical grid cells).

Duncan [Dun99] and some other authors offered data-structure (called the BAR-tree) with similar query time, but it is seems to be inferior, in practice, to Arya *et al.* work, for the reason that while the regions the nodes correspond to are convex, they have higher descriptive complexity, and it is harder to compute the distance of the query point to a cell.

Faster query time. One can improve the query time if one is willing to specify ε during the construction of the data-structure, resulting in a trade off between space for query time. In particular, Clarkson [Cla94] showed that one can construct a data-structure of (roughly) size $O(n/\varepsilon^{(d-1)/2})$, and query time $O(\varepsilon^{-(d-1)/2} \log n)$. Chan simplified and cleaned up this result [Cha98] and presented also some other results.

> **Details on Faster Query Time.** A set of points Q is $\sqrt{\varepsilon}$-*far* from a query point q, if the $\left\|p - c_Q\right\| \geq \operatorname{diam}(Q)/\sqrt{\varepsilon}$, where c_Q is some point of Q. It is easy to verify that if we partition space around c_Q into cones with central angle $O(\sqrt{\varepsilon})$ (this requires $O(1/\varepsilon^{(d-1)/2})$ cones), then the most extreme point of Q in such a cone ψ, furthest away from c_Q, is the $(1 + \varepsilon)$-approximate nearest neighbor for any query point inside ψ which is $\sqrt{\varepsilon}$-far. Namely, we precompute the ANN inside each cone, if the point is far enough. Furthermore, by careful implementation (i.e., grid in the angles space), we can decide, in constant time, which cone the query point lies in. Thus, using $O(1/\varepsilon^{(d-1)/2})$ space, we can answer $(1 + \varepsilon)$-ANN queries for q, if the query point is $\sqrt{\varepsilon}$-far, in constant time.
>
> Next, construct this data-structure for every set P_v, for $v \in \widehat{T}(\mathsf{P})$. This results in a data-structure of size $O(n/\varepsilon^{(d-1)/2})$. Given a query point q, we use the algorithm of Theorem 11.3.3, and stop as soon as for a node v, P_v is $\sqrt{\varepsilon}$-far, and then we use the secondary data-structure for P_v. It is easy to verify that the algorithm would stop as soon as $\operatorname{diam}(\square_v) = O(\sqrt{\varepsilon}\mathbf{d}_{\mathsf{P}}(\mathsf{q}))$. As such, the number of nodes visited would be $O(\log n + 1/\varepsilon^{d/2})$, and identical query time.
>
> Note, that we omitted the construction time (which requires some additional work to be done efficiently), and our query time is slightly worse than the best known. The interested reader can check out the work by Chan [Cha98], which is somewhat more complicated than what is outlined here.

The first to achieve $O(\log(n/\varepsilon))$ query time (using near linear space), was Har-Peled [Har01b], using space roughly $O(n\varepsilon^{-d} \log^2 n)$. This was later simplified and improved by Arya and Malamatos [AM02], which present a data-structure with the same query time,

and of size $O(n/\varepsilon^d)$. Those data-structure relies on the notion of computing approximate Voronoi diagrams and performing point location queries in those diagrams. By extending the notion of approximate Voronoi diagrams, Arya, Mount and Malamatos [AMM02] showed that one can answer $(1+\varepsilon)$-ANN queries in $O(\log(n/\varepsilon))$ time, using $O(n/\varepsilon^{(d-1)})$ space. On the other end of the spectrum, they showed that one can construct a data-structure of size $O(n)$ and query time $O(\log n + 1/\varepsilon^{(d-1)/2})$ (note, that for this data-structure $\varepsilon > 0$ has to be specified in advance). In particular, the later result breaks a space/query time tradeoff that all other results suffers from (i.e., the query time multiplied by the construction time has dependency of $1/\varepsilon^d$ on ε).

Practical Considerations Arya *et al.* [AMN+98] implemented their algorithm. For most inputs, it is essentially a *kd*-tree. The code of their library was carefully optimized and is very efficient. In particular, in practice, I would expect it to beat most of the algorithms mentioned above. The code of their implementation is available online as a library [AM98].

Higher Dimensions. All our results have exponential dependency on the dimension, in query and preprocessing time (although the space can be probably be made subexponential with careful implementation). Getting a subexponential algorithms requires a completely different techniques, and would be discussed in detail at some other point.

Stronger computation models. If one assume that the points have integer coordinates, in the range $[1, U]$, then approximate nearest-neighbor queries can be answered in (roughly) $O(\log\log U + 1/\varepsilon^d)$ time [AEIS99], or even $O(\log\log(U/\varepsilon))$ time [Har01b]. The algorithm of Har-Peled [Har01b] relies on computing a compressed quadtree of height $O(\log(U/\varepsilon))$, and performing fast point-location query in it. This only requires using the floor function and hashing (note, that the algorithm of Theorem 11.3.3 uses the floor function and hashing during the construction, but it is not used during the query). In fact, if one is allowed to slightly blowup the space (by a factor U^δ, where $\delta > 0$ is an arbitrary constant), the ANN query time can be improved to constant [HM04].

By shifting quadtrees, and creating $d+1$ quadtrees, one can argue that the approximate nearest neighbor must lie in the same cell (and of the "right" size) of the query point in one of those quadtrees. Next, one can map the points into a real number, by using the natural space filling curve associated with each quadtree. This results in $d+1$ lists of points. One can argue that a constant approximate neighbor must be adjacent to the query point in one of those lists. This can be later improved into $(1+\varepsilon)$-ANN by spreading $1/\varepsilon^d$ points. This simple algorithm is due to Chan [Cha02].

The reader might wonder why we bothered with a considerably more involved algorithm. There are several reasons: (i) This algorithm requires the numbers to be integers of limited length (i.e., $O(\log U)$ bits), and (ii) it requires shuffling of bits on those integers (i.e., for computing the inverse of the space filling curve) in constant time, and (iii) the assumption is that one can combine d such integers into a single integer and perform XOR on their bits in constant time. The last two assumptions are not reasonable when the input is made out of floating point numbers.

Further research. At least (and only) in low dimensions, the ANN problem seems to be essentially solved both in theory and practice (such proclamations are inherently dangerous, and should be taken with considerable amount of healthy skepticism). Indeed, for $\varepsilon > 1/\log^{1/d} n$, the current data structure of Theorem 11.3.3 provide logarithmic query time. Thus, ε has to be quite small for the query time to become bad enough that one would wish to speed it up.

Main directions for further research seems to be working on this problem in higher dimensions, and solving it in other computation models.

Surveys. A survey on approximate nearest neighbor search in high dimensions is by Indyk [Ind04]. In low dimensions, there is a survey by Arya and Mount [AM04].

11.6 Exercises

Exercise 11.6.1 (Better Ring Tree) [10 Points]

Let P be a set of n points in $\mathbb{R}^d$. Show how to build a ring tree, of linear size, that can answer $O(\log n)$-ANN queries in $O(\log n)$ time. [**Hint:** Show, that there is always a ring containing $O(n/\log n)$ points, such that it is of width w, and its interior radius is $O(w \log n)$. Next, build a ring tree, replicating the points in both children of this ring node. Argue that the size of the resulting tree is linear, and prove the claimed bound on the query time and quality of approximation.**]**

Chapter 12

Approximate Nearest Neighbor via Point-Location among Balls

> Today I know that everything watches, that nothing goes unseen, and that even wallpaper has a better memory than ours. It isn't God in His heaven that sees all. A kitchen chair, a coat-hanger a half-filled ash tray, or the wood replica of a woman name Niobe, can perfectly well serve as an unforgetting witness to every one of our acts.
>
> – – The tin drum, Gunter Grass

12.1 Hierarchical Representation of Points

In the following, it would be convenient to carry out our discussion in a more generalized settings than just low dimensional Euclidean space.

Definition 12.1.1 A *metric space* $\mathcal{M}$ is a pair $(X, \mathbf{d})$ where X is a set and $\mathbf{d} : X \times X \to [0, \infty)$ is a *metric*, satisfying the following axioms: (i) $\mathbf{d}(x, y) = 0$ iff $x = y$, (ii) $\mathbf{d}(x, y) = \mathbf{d}(y, x)$, and (iii) $\mathbf{d}(x, y) + \mathbf{d}(y, z) \geq \mathbf{d}(x, z)$ (triangle inequality).

For example, $\mathbb{R}^2$ with the regular euclidean distance is a metric space.

In the following, we are going to assume that we are provided with a black box, such that given two points $x, y \in X$, we can compute the distances $\mathbf{d}(x, y)$ in constant time.

12.1.1 Low Quality Approximation by HST

We will use the following special type of metric spaces:

Definition 12.1.2 Let P be a set of elements, and T a tree having the elements for P as leaves. The tree T defines a *hierarchically well-separated tree* (HST) over the points of P, if to each vertex $u \in T$ there is associated a label $\Delta_u \geq 0$, such that $\Delta_u = 0$ if and only if u is a leaf of T. The labels are such that if a vertex u is a child of a vertex v then $\Delta_u \leq \Delta_v$. The distance between two leaves $x, y \in T$ is defined as $\Delta_{\mathrm{lca}(x,y)}$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T.

If every internal node of T has exactly two children, we will refer to it as being a *binary HST* (BHST).

For convenience, we will assume that the underlying tree is binary (any HST can be converted to binary HST in linear time, while retaining the underlying distances). We will also associate with every vertex $u \in T$, an arbitrary leaf rep_u of the subtree rooted at u. We also require that $\mathrm{rep}_u \in \left\{ \mathrm{rep}_v \;\middle|\; v \text{ is a child of } u \right\}$.

A metric N is said to t-approximate the metric $\mathcal{M}$, if they are on the same set of points, and $\mathbf{d}_{\mathcal{M}}(u, v) \leq d_N(u, v) \leq t \cdot \mathbf{d}_{\mathcal{M}}(u, v)$, for any $u, v \in \mathcal{M}$.

It is not hard to see that any n-point metric is $(n-1)$-approximated by some HST.

Lemma 12.1.3 *Given a weighted connected graph G on n vertices and m edges, it is possible to construct, in $O(n \log n + m)$ time, a binary HST $\mathcal{H}$ that $(n-1)$-approximates the shortest path metric on G.*

Proof: Compute the minimum spanning tree of G in $O(n \log n + m)$ time, and let T denote this tree.

Sort the edges of T in non-decreasing order, and add them to the graph one by one. The HST is built bottom up. At each point we have a collection of HSTs, each corresponds to a connected component of the current graph. When an added edge merges two connected components, we merge the two corresponding HSTs into one by adding a new common root for the two HST, and labeling this root with the edge's weight times $n-1$. This algorithm is only slight variation on Kruskal algorithm, and has the same running time.

We next estimate the approximation factor. Let x, y be two vertices of G. Denote by e the first edge that was added in the process above that made x and y in the same connected component C. Note that at that point of time e is the heaviest edge in C, so $w(e) \leq d_G(x,y) \leq (|C|-1)\, w(e) \leq (n-1)\, w(e)$. Since $d_H(x,y) = (n-1)\, w(e)$, we are done. ■

Corollary 12.1.4 *For a set P of n points in $\mathbb{R}^d$, one can construct, in $O(n \log n)$ time, a BHST $\mathcal{H}$ that $(2n-2)$-approximates the distances of points in P. That is, for any $p, q \in P$, we have $\mathbf{d}_{\mathcal{H}}(p,q)/(2n-2) \leq \|pq\| \leq \mathbf{d}_{\mathcal{H}}(p,q)$.*

Proof: We remind the reader, that in $\mathbb{R}^d$, one can compute a 2-spanner for P of size $O(n)$, in $O(n \log n)$ time (see Theorem 3.2.1). Let G be this spanner, and apply Lemma 12.1.3 to this spanner. Let $\mathcal{H}$ be the resulting metric. For any $p, q \in P$, we have $\|pq\| \leq \mathbf{d}_{\mathcal{H}}(p,q) \leq (n-1)\mathbf{d}_G(p,q) \leq 2(n-1)\|pq\|$. ■

Corollary 12.1.4 is unique to $\mathbb{R}^d$ since for general metric spaces, no HST can be computed in subquadratic time, see Exercise 12.5.1.

Corollary 12.1.5 *For a set P of n points in a metric space $\mathcal{M}$, one can compute a HST $\mathcal{H}$ that $(n-1)$-approximates the metric $\mathbf{d}_{\mathcal{M}}$.*

12.2 ANN using Point-Location Among Balls

In the following, let P be a set of n points in $\mathcal{M}$, where $\mathcal{M}$ is a metric space.

Definition 12.2.1 For a set of balls $\mathcal{B}$ such that $\bigcup_{b \in \mathcal{B}} b = \mathcal{M}$ (i.e., one of the balls might be of infinite radius and it covers the whole space $\mathcal{M}$), and a query point $q \in \mathcal{M}$, the *target ball of q in $\mathcal{B}$*, denoted by $\odot_{\mathcal{B}}(q)$, is the smallest ball of $\mathcal{B}$ that contains q (if several equal radius balls contain q we resolve this in an arbitrary fashion).

Our objective, is to show that $(1+\varepsilon)$-ANN queries can be reduced to target ball queries, among a near linear size set of balls. But let us first start from a "silly" result, to get some intuition about the problem.

Lemma 12.2.2 *Let $\mathcal{B} = \cup_{i=-\infty}^{\infty} \mathcal{B}(P, (1+\varepsilon)^i)$, where $\mathcal{B}(P,r) = \cup_{p \in P} \mathbf{b}(p,r)$. For $q \in \mathbb{R}^d$, let $\mathbf{b} = \odot_{\mathcal{B}}(q)$, and let $p \in P$ be the center of $\mathbf{b}$. Then, p is $(1+\varepsilon)$-ANN to q.*

Proof: Let $\widehat{q}$ be the nearest neighbor to q in P, and let $r = \mathbf{d}_P(q)$. Let i be such that $(1+\varepsilon)^i < r \leq (1+\varepsilon)^{i+1}$. Clearly, $\text{radius}(\mathbf{b}) > (1+\varepsilon)^i$. On the other hand, $p \in \mathbf{b}(\widehat{q}, (1+\varepsilon)^{i+1})$. It follows that, $\|qp\| \leq (1+\varepsilon)^{i+1} \leq (1+\varepsilon)\mathbf{d}_P(q)$. Implying that p is $(1+\varepsilon)$-ANN to P. ■

Remark 12.2.3 Here is an intuitive construction of a set of balls of polynomial size, such that target queries answer $(1+\varepsilon)$-ANN correctly. Indeed, consider two points $u, v \in P$. As far as correctness, we care if the ANN returns either u or v, for a query point q, only if $\mathbf{d}_P(q) \in [\mathbf{d}_{\mathcal{M}}(u,v)/4, 2\mathbf{d}_{\mathcal{M}}(u,v)/\varepsilon]$ (for shorter distances, either u or v are the unique ANN, and for longer distances either one of them is ANN for the set $\{u, v\}$).

Next, consider a range of distances $[(1+\varepsilon)^i, (1+\varepsilon)^{i+1}]$ to be active, if there is $u, v \in P$, such that $\varepsilon(1+\varepsilon)^i \leq \mathbf{d}_{\mathcal{M}}(u,v) \leq 4(1+\varepsilon)^{i+1}/\varepsilon$. Clearly, the number of active intervals is $O(n^2 \varepsilon^{-1} \log(1/\varepsilon))$ (one can prove a better bound). Generate a ball for each point of P, for each active range. Clearly, the resulting number of balls is polynomial, and can be used to resolve ANN queries.

Getting the number of balls to be near linear, requires to be more careful, and the details are provided below.

12.2.1 Handling a Range of Distances

Definition 12.2.4 For a real number $r > 0$, a *near-neighbor data structure*, denoted by $\text{NNbr} = \text{NNbr}(P, r)$, is a data-structure, such that given a query point q, it can decide whether $\mathbf{d}_P(q) \leq r$, or $\mathbf{d}_P(q) > r$. If $\mathbf{d}_P(q) \leq r$, it also returns as a witness a point $p \in P$, such that $\mathbf{d}(p,q) \leq r$.

The data-structure $\text{NNbr}(P, r)$ can be realized by just a set n balls around the points of P of radius r, and performing target ball queries on this set. For the time being, the reader can consider $\text{NNbr}(P, r)$ as just being this set of n balls.

Definition 12.2.5 One can in fact, resolve ANN queries on a range of distances, $[a, b]$, by building NNbr data structures, with exponential jumps on this range. Formally, let $\mathcal{N}_i = \text{NNbr}(P, r_i)$, where $r_i = (1+\varepsilon)^i a$, for $i = 0, \ldots, M-1$, where $M = \lceil \log_{1+\varepsilon}(b/a) \rceil$. And let $\mathcal{N}_M = \text{NNbr}(P, M)$, where $r_M = b$. We will denote this set of data-structures by $\widehat{\mathcal{I}}(P, a, b, \varepsilon) = \{\mathcal{N}_0, \ldots, \mathcal{N}_M\}$. We refer to $\widehat{\mathcal{I}}$ is *interval near-neighbor* data structure.

Lemma 12.2.6 *Given P as above, and parameters $a \le b$ and $\varepsilon > 0$. We have: (i) $\widehat{\mathcal{I}}(P, a, b, \varepsilon)$ is made out of $O(\varepsilon^{-1} \log(b/a))$* NNbr *data structures, and (ii) $\widehat{\mathcal{I}}(P, a, b, \varepsilon)$ contains $O(\varepsilon^{-1} n \log(b/a))$ balls overall.*

Furthermore, one can decide if either of the following options holds: (i) $\mathbf{d}_P(q) < a$, (ii) $\mathbf{d}_P(q) > b$, (iii) or return a number r and a point $p \in P$, such that $\|pq\| \le \mathbf{d}_P(q) \le (1+\varepsilon)\|pq\|$. This requires two NNbr *queries if (i) or (ii) holds, and $O(\log(\varepsilon^{-1} \log(b/a)))$ otherwise.*

Proof: Given a query point q, we first check if $\mathbf{d}_P(q) \le a$, by querying $\mathcal{N}_0$, and if so, the algorithm returns "$\mathbf{d}_P(q) \le a$". Otherwise, we check if $\mathbf{d}_P(q) > b$, by querying $\mathcal{N}_M$, and if so, the algorithm returns "$\mathbf{d}_P(q) > b$".

Otherwise, let $X_i = 1$ if and only if $\mathbf{d}_P(q) \le r_i$, for $i = 0, \ldots, m$. We can determine the value of X_i by performing a query in the data-structure $\mathcal{N}_i$. Clearly, $X_0, X_1, \ldots, X_M$ is a sequence of zeros, followed by a sequence of ones. As such, we can find the i, such that $X_i = 0$ and $X_{i+1} = 1$, by performing a binary search. This would require $O(\log M)$ queries. In this case, we have that $r_i < \mathbf{d}_P(q) \le r_{i+1} \le (1+\varepsilon) r_i$. Namely, the ball in $\text{NNbr}(P, r_{i+1})$ covering q, corresponds to $(1+\varepsilon)$-ANN to q.

To get the state bounds, observe that by the Taylor's expansion $\ln(1+x) = x - x^2/2 + x^3/3 + \cdots \ge x/2$, for $x \ge 1$. Thus,

$$M = \lceil \log_{1+\varepsilon}(b/a) \rceil = O\left(\frac{\ln(b/a)}{\ln(1+\varepsilon)} \right) = O(\log(b/a)/\varepsilon),$$

since $1 \ge \varepsilon > 0$. ■

Corollary 12.2.7 *Let P be a set of n points in $\mathcal{M}$, and let $a < b$ be real numbers. For a query point $q \in \mathcal{M}$, such that $\mathbf{d}_P(q) \in [a, b]$, the target query over the set of balls* $\text{NNbr}(P, a, b, \varepsilon)$ *returns a ball centered at $(1+\varepsilon)$-ANN to q.*

Lemma 12.2.6 implies that we can "cheaply" resolve $(1+\varepsilon)$-ANN over intervals which are not too long.

Definition 12.2.8 For a set P of n points in a metric space $\mathcal{M}$, let $\mathcal{U}_{\text{balls}}(P, r) = \bigcup_{p \in P} \mathbf{b}(p, r)$ denote the *union of balls of radius r* around the points of P.

Lemma 12.2.9 *Let Q be a set of m points in $\mathcal{M}$ and $r > 0$ a real number, such that $\mathcal{U}_{\text{balls}}(Q, r)$ is a connected set. Then: (i) Any two points $p, q \in Q$ are in distance $\le 2r(m-1)$ from each other. (ii) For $q \in \mathcal{M}$ a query point, such that $\mathbf{d}_Q(q) > 2mr/\delta$, then any point of Q is $(1+\delta)$-ANN of q.*

Proof: (i) Since $\mathcal{U}_{\text{balls}}(Q, r)$ is a connected set, there is a path of length $\le (m-1)2r$ between any two points x, y of P. Indeed, consider the graph G, which connects two vertices $u, v \in P$, if $\mathbf{d}_{\mathcal{M}}(u, v) \le 2r$. Since $\mathcal{U}_{\text{balls}}(Q, r)$ is a connected set, it follows that G is connected. As such, there is a path of length $m-1$ between x and y in G. This corresponds to a path of length $\le (m-1)2r$ connecting x to y in $\mathcal{M}$, and this path lies inside $\mathcal{U}_{\text{balls}}(Q, r)$.

(ii) For any $p \in Q$, we have

$$\frac{2mr}{\delta} \le \mathbf{d}_{\mathcal{M}}(q, p) \le \mathbf{d}_{\mathcal{M}}(q, \widehat{q}) + \mathbf{d}_{\mathcal{M}}(\widehat{q}, p) \le \mathbf{d}_{\mathcal{M}}(q, \widehat{q}) + 2mr \le (1+\delta)\mathbf{d}_{\mathcal{M}}(q, \widehat{q}),$$

where $\widehat{q}$ is the nearest-neighbor of q in Q. ■

Lemma 12.2.9 implies that for faraway query points, a cluster points Q which are close together can be treated as a single point.

12.2.2 The General Case

Theorem 12.2.10 *Given a set P of n points in $\mathcal{M}$, then one can construct data-structures $\mathcal{D}$ that answers $(1+\varepsilon)$-ANN queries, by performing $O(\log(n/\varepsilon))$* NNbr *queries. The total number of balls stored at $\mathcal{D}$ is $O(n\varepsilon^{-1} \log(n/\varepsilon))$.*

Let $\mathcal{B}$ be the set of all the balls stored at $\mathcal{D}$. Then a target query on $\mathcal{B}$ answers $(1+\varepsilon)$-ANN query.

Proof: We are going to build a tree $\mathcal{D}$, such that each node v would have an interval near-neighbor data-structure $\widehat{\mathcal{I}}_v$ associated with it. As we traverse down the tree, we will use those data-structure to decide to what child to continue the search into.

Compute the minimum value $r > 0$ such that $\mathcal{U}_{\text{balls}}(P, r)$ is made out of $\lceil n/2 \rceil$ connected components. We set $\widehat{\mathcal{I}}_{\text{root}(T)} = \widehat{\mathcal{I}}(P, r, R, \varepsilon/4)$, where $R = 2c\mu nr/\varepsilon$, μ is a global parameter and $c > 1$ is an appropriate constant, both to be determined shortly. For each connected component $\mathcal{C}$ of $\mathcal{U}_{\text{balls}}(P, r)$, we build recursively a tree for $\mathcal{C} \cap P$ (i.e., the points corresponding to $\mathcal{C}$), and hung it on root(T). Furthermore, from each such connected component $\mathcal{C}$, we pick one representative point $q \in \mathcal{C} \cap P$, and let Q be this set

of points. We also build (recursively) a tree for Q, and hang it on root(T). We will refer to the child of root(T) corresponding to Q, as the *outer child* of root(T).

Given a query point $q \in \mathcal{M}$, use $\widehat{\mathcal{I}}_{\text{root}(T)} = \widehat{\mathcal{I}}(P, r, R, \varepsilon/4)$ to determine, if $\mathbf{d}_P(q) \le r$. If so, we continue the search recursively in the relevant child built for the connected component of $\mathcal{U}_{\text{balls}}(P, r)$ containing q (we know which connected component it is, because $\widehat{\mathcal{I}}_{\text{root}(T)}$ also returns a point of P in distance $\le r$ from q). If $\mathbf{d}_P(q) \in [r, R]$, then we will find its $(1+\varepsilon)$-ANN from $\widehat{\mathcal{I}}_{\text{root}(T)}$. 'Otherwise, $\mathbf{d}_P(q) > R$, and we continue the search recursively in the outer child of root(T).

Observe, that in any case, we continue the search on a set of balls of size $\le n/2 + 1$. As such, after number of steps $\le \log_{3/2} n$ the search halts.

correctness. If during the search the algorithm traverse from a node v down to one of the connected components which is a child of $\mathcal{U}_{\text{balls}}(P_v, r_v)$, then no error is introduced, where P_v is the point set used in constructing v, and r_v is the value of r used in the construction. If the query is resolved by $\widehat{\mathcal{I}}_v$, then a $(1 + \varepsilon/4)$ error is introduced into the quality of approximation. If the algorithm continues the search in the outer child of v, then an error of $1 + \delta_v$ is introduced in the answer, where $\delta_v = \varepsilon/(c\mu)$, by Lemma 12.2.9 (ii). Thus, the overall quality of the ANN returned in the worst case is

$$t \quad \le \quad \left(1 + \frac{\varepsilon}{4}\right) \prod_{i=1}^{\log_{3/2} n} \left(1 + \frac{\varepsilon}{c\mu}\right) \le \exp\left(\frac{\varepsilon}{4}\right) \prod_{i=1}^{\log_{3/2} n} \exp\left(\frac{c\varepsilon}{c\mu}\right),$$

since $x \le e^x$ for $x \le 1$. Thus, setting $\mu = \left\lceil \log_{3/2} n \right\rceil$ and c to be a sufficiently large constant, we have $t \le \exp\left(\varepsilon/4 + \sum_{i=1}^{\log_{3/2} n} \varepsilon/c\mu\right) \le \exp(\varepsilon/2) \le 1 + \varepsilon$, since $\varepsilon < 1/2$. We used the fact that $e^x \le (1 + 2x)$ for $x \le 1/2$, as can be easily verified.

Number of queries. As the search algorithm proceeds down the tree $\mathcal{D}$, at most two NNbr queries are performed at each node. At last node of the traversal, the algorithm performs $O(\log(\varepsilon^{-1} \log(n/\varepsilon))) = O(\log(n/\varepsilon))$ queries, by Lemma 12.2.6

Number of balls. We need a new interpretation of the construction algorithm. In particular, let $\mathcal{H}$ be the HST constructed for P using the exact distances. It is easy to observe, that a connected component of $\mathcal{U}_{\text{balls}}(P, r)$ is represented by a node v and its subtree in $\mathcal{H}$. In particular, the recursive construction for each connected component, is essentially calling the algorithm recursively on a subtree of $\mathcal{H}$. Let V be the set of nodes of $\mathcal{H}$ which represent connected components of $\mathcal{U}_{\text{balls}}(P, r)$.

Similarly, the outer recursive call, can be charged to the upper tree of $\mathcal{H}$, having the nodes of V for leafs. Indeed, the outer set of points, is the set $\text{rep}(V) = \left\{ \text{rep}_v \;\middle|\; v \in V \right\}$. Let $\widehat{L}$ be this collection of subtrees of $\mathcal{H}$. Clearly, those subtrees are not disjoint in their vertices, but they are disjoint in their edges. The total number of edges is $O(n)$, and $\left|\widehat{L}\right| \ge n/2$.

Namely, we can interpret the algorithm (somewhat counter intuitively) as working on $\mathcal{H}$. At every stage, we break the current HST into subtrees, and recursively continue the construction on those connected subtrees.

In particular, for a node $v \in T$, let n_v be the number of children of v. Clearly, $|P_v| = O(n_v)$, and since we can charge each such child to the fact that we are disconnecting the edges of $\mathcal{H}$ from each other, we have that $\sum_{v in \mathcal{D}} n_v = O(n)$.

At a node $v \in \mathcal{D}$, we have that the data-structure $\widehat{\mathcal{I}}_v$ requires storing $m_v = O(\varepsilon^{-1} |P_v| \log(R_v/r_v)) = O(\varepsilon^{-1} n_v \log(\mu n_v/\varepsilon))$ balls. In particular, $m_v = O(\varepsilon^{-1} n_v \log(\mu n_v/\varepsilon))$. We conclude, that the overall number of balls stored in $\mathcal{D}$ is

$$\sum_{v \in \mathcal{D}} O\left(\frac{n_v}{\varepsilon} \log \frac{\mu n_v}{\varepsilon}\right) = O\left(\frac{n}{\varepsilon} \log \frac{n \log n}{\varepsilon}\right) = O\left(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon}\right).$$

A single target query. The claim that a target query on $\mathcal{B}$ answers $(1 + \varepsilon)$-ANN query on P, follows by an inductive proof on the algorithm execution. Indeed, if the algorithm is at node v, and let $\mathcal{B}_v$ be the set of balls stored in the subtree of v. We claim, that if the algorithm continue the search in w, then all the balls of $U = \mathcal{B}_v \setminus \mathcal{B}_w$ are not relevant for the target query.

Indeed, if w is the outer child of v, then all the balls in U are too small, and none of them contains q. Otherwise, $q \in \mathcal{U}_{\text{balls}}(P_v, r_V)$. As such, all the balls of $\mathcal{B}_v$ that are bigger than the balls of $\mathcal{U}_{\text{balls}}(P_v, r_v)$, and as such they can be ignored. Furthermore, all the other balls stored in other children of v, are of radius $\le r_v$, and are not in the same connected component as $\mathcal{U}_{\text{balls}}(P_w, r_v)$ in $\mathcal{U}_{\text{balls}}(P_v, r_v)$, as such, none of them is relevant for the target query.

The only other case, is when the algorithm stop the search at v. But at this case, we have $r_v \le \mathbf{d}_P(q) \le R_v$, and then all the balls in the children of v are either too big (i.e., the balls stored at the outer child are of radius $> R_v$), or too small (i.e., the balls stored at the regular children are of radius $< r_v$). Thus, only the balls of $\widehat{\mathcal{I}}(P_v, r_v, R_v, \varepsilon/4)$ are relevant, and there we know that the returend ball is a $(1 + \varepsilon/4)$-ANN by Corollary 12.2.7.

Thus, the point returned by the target query on $\mathcal{B}$, is identical to running the search algorithm on $\mathcal{D}$, and as such, by the above prove, the result is correct. ■

12.2.2.1 Efficient Construction

Theorem 12.2.10 does not provide any bounds on the construction time, since it requires quadratic time in the worst case.

Lemma 12.2.11 *Given a set P of n points in $\mathcal{M}$ and $\mathcal{H}$ be a HST of P that t-approximates $\mathcal{M}$. Then one can construct data-structures $\mathcal{D}$ that answers $(1+\varepsilon)$-ANN queries, by performing $O(\log(n/\varepsilon))$ NNbr queries. The total number of balls stored at $\mathcal{D}$ is $O(n\varepsilon^{-1}\log(tn/\varepsilon))$.*

The construction time is $O(n\varepsilon^{-1}\log(tn/\varepsilon))$.

Proof: We reimplement the algorithm of Theorem 12.2.10, by doing the decomposition directly on the HST $\mathcal{H}$. Indeed, let $U = \left\{\Delta_v \;\middle|\; v \in \mathcal{H}, \text{and } v \text{ is an internal node}\right\}$. Let ℓ be the median value of U. Let V be the set of all the nodes v of $\mathcal{H}$, such that $v \in V$, if $\Delta_v \le \ell$ and $\Delta_{\overline{p}(v)} > \ell$. Next, build an $\widehat{\mathcal{I}} = \widehat{\mathcal{I}}(P, r, R)$, where

$$r = \ell/(2tn) \text{ and } R = (2cnr\log n)/\varepsilon, \tag{12.1}$$

where c is a large enough constant. As in the algorithm of Theorem 12.2.10, the set V breaks $\mathcal{H}$ into $\lceil n/2\rceil + 1$ subtrees, and we continue the construction on each such connected component. In particular, for the new root node we create, set $r_{\mathrm{root}(\mathcal{D})} = r$, $R_{\mathrm{root}(\mathcal{D})} = R$, and $\ell_{\mathrm{root}(\mathcal{D})} = \ell$.

Observe, that for a query point $q \in \mathcal{M}$, if $\mathbf{d}_Q(p) \le r$, then $\widehat{\mathcal{I}}$ would return a ball, which in turn would correspond to a point stored in one of the subtrees rooted at a node $v \in V$. Let $\mathcal{C}$ be the connected component of $\mathcal{U}_{\mathrm{balls}}(P, r)$ that contains q, and let $Q = P \cap \mathcal{C}$. It is easy to verify that $Q \subseteq P_v$. Indeed, since $\mathcal{H}$ t-approximates $\mathbf{d}_{\mathcal{M}}$, and Q is a connected component of $\mathcal{U}_{\mathrm{balls}}(P, r)$, it follows that Q must be in the same connected component of $\mathcal{H}$, when considering distances $\le t \cdot r < \ell$. But such a connected component of $\mathcal{H}$, is no more than a node $v \in \mathcal{H}$, and the points of P stored in the subtree v in $\mathcal{H}$. However, such a node v is either in V, or one of its ancestors is in V.

For a node v, the number of balls of $\widehat{\mathcal{I}}_V$ is $O(\varepsilon^{-1} n_v \log((\log n) n_v y/\varepsilon))$. Thus, the overall number of balls in the data-structure is as claimed.

As for the construction time, it is dominated by the size of $\widehat{\mathcal{I}}$ data-structures, and as such the same bound holds. ■

Using Corollary 12.1.4 with Lemma 12.2.11 we get the following.

Theorem 12.2.12 *Let P be a set of n points in $\mathbb{R}^d$, and $\varepsilon > 0$ a parameter. One can compute an a set of $O(n\varepsilon^{-1}\log(tn/\varepsilon))$ balls (the same bound holds for the running time), such that a ANN can be resolved by a target ball query on this set of balls.*

Alternatively, one can construct a data-structure where $(1+\varepsilon)$-ANN can be resolved by $O(\log(n/\varepsilon))$ NNbr queries.

12.3 ANN using Point-Location Among Approximate Balls

While the results of Theorem 12.2.10 and Theorem 12.2.12 might be surprising, they can not be used immediately to solve ANN, since they reduce the ANN problem into answering near neighbor queries, which seems to be also a hard problem to solve efficiently.

The key observation is, that we do not need to use exact balls. We are allowed to slightly deform the balls. This approximate near neighbor problem is considerably easier.

Definition 12.3.1 For a ball $\mathbf{b} = \mathbf{b}(p, r)$, a set $\mathbf{b}_\approx$ is an $(1+\varepsilon)$-*approximation to* $\mathbf{b}$, if $\mathbf{b} \subseteq \mathbf{b}_\approx \subseteq \mathbf{b}(p, (1+\varepsilon)r)$.

For a set of balls $\mathcal{B}$, the set $\mathcal{B}_\approx$ is an $(1+\varepsilon)$-*approximation* to $\mathcal{B}$, if for any ball $\mathbf{b} \in \mathcal{B}$ there is a corresponding $(1+\varepsilon)$-approximation $\mathbf{b}_\approx \in \mathcal{B}_\approx$. For a set $\mathbf{b}_\approx \in \mathcal{B}_\approx$, let $\mathbf{b} \in \mathcal{B}$ denote the ball corresponding to $\mathbf{b}_\approx$, $r_{\mathbf{b}}$ be the radius of $\mathbf{b}$, and let $p_{\mathbf{b}} \in P$ denote the center of $\mathbf{b}$.

For a query point $q \in \mathcal{M}$, the *target set* of $\mathcal{B}_\approx$ in q, is the set $\mathbf{b}_\approx$ of $\mathcal{B}_\approx$ that contains q and has the smallest radius $r_{\mathbf{b}}$.

Luckily, the interval near-neighbor data-structure still works in this settings.

Lemma 12.3.2 *Let $\mathcal{I}_\approx = \mathcal{I}_\approx(P, r, R, \varepsilon/16)$ be a $(1+\varepsilon/16)$-approximation to $\widehat{\mathcal{I}}(P, r, R, \varepsilon/16)$. For a query point, $q \in \mathcal{M}$, if $\mathcal{I}_\approx$ returns a ball centered at $p \in P$ of radius α, and $\alpha \in [r, R]$ then p is $(1+\varepsilon/4)$-ANN to q.*

Proof: The data-structure of $\mathcal{I}_\approx$ returns p as an ANN to q, if and only if there are two consecutive indices, such that q is inside the union of the approximate balls of $\mathcal{N}_{i+1}$ but not inside the balls of $\mathcal{N}_i$. Thus, $r(1+\varepsilon/16)^i \le \mathbf{d}_P(q) \le \mathbf{d}(p, q) \le r(1+\varepsilon/16)^{i+1}(1+\varepsilon/16) \le (1+\varepsilon/4)r$. Thus p is indeed $(1+\varepsilon/4)$-ANN. ■

Lemma 12.3.3 *Let P be a set of n points in a metric space $\mathcal{M}$, and let $\mathcal{B}$ be a set of balls with centers at P, computed by the algorithm of Lemma 12.2.11, such that one can answer $(1+\varepsilon/16)$-ANN queries on P, by performing a single target query in $\mathcal{B}$.*

Let $\mathcal{B}_\approx$ be a $(1+\varepsilon/16)$-approximation to $\mathcal{B}$. A target query on $\mathcal{B}_\approx$, for a query point q, returns a $(1+\varepsilon)$-ANN to q in P.

Proof: Let $\mathcal{D}$ be the tree computed by Lemma 12.2.11. We prove the correctness of the algorithm by an inductive proof over the height of $\mathcal{D}$, similar in nature to the proofs of Lemma 12.2.11 and Theorem 12.2.10.

Indeed, for a node $v \in \mathcal{D}$, let $\mathcal{I}_{\approx v} = \mathcal{I}_{\approx}(P_v, r_v, R_v, \varepsilon/16)$ be the set of $(1+\varepsilon/16)$-approximate balls of $\mathcal{B}_{\approx}$ that corresponds to the set of balls stored in $\widehat{\mathcal{I}}_v = \widehat{\mathcal{I}}(P_v, r_v, R_v, \varepsilon/4)$. If the algorithm stops at v, we know by Lemma 12.3.2 that we returned a $(1+\varepsilon/4)$-ANN to q in P_v.

Otherwise, if we continued the search into the outer child of v using $\widehat{\mathcal{I}}_v$, then we would also continue the search into the this node when using $\mathcal{I}_{\approx v}$. As such, by induction the result is correct.

Otherwise, we continue the search into a node w, where $q \in \mathcal{U}_{\text{balls}}(P_w, (1+\varepsilon/16)r_v)$. We observe that because of the factor 2 slackness of Eq. (12.1), we are guaranteed to continue the search in the right connected component of $\mathcal{U}_{\text{balls}}(P_v, \ell_v)$.

Thus, the quality of approximation argument of Lemma 12.2.11 and Theorem 12.2.10 still holds, and require easy adaption to this case (we omit the straightforward by tedios details). We conclude, that the result returned is correct. ■

12.4 Bibliographical notes

Finite metric spaces would receive more attention later in the course. HSTs, despite their simplicity are a powerful tool in giving a compact (but approximate) representation of metric spaces.

Indyk and Motwani observed that ANN can be reduced to small number of near neighbor queries (i.e., this is the PLEB data-structure of [IM98]). In particular, the elegant argument in Remark 12.2.3 is due to Indyk (personal communications). Indyk and Motwani also provided a rather involved reduction showing that a near linear number of balls is sufficient. A considerably simpler reduction was provided by Har-Peled [Har01b], and we loosely follow his exposition in Section 12.2, although our bound on the number of balls is better by a logarithmic factor than the bounded provided by Har-Peled. This improvement was pointed out by Sabharwal *et al.* [SSS02]. They also provide a more involved construction, which achieves a linear dependency on n.

Open problems. The argument showing that one can use approximate near-neighbor data-structure instead of exact (i.e., Lemma 12.3.3), is tedious and far from being elegant; see Exercise 12.5.2. A natural question for further research, is to try and give a simple concrete condition on the set of balls, such that using approximate near neighbor data-structure still give the correct result. Curretly, it seems that what we need is somekind of separability property. However, it would be nice to give a simpelr direct condition.

As mentioned above, Sabharwal *et al.* [SSS02] showed a reduction from ANN to linear number of balls (ignoring the dependency on ε). However, it seems like a simpler construction should work.

12.5 Exercises

Exercise 12.5.1 (Lower bound on computing HST) [10 Points]

Show, that by adversarial argument, that for any $t > 1$, we have: Any algorithm computing a HST $\mathcal{H}$ for n points in a metric space $\mathcal{M}$, that t approximates $\mathbf{d}_{\mathcal{M}}$, must in the worst case inspect all $\binom{n}{2}$ distances in the metric. Thus, computing a HST requires quadratic time in the worst case.

Exercise 12.5.2 (No direct approximation to ball set) [10 Points]

Lemma 12.3.3 is not elegant. A more natural conjecture would be the following:

> **Conjecture 12.5.3** *Let P be a set of n points in the plane, and let $\mathcal{B}$ be a set of balls such that one can answer $(1+\varepsilon/c)$-ANN queries on P, by performing a single target query in $\mathcal{B}$. Now, let $\mathcal{B}_{\approx}$ be a $(1+\varepsilon/c)$-approximation to $\mathcal{B}$. Then a target query on $\mathcal{B}_{\approx}$, answers $(1+\varepsilon)$-ANN on P, for c large enough constant.*

Give a counter example to the above conjecture, showing that it is incorrect.

Chapter 13

Approximate Voronoi Diagrams

> "She had given him a smile, first because that was more or less what she was there for, and then because she had never seen him before and she had a prejudice in favor of people she did not know."
>
> – – The roots of heaven, Romain Gary

13.1 Introduction

A Voronoi diagram of a point set $P \subseteq \mathbb{R}^d$ is a partition of space into regions, such that the cell of $p \in P$, is $V(p, P) = \left\{ x \;\middle|\; \|xp\| \leq \|xp'\| \text{ for all } p' \in P \right\}$. Vornoi diagrams are a powerful tool and have numerous applications [Aur91].

One problem with Vornoi diagrams is that their descriptive complexity is $O(n^{\lceil d/2 \rceil})$ in $\mathbb{R}^d$, in the worst case. See Figure 13.1 for an exmaple in three dimensions. It is a natrual question to ask, whether one can reduce the complexity to linear (or near linear) by allow some approximation.

Definition 13.1.1 (Approximate Voronoi Diagram.) Given a set P of n points in $\mathbb{R}^d$, and parameter $\varepsilon > 0$, an $(1+\varepsilon)$-*approximated Voronoi diagram* of P, is a partition $\mathcal{V}$ of space into regions, such that for any region $\varphi \in \mathcal{V}$, there is an associated point $\mathrm{rep}_\varphi \in P$, such that for any $x \in \varphi$, we have that rep_φ is a $(1+\varepsilon)$-ANN for x in P. We will refer to $\mathcal{V}$ as $(1+\varepsilon)$-*AVD*.

13.2 Fast ANN in $\mathbb{R}^d$

In the following, assume P is a set of points contained in the hypercube $[0.5 - \varepsilon/d, 0.5 + \varepsilon/d]^d$. This can be easily guaranteed by affine linear transformation T which preserves relative distances (i.e., computing the ANN for q on P, is equivalent to computing the ANN for $T(q)$ on $T(P)$).

In particular, if the query point is outside the unit hypercube $\left[0, 1\right]^d$ then any point of P is $(1+\varepsilon)$-ANN, and we are done. Thus, we need to answer ANN only for points inside the unit hypercube.

We are going to use the reduction we saw between ANN and target queries among balls. In particular, one can compute the set of (exact) balls of Lemma 12.3.3 using the algorithm of Theorem 12.2.12. Let $\mathcal{B}$ be this set of balls. This takes $O(n\varepsilon^{-1}\log(n/\varepsilon))$ time. Next, we approximate each ball $\mathbf{b} \in \mathcal{B}$ of radius r, by the set of grid cells of G_{2^i} that intersects it, where $\sqrt{d}2^i \leq (\varepsilon/16)r$; namely, $i = \left\lfloor \lg(\varepsilon r/\sqrt{d}) \right\rfloor$. In particular, let $\mathbf{b}_\approx$ denote the region corresponding to the grid cells intersected by $\mathbf{b}$. Clearly, $\mathbf{b}_\approx$ is an approximate ball of $\mathbf{b}$.

Let $\mathcal{B}_\approx$ be the resulting set of approximate balls for $\mathcal{B}$, and let C' be the associated (multi) set of grid cells formed by those balls. The multi-set C' is a collection of canonical grid cells from different resolutions. We create a set out of C', by picking from all the instances of the same cell $\square \in C'$, the instance associated with the smallest ball of $\mathcal{B}$. Let C be the resulting set of canonical grid cells.

Thus, by Lemma 12.3.3, answering $(1+\varepsilon)$-ANN is no more than performing a target query on $\mathcal{B}_\approx$. This in turn, just finding the smallest canonical grid cell of C which contains the query point q. In previous lecture, we showed that one can construct a compressed quadtree $\widehat{T}$ that has all the nodes of C appear as nodes of $\widehat{T}$. The construction of this compressed quadtree takes $O(|C|\log|C|)$ time, by Lemma 2.2.5.

By performing a point-location query in $\widehat{T}$, we can compute the smallest grid cell of C that contains the query point in $O(\log|C|)$ time. Specificly, the query returns a leaf v of $\widehat{T}$. We need to find, along the path of v to the root, the smallest ball associated with those nodes. This information can be propogated down the compressed quadtree during the preprocessing stage, and as such, once we have v at hand, one can answer the ANN query in constant time. We conclude:

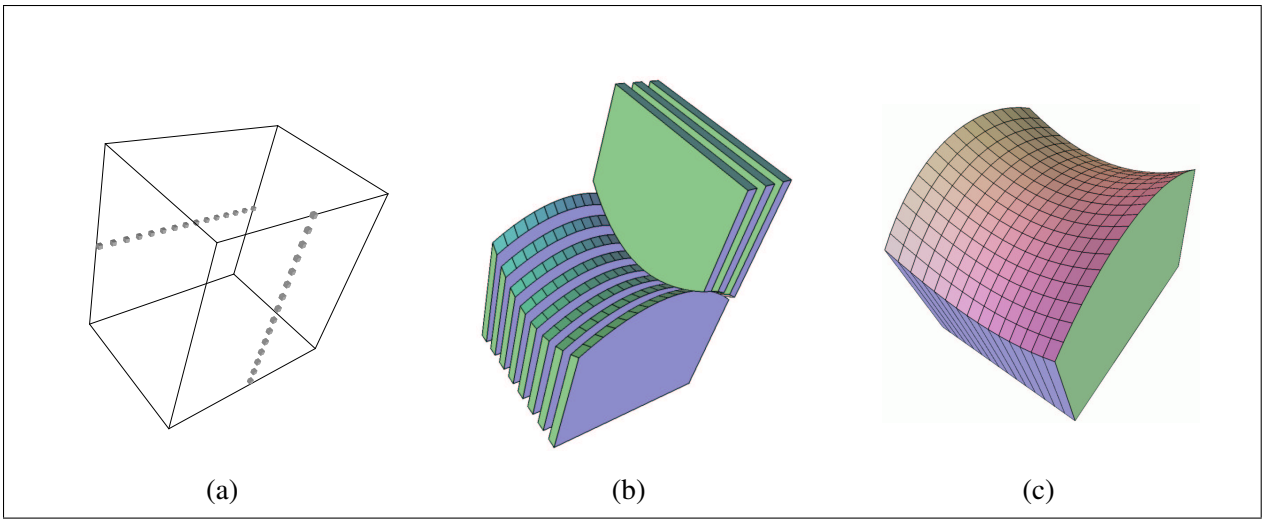


Figure 13.1: (a) The point-set in 3D inducting a Voronoi diagram of quadratic complexity. (b) Some cells in this Voronoi diagram. Note that the cells are thin and flat, and every cell from the lower part touches the cells on the upper part. (c) The contact surface between the two parts of the Voronoi diagram has quadratic complexity, and thus the Voronoi diagram itself has quadratic complexity.

Theorem 13.2.1 *Let P be a set of n points in $\mathbb{R}^d$. One can build a compressed quadtree $\widehat{T}$, in $O(n\varepsilon^{-d}\log^2(n/\varepsilon))$ time, of size $O(n\varepsilon^{-d}\log(n/\varepsilon))$, such that $(1+\varepsilon)$-ANN query on P, can be answered by a single point-location query in $\widehat{T}$. Such a point-location query takes $O(\log(n/\varepsilon))$ time.*

Proof: The construction is described above. We only need to prove the bounds on the running time. It takes $O(n\varepsilon^{-1}\log(n/\varepsilon))$ time to compute $\mathcal{B}$. For every such ball, we generate $O(1/\varepsilon^d)$ canonical grid cells that cover it. Let C be the resulting set of grid cells (after filtering multiple instance of the same grid cell). Naively, the size of C is bounded by $O\left(|\mathcal{B}|/\varepsilon^d\right)$. However, it is easy to verify that $\mathcal{B}$ has a large number of balls of similar size centered at the same point (because the set $\widehat{\mathcal{I}}$ has a lot of such balls). In particular, if we have the set of balls $\widehat{\mathcal{I}}(\{p\}, r, 2r, \varepsilon/16)$, it requires only $O(1/\varepsilon^d)$ canonical grid cells to approximate it. Thus, we can bound $|C|$ by $N = O\left(|\mathcal{B}|/\varepsilon^{d-1}\right) = O(n\varepsilon^{-d}\log(n/\varepsilon))$. We can also compute C in this time, by careful implementation (we omit the straightforward and tedious details).

Constructing $\widehat{T}$ for C takes $O(N\log N) = O(n\varepsilon^{-d}\log^2(n/\varepsilon))$ time (Lemma 2.2.5). The resulting compressed quadtree is of size $O(N)$. Point location queries at $\widehat{T}$ takes $O(\log N) = O(\log(n/\varepsilon))$ time. Given a query point, and the leaf v, such that $q \in \square_v$, we need to find the first ancestor above v that has a point associated with it. This can be done in constant ti1me, by preprocessing $\widehat{T}$, by propagating down the compressed quadtree the nodes they are associated with. ■

13.3 A Direct Construction of AVD

Intuitively, constructing an approximate Voronoi diagram, can be interpret as a meshing problem. Indeed, consider the function $\mathbf{d}_P(q)$, for $q \in \mathbb{R}^d$,and a cell $\square \subseteq \mathbb{R}^d$ such that $\operatorname{diam}(\square) \le (\varepsilon/4)\min_{q\in\square}\mathbf{d}_P(q)$. Since for any $x, y \in \mathbb{R}^d$, we have $\mathbf{d}_P(y) \le \mathbf{d}_P(x) + \|xy\|$. It follows that

$$\forall x, y \in \square \quad \mathbf{d}_P(x) \le (1+\varepsilon/4)\mathbf{d}_P(y). \tag{13.1}$$

Namely, the distance function $\mathbf{d}_P(\cdot)$ is essentially a "constant" in such a cell. Of course, if there is only a unique nearest neighbor to all the points in $\square$, we do not need this condition.

Namely, we need to partition space into cells, such that for each cell, either it has a unique nearest neighbor (no problem), or alternatively, Eq. (13.1) holds in the cell. Unfortunately, forcing Eq. (13.1) globally is too expensive. Alternatively, we sill force Eq. (13.1) only in "critical" regions where there are two points which are close to being ANN to a cell.

The general idea, as before is to generate a set of canonical grid cells. We are guaranteed, that when generating the compressed quadtree for the point-set, and considering the space partition induced by the leafs, then necessarily those cells would be smaller than the input grid cells.

Definition 13.3.1 (Exponential Grid.) For a point $p \in \mathbb{R}^d$, and parameters r, R and $\varepsilon > 0$, let $\mathsf{G}_E(p, r, R, \varepsilon)$ denote an *exponential grid* centered at p.

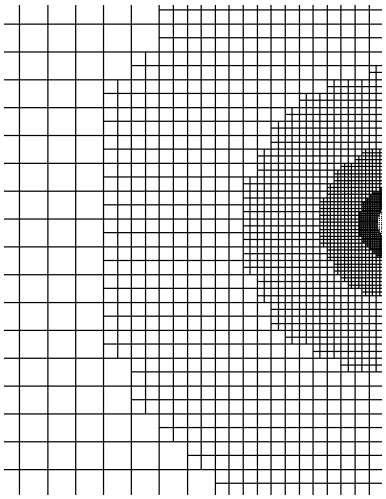


Figure 13.2: Exponential grid.

let $\mathbf{b}_i = \mathbf{b}(p, r_i)$, for $i = 0, \ldots, \lceil \lg R/r \rceil$, where $r_i = r2^i$. Next, let G'_i be the set of cells of the canonical grid G_{α_i} that intersects $\mathbf{b}_i$, where $\alpha_i = 2^{\lfloor \lg(\varepsilon r_i/(16d)) \rfloor}$. Clearly, $\left|G'_i\right| = O(1/\varepsilon^d)$. We remove from G'_i all the canonical cells completely covered by cells of G_{i-1}. Similarly, for cells the that are partially covered in G'_i by cells in G'_{i-1}, we replace them by the cells covering them in $\mathsf{G}_{\alpha_{i-1}}$. Let G_i be the resulting set of canonical grid cells. And let $\mathsf{G}_E(p, r, R, \varepsilon) = \cup_i G_i$. We have $|\mathsf{G}_E(p, r, R, \varepsilon)| = O(\varepsilon^{-d} \log(R/r))$. Furthermore, it can be computed in linear time in its size; see Figure 13.2.

Let P be a set of n points, and let $0 < \varepsilon < 1/2$. As before, we assume that $P \subseteq [0.5 - \varepsilon/d, 0.5 + \varepsilon/d]^d$.

Compute a $(1/(8d))^{-1}$-WSPD $\mathcal{W}$ of P. Note that $|\mathcal{W}| = O(n)$. For every pair $X = \{u, v\} \in \mathcal{W}$, let $\ell_{uv} = \|uv\|$, and consider the set of canonical cells

$$W(u, v) = \mathsf{G}_E(\mathrm{rep}_u, \ell_{uv}/4, 4\ell_{uv}/\varepsilon, \varepsilon) \cup \mathsf{G}_E(\mathrm{rep}_v, \ell_{uv}/4, 4\ell_{uv}/\varepsilon, \varepsilon).$$

For a query point q, if $\mathbf{d}_P(q) \in [\ell_{uv}/4, \ell_{uv}/\varepsilon]$, and the nearest neighbor of q is in $P_u \cup P_v$, then the cell $\square \in W(u, v)$ containing q is of the right size, it comply with Eq. (13.1), and as such any $(1 + \varepsilon/4)$-ANN to any point of $\square$ is a $(1 + \varepsilon)$-ANN to q.

Thus, let $W = \cup_{\{u,v\} \in \mathcal{W}} W(u, v) \cup [0, 1]^{\mathsf{d}}$. Let $\widehat{T}$ be a compressed quadtree constructed so that it contains all the canonical nodes of W as nodes (we remind the reader that this can be done in $O(|W| \log |W|)$ time; see Lemma 2.2.5). Next, let U be the space decomposition induced by the leafs of $\widehat{T}$.

For each cell $\square \in U$, take an arbitrary point inside it $\mathrm{rep}_\square$, and compute a $(1 + \varepsilon/4)$-ANN to p. Let $\widetilde{\mathrm{rep}_\square} \in P$ be this ANN, and store it together with $\square$.

Claim 13.3.2 *The set U is a $(1 + \varepsilon)$-AVD.*

Proof: Let q be am arbitrary query point, and let $\square \in U$ be the cell containing q. Let $\mathrm{rep}_\square \in \square$ be its representative, and let $\widetilde{\mathrm{rep}_\square} \in P$ be the $(1 + \varepsilon/4)$-ANN to $\mathrm{rep}_\square$ stored at $\square$. Also, let $\widehat{q}$ be the nearest neighbor of q in P. If $\widetilde{\mathrm{rep}_\square} = \widehat{q}$ then we are done. Otherwise, consider the pair $\{u, v\} \in \mathcal{W}$ such that $\widetilde{\mathrm{rep}_\square} \in P_u$ and $\widehat{q} \in P_v$. Finally, let $\ell = \left\|\widetilde{\mathrm{rep}_\square}\, \widehat{q}\right\|$.

If $\left\|q\widehat{q}\right\| > \ell/\varepsilon$ then $\widetilde{\mathrm{rep}_\square}$ is $(1 + \varepsilon)$-ANN since $\left\|q\, \widetilde{\mathrm{rep}_\square}\right\| \leq \left\|q\widehat{q}\right\| + \left\|\widehat{q}\, \widetilde{\mathrm{rep}_\square}\right\| \leq (1 + \varepsilon)\left\|q\widehat{q}\right\|$.

If $\left\|q\widehat{q}\right\| < \ell/4$, then by the construction of $W(u, v)$, we have that $\mathrm{diam}(\square) \leq (\varepsilon/16)\left\|\mathrm{rep}_u \mathrm{rep}_v\right\| \leq \varepsilon\ell/8$. This holds by the construction of the WSPD, where $\mathrm{rep}_u, \mathrm{rep}_v$ are the representative from the WSPD construction of P_u and P_v, respecitvely. See Figure 13.3. But then,

$$\left\|\widehat{q}\, \mathrm{rep}_\square\right\| \leq \left\|q\widehat{q}\right\| + \mathrm{diam}(\square) \leq \frac{\ell}{4} + \frac{\ell\varepsilon}{8} < \frac{5}{16}\ell,$$

for $\varepsilon \leq 1/2$. On the other hand,

$$\begin{aligned} \left\|\mathrm{rep}_\square\, \widetilde{\mathrm{rep}_\square}\right\| &\geq \left\|\mathrm{rep}_u \mathrm{rep}_v\right\| - \left\|\mathrm{rep}_v \widehat{q}\right\| - \left\|\widehat{q}q\right\| - \mathrm{diam}(\square) - \left\|\mathrm{rep}_u \widetilde{\mathrm{rep}_\square}\right\| \\ &\geq \ell - \ell/8 - \ell/4 - \varepsilon\ell/8 - \ell/8 \geq (7/16)\ell. \end{aligned}$$

But then, $(1 + \varepsilon/4)\left\|\mathrm{rep}_\square \widehat{q}\right\| \leq (5/16)(9/8)\ell < \ell/2 \leq \left\|\mathrm{rep}_\square\, \widetilde{\mathrm{rep}_\square}\right\|$. A contradiction, because $\widetilde{\mathrm{rep}_\square}$ is not $(1 + \varepsilon/4)$-ANN in P for $\mathrm{rep}_\square$. See Figure 13.3.

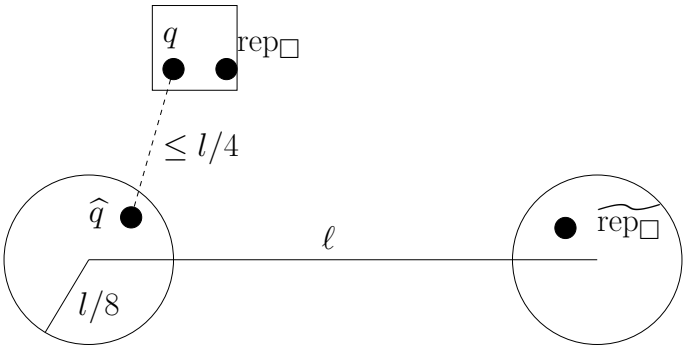


Figure 13.3: Illustration of the proof of Claim 13.3.2.

If $\|\widehat{qq}\| \in [\ell/4, \ell/\varepsilon]$, then by the construction of $W(u, v)$, we have that $\mathrm{diam}(\square) \leq (\varepsilon/4)\mathbf{d}_{P_u \cup P_v}(q)$. We have $\mathbf{d}_P(z) \leq (1 + \varepsilon/4)\mathbf{d}_P(q)$ and

$$\begin{aligned} \|q\,\widetilde{\mathrm{rep}_\square}\| &\leq \|q\,\mathrm{rep}_\square\| + \|\mathrm{rep}_\square\,\widetilde{\mathrm{rep}_\square}\| \leq \mathrm{diam}(\square) + (1 + \varepsilon/4)\mathbf{d}_P(\mathrm{rep}_\square) \\ &\leq (\varepsilon/4)\mathbf{d}_P(q) + (1 + \varepsilon/4)(1 + \varepsilon/4)\mathbf{d}_P(q) \leq (1 + \varepsilon)\mathbf{d}_P(q), \end{aligned}$$

as required. ■

Theorem 13.3.3 *Given a set P of n points in $\mathbb{R}^d$ one can compute, in $O(n/\varepsilon^d \log(1/\varepsilon)(\varepsilon^{-d} + \log(n/\varepsilon)))$ time, a $(1 + \varepsilon)$-AVD of P. The AVD is of complexity $O(n\varepsilon^{-d}\log(1/\varepsilon))$.*

Proof: The total number of cubes in W is $O(n\varepsilon^{-d}\log(1/\varepsilon))$, and W can also be computed in this time, as described above. For each node of W we need to perform a $(1 + \varepsilon/4)$-ANN query on P. After $O(n \log n)$ preprocessing, such queries can be answered in $O(\log n + 1/\varepsilon^d)$ time (Theorem 11.3.3). Thus, we can answer those queries, and built the overall data-structure, in the time stated in the theorem. ■

13.4 Bibliographical notes

The realization that point-location among approximate balls is sufficient for ANN, was realized by Indyk and Motwani [IM98]. The idea of first building a global set of balls, and using a compressed quadtree on the associated set of approximate balls, is due to Har-Peled [Har01b]. The space used by the data-structure of Theorem 13.2.1 was further improved (by a logarithmic factor) by Sabharwal *et al.* [SSS02] and Arya and Malamatos [AM02]. As mentioned before, Sabharwal *et al.* improved the number of balls needed (for the general metric case), while Arya and Malamatos showed a direct construction for the low-dimensional euclidean space using $O(n/\varepsilon^d)$ balls.

The direct construction (Section 13.3) of AVD is due to Arya and Malamatos [AM02]. The reader might wonder why we went through the excruciating pain of first approximating the metric by balls, and then using it to construct AVD. The reduction to point location among approximate balls, would prove to be useful when dealing with proximity in high dimensions. Of course, once we have the fact about ANN via point-location among approximate balls, the AVD construction is relatively easy.

Chapter 14

The Johnson-Lindenstrauss Lemma

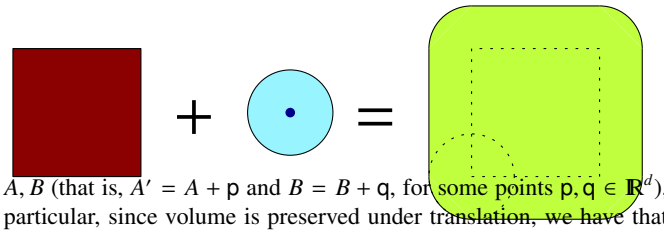
In this chapter, we will prove that given a set P of n points in $\mathbb{R}^d$, one can reduce the dimension of the points to $k = O(\varepsilon^{-2} \log n)$ and distances are $1 \pm \varepsilon$ reserved. Surprisingly, this reduction is done by randomly picking a subspace of k dimensions and projecting the points into this random subspace. One way of thinking about this result is that we are "compressing" the input of size nd (i.e., n points with d coordinates) into size $O(n\varepsilon^{-2} \log n)$, while (approximately) preserving distances.

14.1 The Brunn-Minkowski inequality

For a set $A \subseteq \mathbb{R}^d$, an a point $\mathsf{p} \in \mathbb{R}^d$, let $A + \mathsf{p}$ denote the translation of A by p. Formally, $A + \mathsf{p} = \left\{ \mathsf{q} + \mathsf{p} \;\middle|\; \mathsf{q} \in A \right\}$.

Definition 14.1.1 For two sets A and B in $\mathbb{R}^n$, let $A + B$ denote the ***Minkowski sum*** of A and B. Formally,

$$A + B = \left\{ a + b \;\middle|\; a \in A, b \in B \right\} = \cup_{\mathsf{p} \in A} (\mathsf{p} + B).$$



It is easy to verify that if A', B' are translated copies of A, B (that is, $A' = A + \mathsf{p}$ and $B = B + \mathsf{q}$, for some points $\mathsf{p}, \mathsf{q} \in \mathbb{R}^d$), respectively, then $A' + B'$ is a translated copy of $A + B$. In particular, since volume is preserved under translation, we have that $\mathrm{Vol}(A' + B') = \mathrm{Vol}((A + B) + \mathsf{p} + \mathsf{q}) = \mathrm{Vol}(A + B)$.

Theorem 14.1.2 (Brunn-Minkowski inequality) *Let A and B be two non-empty compact sets in $\mathbb{R}^n$. Then*

$$\mathrm{Vol}(A + B)^{1/n} \geq \mathrm{Vol}(A)^{1/n} + \mathrm{Vol}(B)^{1/n}.$$

Definition 14.1.3 A set $A \subseteq \mathbb{R}^n$ is a ***brick set*** if it is the union of finitely many (close) axis parallel boxes with disjoint interiors.

It is intuitively clear, by limit arguments, that proving Theorem 14.1.2 for brick sets will imply it for the general case.

Lemma 14.1.4 (Brunn-Minkowski inequality for Brick Sets) *Let A and B be two non-empty brick sets in $\mathbb{R}^n$. Then*

$$\mathrm{Vol}(A + B)^{1/n} \geq \mathrm{Vol}(A)^{1/n} + \mathrm{Vol}(B)^{1/n}$$

Proof: By induction on the number k of bricks in A and B. If $k = 2$ then A and B are just bricks, with dimensions $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, respectively. In this case, the dimensions of $A + B$ are $a_1 + b_1, \ldots, a_n + b_n$, as can be easily verified. Thus, we need to prove that $(\prod_{i=1}^n a_i)^{1/n} + (\prod_{i=1}^n b_i)^{1/n} \leq (\prod_{i=1}^n (a_i + b_i))^{1/n}$. Dividing the left side by the right side, we have

$$\left(\prod_{i=1}^n \frac{a_i}{a_i + b_i} \right)^{1/n} + \left(\prod_{i=1}^n \frac{b_i}{a_i + b_i} \right)^{1/n} \leq \frac{1}{n} \sum_{i=1}^n \frac{a_i}{a_i + b_i} + \frac{1}{n} \sum_{i=1}^n \frac{b_i}{a_i + b_i} = 1,$$

by the generalized arithmetic-geometric mean inequality[④], and the claim follows for this case.

[④] Here is a proof of this generalized form: Let $x_1, \ldots, x_n$ be n positive real numbers. Consider the quantity $R = x_1 x_2 \cdots x_n$. If we fix the sum of the n numbers to be equal α, then R is maximized when all the x_is are equal. Thus, $\sqrt[n]{x_1 x_2 \cdots x_n} \leq \sqrt[n]{(\alpha/n)^n} = \alpha/n = (x_1 + \cdots + x_n)/n$.

Now let $k > 2$ and suppose that the Brunn-Minkowski inequality holds for any pair of brick sets with fewer than k bricks (together). Let A, B be a pair of sets having k bricks together, and A has at least two (disjoint) bricks. However, this implies that there is an axis parallel hyperplane h that separates between the interior of one brick of A and the interior of another brick of A (the hyperplane h might intersect other bricks of A). Assume that h is the hyperplane $x_1 = 0$ (this can be achieved by translation and renaming of coordinates).

Let $\overline{A^+} = A \cap h^+$ and $\overline{A^-} = A \cap h^-$, where h^+ and h^- are the two open half spaces induced by h. Let A^+ and A^- be the closure of $\overline{A^+}$ and $\overline{A^-}$, respectively. Clearly, A^+ and A^- are both brick sets with (at least) one fewer brick than A.

Next, observe that the claim is translation invariant, and as such, let us translate B so that its volume is split by h in the same ratio A's volume is being split. Denote the two parts of B by B^+ and B^-, respectively. Let $\rho = \mathrm{Vol}(A^+)/\mathrm{Vol}(A) = \mathrm{Vol}(B^+)/\mathrm{Vol}(B)$ (if $\mathrm{Vol}(A) = 0$ or $\mathrm{Vol}(B) = 0$ the claim trivially holds).

Observe, that $A^+ + B^+ \subseteq A + B$, and it lies on one side of h, and similarly $A^- + B^- \subseteq A + B$ and it lies on the other side of h. Thus, by induction, we have

$$\begin{aligned}
\mathrm{Vol}(A+B) &\geq \mathrm{Vol}(A^+ + B^+) + \mathrm{Vol}(A^- + B^-) \\
&\geq \Big(\mathrm{Vol}(A^+)^{1/n} + \mathrm{Vol}(B^+)^{1/n}\Big)^n + \Big(\mathrm{Vol}(A^-)^{1/n} + \mathrm{Vol}(B^-)^{1/n}\Big)^n \\
&= \Big[\rho^{1/n}\mathrm{Vol}(A)^{1/n} + \rho^{1/n}\mathrm{Vol}(B)^{1/n}\Big]^n \\
&\quad + \Big[(1-\rho)^{1/n}\mathrm{Vol}(A)^{1/n} + (1-\rho)^{1/n}\mathrm{Vol}(B)^{1/n}\Big]^n \\
&= (\rho + (1-\rho))\Big[\mathrm{Vol}(A)^{1/n} + \mathrm{Vol}(B)^{1/n}\Big]^n \\
&= \Big[\mathrm{Vol}(A)^{1/n} + \mathrm{Vol}(B)^{1/n}\Big]^n,
\end{aligned}$$

establishing the claim. ■

Proof of ***Theorem 14.1.2***: Let $A_1 \subseteq A_2 \subseteq \cdots A_i \subseteq$ be a sequence of brick sets, such that $\bigcup_i A_i = A$, and similarly let $B_1 \subseteq B_2 \subseteq \cdots B_i \subseteq \cdots$ be a sequence of finite brick sets, such that $\bigcup_i B_i = B$. It is well known fact in measure theory, that $\lim_{i\to\infty} \mathrm{Vol}(A_i) = \mathrm{Vol}(A)$ and $\lim_{i\to\infty} Vol(B_i) = \mathrm{Vol}(B)$.

We claim that $\lim_{i\to\infty} \mathrm{Vol}(A_i + B_i) = \mathrm{Vol}(A + B)$. Indeed, consider any point $z \in A + B$, and let $u \in A$ and $v \in B$ be such that $u + v = z$. By definition, there exists i, such that for all $j > i$ we have $u \in A_j$, $v \in B_j$, and as such $z \in A_i + B_i$. Thus, $\cup_i (A_i + B_i) = A + B$.

Furthermore, for any $i > 0$, since A_i and B_i are brick sets, we have

$$\mathrm{Vol}(A_i + B_i)^{1/n} \geq \mathrm{Vol}(A_i)^{1/n} + \mathrm{Vol}(B_i)^{1/n},$$

by Lemma 14.1.4. Thus,

$$\begin{aligned}
\mathrm{Vol}(A+B) &= \lim_{i\to\infty} \mathrm{Vol}(A_i + B_i)^{1/n} \geq \lim_{i\to\infty} \Big(\mathrm{Vol}(A_i)^{1/n} + \mathrm{Vol}(B_i)^{1/n}\Big) \\
&= \mathrm{Vol}(A)^{1/n} + \mathrm{Vol}(B)^{1/n}.
\end{aligned}$$

■

Theorem 14.1.5 (Brunn-Minkowski for slice volumes.) *Let $\mathcal{P}$ be a convex set in $\mathbb{R}^{n+1}$, and let $A = \mathcal{P} \cap (x_1 = a)$, $B = \mathcal{P} \cap (x_1 = b)$ and $C = \mathcal{P} \cap (x_1 = c)$ be three slices of A, for $a < b < c$. We have $\mathrm{Vol}(B) \geq \min(\mathrm{Vol}(A), \mathrm{Vol}(C))$.*

In fact, consider the function

$$v(t) = (\mathrm{Vol}(\mathcal{P} \cap (x_1 = t)))^{1/n},$$

and let $\mathcal{I} = [t_{min}, t_{max}]$ be the interval where the hyperplane $x_1 = t$ intersects $\mathcal{P}$. Then, $v(t)$ is concave in I.

Proof: If a or c are outside $\mathcal{I}$, then $\mathrm{Vol}(A) = 0$ or $\mathrm{Vol}(C) = 0$, respectively, and then the claim trivially holds.

Otherwise, let $\alpha = (b-a)/(c-a)$. We have that $b = (1-\alpha)\cdot a + \alpha \cdot c$, and by the convexity of $\mathcal{P}$, we have $(1-\alpha)A + \alpha C \subseteq B$. Thus, by Theorem 14.1.2 we have

$$\begin{aligned}
v(b) = \mathrm{Vol}(B)^{1/n} &\geq \mathrm{Vol}((1-\alpha)A + \alpha C)^{1/n} \geq \mathrm{Vol}((1-\alpha)A)^{1/n} + \mathrm{Vol}(\alpha C)^{1/n} \\
&= (1-\alpha)\cdot \mathrm{Vol}(A)^{1/n} + \alpha \cdot \mathrm{Vol}(C)^{1/n} \\
&\geq (1-\alpha)v(a) + \alpha v(c).
\end{aligned}$$

Namely, $v(\cdot)$ is concave on $\mathcal{I}$, and in particular $v(b) \geq \min(v(a), v(c))$, which in turn implies that $\mathrm{Vol}(B) = v(b)^n \geq \min(\mathrm{Vol}(A), \mathrm{Vol}(B))$, as claimed. ■

Corollary 14.1.6 *For A and B compact sets in $\mathbb{R}^n$, we have $\mathrm{Vol}((A+B)/2) \geq \sqrt{\mathrm{Vol}(A)\,\mathrm{Vol}(B)}$.*

Proof: $\mathrm{Vol}((A+B)/2)^{1/n} = \mathrm{Vol}(A/2 + B/2)^{1/n} \geq \mathrm{Vol}(A/2)^{1/n} + \mathrm{Vol}(B/2)^{1/n} = (\mathrm{Vol}(A)^{1/n} + \mathrm{Vol}(B)^n)/2 \geq \sqrt{\mathrm{Vol}(A)^{1/n}\,\mathrm{Vol}(B)^{1/n}}$ by Theorem 14.1.2, and since $(a+b)/2 \geq \sqrt{ab}$ for any $a, b \geq 0$. The claim now follows by raising this inequality to the power n. ■

14.1.1 The Isoperimetric Inequality

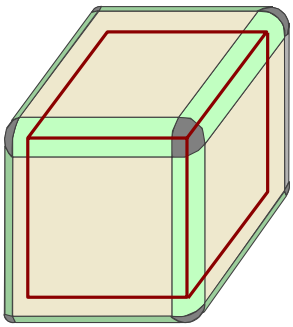
Useless Stuning!!!

The following is not used anywhere else and is provided because of its mathematical elegance. The skip-able reader can thus skip this section.

The ***isoperimetric inequality*** states that among all convex bodies of a fixed surface area, the ball has the largest volume (in particular, the unit circle is the largest area planar region with perimeter 2π). This problem can be traced back to antiquity, in particular Zenodorus (200–140 BC) wrote a monograph (which was lost) that seemed to have proved the claim in the plane for some special cases. The first formal proof for the planar case was done by Steiner in 1841. Interestingly, the more general claim is an easy consequence of the Brunn-Minkowski inequality.

Let K be a convex body in $\mathbb{R}^n$ and $\mathbf{b} = \mathbf{b}^n$ be the n dimensional ball of radius one centered at the origin. Let $\mathsf{S}(X)$ denote the surface area of a compact set $X \subseteq \mathbb{R}^n$. The ***isoperimetric inequality*** states that

$$\left(\frac{\mathrm{Vol}(K)}{\mathrm{Vol}(\mathbf{b})}\right)^{1/n} \leq \left(\frac{\mathsf{S}(K)}{\mathsf{S}(\mathbf{b})}\right)^{1/(n-1)}. \tag{14.1}$$



Namely, the left side is the radius of a ball having the same volume as K, and the right side is the radius of a sphere having the same surface area as K. In particular, if we scale K so that its surface area is the same as $\mathbf{b}$, then the above inequality implies that $\mathrm{Vol}(K) \leq \mathrm{Vol}(\mathbf{b})$.

To prove Eq. (14.1), observe that $\mathrm{Vol}(\mathbf{b}) = \mathsf{S}(\mathbf{b})/n$[②]. Also, observe that $K + \varepsilon\mathbf{b}$ is the body K together with a small "atmosphere" around it of thickness ε. In particular, the volume of this "atmosphere" is (roughly) $\varepsilon\,\mathsf{S}(K)$ (in fact, Minkowski defined the surface area of a convex body to be the limit stated next). Formally, we have

$$\mathsf{S}(K) = \lim_{\varepsilon\to 0+} \frac{\mathrm{Vol}(K+\varepsilon\mathbf{b}) - \mathrm{Vol}(K)}{\varepsilon} \geq \lim_{\varepsilon\to 0+} \frac{\left(\mathrm{Vol}(K)^{1/n} + \mathrm{Vol}(\varepsilon\mathbf{b})^{1/n}\right)^n - \mathrm{Vol}(K)}{\varepsilon},$$

by the Brunn-Minkowski inequality. Now $\mathrm{Vol}(\varepsilon\mathbf{b})^{1/n} = \varepsilon\,\mathrm{Vol}(\mathbf{b})^{1/n}$, and as such

$$\begin{aligned}\mathsf{S}(K) &\geq \lim_{\varepsilon\to 0+} \frac{\mathrm{Vol}(K) + \binom{n}{1}\varepsilon\,\mathrm{Vol}(K)^{(n-1)/n}\,\mathrm{Vol}(\mathbf{b})^{1/n} + \binom{n}{2}\varepsilon^2\,\langle\text{whatever}\rangle\cdots + \varepsilon^n\,\mathrm{Vol}(\mathbf{b}) - \mathrm{Vol}(K)}{\varepsilon} \\ &= \lim_{\varepsilon\to 0+} \frac{n\varepsilon\,\mathrm{Vol}(K)^{(n-1)/n}\,\mathrm{Vol}(\mathbf{b})^{1/n}}{\varepsilon} = n\,\mathrm{Vol}(K)^{(n-1)/n}\,\mathrm{Vol}(\mathbf{b})^{1/n}.\end{aligned}$$

Dividing both sides by $\mathsf{S}(\mathbf{b}) = n\,\mathrm{Vol}(\mathbf{b})$, we have

$$\frac{\mathsf{S}(K)}{\mathsf{S}(\mathbf{b})} \geq \frac{\mathrm{Vol}(K)^{(n-1)/n}}{\mathrm{Vol}(\mathbf{b})^{(n-1)/n}} \quad\Rightarrow\quad \left(\frac{\mathsf{S}(K)}{\mathsf{S}(\mathbf{b})}\right)^{1/(n-1)} \geq \left(\frac{\mathrm{Vol}(K)}{\mathrm{Vol}(\mathbf{b})}\right)^{1/n},$$

establishing the isoperimetric inequality.

14.2 Measure Concentration on the Sphere

Let $\mathbb{S}^{(n-1)}$ be the unit sphere in $\mathbb{R}^n$. We assume there is a uniform probability measure defined over $\mathbb{S}^{(n-1)}$, such that its total measure is 1. Surprisingly, most of the mass of this measure is near the equator. In fact, as the dimension increases, the width of the strip around the equator $(x_1 = 0) \cap \mathbb{S}^{(n-1)}$ contains, say, 90% of the measure is of width $\approx c/n$, for some constant c. Counter intuitively, this is true for *any* equator. We are going to show that a stronger result holds: The mass is concentrated close to the boundary of any set $A \subseteq \mathbb{S}^{(n-1)}$ such that $\mathbf{Pr}[A] = 1/2$.

Before proving this someswhat surprising theorem, we will first try to get an intuition about the behaviour of the hypersphere in high dimensions.

14.2.1 The strange and curious life of the hypersphere

Consider the ball of radius r denoted by $r\,\mathbf{b}^n$, where $\mathbf{b}^n$ is the unit radius ball centered at the origin. Clearly, $\mathrm{Vol}(r\,\mathbf{b}^n) = r^n\,\mathrm{Vol}(\mathbf{b}^n)$. Now, even if r is very close to 1, the quantity r^n might be very close to zero if n is sufficiently large. Indeed, if $r = 1 - \delta$, then $r^n \leq (1-\delta)^n \leq \exp(-\delta n)$, which is very small if $\delta \gg 1/n$. (Here, we used the fact that $1 - x \leq e^x$, for $x \geq 0$.) Namely, for the ball in high dimensions, its mass is concentrated in a very thin shell close to its surface.

[②] Indeed, $\mathrm{Vol}(\mathbf{b}) = \int_{r=0}^{1} \mathsf{S}(\mathbf{b})\,r^{n-1}dr = \mathsf{S}(\mathbf{b})/n$.

The volume of a ball and the surface area of hypersphere. In fact, let $\mathrm{Vol}(r\mathbf{b}^n)$ denote the volume of the ball of radius r in $\mathbb{R}^n$, $\mathrm{Area}\left(r\mathbb{S}^{(n)}\right)$ denote the surface area of its boundry (i.e., the surface area of $r\mathbb{S}^{(n-1)}$). It is known that

$$\mathrm{Vol}(r\mathbf{b}^n) = \frac{\pi^{n/2} r^n}{\Gamma(n/2+1)} \quad \text{and} \quad \mathrm{Area}\left(r\mathbb{S}^{(n-1)}\right) = \frac{2\pi^{n/2} r^{n-1}}{\Gamma(n/2)},$$

where the $\Gamma(\cdot)$ is an extension of the factorial function. Specifically, if n is even then $\Gamma(n/2+1) = (n/2)!$, and for n odd $\Gamma(n/2+1) = \sqrt{\pi}(n!!)/2^{(n+1)/2}$, where $n!! = 1 \cdot 3 \cdot 5 \cdots n$ is the ***double factorial***. The most surprising implication of these two formulas is that the volume of the unit ball increases (till dimension 5 in fact) and then it starts decreasing to zero. Similarly, the surface area of the unit sphere $\mathbb{S}^{(n-1)}$ in $\mathbb{R}^n$ tends to zero as the dimension increases. To see this compute the volume of the unit ball using an integral of its slice volume, when it is being sliced by a hyperplanes perpendicular to the nth coordinate. We have

$$\mathrm{Vol}(\mathbf{b}^n) = \int_{x_n=-1}^{1} \mathrm{Vol}\left(\sqrt{1-x_n^2}\,\mathbf{b}^{n-1}\right) dx_n = \mathrm{Vol}\left(\mathbf{b}^{n-1}\right) \int_{x_n=-1}^{1} \left(1-x_n^2\right)^{(n-1)/2} dx_n.$$

Now, the integral on the right side tends to zero as n increases. In fact, for n very large, the term $\left(1-x_n^2\right)^{(n-1)/2}$ is very close to 0 everywhere except for a small interval around 0. This implies that the main contribution of the volume of the ball happens when we consider slices of the ball by hyperplanes of the form $x_n = \delta$, where δ is small.

If one has to visualize how such a ball in high dimesions looks like, it might be best to think about it as a star-like creature: It has very little mass close to the tips of any set of orthogonal directions we pick, and most of its mass somehow lies close to its center.[3]

14.2.2 Measure Concentration on the Sphere

Theorem 14.2.1 (Measure concentration on the sphere.) *Let $A \subseteq \mathbb{S}^{(n-1)}$ be a measurable set with $\mathbf{Pr}[A] \geq 1/2$, and let A_t denote the set of points of $\mathbb{S}^{(n-1)}$ in distance at most t from A, where $t \leq 2$. Then $1 - \mathbf{Pr}[A_t] \leq 2\exp\left(-nt^2/2\right)$.*

Proof: We will prove a slightly weaker bound, with $-nt^2/4$ in the exponent. Let

$$\widehat{A} = \left\{ \alpha x \;\middle|\; x \in A, \alpha \in [0,1] \right\} \subseteq \mathbf{b}^n,$$

where $\mathbf{b}^n$ is the unit ball in $\mathbb{R}^n$. We have that $\mathbf{Pr}[A] = \mu\left(\widehat{A}\right)$, where $\mu\left(\widehat{A}\right) = \mathrm{Vol}\left(\widehat{A}\right) / \mathrm{Vol}(\mathbf{b}^n)$[4]

Let $B = \mathbb{S}^{(n-1)} \setminus A_t$. We have that $\|a-b\| \geq t$ for all $a \in A$ and $b \in B$.

Lemma 14.2.2 *For any $\widehat{a} \in \widehat{A}$ and $\widehat{b} \in \widehat{B}$, we have $\left\|\frac{a+b}{2}\right\| \leq 1 - \frac{t^2}{8}$.*

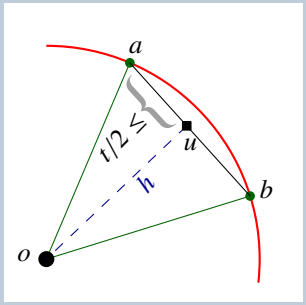
Proof: Let $\widehat{a} = \alpha a$ and $\widehat{b} = \beta b$, where $a \in A$ and $b \in B$. We have

$$\|u\| = \left\|\frac{a+b}{2}\right\| = \sqrt{1^2 - \left\|\frac{a-b}{2}\right\|^2} \leq \sqrt{1-\frac{t^2}{4}} \leq 1 - \frac{t^2}{8}, \tag{14.2}$$

since $\|a-b\| \geq t$. As for $\widehat{a}$ and $\widehat{b}$, assume that $\alpha \leq \beta$, and observe that the quantity $\left\|\widehat{a}+\widehat{b}\right\|$ is maximized when $\beta = 1$. As such, by the triangle inequality, we have

$$\begin{aligned}\left\|\frac{\widehat{a}+\widehat{b}}{2}\right\| &= \left\|\frac{\alpha a + b}{2}\right\| \leq \left\|\frac{\alpha(a+b)}{2}\right\| + \left\|(1-\alpha)\frac{b}{2}\right\| \\ &\leq \alpha\left(1-\frac{t^2}{8}\right) + (1-\alpha)\frac{1}{2} = \tau,\end{aligned}$$

by Eq. (14.2) and since $\|b\| = 1$. Now, τ is a convex combination of the two numbers $1/2$ and $1 - t^2/8$. In particular, we conclude that $\tau \leq \max(1/2, 1 - t^2/8) \leq 1 - t^2/8$, since $t \leq 2$. ■



[3] In short, it looks like a Boojum [Car76].

[4] This is one of these "trivial" claims that might give the reader a pause, so here is a formal proof. Pick a random point p uniformly inside the ball $\mathbf{b}^n$. Let ψ be the probability that $\mathsf{p} \in \widehat{A}$. Clearly, $\mathrm{Vol}(\widehat{A}) = \psi \mathrm{Vol}(\mathbf{b}^n)$. So, consider the normalized point $\mathsf{q} = \mathsf{p}/\|\mathsf{p}\|$. Clearly, $\mathsf{p} \in \widehat{A}$ if and only if $\mathsf{q} \in A$, by the definition of $\widehat{A}$. Thus, $\mu\left(\widehat{A}\right) = \mathrm{Vol}(\widehat{A})/\mathrm{Vol}(\mathbf{b}^n) = \psi = \mathbf{Pr}\left[\mathsf{p} \in \widehat{A}\right] = \mathbf{Pr}[\mathsf{q} \in A] = \mathbf{Pr}[A]$, since q has a uniform distribution on the hypersphere by the symmetry of $\mathbf{b}^n$.

By Lemma 14.2.2, the set $\left(\widehat{A}+\widehat{B}\right)/2$ is contained in a ball of radius $\leq 1-t^2/8$ around the origin. Applying the Brunn-Minkowski inequality in the form of Corollary 14.1.6, we have

$$\left(1-\frac{t^2}{8}\right)^n \geq \mu\left(\frac{\widehat{A}+\widehat{B}}{2}\right) \geq \sqrt{\mu\left(\widehat{A}\right)\mu\left(\widehat{B}\right)} = \sqrt{\mathbf{Pr}[A]\,\mathbf{Pr}[B]} \geq \sqrt{\mathbf{Pr}[B]\,/2}.$$

Thus, $\mathbf{Pr}[B] \leq 2(1-t^2/8)^{2n} \leq 2\exp(-2nt^2/8)$, since $1-x \leq \exp(-x)$, for $x \geq 0$. ■

14.3 Concentration of Lipschitz Functions

Consider a function $f : \mathbb{S}^{(n-1)} \to \mathbb{R}$. Furthermore, imagine that we have a probability density defined over the sphere. Let $\mathbf{Pr}[f \leq t] = \mathbf{Pr}\left[\left\{x \in S^{n-1} \;\middle|\; f(x) \leq t\right\}\right]$. We define the ***median*** of f, denoted by $\mathrm{med}(f)$, to be the $\sup t$, such that $\mathbf{Pr}[f \leq t] \leq 1/2$.

Lemma 14.3.1 *We have* $\mathbf{Pr}[f < \mathrm{med}(f)] \leq 1/2$ *and* $\mathbf{Pr}[f > \mathrm{med}(f)] \leq 1/2$.

Proof: Since $\bigcup_{k\geq 1}(-\infty, \mathrm{med}(f) - 1/k] = (-\infty, \mathrm{med}(f))$, we have

$$\mathbf{Pr}[f < \mathrm{med}(f)] \quad = \quad \sup_{k\geq 1} \mathbf{Pr}\left[f \leq \mathrm{med}(f) - \frac{1}{k}\right] \leq \frac{1}{2}.$$

■

Definition 14.3.2 (*c*-*Lipschitz*) A function $f : A \to B$ is ***c*-*Lipschitz*** if, for any $x, y \in A$, we have $\|f(x) - f(y)\| \leq c\,\|x - y\|$.

Theorem 14.3.3 (Lévy's Lemma.) *Let $f : \mathbb{S}^{(n-1)} \to \mathbb{R}$ be 1-Lipschitz. Then for all $t \in [0, 1]$,*

$$\mathbf{Pr}[f > \mathrm{med}(f) + t] \leq 2\exp\left(-t^2 n/2\right) \text{ and } \mathbf{Pr}[f < \mathrm{med}(f) - t] \leq 2\exp\left(-t^2 n/2\right).$$

Proof: We prove only the first inequality, the second follows by symmetry. Let

$$A = \left\{x \in \mathbb{S}^{(n-1)} \;\middle|\; f(x) \leq \mathrm{med}(f)\right\}.$$

By Lemma 14.3.1, we have $\mathbf{Pr}[A] \geq 1/2$. Since f is 1-Lipschitz, we have $f(x) \leq \mathrm{med}(f) + t$, for any $x \in A_t$. Thus, by Theorem 14.2.1, we get $\mathbf{Pr}[f > \mathrm{med}(f) + t] \leq 1 - \mathbf{Pr}[A_t] \leq 2\exp\left(-t^2 n/2\right)$. ■

14.4 The Johnson-Lindenstrauss Lemma

Lemma 14.4.1 *For a unit vector $x \in \mathbb{S}^{(n-1)}$, let*

$$f(x) = \sqrt{x_1^2 + x_2^2 + \cdots + x_k^2}$$

be the length of the projection of x into the subspace formed by the first k coordinates. Let x be a vector randomly chosen with uniform distribution from $\mathbb{S}^{(n-1)}$. Then $f(x)$ is sharply concentrated. Namely, there exists $m = m(n, k)$ such that

$$\mathbf{Pr}[f(x) \geq m + t] \leq 2\exp(-t^2 n/2) \quad \text{and} \quad \mathbf{Pr}[f(x) \leq m - t] \leq 2\exp(-t^2 n/2).$$

Furthermore, for $k \geq 10\ln n$, we have $m \geq \frac{1}{2}\sqrt{k/n}$.

Proof: The orthogonal projection $p : \ell_2^n \to \ell_2^k$ given by $p(x_1, \ldots, x_n) = (x_1, \ldots, x_k)$ is 1-Lipschitz (since projections can only shrink distances, see Exercise 14.7.4). As such, $f(x) = \|p(x)\|$ is 1-Lipschitz, since for any x, y we have

$$|f(x) - f(y)| = \Big|\,\|p(x)\| - \|p(y)\|\,\Big| \leq \|p(x) - p(y)\| \leq \|x - y\|,$$

by the triangle inequality and since p is 1-Lipschitz. Theorem 14.3.3 (i.e., Lévy's lemma) gives the required tail estimate with $m = \mathrm{med}(f)$.

Thus, we only need to prove the lower bound on m. For a random $x = (x_1, \ldots, x_n) \in \mathbb{S}^{(n-1)}$, we have $\mathbf{E}\left[\|x\|^2\right] = 1$. By linearity of expectations, and symmetry, we have $1 = \mathbf{E}\left[\|x\|^2\right] = \mathbf{E}\left[\sum_{i=1}^n x_i^2\right] = \sum_{i=1}^n \mathbf{E}\left[x_i^2\right] = n\,\mathbf{E}\left[x_j^2\right]$, for any $1 \leq j \leq n$. Thus, $\mathbf{E}\left[x_j^2\right] = 1/n$, for $j = 1, \ldots, n$. Thus, $\mathbf{E}\left[(f(x))^2\right] = k/n$. We next use the fact that f is concentrated, to show that f^2 is also relatively concentrated.

For any $t \geq 0$, we have

$$\frac{k}{n} = \mathbf{E}\left[f^2\right] \leq \mathbf{Pr}[f \leq m + t]\,(m + t)^2 + \mathbf{Pr}[f \geq m + t] \cdot 1 \leq 1 \cdot (m + t)^2 + 2\exp(-t^2 n/2),$$

since $f(x) \leq 1$, for any $x \in \mathbb{S}^{(n-1)}$. Let $t = \sqrt{k/5n}$. Since $k \geq 10 \ln n$, we have that $2\exp(-t^2 n/2) \leq 2/n$. We get that

$$\frac{k}{n} \leq \left(m + \sqrt{k/5n}\right)^2 + 2/n.$$

Implying that $\sqrt{(k-2)/n} \leq m + \sqrt{k/5n}$, which in turn implies that $m \geq \sqrt{(k-2)/n} - \sqrt{k/5n} \geq \frac{1}{2}\sqrt{k/n}$. ■

At this point, we would like to flip Lemma 14.4.1 around, and instead of randomly picking a point and projecting it down to the first k-dimensional space, we would like x to be fixed, and randomly pick the k-dimensional subspace. However, we need to pick this k-dimensional space carefully, so that if we rotate this random subspace, by a transformation T, so that it occupies the first k dimensions, then the point $T(x)$ is uniformly distributed on the hypersphere.

To this end, we would like to randomly pick a random rotation of $\mathbb{R}^n$. This is an orthonormal matrix with determinant 1. We can generate such a matrix, by randomly picking a vector $e_1 \in \mathbb{S}^{(n-1)}$. Next, we set e_1 is the first column of our rotation matrix, and generate the other $n-1$ columns, by generating recursively $n-1$ orthonormal vectors in the space orthogonal to e_1.

Generating a random vector from the unit hypersphere, and a random rotation. At this point, the reader might wonder how do we pick a point uniformly from the unit hypersphere. The idea is to pick a point from the multi-dimensional normal distribution $N^d(0,1)$, and normalizing it to have length 1. Since the multi-dimensional normal distribution has the density function

$$(2\pi)^{-n/2} \exp\left(-(x_1^2 + x_2^2 + \cdots + x_n^2)/2\right),$$

which is symmetric (i.e., all the points in distance r from the origin has the same distribution), it follows that this indeed randomly generates a point randomly and uniformly on $\mathbb{S}^{(n-1)}$.

Generating a vector with multi-dimensional normal distribution, is no more than picking each coordinate according to the normal distribution. Given a source of random numbers according to the uniform distribution, this can be done using a $O(1)$ computations, using the Box-Muller transformation [BM58].

Since projecting down n-dimensional normal distribution to the lower dimensional space yields a normal distribution, it follows that generating a random projection, is no more than randomly picking n vectors according to the multidimensional normal distribution $v_1, \ldots, v_n$. Then, we orthonormalize them, using Graham-Schmidt, where $\widehat{v_1} = v_1/\|v_1\|$, and $\widehat{v_i}$ is the normalized vector of $v_i - w_i$, where w_i is the projection of v_i to the space spanned by $v_1, \ldots, v_{i-1}$.

Taking those vectors as columns of a matrix, generates a matrix A, with determinant either 1 or -1. We multiply one of the vectors by -1 if the determinant is -1. The resulting matrix is a random rotation matrix.

Definition 14.4.2 The mapping $f : \mathbb{R}^n \to \mathbb{R}^k$ is called ***K-bi-Lipschitz*** for a subset $X \subseteq \mathbb{R}^n$ if there exists a constant $c > 0$ such that

$$cK^{-1} \cdot \|\mathsf{p} - \mathsf{q}\| \leq \|f(\mathsf{p}) - f(\mathsf{q})\| \leq c \cdot \|\mathsf{p} - \mathsf{q}\|,$$

for all $\mathsf{p}, \mathsf{q} \in X$.

The least K for which f is K-bi-Lipschitz is called the ***distortion*** of f, and is denoted dist(f). We will refer to f as a ***K-embedding*** of X.

Theorem 14.4.3 (Johnson-Lindenstrauss lemma.) *Let X be an n-point set in a Euclidean space, and let $\varepsilon \in (0, 1]$ be given. Then there exists a $(1+\varepsilon)$-embedding of X into $\mathbb{R}^k$, where $k = O(\varepsilon^{-2} \log n)$.*

Proof: Let $X \subseteq \mathbb{R}^n$ (if X lies in higher dimensions, we can consider it to be lying in the span of its points, if it is in lower dimensions, we can add zero coordinates). Let $k = 200\varepsilon^{-2} \ln n$. Assume $k < n$, and let $\mathcal{F}$ be a random k-dimensional linear subspace of $\mathbb{R}^n$. Let $P_{\mathcal{F}} : \mathbb{R}^n \to \mathcal{F}$ be the orthogonal projection operator of $\mathbb{R}^n$ into $\mathcal{F}$. Let m be the number around which $\|P_{\mathcal{F}}(x)\|$ is concentrated, for $x \in \mathbb{S}^{(n-1)}$, as in Lemma 14.4.1.

Fix two points $x, y \in \mathbb{R}^n$, we prove that

$$\left(1 - \frac{\varepsilon}{3}\right) m \|x - y\| \leq \|P_{\mathcal{F}}(x) - P_{\mathcal{F}}(y)\| \leq \left(1 + \frac{\varepsilon}{3}\right) m \|x - y\|$$

holds with probability $\geq 1 - n^{-2}$. Since there are $\binom{n}{2}$ pairs of points in X, it follows that with constant probability this holds for all pair of points of X. In such a case, the mapping p is D-embedding of X into $\mathbb{R}^k$ with $D \leq \frac{1+\varepsilon/3}{1-\varepsilon/3} \leq 1 + \varepsilon$, for $\varepsilon \leq 1$.

Let $u = x - y$, we have $P_{\mathcal{F}}(u) = P_{\mathcal{F}}(x) - P_{\mathcal{F}}(y)$ since $P_{\mathcal{F}}(\cdot)$ is a linear operator. Thus, the condition becomes $\left(1 - \frac{\varepsilon}{3}\right) m \|u\| \leq \|P_{\mathcal{F}}(u)\| \leq \left(1 + \frac{\varepsilon}{3}\right) m \|u\|$. Since this condition is scale independent, we can assume $\|u\| = 1$. Namely, we need to show that

$$\left| \|P_{\mathcal{F}}(u)\| - m \right| \leq \frac{\varepsilon}{3} m.$$

By Lemma 14.4.1 (exchanging the random space with the random vector), for $t = \varepsilon m/3$, we have that the probablity that this does not hold is bounded

$$4\exp\left(-\frac{t^2 n}{2}\right) = 4\exp\left(\frac{-\varepsilon^2 m^2 n}{18}\right) \leq 4\exp\left(-\frac{\varepsilon^2 k}{72}\right) < n^{-2},$$

since $m \geq \frac{1}{2}\sqrt{k/n}$. ■

14.5 An alternative proof of the Johnson-Lindenstrauss lemma

14.5.1 Some Probability

Definition 14.5.1 Let $N(0,1)$ denote the one dimensional ***normal distribution***. This distribution has density $n(x) = e^{-x^2/2}/\sqrt{2\pi}$.

Let $N^d(0,1)$ denote the d-dimensional Gaussian distribution, induced by picking each coordinate independently from the standard normal distribution $N(0,1)$.

Let $\text{Exp}(\lambda)$ denote the ***exponential distribution***, with parameter λ. The density function of the exponential distribution is $f(x) = \lambda\exp(-\lambda x)$.

Let $\Gamma_{\lambda,k}$ denote the ***gamma distribution***, with parameters λ and k. The density function of this distribution is $g_{\lambda,k}(x) = \lambda\frac{(\lambda x)^{k-1}}{(k-1)!}\exp(-\lambda x)$. The cumulative distribution function of $\Gamma_{\lambda,k}$ is $G_{\lambda,k}(x) = 1 - \exp(-\lambda x)\left(1 + \frac{\lambda x}{1!} + \cdots + \frac{(\lambda x)^i}{i!} + \cdots + \frac{(\lambda x)^{k-1}}{(k-1)!}\right)$. As we prove below, gamma distribution is how much time one has to wait till k experiments succeed, where an experiment duration distributes according to the exponential distribution.

A random variable X has the ***Poisson distribution***, with parameter $\eta > 0$, which is a discrete distribution, if $\mathbf{Pr}[X = i] = e^{-\eta}\frac{\eta^i}{i!}$.

Lemma 14.5.2 *The following properties hold for the d dimensional Gaussian distribution $N^d(0,1)$:*

(i) The distribution $N^d(0,1)$ is centrally symmetric around the origin.

(ii) If $X \sim N^d(0,1)$ and u is a unit vector, then $X \cdot u \sim N(0,1)$.

(iii) If $X, Y \sim N(0,1)$ are two independent variables, then $Z = X^2 + Y^2$ follows the exponential distribution with parameter $\lambda = \frac{1}{2}$.

(iv) Given k independent variables $X_1, \ldots, X_k$ distributed according to the exponential distribution with parameter λ, then $Y = X_1 + \cdots + X_k$ is distributed according to the Gamma distribution $\Gamma_{\lambda,k}(x)$.

Proof: (i) Let $x = (x_1, \ldots, x_d)$ be a point picked from the Gaussian distribution. The density $\phi_d(x) = \phi(x_1)\phi(x_2)\cdot\phi(x_d)$, where $\phi(x_i)$ is the normal distribution density function, which is $\phi(x_i) = \exp(-x_i^2/2)/\sqrt{2\pi}$. Thus $\phi_d(x) = (2\pi)^{-n/2}\exp(-(x_1^2\cdots + x_d^2)/2)$. Consider any two points $x, y \in \mathbb{R}^n$, such that $r = \|x\| = \|y\|$. Clearly, $\phi_d(x) = \phi_d(y)$. Namely, any two points of the same distance from the origin, have the same density (i.e., "probability"). As such, the distribution $N^d(0,1)$ is centrally symmetric around the origin.

(ii) Consider $e_1 = (1, 0, \ldots, 0) \in \mathbb{R}^n$. Clearly, $x \cdot e_1 = x_1$, which is distributed $N(0,1)$. Now, by the symmetry of $N^d(0,1)$, this implies that $x \cdot u$ is distributed $N(0,1)$. Formally, let R be a rotation matrix that maps u to e_1. We know that Rx is distributed $N^d(0,1)$ (since $N^d(0,1)$ is centrally symmetric). Thus $x \cdot u$ has the same distribute as $Rx \cdot Ru$, which has the same distribution as $x \cdot e_1$, which is $N(0,1)$.

(iii) If $X, Y \sim N(0,1)$, and consider the integral of the density function

$$A = \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty}\frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right)dx\,dy.$$

We would like to change the integration variables to $x(r,\alpha) = \sqrt{r}\sin\alpha$ and $y(r,\alpha) = \sqrt{r}\cos\alpha$. The Jacobian of this change of variables is

$$I(r,\alpha) = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \alpha} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \alpha} \end{vmatrix} = \begin{vmatrix} \frac{\sin\alpha}{2\sqrt{r}} & \sqrt{r}\cos\alpha \\ \frac{\cos\alpha}{2\sqrt{r}} & -\sqrt{r}\sin\alpha \end{vmatrix} = -\frac{1}{2}\left(\sin^2\alpha + \cos^2\alpha\right) = -\frac{1}{2}.$$

As such, we have

$$\begin{aligned}
\mathbf{Pr}[Z = z] &= \int_{x^2+y^2=\alpha}\frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right) \\
&= \int_{\alpha=0}^{2\pi}\frac{1}{2\pi}\exp\left(-\frac{x(\sqrt{z},\alpha)^2 + y(\sqrt{z},\alpha)^2}{2}\right)\cdot|I(r,\alpha)| \\
&= \frac{1}{2\pi}\cdot\frac{1}{2}\cdot\int_{\alpha=0}^{2\pi}\exp\left(-\frac{z}{2}\right) = \frac{1}{2}\exp\left(-\frac{z}{2}\right).
\end{aligned}$$

As such, Z has an exponential distribution with $\lambda = 1/2$.

(iv) For $k = 1$ the claim is trivial. Otherwise, let $g_{k-1}(x) = \lambda \frac{(\lambda x)^{k-2}}{(k-2)!} \exp(-\lambda x)$. Observe that

$$\begin{aligned} g_k(t) &= \int_0^t g_{k-1}(t-x) g_1(x)\, dx = \int_0^t \left(\lambda \frac{(\lambda(t-x))^{k-2}}{(k-2)!} \exp(-\lambda(t-x))\right)(\lambda \exp(-\lambda x))\, dx \\ &= \int_0^t \lambda^2 \frac{(\lambda(t-x))^{k-2}}{(k-2)!} \exp(-\lambda t)\, dx \\ &= \lambda \exp(-\lambda t) \int_0^t \lambda \frac{(\lambda x)^{k-2}}{(k-2)!}\, dx = \lambda \exp(-\lambda t) \frac{(\lambda t)^{k-1}}{(k-1)!} = g_k(x). \end{aligned}$$

■

14.5.2 The proof

Lemma 14.5.3 *Let u be a unit vector in $\mathbb{R}^d$. For any even positive integer k, let $U_1, \ldots, U_k$ be random vectors chosen independently from the d-dimensional Gaussian distribution $N^d(0,1)$. For $X_i = u \cdot U_i$, define $W = W(u) = (X_1, \ldots, X_k)$ and $L = L(u) = \|W\|^2$. Then, for any $\beta > 1$, we have:*

1. $\mathbf{E}[L] = k$.
2. $\mathbf{Pr}[L \geq \beta k] \leq \frac{k+3}{2} \exp\left(-\frac{k}{2}(\beta - (1 + \ln \beta))\right)$.
3. $\mathbf{Pr}[L \leq k/\beta] \leq 6k \exp\left(-\frac{k}{2}(\beta^{-1} - (1 - \ln \beta))\right)$.

Proof: By Lemma 14.5.2 (ii) each X_i is distributed as $N(0,1)$, and $X_1, \ldots, X_k$ are independent. Define $Y_i = X_{2i-1}^2 + X_{2i}^2$, for $i = 1, \ldots, \tau$, where $\tau = k/2$. By Lemma 14.5.2 (iii) Y_i follows the exponential distribution with parameter $\lambda = 1/2$. Let $L = \sum_{i=1}^{\tau} Y_i$. By Lemma 14.5.2 (iv), the variable L follows the Gamma distribution $(k/2, 1/2)$, and its expectation is $\mathbf{E}[L] = \sum_{i=1}^{k/2} \mathbf{E}[Y_i] = 2\tau = k$.

Now, let $\eta = \beta\tau = \beta k/2$, we have

$$\mathbf{Pr}[L \geq \beta k] = 1 - \mathbf{Pr}[L \leq \beta k] = 1 - G_{1/2,\tau}(\beta k) = \sum_{i=0}^{\tau} e^{-\eta} \frac{\eta^i}{i!} \leq (\tau + 1) e^{-\eta} \frac{\eta^\tau}{\tau!},$$

since $\eta = \beta\tau > \tau$, as $\beta > 1$. Now, since $\tau! \geq (\tau/e)^\tau$, as can be easily verified[5], and thus

$$\begin{aligned} \mathbf{Pr}[L \geq \beta k] &\leq (\tau+1) e^{-\eta} \frac{\eta^\tau}{\tau^\tau / e^\tau} = (\tau+1) e^{-\eta} \left(\frac{e\eta}{\tau}\right)^\tau = (\tau+1) e^{-\beta\tau} \left(\frac{e\beta\tau}{\tau}\right)^\tau \\ &= (\tau+1) e^{-\beta\tau} \cdot \exp(\tau \ln(e\beta)) = (\tau+1) \exp(-\tau(\beta - (1 + \ln\beta))) \\ &= \frac{k+3}{2} \exp\left(-\frac{k}{2}(\beta - (1 + \ln\beta))\right). \end{aligned}$$

Arguing in a similar fashion, we have, for $\nu = \lceil 2e\tau \rceil$, that

$$\begin{aligned} \mathbf{Pr}[L \leq k/\beta] &= \sum_{i=\tau}^{\infty} e^{-\tau/\beta} \frac{(\tau/\beta)^i}{i!} \leq e^{-\tau/\beta} \sum_{i=\tau}^{\infty} \left(\frac{e\tau}{i\beta}\right)^i = e^{-\tau/\beta} \left[\sum_{i=\tau}^{\nu} \left(\frac{e\tau}{i\beta}\right)^i + \sum_{i=\nu+1}^{\infty} \left(\frac{e\tau}{i\beta}\right)^i\right] \\ &\leq e^{-\tau/\beta} \left[\sum_{i=\tau}^{\nu} \left(\frac{e\tau}{i\beta}\right)^i + \frac{1}{(2\beta)^\nu}\right] \leq 2 e^{-\tau/\beta} \sum_{i=\tau}^{\nu} \left(\frac{e\tau}{i\beta}\right)^i, \end{aligned}$$

since $(e\tau/\tau\beta)^\tau \geq 1/(2\beta)^\nu$. As the sequence $(e\tau/i\beta)^i$ is decreasing for $i > \tau/\beta$, as can be easily verified[6], we can bound the (decreasing) summation above by

$$\sum_{i=\tau}^{\nu} \left(\frac{e\tau}{i\beta}\right)^i \leq \nu \left(\frac{e}{\beta}\right)^\tau = \lceil 2e\tau \rceil \exp(\tau(1 - \ln\beta)).$$

We conclude

$$\mathbf{Pr}[L \leq k/\beta] \leq 2 \lceil 2e\tau \rceil \exp(-\tau/\beta + \tau(1 - \ln\beta)) \leq 6k \exp\left(-\frac{k}{2}(\beta^{-1} - (1 - \ln\beta))\right).$$

■

[5] Indeed, $\ln \tau! = \sum_{i=1}^{\tau} \ln i \geq \int_{x=1}^{n} \ln x\, dx = \Big[x \ln x - x\Big]_{x=1}^{n} = n \ln n - n + 1 \geq n \ln n - n = \ln((n/e)^n)$.

[6] Indeed, consider the function $f(x) = x \ln(c/x)$, its derivative is $f'(x) = \ln(c/x) - 1$, and as such $f'(x) = 0$, for $x = c/e$. Namely, for $c = e\tau/\beta$, the function $f(x)$ achieves its maximum at $x = \tau/\beta$, and from this point on the function is decreasing.

Next, we show how to interpret the inequalities of Lemma 14.5.3 in a somewhat more intuitive way. Let $\beta = 1 + \varepsilon$, for ε such that $1 > \varepsilon > 0$. From the Taylor expansion of $\ln(1 + x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{i+1} x^{i+1}$, it follows that $\ln\beta \leq \varepsilon - \varepsilon^2/2 + \varepsilon^3/3$. By plugging it into the upper bound for $\mathbf{Pr}[L \geq \beta k]$ we get

$$\mathbf{Pr}[L \geq \beta k] \quad \leq \quad \frac{k+3}{2} \exp\left(-\frac{k}{2}(1 + \varepsilon - 1 - \varepsilon + \varepsilon^2/2 - \varepsilon^3/3)\right) \leq \frac{k+3}{2} \exp\left(-\frac{k}{2}(\varepsilon^2/2 - \varepsilon^3/3)\right)$$

On the other hand, since $\ln\beta \geq \varepsilon - \varepsilon^2/2$, we have $\mathbf{Pr}[L \leq k/\beta] \leq 6k \exp(\Delta)$, where

$$\begin{aligned} \Delta &= -\frac{k}{2}(\beta^{-1} - (1 - \ln\beta)) \leq -\frac{k}{2}\left(\frac{1}{1+\varepsilon} - 1 + \varepsilon - \frac{\varepsilon^2}{2}\right) \\ &\leq -\frac{k}{2}\left(\frac{\varepsilon^2}{1+\varepsilon} - \frac{\varepsilon^2}{2}\right) = -\frac{k}{2} \cdot \frac{\varepsilon^2 - \varepsilon^3}{2(1+\varepsilon)} \end{aligned}$$

Thus, the probability that a given unit vector gets distorted by more than $(1+\varepsilon)$ in any direction[7] grows roughly as $\exp(-k\varepsilon^2/4)$, for small $\varepsilon > 0$. Therefore, if we are given a set P of n points in l_2, we can set k to roughly $8\ln(n)/\varepsilon^2$ and make sure that with non-zero probability we obtain projection which does not distort distances[8] between *any* two different points from P by more than $(1 + \varepsilon)$ in each direction.

Theorem 14.5.4 *Let P be a set of n points in $\mathbb{R}^d$, $0 < \varepsilon, \delta < 1/2$, and $k = 16\ln(n/\delta)/\varepsilon^2$. Let $U_1, \ldots, U_k$ be random vectors chosen independently from the d-dimensional Gaussian distribution $N^d(0, 1)$, and let $T(x) = (U_1 \cdot x, \ldots, U_k \cdot x)$ be a linear transformation. Then, with probablity $\geq 1 - \delta$, for any $\mathsf{p}, \mathsf{q} \in \mathsf{P}$, we have that*

$$\frac{1}{(1+\varepsilon)} \sqrt{k}\, \|\mathsf{p} - \mathsf{q}\| \leq \|T(\mathsf{p}) - T(\mathsf{q})\| \leq (1+\varepsilon)\sqrt{k}\, \|\mathsf{p} - \mathsf{q}\|.$$

Sometime it is useful to be able to handle high distortion.

Corollary 14.5.5 *Let k be the target dimension of the transformation T of Theorem 14.5.4 and $\beta \geq 3$ a parameter. We have that*

$$\|T(\mathsf{p}) - T(\mathsf{q})\| \leq \beta\sqrt{k}\, \|\mathsf{p} - \mathsf{q}\|,$$

for any two points $\mathsf{p}, \mathsf{q} \in \mathsf{P}$, and this holds with probability $\geq 1 - \exp\left(-\frac{k\beta^2}{32\ln n}\right)$.

14.6 Bibliographical notes

Our presentation follows Matoušek [Mat02]. The Brunn-Minkowski inequality is a powerful inequality which is widely used in mathematics. A nice survey of this inequality and its applications is provided by Gardner [Gar02]. Gardner says: "In a sea of mathematics, the Brunn-Minkowski inequality appears like an octopus, tentacles reaching far and wide, its shape and color changing as it roams from one area to the next." However, Gardner is careful in claiming that the Brunn-Minkowski inequality is one of the most powerful inequalities in mathematics since as a wit put it "the most powerful inequality is $x^2 \geq 0$, since all inequalities are in some sense equivalent to it."

A striking application of the Brunn-Minkowski inequality is the proof that in any partial ordering of n elements, there is a single comparison that knowing its result, reduces the number of linear extensions that are consistent with the partial ordering, by a constant fraction. This immediately implies (the uninteresting result) that one can sort n elements in $O(n \log n)$ comparisons. More interestingly, it implies that if there are m linear extensions of the current partial ordering, we can *always* sort it using $O(\log m)$ comparisons. A nice exposition of this surprising result is provided by Matoušek [Mat02, Section 12.3].

The probability review of Section 14.5.1 can be found in Feller [Fel71]. The alternative proof of the Johnson-Lindenstrauss lemma of Section 14.5.2 is described by Indyk and Motwani [IM98] but earlier proofs are known [Dur95]. It exposes the fact that the Johnson-Lindenstrauss lemma is no more than yet another instance of the concentration of mass phenomena (i.e., like the Chernoff inequality). The alternative proof is provided since it is conceptually simpler (although the computations are more involved), and it is technically easier to use. Another alternative proof is provided by Dasgupta and Gupta [DG03].

Interestingly, it is enough to pick each entry in the dimension reducing matrix randomly out of $-1, 0, 1$. This requires more involved proof [Ach01]. This is useful when one care about storing this dimension reduction transformation efficiently.

Magen [Mag02] observed that in fact the JL lemma preserves angles, and in fact can be used to preserve any "k dimensional angle", by projecting down to dimension $O(k\varepsilon^{-2}\log n)$. In particular, Exercise 14.7.5 is taken from there.

[7] Note that this implies distortion $(1 + \varepsilon)^2$ if we require the mapping to be a contraction.

[8] In fact, this statement holds even for the *square* of the distances.

In fact, the random embedding preserves much more structure than just distances between points. It preserves the structure and distances of surfaces as long as they are low dimensional and "well behaved", see [AHY07] for some results in this direction.

Dimension reduction is crucial in learning, AI, databases, etc. One common technique that is being used in practice is to do PCA (i.e., principal component analysis) and take the first few main axises. Other techniques include independent component analysis, and MDS (multidimensional scaling). MDS tries to embed points from high dimensions into low dimension ($d = 2$ or 3), which preserving some properties. Theoretically, dimension reduction into really low dimensions is hopeless, as the distortion in the worst case is $\Omega(n^{1/(k-1)})$, if k is the target dimension [Mat90].

14.7 Exercises

Exercise 14.7.1 (Boxes can be separated.) [1 Points]

(Easy.) Let A and B be two axis-parallel boxes that are interior disjoint. Prove that there is always an axis-parallel hyperplane that separates the interior of the two boxes.

Exercise 14.7.2 (Brunn-Minkowski inequality slight extension.) [4 Points]

> **Corollary 14.7.3** *For A and B compact sets in $\mathbb{R}^n$, we have for any $\lambda \in [0,1]$ that* $\mathrm{Vol}(\lambda A + (1-\lambda)B) \geq \mathrm{Vol}(A)^{\lambda} \mathrm{Vol}(B)^{1-\lambda}$.

Exercise 14.7.4 (Projections are contractions.) [1 Points]

(Easy.) Let $\mathcal{F}$ be a k-dimensional affine subspace, and let $P_{\mathcal{F}} : \mathbb{R}^d \to \mathcal{F}$ be the projection that maps every point $x \in \mathbb{R}^d$ to its nearest neighbor on $\mathcal{F}$. Prove that p is a contraction (i.e., 1-Lipschitz). Namely, for any $\mathsf{p}, \mathsf{q} \in \mathbb{R}^d$, it holds that $\|P_{\mathcal{F}}(\mathsf{p}) - P_{\mathcal{F}}(\mathsf{q})\| \leq \|\mathsf{p} - \mathsf{q}\|$.

Exercise 14.7.5 (JL Lemma works for angles.) [10 Points]

Show that the Johnson-Lindenstrauss lemma also $(1 \pm \varepsilon)$-preserves angles among triples of points of P (you might need to increase the target dimension however by a constant factor). [**Hint:** For every angle, construct a equilateral triangle that its edges are being preserved by the projection (add the vertices of those triangles [conceptually] to the point set being embedded). Argue, that this implies that the angle is being preserved.**]**

Chapter 15

ANN in High Dimensions

15.1 ANN on the Hypercube

15.1.1 Hypercube and Hamming distance

Definition 15.1.1 The set of points $\mathcal{H}^d = \{0,1\}^d$ is the d-dimensional hypercube. A point $p = (p_1, \ldots, p_d) \in \mathcal{H}^d$ can be interpreted, naturally, as a binary string $p_1 p_2 \ldots p_d$. The *Hamming distance* $\mathbf{d}_H(p,q)$ between $p, q \in \mathcal{H}^d$, is the number of coordinates where p and q disagree.

It is easy to verify that the Hamming distance comply with the triangle inequality, and is as such a metric.

As we saw in previous lectures, all we need to solve $(1+\varepsilon)$-ANN, is it is enough to efficiently solve the *approximate near neighbor* problem. Namely, given a set P of n points in $\mathcal{H}^d$, and radius $r > 0$ and parameter $\varepsilon > 0$, we want to decide for a query point q whether $\mathbf{d}_H(q,P) \le r$ or $\mathbf{d}_H(q,P) \ge (1+\varepsilon)r$.

Definition 15.1.2 For a set P of points, a data-structure $\mathrm{NNbr}_{\approx}(P, r, (1+\varepsilon)r)$ solves the *approximate near neighbor* problem, if given a query point q, the data-structure works as follows.

- If $\mathbf{d}(q,P) \le r$ then $\mathrm{NNbr}_{\approx}$ outputs a point $p \in P$ such that $\mathbf{d}(p,q) \le (1+\varepsilon)r$.
- If $\mathbf{d}(q,P) \ge (1+\varepsilon)r$, in this case $\mathrm{NNbr}_{\approx}$ outputs that "$\mathbf{d}(q,P) \ge r$".
- If $r \le \mathbf{d}(q,P) \le (1+\varepsilon)r$, either of the above answers is acceptable.

Given such a data-structure $\mathrm{NNbr}_{\approx}(P, r, (1+\varepsilon)r)$, one can construct a data-structure that answers ANN using $O(\log(n/\varepsilon))$ queries.

15.1.2 Constructing NNbr for the Hamming cube

Let $P = \{p_1, \ldots, p_n\}$ be a subset of vertices of the hypercube in d dimensions. In the following we assume that $d = n^{O(1)}$. Let $r, \varepsilon > 0$ be two prespecified parameters. We are interested in building an $\mathrm{NNbr}_{\approx}$ for balls of radius r in the Hamming distance.

Definition 15.1.3 Let U be a (small) positive integer. A family $\mathcal{F} = \{h : S \to [0, U]\}$ of functions, is an (r, R, α, β)-sensitive if for any $u, q \in S$, we have:

- If $u \in \mathbf{b}(q, r)$ then $\mathbf{Pr}[h(u) = h(v)] \ge \alpha$.
- If $u \notin \mathbf{b}(q, R)$ then $\mathbf{Pr}[h(u) = h(v)] \le \beta$,

where h is randomly picked from $\mathcal{F}$, $r < R$, and $\alpha > \beta$.

Intuitively, if we can construct a (r, R, α, β)-sensitive family, then we can distinguish between two points which are close together, and two points which are far away from each other. Of course, the probabilities α and β might be very close to each other, and we need a way to do amplification.

Lemma 15.1.4 *For the hypercube $\mathcal{H}^d = \{0,1\}^d$, and a point $b = (b_1, \ldots, b_d) \in \mathcal{H}^d$, let $\mathcal{F}$ be the set of functions*

$$\left\{ h_i(b) = b_i \;\middle|\; b = (b_1, \ldots, b_d) \in \mathcal{H}^d, \text{ for } i = 1, \ldots, d \right\}.$$

Then for any r, ε, the family $\mathcal{F}$ is $\left(r, (1+\varepsilon)r, 1 - \frac{r}{d}, 1 - \frac{r(1+\varepsilon)}{d}\right)$-sensitive.

Proof: If $u, v \in \{0,1\}^d$ are in distance smaller than r from each other (under the Hamming distance), then they differ in at most r coordinates. The probability that $h \in \mathcal{F}$ would project into a coordinate that u and v agree on is $\geq 1 - r/d$.

Similarly, if $\mathbf{d}_H(u,v) \geq (1+\varepsilon)r$ then the probability that h would map into a coordinate that u and v agree on is $\leq 1-(1+\varepsilon)r/d$. ■

Let k be a parameter to be specified shortly. Let

$$\mathcal{G}(\mathcal{F}) = \left\{ g : \{0,1\}^d \to \{0,1\}^k \;\middle|\; g(u) = (h^1(u), \ldots, h^k(u)), \text{ for } h^1, \ldots, h^k \in \mathcal{F} \right\}.$$

Intuitively, $\mathcal{G}$ is a family that extends $\mathcal{F}$ by probing into k coordinates instead of only one coordinate.

15.1.3 Construction the near-neighbor data-structure

Let τ be (yet another) parameter to be specified shortly. We pick $g_1, \ldots, g_\tau$ functions randomly and uniformly from $\mathcal{G}$. For each point $u \in P$ compute $g_1(u), \ldots, g_\tau(u)$. We construct a hash table H_i to store all the values of $g_i(p_1), \ldots, g_i(p_n)$, for $i = 1, \ldots, \tau$.

Given a query point $q \in \mathcal{H}^d$, we compute $p_1(q), \ldots, p_\tau(q)$, and retrieve all the points stored in those buckets in the hash tables $H_1, \ldots, H_\tau$, respectively. For every point retrieved, we compute its distance to q, and if this distance is $\leq (1+\varepsilon)r$, we return it. If we encounter more than 4τ points we abort, and return '*fail*'. If no "close" point is encountered, the search returns '*fail*'.

We choose k and τ so that with constant probability (say larger than half) we have the following two properties:

(1) If there is a point $u \in P$, such that $\mathbf{d}_H(u,q) \leq r$, then $g_j(u) = g_j(q)$ for some j.

(2) Otherwise (i.e., $\mathbf{d}_H(u,q) \geq (1+\varepsilon)r$), the total number of points colliding with q in the τ hash tables, is smaller than 4τ.

Given a query point, $q \in \mathcal{H}^d$, we need to perform τ probes into τ hash tables, and retrieve at most 4τ results. Overall this takes $O(d^{O(1)}\tau)$ time.

Lemma 15.1.5 *If there is a $(r, (1+\varepsilon)r, \alpha, \beta)$-sensitive family $\mathcal{F}$ of functions for the hypercube, then there exists a $\mathrm{NNbr}_{\approx}(P, r, (1+\varepsilon)r)$ which uses $O\left(dn + n^{1+\rho}\right)$ space and $O(n^\rho)$ hash probes for each query, where*

$$\rho = \frac{\ln 1/\alpha}{\ln 1/\beta}.$$

This data-structure succeeds with constant probability.

Proof: It suffices to ensure that properties (1) and (2) holds with probability larger than half.

Set $k = \log_{1/\beta} n = \frac{\ln n}{\ln(1/\beta)}$, then the probability that for a random hash function $g \in \mathcal{G}(\mathcal{F})$, we have $g(p) = g(q)$ for $p \in P \setminus \mathbf{b}(q, (1+\varepsilon)r)$ is at most

$$\mathbf{Pr}[g(p') = g(q)] \leq \beta^k \leq \exp\left(\ln(\beta) \cdot \tfrac{\ln n}{\ln(1/\beta)}\right) \leq \frac{1}{n}.$$

Thus, the expected number of elements from $P \setminus \mathbf{b}(q, (1+\varepsilon)r)$ colliding with q in the jth hash table H_j is bounded by one. In particular, the overall expected number of such collisions in $H_1, \ldots, H_\tau$ is bounded by τ. By the Markov inequality we have that the probability that the collusions number exceeds 4τ is less than $1/4$; therefore the probability that the property (2) holds is $\geq 3/4$.

Next, for a point $p \in \mathbf{b}(q, r)$, consider the probability of $g_j(p) = g_j(q)$, for a fixed j. Clearly, it is bounded from below by

$$\geq \alpha^k = \alpha^{\log_{1/\beta} n} = n^{-\frac{\ln 1/\alpha}{\ln 1/\beta}} = n^{-\rho}.$$

Thus the probability that such a g_j exists is at least $1 - (1 - n^{-\rho})^\tau$. By setting $\tau = 2n^\rho$ we get property (1) holds with probability $\geq 1 - 1/e^2 > 4/5$. The claim follows. ■

Claim 15.1.6 *For $x \in [0,1)$ and $t \geq 1$ such that $1 - tx > 0$ we have* $\dfrac{\ln(1-x)}{\ln(1-tx)} \leq \dfrac{1}{t}$.

Proof: Since $\ln(1-tx) < 0$, it follows that the claim is equivalent to $t\ln(1-x) \geq \ln(1-tx)$. This in turn is equivalent to

$$g(x) \equiv (1-tx) - (1-x)^t \leq 0.$$

This is trivially true for $x = 0$. Furthermore, taking the derivative, we see $g'(x) = -t + t(1-x)^{t-1}$, which is non-positive for $x \in [0,1)$ and $t \geq 1$. Therefore, g is non-increasing in the region in which we are interested, and so $g(x) \leq 0$ for all values in this interval. ■

Lemma 15.1.7 *There exists a $\mathrm{NNbr}_{\approx}(r, (1+\varepsilon)r)$ which uses $O\left(dn + n^{1+1/(1+\varepsilon)}\right)$ space and $O(n^{1/(1+\varepsilon)})$ hash probes for each query. The probability of success (i.e., there is a point $u \in P$ such that $\mathbf{d}_H(u,q) \leq r$, and we return a point $v \in P$ such that $\|uv\| \leq (1+\varepsilon)r$) is a constant.*

Proof: By Lemma 15.1.4, we have a $(r, (1+\varepsilon)r, \alpha, \beta)$-sensitive family of hash functions, where $\alpha = 1 - \frac{r}{d}$ and $\beta = 1 - \frac{r(1+\varepsilon)}{d}$. As such

$$\rho = \frac{\ln 1/\alpha}{\ln 1/\beta} = \frac{\ln \alpha}{\ln \beta} = \frac{\ln \frac{d-r}{d}}{\ln \frac{d-(1+\varepsilon)r}{d}} = \frac{\ln\left(1 - \frac{r}{d}\right)}{\ln\left(1 - (1+\varepsilon)\frac{r}{d}\right)} \leq \frac{1}{1+\varepsilon},$$

by Claim 15.1.6. ■

By building $O(\log n)$ structures of Lemma 15.1.7, we can do probability amplification and get a correct result with High probability.

Theorem 15.1.8 *Given a set P of n points on the hypercube $\mathcal{H}^d$, parameters $\varepsilon > 0$ and $r > 0$, one can build a $\mathrm{NNbr}_{\approx} = \mathrm{NNbr}_{\approx}(P, r, (1+\varepsilon)r)$, such that given a query point q, one can decide if:*

- *$\mathbf{b}(q, r) \cap P \neq \emptyset$, then $\mathrm{NNbr}_{\approx}$ returns a point $u \in P$, such that $\mathbf{d}_H(u, q) \leq (1+\varepsilon)r$.*
- *$\mathbf{b}(q, (1+\varepsilon)r) \cap P = \emptyset$ then $\mathrm{NNbr}_{\approx}$ returns that no point is in distance $\leq r$ from q.*

In any other case, any of the answers is correct. The query time is $O(dn^{1/(1+\varepsilon)} \log n)$ and the space used is $O\left(dn + n^{1+1/(1+\varepsilon)} \log n\right)$. The result returned is correct with high probability.

Proof: Note, that every point can be stored only once. Any other reference to it in the data-structure can be implemented with a pointer. Thus, the $O(dn)$ requirement on the space. The other term follows by repeating the space requirement of Lemma 15.1.7 $O(\log n)$ times. ■

In the hypercube case, when $d = n^{O(1)}$, we can just build $M = O(\varepsilon^{-1} \log n)$ such data-structures such that $(1+\varepsilon)$-ANN can be answered using binary search on those data-structures, which corresponds to radiuses $r_1, \ldots, r_M$, where $r_i = (1+\varepsilon)^i$.

Theorem 15.1.9 *Given a set P of n points on the hypercube $\mathcal{H}^d$ (where $d = n^{O(1)}$) parameters $\varepsilon > 0$ and $r > 0$, one can build ANN data-structure using $O\left((d + n^{1/(1+\varepsilon)})\varepsilon^{-1} n \log^2 n\right)$ space, such that given a query point q, one can returns an ANN in P (under the Hamming distance) in $O(dn^{1/(1+\varepsilon)} \log(\varepsilon^{-1} \log n))$ time. The result returned is correct with high probability.*

15.2 LSH and ANN in Euclidean Space

15.2.1 Preliminaries

Lemma 15.2.1 *Let $X = (X_1, \ldots, X_d)$ be a vector of d independent variables which have distribution $N(0, 1)$, and let $v = (v_1, \ldots, v_d) \in \mathbb{R}^d$. We have that $v \cdot X = \sum_i v_i X_i$ is distributed as $\|v\| Z$, where $Z \sim N(0, 1)$.*

Proof: If $\|v\| = 1$ then this holds by the symmetry of the normal distribution. Indeed, let $e_1 = (1, 0, \ldots, 0)$. By the symmetry of the d-dimensional normal distribution, we have that $v \cdot X \sim e_1 \cdot X = X_1 \sim N(0, 1)$.

Otherwise, $v \cdot X / \|V\| \sim N(0, 1)$, and as such $v \cdot X \sim N\left(0, \|v\|^2\right)$, which is indeed the distribution of $\|v\| Z$. ■

A d-dimensional distribution that has the property of Lemma 15.2.1, is called a 2-*stable distribution*.

15.2.2 Locality Sensitive Hashing

Let q, r be two points in $\mathbb{R}^d$. We want to perform an experiment to decide if $\|\mathsf{q} - \mathsf{r}\| \leq 1$ or $\|\mathsf{q} - \mathsf{r}\| \geq \eta$, where $\eta = 1 + \varepsilon$. We will randomly choose a vector $\vec{v}$ from the d-dimensional normal distribution $N^d(0, 1)$ (which is 2-stable). Next, let r be a parameter, and let t be a random number chosen uniformly from the interval $[0, r]$. For $p \in \mathbb{R}^d$, and consider the random hash function

$$h(p) = \left\lfloor \frac{p \cdot \vec{v} + t}{r} \right\rfloor. \tag{15.1}$$

If p and q are in distance η from each other, and when we project to $\vec{v}$, the distance between the projection is t, then the probability that they get the same hash value is $1 - t/r$, since this is the probability that the random sliding will not separate them. As such, we have that the probability of collusion is

$$\alpha(\eta) = \mathbf{Pr}[h(p) = h(q)] = \int_{t=0}^{r} \mathbf{Pr}\left[\left|p \cdot \vec{v} - q \cdot \vec{v}\right| = t\right]\left(1 - \frac{t}{r}\right) dt.$$

However, since $\vec{v}$ is chosen from a 2-stable distribution, we have that $p \cdot \vec{v} - q \cdot \vec{v} = (p - q) \cdot \vec{v} \sim N(0, \|pq\|^2)$. Since we are considering the absolute value of the variable, we need to multiply this by two. Thus, we have

$$\alpha(\eta, r) = \int_{t=0}^{r} \frac{2}{\sqrt{2\pi}\eta} \exp\left(-\frac{t^2}{2\eta^2}\right)\left(1 - \frac{t}{r}\right) dt.$$

Intuitively, we care about the difference $\alpha(1+\varepsilon, r) - \alpha(1, r)$, and we would like to maximize it as much as possible (by choosing the right value of r). Unfortunately, this integral is unfriendly, and we have to resort to numerical computation.

In fact, if are going to use this hashing scheme for constructing locality sensitive hashing, like in Lemma 15.1.5, then we care about the ratio

$$\rho(1+\varepsilon) = \min_r \frac{\log(1/\alpha(1))}{\log(1/\alpha(1+\varepsilon))}.$$

The following is verified using numerical computations on a computer,

Lemma 15.2.2 ([DNIM04]) *One can choose r, such that $\rho(1+\varepsilon) \leq \frac{1}{1+\varepsilon}$.*

Lemma 15.2.2 implies that the hash functions defined by Eq. (15.1) are $(1, 1+\varepsilon, \alpha', \beta')$-sensitive, and furthermore, $\rho = \frac{\log(1/\alpha')}{\log(1/\beta')} \leq \frac{1}{1+\varepsilon}$, for some values of α' and β'. As such, we can use this hashing family to construct $\mathrm{NNbr}_{\approx}$ for the set P of points in $\mathbb{R}^d$. Following the same argumentation of Theorem 15.1.8, we have the following.

Theorem 15.2.3 *Given a set P of n points in $\mathbb{R}^d$, parameters $\varepsilon > 0$ and $r > 0$, one can build a $\mathrm{NNbr}_{\approx} = \mathrm{NNbr}_{\approx}(P, r, (1+\varepsilon)r)$, such that given a query point q, one can decide if:*

- *$\mathbf{b}(q, r) \cap P \neq \emptyset$, then $\mathrm{NNbr}_{\approx}$ returns a point $u \in P$, such that $\mathbf{d}_H(u, q) \leq (1+\varepsilon)r$.*
- *$\mathbf{b}(q, (1+\varepsilon)r) \cap P = \emptyset$ then $\mathrm{NNbr}_{\approx}$ returns that no point is in distance $\leq r$ from q.*

In any other case, any of the answers is correct. The query time is $O(dn^{1/(1+\varepsilon)} \log n)$ and the space used is $O\left(dn + n^{1/(1+\varepsilon)} n \log n\right)$. The result returned is correct with high probability.

15.2.3 ANN in High Dimensional Euclidean Space

Unlike the hypercube case, where we could just do direct binary search on the distances. Here we need to use the reduction from ANN to near-neighbor queries. We will need the following result (which follows from what we had seen in previous lectures).

Theorem 15.2.4 *Given a set P of n points in $\mathbb{R}^d$, then one can construct data-structures $\mathcal{D}$ that answers $(1+\varepsilon)$-ANN queries, by performing $O(\log(n/\varepsilon))$ $\mathrm{NNbr}_{\approx}$ queries. The total number of points stored at $\mathrm{NNbr}_{\approx}$ data-structures of $\mathcal{D}$ is $O(n\varepsilon^{-1} \log(n/\varepsilon))$.*

Constructing the data-structure of Theorem 15.2.4 requires building a low quality HST. Unfortunately, the previous construction seen for HST are exponential in the dimension, or take quadratic time. We next present a faster scheme.

15.2.3.1 Low quality HST in high dimensional Euclidean space

Lemma 15.2.5 *Let P be a set of n in $\mathbb{R}^d$. One can compute a nd-HST of P in $O(nd \log^2 n)$ time (note, that the constant hidden by the O notation does not depend on d).*

Proof: Our construction is based on a recursive decomposition of the point-set. In each stage, we split the point-set into two subsets. We recursively compute a nd-HST for each point-set, and we merge the two trees into a single tree, by creating a new vertex, assigning it an appropriate value, and hung the two subtrees from this node. To carry this out, we try to separate the set into two subsets that are furthest away from each other.

Let $R = R(P)$ be the minimum axis parallel box containing P, and let $\nu = l(P) = \sum_{i=1}^{d} \|I_i(R)\|$, where $I_i(R)$ is the projection of R to the ith dimension.

Clearly, one can find an axis parallel strip H of width $\geq \nu/((n-1)d)$, such that there is at least one point of P on each of its sides, and there is no points of P inside H. Indeed, to find this strip, project the point-set into the ith dimension, and find the longest interval between two consecutive points. Repeat this process for $i = 1, \ldots, d$, and use the longest interval encountered. Clearly, the strip H corresponding to this interval is of width $\geq \nu/((n-1)d)$. On the other hand, $\mathrm{diam}(P) \leq \nu$.

Now recursively continue the construction of two trees T^+, T^-, for P^+, P^-, respectively, where P^+, P^- is the splitting of P into two sets by H. We hung T^+ and T^- on the root node v, and set $\Delta_v = \nu$. We claim that the resulting tree T is a nd-HST. To this end, observe that $\mathrm{diam}(P) \leq \Delta_v$, and for a point $p \in P^-$ and a point $q \in P^+$, we have $\|pq\| \geq \nu/((n-1)d)$, which implies the claim.

To construct this efficiently, we use an efficient search trees to store the points according to their order in each coordinate. Let $D_1, \ldots, D_d$ be those trees, where D_i store the points of P in ascending order according to the ith axis, for $i = 1, \ldots, d$. We modify them, such that for every node $v \in D_i$, we know what is the largest empty interval along the ith axis for the points P_v (i.e., the points stored in the subtree of v in D_i). Thus, finding the largest strip to split along, can be done in $O(d \log n)$ time. Now, we need to split the d trees into two families of d trees. Assume we split according to the first axis. We can split D_1 in $O(\log n)$ time using the splitting operation provided by the search tree (Treaps for example can do this split in $O(\log n)$ time). Let assume that this split P into two sets L and R, where $|L| < |R|$.

We still need to split the other $d-1$ search trees. This is going to be done by deleting all the points of L from those trees, and building $d-1$ new search trees for L. This takes $O(|L|\, d \log n)$ time. We charge this work to the points of L.

Since in every split, only the points in the smaller portion of the split get charged, it follows that every point can be charged at most $O(\log n)$ time during this construction algorithm. Thus, the overall construction time is $O(dn \log^2 n)$ time. ■

15.2.3.2 The overall result

Plugging Theorem 15.2.3 into Theorem 15.2.4, we have:

Theorem 15.2.6 *Given a set P of n points in $\mathbb{R}^d$, parameters $\varepsilon > 0$ and $r > 0$, one can build ANN data-structure using*

$$O\Big(dn + n^{1+1/(1+\varepsilon)}\varepsilon^{-2}\log^3(n/\varepsilon)\Big)$$

space, such that given a query point q, one can returns an $(1+\varepsilon)$-ANN in P in

$$O\Big(dn^{1/(1+\varepsilon)}(\log n)\log\frac{n}{\varepsilon}\Big)$$

time. The result returned is correct with high probability.

The construction time is $O\Big(dn^{1+1/(1+\varepsilon)}\varepsilon^{-2}\log^3(n/\varepsilon)\Big)$.

Proof: We compute the low quality HST using Lemma 15.2.5. This takes $O(nd\log^2 n)$ time. Using this HST, we can construct the data-structure $\mathcal{D}$ of Theorem 15.2.4, where we do not compute the $\mathrm{NNbr}_{\approx}$ data-structures. We next traverse the tree $\mathcal{D}$, and construct the $\mathrm{NNbr}_{\approx}$ data-structures using Theorem 15.2.3.

We only need to prove the bound on the space. Observe, that we need to store each point only once, since other place can refer to the point by a pointer. Thus, this is the $O(nd)$ space requirement. The other term comes from plugging the bound of Theorem 15.2.4 into the bound of Theorem 15.2.3. ■

15.3 Bibliographical notes

Section 15.1 follows the exposition of Indyk and Motwani [IM98]. The fact that one can perform approximate nearest neighbor in high dimensions in time and space polynomial in the dimension is quite surprising, One can reduce the approximate near-neighbor in euclidean space to the same question on the hypercube (we show the details below). This implies together with the reduction from ANN to approximate near-neighbor (seen in previous lectures) that one can answer ANN in high dimensional euclidean space with similar performance. Kushilevitz, Ostrovsky and Rabani [KOR00] offered an alternative data-structure with somewhat inferior performance.

The value of the results showed in this write-up depend to large extent on the reader perspective. Indeed, for small value of $\varepsilon > 0$, the query time $O(dn^{1/(1+\varepsilon)})$ is very close to linear dependency on n, and is almost equivalent to just scanning the points. Thus, from low dimension perspective, where ε is assumed to be small, this result is slightly sublinear. On the other hand, if one is willing to pick ε to be large (say 10), then the result is clearly better than the naive algorithm, suggesting running time for an ANN query which takes (roughly) $n^{1/11}$.

The idea of doing locality sensitive hashing directly on the Euclidean space, as done in Section 15.2 is not shocking after seeing the Johnson-Lindenstrauss lemma. It is taken from a recent paper of Datar *et al.* [DNIM04]. In particular, the current analysis which relies on computerized estimates is far from being satisfactory. It would be nice to have a simpler and more elegant scheme for this case. This is an open problem for further research. Another open problem is to improve the performance of the LSH scheme.

The low-quality high-dimensional HST construction of Lemma 15.2.5, is taken from [Har01b]. The running time of this lemma can be further improved to $O(dn\log n)$ by more careful and involved implementation, see [CK95] for details.

From approximate near-neighbor in $\mathbb{R}^d$ to approximate near-neighbor on the hypercube. The reduction is quite involved, and we only sketch the details. Let P Be a set of n points in $\mathbb{R}^d$. We first reduce the dimension to $k = O(\varepsilon^{-2} \log n)$ using the Johnson-Lindenstrauss lemma. Next, we embed this space into $\ell_1^{k'}$ (this is the space $\mathbb{R}^k$, where distances are the L_1 metric instead of the regular L_2 metric), where $k' = O(k/\varepsilon^2)$. This can be done with distortion $(1 + \varepsilon)$.

Let Q the resulting set of points in $\mathbb{R}^{k'}$. We want to solve $\text{NNbr}_{\approx}$ on this set of points, for radius r. As a first step, we partition the space into cells by taking a grid with sidelength (say) $k'r$, and randomly translating it, clipping the points inside each grid cell. It is now sufficient to solve the $\text{NNbr}_{\approx}$ inside this grid cell (which has bounded diameter as a function of r), since with small probability that the result would be correct. We amplify the probability by repeating this polylogarithmic number of times.

Thus, we can assume that P is contained inside a cube of side length $\leq k'nr$, and it is in $\mathbb{R}^{k'}$, and the distance metric is the L_1 metric. We next, snap the points of P to a grid of sidelength (say) $\varepsilon r/k'$. Thus, every point of P now has an integer coordinate, which is bounded by a polynomial in $\log n$ and $1/\varepsilon$. Next, we write the coordinates of the points of P using unary notation. (Thus, a point $(2, 5)$ would be written as $(010, 101)$ assuming the number of bits for each coordinates is 3.) It is now easy to verify that the hamming distance on the resulting strings, is equivalent to the L_1 distance between the points.

Thus, we can solve the near-neighbor problem for points in $\mathbb{R}^d$ by solving it on the hypercube under the Hamming distance.

See Indyk and Motwani [IM98] for more details.

This relationship indicates that the ANN on the hypercube is "equivalent" to the ANN in Euclidean space. In particular, making progress on the ANN on the hypercube would probably lead to similar progress on the Euclidean ANN problem.

We had only scratched the surface of proximity problems in high dimensions. The interested reader is referred to the survey by Indyk [Ind04] for more information.

Chapter 16

Approximating a Convex Body by An Ellipsoid

16.1 Some Linear Algebra

In the following, we cover some material from linear algebra. Proofs of those facts can be found in any text on linear algebra, for example [Leo98].

For a matrix A, let A^T denote the transposed matrix. We remind the reader that for two matrices A and B, we have $(AB)^T = B^T A^T$. Furthermore, for any three matrices A, B and C, we have $(AB)C = A(BC)$.

A matrix $A \in \mathbb{R}^{n\times n}$ is *symmetric* if $A^T = A$. All the eigenvalues of a symmetric matrix are real numbers. A matrix A is *positive definite* if $x^T Ax > 0$, for all $x \in \mathbb{R}^n$. Among other things this implies that A is non-singular. If A is symmetric then it is positive definite if and only if all its eigenvalues are positive numbers.

In particular, if A is symmetric positive definite then $\det(A) > 0$.

For two vectors $u, v \in \mathbb{R}^n$, let $\langle u, v\rangle = u^T v$ denote their dot product.

16.2 Ellipsoids

Definition 16.2.1 Let $\mathbf{b} = \left\{ x \;\middle|\; \|x\| \le 1 \right\}$ be the unit ball. Let $a \in \mathbb{R}^n$ be a vector and let $T : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible linear transformation. The set

$$\mathcal{E} = T(\mathbf{b}) + a$$

is called an *ellipsoid* and a is its *center*.

Alternatively, we can write

$$\mathcal{E} = \left\{ x \in \mathbb{R}^n \;\middle|\; \left\| T^{-1}(x-a) \right\| \le 1 \right\}.$$

However,

$$\left\| T^{-1}(x-a) \right\| = \left\langle T^{-1}(x-a), T^{-1}(x-a) \right\rangle = (T^{-1}(x-a))^T T^{-1}(x-a) = (x-a)^T \left(T^{-1}\right)^T T^{-1}(x-a).$$

In particular, let $Q = \left(T^{-1}\right)^T T^{-1}$. Observe that Q is symmetric, and that it is positive definite. Thus,

$$\mathcal{E} = \left\{ x \in \mathbb{R}^n \;\middle|\; (x-a)^T Q(x-a) \le 1 \right\}.$$

If we change the basis of $\mathbb{R}^n$ to be the set of unit eigenvectors of $\mathcal{E}$, then Q becomes a diagonal matrix, and we have that

$$\mathcal{E} = \left\{ (y_1, \ldots, y_n) \in \mathbb{R}^n \;\middle|\; \lambda_1 (y_1 - a_1)^2 + \cdots + \lambda_n (y_n - a_n)^2 \le 1 \right\},$$

where $a = (a_1, \ldots, a_n)$ and $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of Q. In particular, this implies that the point $\left(a_1, \ldots, a_i \pm 1/\sqrt{\lambda_i}, \ldots, a_n\right) \in \partial\mathcal{E}$, for $i = 1, \ldots, n$. In particular,

$$\mathrm{Vol}(\mathcal{E}) = \frac{\mathrm{Vol}(\mathbf{b})}{\sqrt{\lambda_1} \cdots \sqrt{\lambda_n}} = \frac{\mathrm{Vol}(\mathbf{b})}{\sqrt{\det(Q)}}.$$

For a convex body K (i.e., a convex and bounded set), let $\mathcal{E}$ be a largest volume ellipsoid contained inside K. One can show that $\mathcal{E}$ is unique. Namely, there is a single maximum volume ellipsoid inside K.

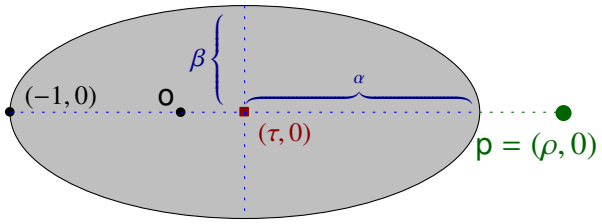
Theorem 16.2.2 *Let $\mathsf{K} \subset \mathbb{R}^n$ be a convex body, and let $\mathcal{E} \subseteq \mathsf{K}$ be its maximum volume ellipsoid. Suppose that $\mathcal{E}$ is centered at the origin, then $\mathsf{K} \subseteq n\,\mathcal{E} = \left\{ nx \;\middle|\; x \in \mathcal{E} \right\}$.*

Proof: By applying a linear transformation, we can assume that $\mathcal{E}$ is the unit ball $\mathbf{b}$. And assume for the sake of contradiction that there is a point $\mathsf{p} \in \mathsf{K}$, such that $\|\mathsf{p}\| > n$. Consider the set C which is the convex-hull of $\{\mathsf{p}\} \cup \mathbf{b}$. Since K is convex, we have that $C \subseteq \mathsf{K}$.

We will reach a contradiction, by finding an ellipsoid $\mathcal{G}$ which has volume larger than $\mathbf{b}$ and is enclosed inside C.

By rotating space, we can assume that the apex p of C is the point $(\rho, 0, \ldots, 0)$, for $\rho > n$. We consider ellipsoids of the form

$$\mathcal{G} = \left\{ (y_1, \ldots, y_n) \;\middle|\; \frac{(y_1 - \tau)^2}{\alpha^2} + \frac{1}{\beta^2} \sum_{i=2}^{n} y_i^2 \leq 1 \right\}.$$



We need to pick the values of τ, α and β such that $\mathcal{G} \subseteq C$. Observe that by symmetry, it is enough to enforce that $\mathcal{G} \subseteq C$ in the first two dimensions.

Thus, we can consider C and $\mathcal{G}$ to be in two dimensions. Now, the center of $\mathcal{G}$ is going to be on the x-axis, at the point $(\tau, 0)$. The set $\mathcal{E}$ is just an ellipse with axises parallel to x and y. In particular, we require that $(-1, 0)$ would be on the boundary of $\mathcal{G}$. This implies that $(-1 - \tau)^2 = \alpha^2$ and that $0 \leq \tau \leq (\rho + (-1))/2$. Namely,

$$\alpha = 1 + \tau. \tag{16.1}$$

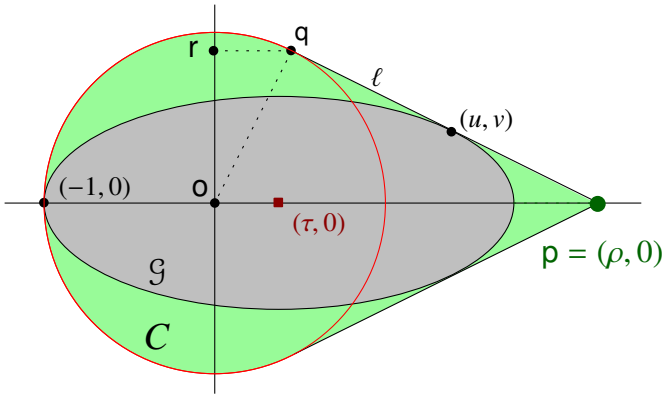
In particular, the equation of curve forming (the boundary) of $\mathcal{G}$ is

$$F(x, y) = \frac{(x - \tau)^2}{(1 + \tau)^2} + \frac{y^2}{\beta^2} - 1 = 0.$$

We next compute the value of β^2. Consider the tangent ℓ to the unit circle that passes through $\mathsf{p} = (\rho, 0)$. Let q be the point where ℓ touches the unit circle. See figure on the right. We have $\triangle \mathsf{poq} \triangleq \triangle \mathsf{oqr}$. As such

$$\frac{\|\mathsf{r} - \mathsf{q}\|}{\|\mathsf{q} - \mathsf{o}\|} = \frac{\|\mathsf{q} - \mathsf{o}\|}{\|\mathsf{o} - \mathsf{p}\|}.$$

Since $\|\mathsf{q} - \mathsf{o}\| = 1$, we have $\|\mathsf{q} - \mathsf{r}\| = 1/\rho$. Furthermore, since q is on the unit circle, we have



$$\mathsf{q} = \left(1/\rho, \sqrt{1 - 1/\rho^2}\right).$$

As such, the equation of the line ℓ is

$$\left\langle \mathsf{q}, (x, y) \right\rangle = 1 \quad \Rightarrow \quad \frac{x}{\rho} + \sqrt{1 - 1/\rho^2}\, y = 1 \quad \Rightarrow \quad \ell \equiv y = -\frac{1}{\sqrt{\rho^2 - 1}} x - \frac{\rho}{\sqrt{\rho^2 - 1}}.$$

Next, consider the tangent ℓ' to $\mathcal{G}$ at (u, v). We will drive a formula for ℓ' as a function of (u, v) and then require that $\ell = \ell'$. The slope of the ℓ' is the slope of the tangent to $\mathcal{G}$ at (u, v), which is

$$\frac{dy}{dx} = -\frac{F_u(u, v)}{F_v(u, v)} = -\left(\frac{2(u - \tau)}{(1 + \tau)^2}\right) \Big/ \left(\frac{2v}{\beta^2}\right) = -\frac{\beta^2 (u - \tau)}{v(1 + \tau)^2}$$

by computing the derivatives of the implicit function F. The line ℓ' is just

$$\frac{(u - \tau)}{\alpha^2}(x - \tau) + \frac{v}{\beta^2} y = 1,$$

since it has the required slope and it passes through (u, v). Namely $\ell' \equiv y = -\dfrac{\beta^2 (u - \tau)}{v\alpha^2}(x - \tau) + \dfrac{\beta^2}{v}$. Setting $\ell = \ell$, we have that the line ℓ' passes through $(\rho, 0)$. As such,

$$\frac{(\rho - \tau)(u - \tau)}{\alpha^2} = 1 \;\Rightarrow\; \frac{(u - \tau)}{\alpha} = \frac{\alpha}{\rho - \tau} \;\Rightarrow\; \frac{(u - \tau)^2}{\alpha^2} = \frac{\alpha^2}{(\rho - \tau)^2}. \tag{16.2}$$

Since ℓ and ℓ' are the same line, we have that

$$\frac{\beta^2(u-\tau)}{v\alpha^2} = \frac{1}{\sqrt{\rho^2-1}}.$$

However,

$$\frac{\beta^2(u-\tau)}{v\alpha^2} = \frac{\beta^2}{v\alpha}\cdot\frac{(u-\tau)}{\alpha} = \frac{\beta^2}{v\alpha}\cdot\frac{\alpha}{\rho-\tau} = \frac{\beta^2}{v(\rho-\tau)}.$$

Thus,

$$\frac{\beta^2}{v} = \frac{\rho-\tau}{\sqrt{\rho^2-1}}.$$

Squaring and inverting both sides, we have $\frac{v^2}{\beta^4} = \frac{\rho^2-1}{(\rho-\tau)^2}$ and thus $\frac{v^2}{\beta^2} = \beta^2\frac{\rho^2-1}{(\rho-\tau)^2}$. The point $(u,v) \in \partial\mathcal{G}$, and as such $\frac{(u-\tau)^2}{\alpha^2} + \frac{v^2}{\beta^2} = 1$. Using Eq. (16.2) and the above, we get

$$\frac{\alpha^2}{(\rho-\tau)^2} + \beta^2\frac{\rho^2-1}{(\rho-\tau)^2} = 1,$$

and thus

$$\beta^2 = \frac{(\rho-\tau)^2-\alpha^2}{\rho^2-1} = \frac{(\rho-\tau)^2-(\tau+1)^2}{\rho^2-1} = \frac{(\rho-2\tau-1)(\rho+1)}{\rho^2-1} = \frac{\rho-2\tau-1}{\rho-1} = 1-\frac{2\tau}{\rho-1}.$$

Namely, for $n \geq 2$, and $0 \leq \tau < (\rho-1)/2$, we have that the ellipsoid $\mathcal{G}$ is defined by the parameters $\alpha = 1+\tau$ and β. It is contained inside the "cone" C. It holds that $\mathrm{Vol}(\mathcal{G}) = \beta^{n-1}\alpha\,\mathrm{Vol}(\mathbf{b})$, and thus

$$\mu = \ln\frac{\mathrm{Vol}(\mathcal{G})}{\mathrm{Vol}(\mathbf{b})} = (n-1)\ln\beta + \ln\alpha = \frac{n-1}{2}\ln\beta^2 + \ln\alpha.$$

For $\tau > 0$ sufficiently small, we have $\ln\alpha = \ln(1+\tau) = \tau + O(\tau^2)$, because of the Taylor expansion $\ln(1+x) = x - x^2/2 + x^3/3 - \cdots$, for $-1 < x \leq 1$. Similarly, $\ln\beta^2 = \ln\left(1-\frac{2\tau}{\rho-1}\right) = -\frac{2\tau}{\rho-1} + O(\tau^2)$. Thus,

$$\mu = \frac{n-1}{2}\left(-\frac{2\tau}{\rho-1} + O(\tau^2)\right) + \tau + O(\tau^2) = \left(1-\frac{n-1}{\rho-1}\right)\tau + O(\tau^2) > 0,$$

for τ sufficiently small and if $\rho > n$. Thus, $\mathrm{Vol}(\mathcal{G})/\mathrm{Vol}(\mathbf{b}) = \exp(\mu) > 1$ implying that $\mathrm{Vol}(\mathcal{G}) > \mathrm{Vol}(\mathbf{b})$. A contradiction. ■

A convex body K centered at the origin is *symmetric* if $p \in \mathsf{K}$, implies that $-p \in \mathsf{K}$. Interestingly, the constant in Theorem 16.2.2 can be improved to $\sqrt{n}$ in this case. We omit the proof, since it is similar to the proof of Theorem 16.2.2.

Theorem 16.2.3 *Let $\mathsf{K} \subset \mathbb{R}^n$ be a symmetric convex body, and let $\mathcal{E} \subseteq \mathsf{K}$ be its maximum volume ellipsoid. Suppose that $\mathcal{E}$ is centered at the origin, then $\mathsf{K} \subseteq \sqrt{n}\,\mathcal{E}$.*

16.3 Bibliographical notes

We closely follow the exposition of Barvinok [Bar02]. One can approximate the John ellipsoid using the interior point techniques [GLS88]. However, this is not very efficient in low dimensions. In the next lecture, we will show how to do this by other (simpler) techniques, which are faster (and simpler) for point sets in low dimensions.

The maximum volume ellipsoid is sometimes referred to as John's ellipsoid. In particular, Theorem 16.2.2 is known as John's theorem, and was originally proved by Fritz John [Joh48].

There are numerous interesting results in convexity theory about how convex shapes looks like. One of the surprising results is Dvoretsky's theorem, which states that any symmetric convex body K around the origin (in "high enough" dimension) can be cut by a k-dimensional subspace, such that the intersection of K with this k-dimensional space, contains an ellipsoid $\mathcal{E}$ and is contained inside an ellipsoid $(1+\varepsilon)\mathcal{E}$ (k here is an arbitrary parameter). Since one can define a metric in a Banach space by providing a symmetric convex body which defines the unit ball, this implies that any high enough dimensional Banach space contains (up to translation that maps the ellipsoid to a ball) a subspace which is almost Euclidean.

A survey of those results is provided by Ball [Bal97].

Chapter 17

Approximation via Reweighting

> In addition, the sirloin which I threw overboard, instead of drifting off into the void, didn't seem to want to leave the rocket and revolved about it, a second artificial satellite, which produced a brief eclipse of the sun every eleven minutes and four seconds. To calm my nerves I calculated till evening the components of its trajectory, as well as the orbital perturbation caused by the presence of the lost wrench. I figured out that for the next six million years the sirloin, rotating about the ship in circular path, would lead the wrench, then catch up with it from behind and pass it again.
>
> – The Star Diaries, Stanislaw Lem.

In this chapter, we will introduce a powerful technique for "structure" approximation. The basic idea is to perform a search by assigning elements weights, and picking the elements according to their weights. Elements weight indicates their importance. By repeatedly picking elements according to their weights, and updating the weights of objected that are being neglected (i.e., they are more important than the current weights indicate), we end up with a structure that has some desired properties.

We will demonstrate this technique for two problems. In the first problem, we will compute a spanning tree of points that has low stabbing number. In the second problem, we will show how the Set Cover problem can be approximated efficiently in geometric settings, yielding a better bound than the general approximation algorithm for this problem.

17.1 Computing a spanning tree with low stabbing number

Let P be a set of n points in the plane. We would like to compute a tree $\mathcal{T}$ that span the points of P such that every line in the plane crosses the edges of the tree at most $O(\sqrt{n})$ times. If the points are the $\sqrt{n} \times \sqrt{n}$ grid, this easily holds, if we pick any spanning tree that connects only points that are adjacent on the grid. It is not hard to show that one can not do better (see Exercise 17.4.2).

17.1.1 The algorithm

So, consider the set of all separating lines $\widehat{\mathsf{L}}$ of P. Formally, we will consider two lines ℓ and ℓ' be equivalent, if the closed halfplane above ℓ contains the same set of points as the closed halfplane above ℓ' (we assume no two points in P have the same x coordinate, and as such we can ignore vertical lines). Let $\backsim$ denote this equivalent relation. For each equivalent class of $\widehat{\mathsf{L}}$ pick one representative line into a set L. Alternatively, L is just all the set of lines that passes through two points of P (it is easy to see that under this definition the set L might contain some lines that are equivalent under $\backsim$; this is a minor technicality that we shell ignore since L contains representative for each one of the equivalence classes). As such, the set L contains at most $\binom{n}{2}$ lines.

Let $\mathcal{E}_i$ be the set of edges added by the algorithm by the end of the ith iteration. And consider a candidate edge (i.e., segment) $s = \mathsf{qr}$, and consider a line $\ell \in \mathsf{L}$. If ℓ already intersects k edges of $\mathcal{E}_i$ and k is large (i.e., $\Omega(\sqrt{n})$) then we would like to discourage the usage of s in the tree $\mathcal{T}$. The problem is that the desirability of an edge is determined by the lines that intersect them, as every line knows how much load it is carrying, and how close it to the limit. Intuitively, as the load of a line get higher, edges that crosses this line becomes less desirable.

To this end, let us define weights over the lines. The weight ω_ℓ of a line ℓ in the beginning would be initialized to 1. The weight of the line at the beginning of the ith iteration would be $\omega_{\ell,i-1} = 2^{n_{i-1}(\ell)}$, where $n_{i-1}(\ell) = \left|\left\{ s \in \mathcal{E}_{i-1} \;\middle|\; \mathrm{sec} \cap \ell \neq \emptyset \right\}\right|$ is the number of segments of $\mathcal{E}_{i-1}$ that intersects ℓ. Thus, the weight of a segment s, in the beginning of the ith iteration, is

$$\omega_{s,i} = \sum_{\ell \in \mathsf{L}, \ell \cap s \neq \emptyset} \omega_{\ell,i-1}.$$

Clearly, the heavier a segment s is the less desirable it is to be used for the spanning tree. As such, we would always pick an edge qr such that $\mathsf{q}, \mathsf{r} \in \mathsf{P}$ belong to two different connected components of the forest induced by $\mathcal{E}_{i-1}$, and its weight is minimal among all such edges. We repeat this process till we end up with a spanning tree of P. To simplify the implementation of the algorithm, when adding s to the set of edges in the forest, we also remove one its endpoints from P (i.e., every connect component of our forest has a single representative point). Thus, the algorithm terminates when P has a single point in it.

We claim that the resulting spanning tree has the required properties. Clearly, the algorithm can be implemented in polynomial time since it performs $n - 1$ iterations and as such, the largest weight used is $\leq 2^n$. But such numbers can be manipulated in polynomial time, and as such the running time is polynomial.

17.1.2 Proof of correctness

A key concept in our discussion (which was implicit above) is the ***crossing distance***. The crossing distance $\mathbf{d}_{\bowtie}(\mathsf{p}, \mathsf{q})$ between two points p and q is the minimum number of lines one has to cross (i.e., cut) to get from p to q. It is easy to verify that this is a metric (i.e., the triangle inequality holds for it).

Lemma 17.1.1 *Let P be a set of n points in the plane, and let L be a set of lines in the plane with total weight W. One can always find a pair of points q and r in P, such that the total weight of the segment $s = \mathsf{qr}$ (i.e., the total weight of the lines of L intersecting s) is at most $cW/\sqrt{n}$, for some constant c.*

Proof: First, since the weights considered by us are always integers, we can consider all the weights to be 1 by replacing a line ℓ of weight ω_ℓ by ω_ℓ copies of it. Perturb slightly the lines, so that there is no pair of them which is parallel. Next, consider a point $\mathsf{q} \in \mathbb{R}^2$, and all the vertices of the arrangement of $\mathcal{A} = \mathcal{A}(\mathsf{L})$. Consider the ball $\mathbf{b}(\mathsf{q}, r)$ of all vertices of the arrangement $\mathcal{A}$ in crossing distance at most $\leq r$ from q.

We claim that $|\mathbf{b}(\mathsf{q}, r)| \geq r^2/8$. Indeed, one can shoot a ray ζ from q that intersects at least $W/2$ lines of L. Let $\ell_1, \ldots, \ell_{r/2}$ be the first $r/2$ lines hit by the ray ζ, and let $\mathsf{r}_1, \ldots, \mathsf{r}_{r/2}$ be the respective intersection points between these lines and ζ. Now, mark all the intersection points of the arrangement $\mathcal{A}(\mathsf{L})$ along the line ℓ_i that are in distance at most $r/2$ from r_i, for $i = 1, \ldots, r/2$. Clearly, we overall marked at least $(r/2)(r/2)/2$ vertices of the arrangement, since we marked (at least) $r/2$ vertices along each of the lines $\ell_1, \ldots, \ell_{r/2}$. Furthermore, each vertex can be counted in this way at most twice. Now, observe that all these vertices are in distance at most $(r/2) + (r/2)$ from q because of the triangle inequality.

So, consider the set of vertices $X(r) = \cup_{\mathsf{p} \in \mathsf{P}} \mathbf{b}(\mathsf{p}, r)$. Clearly, as long the balls of $X(r)$ are disjoint, the number of vertices of the arrangement $\mathcal{A}$ included in $X(r)$ is at least $nr^2/8$. In particular, the overall number of vertices in the arrangement is $\binom{W}{2}$, and as such it must be two balls of $X(r)$ are not disjoint when $nr^2/8 > \binom{W}{2} = W(W-1)/2$. Namely, when $r^2 > 4W^2/n$; namely, when $r > 2W/\sqrt{n}$. Now, when $r = \lceil 2W/\sqrt{n} \rceil + 1$ there must be a vertex v in the arrangement $\mathcal{A}$ and two points $\mathsf{q}, \mathsf{r} \in \mathsf{P}$, such that $\mathbf{d}_{\bowtie}(\mathsf{q}, v), \mathbf{d}_{\bowtie}(v, \mathsf{r}) \leq r$, and by the triangle inequality, we have that

$$\mathbf{d}_{\bowtie}(\mathsf{q}, \mathsf{r}) \leq \mathbf{d}_{\bowtie}(\mathsf{q}, v) + \mathbf{d}_{\bowtie}(v, \mathsf{r}) \leq O(W/\sqrt{n}).$$

Namely, q and r are within the required crossing distance. ■

Let W_i denote the total weight of the lines in L in the end of the ith iteration. We have that $W_0 \leq \binom{n}{2}$, and since there are $n_i = n - i + 1$ points in P in the beginning of the ith iteration, it follows that the algorithm found a segment s_i of weight at most $cW_{i-1}/\sqrt{n_i}$. We double the weight of all the lines that intersect s_i. Thus, it holds

$$\begin{aligned} W_i &\leq W_{i-1} + cW_{i-1}/\sqrt{n_i} \leq (1 + c/\sqrt{n_i})W_{i-1} \leq \prod_{k=1}^{i} (1 + c/\sqrt{n_k}) W_0 \\ &\leq W_0 \prod_{k=1}^{i} \exp\left(c/\sqrt{n_k}\right) = \exp\left(\sum_{k=1}^{i} \frac{c}{\sqrt{n-k+1}}\right), \end{aligned}$$

since $1 + x \leq e^x$, for all $x \geq 0$. In particular, we have that

$$W_n \leq W_0 \exp\left(\sum_{k=1}^{n} \frac{c}{\sqrt{n-k+1}}\right) = \binom{n}{2} \exp\left(\sum_{k=1}^{n} \frac{c}{\sqrt{k}}\right) \leq n^2 \exp\left(4c\sqrt{n}\right),$$

since $\sum_{k=1}^{n} 1/\sqrt{k} \leq 1 + \int_{x=1}^{n} (1/\sqrt{x})dx = 1 + \left[2\sqrt{x}\right]_{x=1}^{x=n+1} \leq 4\sqrt{n}$. On the other direction, consider the heaviest line l in L in the end of the execution of the algorithm. If it crosses Δ lines then its weight is 2^Δ, and as such

$$2^\Delta = \omega_l \leq W_n \leq n^2 \exp\left(4c\sqrt{n}\right).$$

It follows that $\Delta = O(\log n + \sqrt{n})$, as required. Namely, any line in the plane crosses at most $O(\sqrt{n})$ edges of $\mathcal{T}$.

Theorem 17.1.2 *Given a set* P *of n points in the plane, one can compute a spanning tree* $\mathcal{T}$ *of* P *such that each line crosses at most* $O(\sqrt{n})$ *edges of* $\mathcal{T}$*. The running time is polynomial in n.*

This result also hold in higher dimensions. The proof is left as an exercise (see Exercise 17.4.1).

Theorem 17.1.3 *Given a set* P *of n points in* $\mathbb{R}^d$*, one can compute a spanning tree* $\mathcal{T}$ *of* P *such that each line crosses at most* $O(n^{1-1/d})$ *edges of* $\mathcal{T}$*. The running time is polynomial in n.*

17.1.3 An application - better discrepancy

Before extending this result to more abstract settings, let us quickly outline why this spanning tree of low stabbing number leads to better discrepancy and smallest ε-sample for halfplanes.

Indeed, let us turn $\mathcal{T}$ into a cycle by drawing a contour walking around the edges of $\mathcal{T}$; formally, we double every edge of $\mathcal{T}$, and observe that the resulting graph is Eulerian, and we extract the Eulerian tour from this graph. Clearly, this cycle C has at most twice the stabbing number than the tree.

Next, consider a curve γ in the plane and a segment s with the same endpoints, and observe that a line that intersects s must also intersect γ (but not vice versa!). As such, if we shortcut parts of C by replacing them by straight segments, we are only decreasing the stabbing number of C. To this end, start traversing C. So, we start from some arbitrary point $p_0 \in P$, and start "walking" along C. Whenever we are at point $pin\mathsf{P}$, along the cycle C, we go directly from there (i.e., shortcut) to the next point visited by C that was not visited yet. Let C' be the resulting cycle. Clearly, it visits all the points of P and it has a crossing number which is at most twice the crossing number of $\mathcal{T}$. Now, assuming that $n = |\mathsf{P}|$ is even, pick all the even edges of C' into a prefect matching $\mathcal{M}$ of P. We have:

Lemma 17.1.4 *One can compute a perfect matching* $\mathcal{M}$ *of a set of n points in the plane, such that every line crosses at most* $O(\sqrt{n})$ *edges of the matching.*

(A somewhat similar argument is being used in the 2-approximation algorithm for TSP with the triangle inequality.)

Now, going back to the discrepancy question, we remind the reader that we would like to color the points by $\{-1, 1\}$ such that for any halfplane the 'balance' of the coloring is as close to perfect as possible. To this end, we use the matching of the above lemma, and plug it into Theorem 5.4.1. Since any line ℓ crosses at most $\#_\ell = O(\sqrt{n})$ edges of $\mathcal{M}$, we get the following result.

Theorem 17.1.5 *Let* P *be a set of n points in the plane. One can compute a coloring* χ *of* P *by* $\{-1, 1\}$*, such that for all halfplane h, it holds*

$$|\chi(h)| = O\Big(n^{1/4}\sqrt{\ln n}\Big).$$

Namely, the discrepancy of n points in relation to half planes is $O(n^{1/4}\sqrt{\ln n})$. This also implies that one can construct a better ε-sample in this case. But before dwelling into this, let us prove a more general version of the spanning tree lemma.

17.1.4 Spanning tree for space with bounded shattering dimension

Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space with dual shattering dimension d. We would also assume that its shattering dimension is bounded, say, by d'. Let $\mathsf{P} \subseteq \mathsf{X}$ be a set of n points. Consider a spanning tree $\mathcal{T}$ of P. The tree $\mathcal{T}$ is defined by $n-1$ edges $e_i = \{p_i, q_i\} \subseteq \mathsf{P}$. An edge $\{p, q\}$ ***crosses*** a range $\mathbf{r} \in \mathcal{R}$ if $|\{p, q\} \cap \mathbf{r}| = 1$. Our purpose is to build a spanning tree such that every range of $\mathcal{R}$ crosses a small number of edges of $\mathcal{T}$. The reader can verify that this abstract settings corresponds to the more concrete problem we just solved.

As before, we can concentrate on the ranges of $\mathcal{R}_{|\mathsf{P}}$, which has size $m \leq n^d$. So consider $\mathcal{F}$ be a weighted subset of these m ranges. Our algorithm would work as before: We will find the pair $p, q \subseteq \mathsf{P}$ such that the total weight of the ranges is crosses is minimized. Next, we will double the weight of these ranges, add the edge p, q to the spanning tree, and delete, say, q from P. We repeat this process this we remain with a single point in P. Clearly, we had computed a spanning tree. What remains to be bounded is the crossing number of this tree. As before, the analysis boils down to proving the existence of two "close" points.

Lemma 17.1.6 *Let* P *be a set of n points, and let* $\mathcal{F}$ *be a weighted set of ranges from* $\mathcal{R}_{|\mathsf{P}}$*, with total weight W. Then, there is a pair of points* $e = p, q \subseteq \mathsf{P}$*, such that the total weight of the ranges crossed by e is at most* $O(Wn^{-1/d}\log(n))$*.*

Proof: Let ε be a parameter to be specified shortly. For an edge $u, v \subseteq \mathsf{P}$, consider the set of edges that its crosses

$$C(u, v) = \left\{ \mathbf{r} \;\middle|\; (u \in \mathbf{r} \text{ and } v \notin \mathbf{r}) \quad \text{or} \quad (u \notin \mathbf{r} \text{ and } v \in \mathbf{r}), \quad \text{for} \quad \mathbf{r} \in \mathcal{F} \right\}.$$

So, consider the dual range space $\mathsf{S}^\star = (\mathcal{R}, \mathsf{X}^\star)$[9] This range space has shattering dimension bounded by d, by assumption. Consider the new range space

$$\mathsf{T} = \left(\mathcal{R}, \left\{ \mathbf{r} \oplus \mathbf{r}' \;\middle|\; \mathbf{r}, \mathbf{r}' \in \mathsf{X}^\star \right\}\right)$$

, where $\oplus$ is the symmetric difference of the two sets; namely, $\mathbf{r} \oplus \mathbf{r}' = (\mathbf{r} \setminus \mathbf{r}') \cup (\mathbf{r}' \setminus \mathbf{r})$. By arguing as in Corollary 5.2.7, we have that T has a shattering dimension at most $2d$. Furthermore, the projected range space $\mathsf{T}_{|\mathcal{F}}$ has $C(u, v)$ as a range.

So consider a random sample R of size $O((d/\varepsilon)\log(d/\varepsilon))$ from $\mathcal{F}$ (note that $\mathcal{F}$ is weighted, and the random sampling is done accordingly). By the ε-net theorem (Theorem 5.3.4), we know that with constant probability, this is an ε-net. Namely, a set $C(u, v)$ which do not contain any range of R has weight at most εW.

On the other hand, the range space $\mathsf{S}^\star_{|\mathsf{R}} = (\mathsf{R}, \mathsf{X}^\star_{|\mathsf{R}})$ has at most

$$\mu = O\left(|\mathsf{R}|^d\right) = \left(c\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)^d$$

ranges[2], since the dual range space has shattering dimension d, where c is an appropriate constant. In particular, let us pick ε as large as possible, such that $\mu < n = |\mathsf{P}|$. We are then guaranteed that there are two points $p, q \in \mathsf{P}$ such that $\mathsf{R}_p = \mathsf{R}_q$. In particular, the total weight of $C(u, v)$ is εW. Thus, we left with the task of picking ε.

So, for $\varepsilon = c_1 n^{-1/d}\log(n)$ it holds that $\mu < n$, if c_1 is sufficiently large, as can be verified. ■

To complete the argument, we need to argue about the total weight of the ranges in the end of this process. It is bounded by

$$U = n^{d'}\prod_{i=1}^{n}\left(1 + c_1\frac{\log(i)}{i^{1/d}}\right) \le n^{d'}\exp\left(\sum_i c_1\frac{\log(i)}{i^{1/d}}\right) \le n^{d'}\exp\left(O\left(c_1 n^{1-1/d}\log(n)\right)\right)$$

Now, the crossing number of the resulting tree $\mathcal{T}$, for any range $\mathbf{r} \in \mathcal{R}$ is bounded by $\lg(U) = O(d'\log n + n^{1-1/d}\log(n))$. We thus conclude:

Theorem 17.1.7 *Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with shattering dimension d' and dual shattering dimension d, and a set $\mathsf{P} \subseteq \mathsf{X}$ of n points, one can compute, in polynomial time, a spanning tree $\mathcal{T}$ of P, such that any range of $\mathcal{R}$ crosses at most $O(d'\log n + n^{1-1/d}\log(n))$ edges of $\mathcal{T}$.*

17.2 Geometric Set Cover

Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space with bounded VC dimension. For example, let X be a set of n points in the plane, and let $\mathcal{R}$ be a set of m allowable disks. The question is to find the minimal number of disks of $\mathcal{R}$ one needs to cover the points of X.

This is an instance of Set Cover, and it is in general NP-Hard, and even NP-Hard to approximate within a factor of $\Omega(\log n)$. There is an easy greedy algorithm, which repeatedly pick the set (i.e., disk) that covers the largest number of points not covered yet. It is easy to show that his algorithm would cover the points using $O(k\log n)$ sets, where k is the number of sets used by the optimal solution.

The algorithm. Interestingly, one can do much better if the set system S has bounded dual shattering dimension d. Indeed, let us assign weight 1 to each range of $\mathcal{R}$, and pick a random subset R of size $O((d/\varepsilon)\log(d/\varepsilon))$ from $\mathcal{R}$, where $\varepsilon = 1/(4k)$ (the sample is done according to the weights). If the sets of R covers all the points of $\mathcal{R}$, then we are done. Otherwise, consider a point $\mathsf{p} \in \mathsf{X}$ which is not covered by R. If the total weight of $\mathcal{R}_\mathsf{p}$ (i.e., the set of ranges covering p) is smaller than $\varepsilon W(\mathcal{R}) = \sum_{\mathbf{r}\in\mathcal{R}}\omega_\mathbf{r}$ then we redouble the weight of all the sets of $\mathcal{R}_\mathsf{p}$. In any case, even if doubling is not carried out, we repeat this process till it succeeds.

Details and Intuition. In the above algorithm, if a random sample fails (i.e., there is an uncovered point) then one of the ranges that covers p must be in the optimal solution. In particular, by increasing the weight of the ranges covering p, we improve the probability that p would be covered in the next iteration. Furthermore, with good probability, the sample is an ε-net, and as such the algorithm doubles the weight of a "few" ranges. One of these few ranges must be in the optimal solution. As such, the weight of the optimal solution grows exponentially in a faster rate than the total weight of the universe, implying that at some point the algorithm must terminates, as the weight of the optimal solution exceeds the total weight of all the ranges, which is of course impossible.

[9] We remind the reader that $\mathsf{X}^\star = \left\{\mathcal{R}_\mathsf{p} \;\middle|\; \mathsf{p} \in \mathsf{X}\right\}$ and $\mathcal{R}_\mathsf{p} = \left\{\mathbf{r} \;\middle|\; \mathbf{r} \in \mathcal{R},\ \text{the range } \mathbf{r} \text{ contains } \mathsf{p}\right\}$.

[2] If the ranges of $\mathcal{R}$ are geometric shapes, it means that the arrangement formed by the shapes of R has at most μ faces.

17.2.1 Proof of correctness

In the following, we bound the number of iterations performed by the algorithm. As before, let $W_0 = m$ be the initial weight of the ranges, and W_i would be the weight in the end of the ith iteration. We consider an iteration to be ***successful*** if the doubling stage is being performed in the iteration. Since an iteration is successful if the sample is an ε-net, and by Theorem 5.3.4 the probability for that is at least, say, $1/2$, it follows that we need to bound only the number of successful iterations (indeed, it would be easy to verify that with high probability, using the Chernoff inequality, the number of successful iteration is at least a quarter of all iterations performed).

As before, we know that $W_i \leq (1+\varepsilon)W_{i-1} = (1+\varepsilon)^i m \leq m\exp(\varepsilon i)$. On the other hand, in each iteration the algorithm "hits" at least one of the ranges in the optimal solution. Let $t_i(j)$ be the number of times the weight of the jth range in the optimal solution was doubled, for $j = 1, \ldots, k$, where k is the size of the optimal solution. Clearly, the weight of the universe in the ith iteration is at least

$$\sum_{j=1}^{k} 2^{t_i(j)}.$$

But this quantity is minimized when $t_i(1), \ldots, et_i(k)$ are as equal to each other as possible. (Indeed, $2^a + 2^b \geq 2 \cdot 2^{\lfloor (a+b)/2 \rfloor}$, for any integers $a, b \geq 0$.) As such, we have that

$$k2^{\lfloor i/k \rfloor} \leq \sum_{j=1}^{k} 2^{t_i(j)} \leq W_i \leq m\exp(\varepsilon i)\,.$$

So, consider $i = tk$, for t an integer. We have that $k2^t \leq m\exp(\varepsilon tk) = m\exp(t/4)$, since $\varepsilon = 1/4k$. Namely,

$$\lg k + t \leq \lg m + \frac{t}{4}\lg e \leq \lg m + \frac{t}{2},$$

as $\lg e \leq 1.45$. Namely, $t \leq 2\lg(m/k)$. We conclude that the algorithm performs at most $2\lg(m/k)$ successful iterations, and as such it performs at most $O(\log(m/k))$ iterations overall.

Running time. It is easy to verify that with careful implementation the sampling stage can be carried out in linear time. The size of the resulting approximation is $O((d/\varepsilon)\log(d/\varepsilon)) = O(dk\log dk)$. Checking if all the points are covered by the random sample takes $O(ndk\log dk)$ time, assuming we can in constant time determine if a point is inside a range. Computing the total weight of the ranges covering p takes $O(m)$ time. Thus, each iteration takes $O(m + ndk\log dk)$ time.

Note, that we assumed that k is known to us in advance. This can be easily overcome by doing an exponential search for the right value of k. Given a guess κ to the value of k, we will run the algorithm with κ. If the algorithm exceeds $c\log(m/k)$ iterations without terminating, we know the guess is too small and we continue to try 2κ. We thus get the following.

Theorem 17.2.1 *Given a finite range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with n points and m ranges, and S has a dual shattering dimension d, then one can compute a cover of X, using the ranges of $\mathcal{R}$, such that the cover uses $O(dk\log(dk))$ sets, where k is the size of the optimal set cover.*

The running time of the algorithm is $O((m + ndk\log(dk))\log(m/k)\log n)$ time, with high probability, assuming that in constant time we can decide if a point is inside a range.

17.2.2 Application - Guarding an art gallery

Let $\mathcal{P}$ be a simple polygon (i.e., without holes) with n sides in the plane. We would like to find a minimal number of guards that see the whole polygon. A guard is a point that can see in all directions, but, unlike superman, it can not see through the walls of the polygon. There is a beautiful standard result in computational geometry that shows that such a polygon can always be guarded by $\lfloor n/3 \rfloor$ guards placed on the vertices of the polygons. But of course, this might be arbitrarily bad compared to the optimal solution using k guards.

So, consider the range space $\mathsf{S} = (\mathcal{P}, \mathcal{R})$, where $\mathcal{R}$ are all the possible visibility polygons inside $\mathcal{P}$. We remind the reader that for a point $p \in \mathcal{P}$, the ***visibility polygon*** of p in $\mathcal{P}$ is the polygon

$$\mathcal{V}_{\mathcal{P}}(\mathsf{p}) = \left\{ \mathsf{q} \;\middle|\; \mathsf{q} \in \mathcal{P}, \mathsf{pq} \subseteq \mathcal{P} \right\},$$

where we consider $\mathcal{P}$ to be a closed set.[3]

The following theorem is a bit surprising, see Section 17.2.2.1 for the proof.

Theorem 17.2.2 *The* VC *dimension of the range space formed by all visibility polygons inside a polygon $\mathcal{P}$ (i.e.,* S *above) is a constant.*

[3] As such, under our definition of visibility, one can see through a reflex corner of a polygon.

So, consider a simple polygon $\mathcal{P}$ with n vertices, and let $\widehat{\mathcal{P}}$ be a (finite) set of visibility polygons that covers $\mathcal{P}$ (say, all the visibility polygons induced by vertices of $\mathcal{P}$). The range space induced by $\mathcal{P}$ and $\widehat{\mathcal{P}}$ has finite VC dimension (since this range space is contained inside the range space of Theorem 17.2.2). To make things more concrete, consider placing a point inside each face of the arrangement of the visibility polygons of $\widehat{\mathcal{P}}$ inside $\mathcal{P}$, and let U denote the resulting set of points. Next, consider the projection of this range space into U; namely, consider the range space $\mathsf{S} = \left(U, \left\{ \mathcal{Q} \cap U \;\middle|\; \mathcal{Q} \in \widehat{\mathcal{P}} \right\}\right)$. Clearly, a set cover of minimal size for S corresponds to a minimal number of visibility polygons of $\widehat{\mathcal{P}}$ that covers $\mathcal{P}$. Now, S has size polynomial in $\mathcal{P}$ and $\widehat{\mathcal{P}}$ and it can be clearly be computed efficiently. We can now apply the algorithm of Theorem 17.2.1 to it, and get a "small" set cover. We conclude:

Theorem 17.2.3 *Given a simple polygon $\mathcal{P}$ of size n, and a set of visibility polygons $\widehat{\mathcal{P}}$ (of polynomial size) that covers $\mathcal{P}$, then one can compute, in polynomial time, a cover of $\mathcal{P}$ using $O(k_{\mathrm{opt}} \log k_{\mathrm{opt}})$ polygons of $\widehat{\mathcal{P}}$, where k_{opt} is the smallest number of polygons of $\widehat{\mathcal{P}}$ that completely covers $\mathcal{P}$.*

17.2.2.1 A proof of Theorem 17.2.2

Proof: Let U be a set of k points shattered by the set of visibility polygons inside $\mathcal{P}$. To this end, consider the set $\widehat{\mathcal{P}}$ of k visibility polygons induced by the points of U. Clearly, by the shattering property, for every subset S of U, there exists a point $\mathsf{p} \in \mathcal{P}$ such that the visibility polygon of p contains exactly S and no other points of U. Conversely, in the arrangement $\mathcal{A}(\widehat{\mathcal{P}})$, the point p is contained in a face that is covered by the visibility polygons of S, and is outside all the visibility polygons of $U \setminus S$.

Unfortunately, we can not just bound the complexity of $\mathcal{A}(\widehat{\mathcal{P}})$, and use it to bound the VC dimension of the given range space, since its complexity involves n (it is $O(nk^2)$ in the worst case). Instead, let $T(k, \mathcal{P})$ be the maximum number of different subsets of k fixed points one can see from inside a polygon $\mathcal{Q} \subseteq \mathcal{P}$. Similarly, let $T_{\mathrm{out}}(k, \mathcal{Q})$ be the maximum number of different subsets of k fixed points one can see inside a polygon $\mathcal{Q} \subseteq \mathcal{P}$, where all these points must lie outside $\mathcal{Q}$, where there is a diagonal s such that all the points are one side of s and $\mathcal{Q}$ is on the other side.

Lemma 17.2.4 *It holds $T_{\mathrm{out}}(k, \mathcal{Q}) = O(k^6)$.*

Proof: Let S be the set of k points under consideration, which are on one side of s and $\mathcal{Q}$ is on the other side. Each of these points sees a subinterval of s (inside $\mathcal{P}$), and this interval induces a wedge in the plane on the other side of s (we ignore for the time being the boundary of $\mathcal{P}$). Also, consider the lines passing through pair of points of S. Together, this set of $O(k^2)$ lines, rays and segments, partition the place into $O(k^4)$ different faces, so that for a point inside a face of this arrangement, it sees the same subset of points of S through the segment s in the same radial order. So, consider a point p inside such a face f, and let $\mathsf{q}_1, \ldots, \mathsf{q}_m$ be the (clockwise ordered) list of points of S that p sees. To visualize this list, connect p to each one of this points by a segment. Now, as we now introduce the boundaries of $\mathcal{P}$, some of these segments are no longer realizable since the intersect the boundary of polygon. Observe, however, the set of visible points from p must be a consecutive subsequence of the above list; namely, p can see inside $\mathcal{P}$ the points $\mathsf{q}_L, \ldots, \mathsf{q}_R$, for some $L \leq R$. We conclude that since there are $O(k^2)$ choices for the indices L and R, it follows that inside f there are most $O(k^2)$ different subsets of S that are realizable inside $\mathcal{Q}$. Now, there are $O(k^4)$ such faces in the arrangement, which implies the bound. ■

Now, consider a triangulation of $\mathcal{P}$. There must be a triangle in this triangulation, such that if we remove it, then every one of the remaining pieces $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3$ contains at most $2k/3$ points of U. Let $\triangle$ be this triangle.

Clearly, the complexity of the visibility polygons of $\widehat{\mathcal{P}}$ inside $\triangle$ is $O(k^2)$. Furthermore, inside $\mathcal{Q}_i$ one can see only $T_{\mathrm{out}}(k, \mathcal{Q}_i)$ different subsets of the points of U outside $\mathcal{Q}_i$, for $i = 1, 2, 3$. Thus, the total number of different subsets of U one can see is bounded by

$$T(k, \mathcal{P}) \leq O(k^2) + \sum_{i=1}^{3} T_{\mathrm{out}}(k, \mathcal{Q}_i) \cdot T(|U \cap \mathcal{Q}_i|, \mathcal{Q}_i) = O(k^2) + O(k^6) T(2k/3) = k^{O(\log k)},$$

by Lemma 17.2.4. However, for U to be shattered, we need that $T(k, \mathcal{P}) \geq 2^k$. Namely, we have that

$$2^k \leq k^{O(\log k)}.$$

Namely, $k \leq O(\log^2 k)$, which implies that k is a constant, as desired. ■

17.3 Bibliographical Notes

The stabbing tree result is due to Welzl [Wel92], which is in turn a simplification of the work of Chazelle and Welzl [CW89]. The running time of computing the good spanning tree can be improved to $O(n^{1+\varepsilon})$ [Wel92].

A natural question is whether one can compute a spanning tree which is weight sensitive. Namely, compute a spanning tree of n points in the plane, such that if a line has only k points on one side of it, it would cross at most $O(\sqrt{k})$ edges of the tree. A slightly weaker bound is currently known, see Exercise 17.4.3, which is from [HS06]. This leads to small ε-samples which work for ranges which are heavy enough.

Another natural question is whether given a set of lines and points in the plane, can one compute a spanning tree with overall small number of crossings. Surprisingly, one can do $(1+\varepsilon)$-approximation for the best tree in near linear time, see [HI00]. This result also extends to higher dimensions.

The algorithm we described for Set Cover falls into a general method of multiplicative weights update. Algorithms of this family include Littleston's Winnow algorithm [Lit88], Adaboost [FS97], and many more. In fact, the basic idea can be tracked back to the fifties. See [AHK06] for a nice survey of this method.

17.4 Exercises

Exercise 17.4.1 Prove Theorem 17.1.3.

Exercise 17.4.2 Show that in the worst case, one can pick a point set P of n points in the plane, such that any spanning tree $\mathcal{T}$ of P, there exists a line ℓ, such that ℓ crosses $\Omega(\sqrt{n})$ edges of $\mathcal{T}$.

Exercise 17.4.3 [20 Points] Let P be a set of n points in the plane. For a line ℓ, let w_ℓ^+ (resp., w_ℓ^-) be the number of points of P lying above (resp., below or on) ℓ, and define the *weight* of ℓ, denoted by $\omega\ell$, to be $\min(w_\ell^+, w_\ell^-)$.

Show, that one can construct a spanning tree $\mathcal{T}$ for P such that any line ℓ crosses at most $O(\sqrt{\omega\ell}\log(n/\omega\ell))$ edges of $\mathcal{T}$.

Chapter 18

Approximating the Minimum Volume Bounding Box of a Point Set

> Isn't it an artificial, sterilized, didactically pruned world, a mere sham world in which you cravenly vegetate, a world without vices, without passions without hunger, without sap and salt, a world without family, without mothers, without children, almost without women? The instinctual life is tamed by meditation. For generations you have left to others dangerous, daring, and responsible things like economics, law, and politics. Cowardly and well-protected, fed by others, and having few burdensome duties, you lead your drones' lives, and so that they won't be too boring you busy yourselves with all these erudite specialties, count syllables and letters, make music, and play the Glass Bead Game, while outside in the filth of the world poor harried people live real lives and do real work.
>
> – The Glass Bead Game, Hermann Hesse

18.1 Some Geometry

Let $\mathcal{H} = [0,1]^{\mathsf{d}}$ denote the unit hypercube in $\mathbb{R}^{\mathsf{d}}$. The *width* of a convex body P in $\mathbb{R}^{\mathsf{d}}$ is the minimum distance between two parallel planes that encloses P. Alternatively, the width is the minimum length of the projection of P into a line in $\mathbb{R}^{\mathsf{d}}$. The *diameter* of P is the maximum distance between two points of P. Alternatively, this is the maximum length projection of P into a line.

Lemma 18.1.1 *The width of $\mathcal{H}$ is 1 and its diameter is $\sqrt{d}$.*

Proof: The upper bound on the width is obvious. As for the lower bound, observe that P encloses a ball of radius $1/2$, and as such its projection in any direction is at least 1.

The diameter is just the distance between the two points $(0,\ldots,0)$ and $(1,\ldots,1)$. ■

Lemma 18.1.2 *Let P be a convex body, and let h be a hyperplane cutting P. Let $\mu = \mathrm{Vol}(h \cap P)$, and let $\vec{v}$ be unit vector orthogonal to h. Let ρ be the length of the projection of P into the direction of $\vec{v}$. Then $\mathrm{Vol}(P) \geq \mu \cdot v/\mathsf{d}$.*

Proof: Let $P^+ = P \cap h^+$ and a be the point of maximum distance of P^+ from h, where h^+ denotes the positive half space induced by h.

Let $C = h \cap P$. By the convexity of P, the pyramid $R = \mathcal{CH}(C \cup \{a\})$ is contained inside P, where $\mathcal{CH}(S)$ denotes the convex-hull of S. We explicitly compute the volume of R for the sake of completeness. To this end, let s be the segment connecting a to its projection on h, and let α denote the length of s. Parameterizing s by the interval $[0, \rho^-]$, let $C(t)$ denote the intersection of the hyperplane passing through $s(t)$ and R, where $s(0) = a$, and $s(t) \in h$. Clearly, $\mathrm{Vol}(C(t)) = (t/\alpha)^{d-1}\,\mathrm{Vol}(C) = (t/\alpha)^{d-1}\mu$. (We abuse notations a bit and refer to the $(d-1)$-dimensional volume and d-dimensional volume by the same notation.) Thus,

$$\mathrm{Vol}(R) = \int_{t=0}^{\alpha} \mathrm{Vol}(C(t))\,dt = \int_{t=0}^{\alpha} (t/\alpha)^{d-1}\,dt = \frac{\mu}{\alpha^{d-1}} \int_{t=0}^{\alpha} t^{d-1}\,dt = \frac{\mu}{\alpha^{d-1}} \cdot \frac{\alpha^d}{d} = \frac{\alpha\mu}{d}.$$

Thus, $\mathrm{Vol}(P^+) \geq \alpha\mu/\mathsf{d}$. Similar argumentation can be applied to $P^- = P \cap h^-$. Thus, we have $\mathrm{Vol}(P) \geq \rho\mu/\mathsf{d}$. ■

Lemma 18.1.3 *Let h be any hyperplane. We have $\mathrm{Vol}(h \cap [0,1]^{\mathsf{d}}) \leq \mathsf{d}$.*

Proof: Since the width of $\mathcal{H} = [0,1]^{\mathsf{d}}$ is 1 and thus the projection of $\mathcal{H}$ on the direction perpendicular to h is of length ≥ 1. As such, by Lemma 18.1.2 if $\mathrm{Vol}(h \cap \mathcal{H}) > \mathsf{d}$ then $\mathrm{Vol}(\mathcal{H}) > 1$. A contradiction. ■

Lemma 18.1.4 *Let $P \subseteq [0,1]^{\mathsf{d}}$ be a convex body. Let $\mu = \mathrm{Vol}(P)$. Then $\omega(P) \geq \mu/\mathsf{d}$ and P contains a ball of radius $\mu/(2\mathsf{d}^2)$.*

Proof: By Lemma 18.1.3, any hyperplane cut P in a set of volume at most d. Thus, $\mu = \mathrm{Vol}(P) \leq \omega(P)\mathsf{d}$. Namely, $\omega(P) \geq \mu/\mathsf{d}$.

Next, let $\mathcal{E}$ be the largest volume ellipsoid that is contained inside P. By John's theorem, we have that $P \subseteq d\mathcal{E}$. Let α be the length of the shortest axis of $\mathcal{E}$. Clearly, $\omega(P) \leq 2\mathsf{d}\alpha$, since $\omega(d\mathcal{E}) = 2\mathsf{d}\alpha$. Thus $2\mathsf{d}\alpha \geq \mu/\mathsf{d}$. This implies that $\alpha \geq \mu/(2\mathsf{d}^2)$.

Thus, $\mathcal{E}$ is an ellipsoid with its shortest axis is of length $\alpha \geq \mu/(2\mathsf{d}^2)$. In particular, $\mathcal{E}$ contains a ball of radius α, which is in turn contained inside P. ■

In Lemma 18.1.4 we used the fact that $r(P) \geq \omega(P)/(2\mathsf{d})$ (which we proved using John's theorem), where $r(P)$ is the radius of the largest ball enclosed inside P. Not surprisingly, considerably better bounds are known. In particular, it is known that $\omega(P)/(2\sqrt{\mathsf{d}}) \leq r(p)$ for add dimension, and $\omega(P)/(2(\mathsf{d}+1)/\sqrt{\mathsf{d}+2}) \leq r(P)$. Thus, $r(p) \geq \omega(P)/(2\sqrt{\mathsf{d}+1})$ [GK92]. Plugging this fact into the proof of Lemma 18.1.4, will give us slightly better result.

18.2 Approximating the Diameter

Lemma 18.2.1 *Given a point set S in $\mathbb{R}^{\mathsf{d}}$, one can compute in $O(n\mathsf{d})$ time a pair of points $s, t \in S$, such that $|st| \leq \mathrm{diam}(S) \leq \min(2, \sqrt{\mathsf{d}})|st|$.*

Proof: Let B be the minimum axis-parallel box containing S, and let s and t be the points in S that define the longest edge of B, whose length is denoted by l. By the diameter definition, $\|st\| \leq \mathrm{diam}(S)$, and clearly, $\mathrm{diam}(S) \leq \sqrt{d}\, l \leq \sqrt{d}\, \|st\|$. The points s and t are easily found in $O(n\mathsf{d})$ time.

Alternatively, pick a point $s' \in S$, and compute its furthest point $t' \in S$. Next, let a, b be the two points realizing the diameter. We have $\mathrm{diam}(S) = \|ab\| \leq \|as'\| + \|s'b\| \leq 2\,\|s't;\|$. Thus, $\|s't'\|$ is 2-approximation to the diameter of P. ■

18.3 Approximating the minimum volume bounding box

18.3.1 Constant Factor Approximation

Lemma 18.3.1 *Given a set P of n points in $\mathbb{R}^{\mathsf{d}}$, one can compute in $O(\mathsf{d}^2 n)$ time a bounding box $B(\mathsf{P})$ with $\mathrm{Vol}(B_{\mathrm{opt}}(\mathsf{P})) \leq \mathrm{Vol}(B(\mathsf{P})) \leq 2^{\mathsf{d}}\mathsf{d}!\,\mathrm{Vol}(B_{\mathrm{opt}}(\mathsf{P}))$, where B_{opt} is the minimum volume bounding box of P.*

Furthermore, there exists a vector $v \in \mathbb{R}^{\mathsf{d}}$, such that $c \cdot B + v \subseteq \mathcal{CH}(\mathsf{P})$. constant $c = 1/\left(2^{\mathsf{d}}\mathsf{d}!\mathsf{d}^{5/2}\right)$.

Proof: By using the algorithm of Lemma 18.2.1 we compute in $O(n)$ time two points $s, t \in \mathsf{P}$ which form a 2-approximation of the diameter of P. For the simplicity of exposition, we assume that st is on the x_{d}-axis (i.e., the line $\ell \equiv \cup_x(0, \ldots, 0, x)$), and there is one point of S that lies on the hyperplane $h \equiv x_{\mathsf{d}} = 0$, an that $x_d \geq 0$ for all points of P.

Let Q be the orthogonal projection of P into h, and let I be the shortest interval on ℓ which contain the projection of P into ℓ_s.

By recursion, we can compute a bounding box B' of Q in h. Let the bounding box be $B = B' \times I$. Note, that in the bottom of the recursion, the point-set is one dimensional, and the minimum interval containing the points can be computed in linear time.

Clearly, $\mathsf{P} \subseteq B$, and thus we only need to bound the quality of approximation. We next show that $\mathrm{Vol}(B) \geq \mathrm{Vol}(P)/c_{\mathsf{d}}$, where $C = \mathcal{CH}(\mathsf{P})$, and $c_{\mathsf{d}} = 2^{\mathsf{d}} \cdot \mathsf{d}!$. We prove this by induction on the dimension. For $\mathsf{d} = 1$ the claim trivially holds. Otherwise, by induction, that $\mathrm{Vol}(B') \geq \mathrm{Vol}(C')/c_{\mathsf{d}-1}$, where $C' = \mathcal{CH}(\mathsf{Q})$.

For a point $p \in C'$, let ℓ_p be the line parallel to x_d-axis passing through p. Let $L(p)$ be the minimum value of x_d for the points of ℓ_p lying inside C, and similarly, let $U(p)$ be the maximum value of x_d for the points of ℓ_p lying inside C. That is $\ell_p \cap C = [L[p), U(p)]$. Clearly, since C is convex, the function $L(\cdot)$ is concave, and $U(\cdot)$ is convex. As such, $\gamma(p) = U(p) - L(p)$ is a convex function, being the difference between a convex and a concave function. In particular, $\gamma(\cdot)$ induces the following convex body

$$U = \bigcup_{x \in C'} \Big[(x, 0), (x, \gamma(x))\Big].$$

Clearly, $\mathrm{Vol}(U) = \mathrm{Vol}(C)$. Furthermore, $\gamma((0, \ldots, 0)) \geq \|st\|$ and U is shaped like a "pyramid" its base is on the hyperplane $x_{\mathsf{d}} = 0$ is the set C', and the segment $[(0, \ldots, 0), (0, \ldots, 0, \|st\|)]$ is contained inside it. Thus,

$$\mathrm{Vol}(C) = \mathrm{Vol}(U) \geq |st|\,\mathrm{Vol}(C')/\mathsf{d},$$

by Lemma 18.1.2. Let $r = |I|$ be the length of the projection of S into the line ℓ, we have that $r \leq 2\,\|st\|$. Thus,

$$\mathrm{Vol}(B) = \mathrm{Vol}(B')\,|I| \leq \mathrm{Vol}(B')\,\|st\|\,2 \leq \mathrm{Vol}(C') \cdot c_{\mathsf{d}-1}\,\|st\|\,2.$$

On the other hand,

$$\mathrm{Vol}(B_{\mathrm{opt}}(\mathsf{P})) \geq \mathrm{Vol}(C) \geq \frac{\mathrm{Vol}(C')\,\|st\|}{\mathsf{d}} \geq \frac{(\mathrm{Vol}(B')/c_{\mathsf{d}-1}) \cdot (2\,\|st\|)}{2\mathsf{d}} \geq \frac{\mathrm{Vol}(B')\,|I|}{2c_{\mathsf{d}-1}\mathsf{d}} = \frac{\mathrm{Vol}(B)}{2^{\mathsf{d}}\mathsf{d}!}.$$

Let T be an affine transformation that maps B to the unit hypercube $\mathcal{H} = [0,1]^{\mathsf{d}}$. Observe that $\mathrm{Vol}(T(C)) \geq 1/c_{\mathsf{d}}$. By Lemma 18.1.4, there is ball $\mathbf{b}$ of radius $r \geq 1/(c_{\mathsf{d}} \cdot 2\mathsf{d}^2)$ contained inside $T(C)$. The ball $\mathbf{b}$ contains a hypercube of sidelength $2r/\sqrt{\mathsf{d}} \geq 2/(2\mathsf{d}^2\sqrt{\mathsf{d}}c_{\mathsf{d}})$. Thus, for $c = 1/\left(2^{\mathsf{d}}\mathsf{d}!\mathsf{d}^{5/2}\right)$, there exists a vector $v' \in \mathbb{R}^{\mathsf{d}}$, such that $c\mathcal{H} + v' \subseteq T(C)$. Thus, applying T^{-1} to both sides, we have that there exists a vector $v \in \mathbb{R}^{\mathsf{d}}$, such that $c \cdot B + v = c \cdot T^{-1}(\mathcal{H}) + v \subseteq C$. ■

18.4 Exact Algorithms

18.4.1 An exact algorithm 2d

Let P be a set of points in the plane. We compute the convex hull $C = \mathcal{CH}(\mathsf{P})$. Next, we rotate to lines parallel to each other, which touches the boundary of C. This can be easily done in linear time. We can also rotate two parallel lines which are perpendicular to the first set. Those four lines together induces a rectangle. It is easy to observe that during this rotation, we will encounter the minimum area rectangle. The function those lines define, changes every time the lines changes the vertices they rotate around. This happens $O(n)$ time. When the vertices the lines rotate around are fixed, the area function is a constant size function, and as such its minimum/maximum can be computed in linear time. Thus, the minimum volume bounding box, can be computed in $O(n \log n)$ time.

An important property of the minimum area rectangle, is that one of the edges of the convex hull lie on one of the bounding rectangle edges. We will refer to this edge as being *flush*. Thus, there are only n possibilities we have to check.

18.4.2 An exact algorithm 3d

Let P be a set of points in $\mathbb{R}^3$, our purpose is to compute the minimum volume bounding box B_{opt} of P. It is easy to verify that B_{opt} must touch P on every one of its faces. In fact, consider an edge e for the bounding box B_{opt}. Clearly, if we project the points in the direction of e into a perpendicular plane h, it must hold that the projection of B_{opt} into this plane is a minimum area rectangle. As such, it has one flush edges, which corresponds to a flush edge of the convex hull of P that must lie on a face of B_{opt}. In fact, there must be two adjacent faces of B_{opt} that have flush edges of $\mathcal{CH}(\mathsf{P})$ on them.

Otherwise, consider a face f of B_{opt} that has an edge flush on it. All the four adjacent faces of B_{opt} do not have flush edges on them. But thats not possible, since we can project the points in the direction of the normal of f, and argue that in the projection there must be a flush edge. This flush edge, corresponds to an edge of $\mathcal{CH}(\mathsf{P})$ that lies on one of the faces of B_{opt} that is adjacent to f.

Lemma 18.4.1 *If B_{opt} is the minimum volume bounding box of P, then it has two adjacent faces which are flush.*

This provides us with a natural algorithm to compute the minimum volume bounding box. Indeed, let us check all possible pair of edges $e, e' \in \mathcal{CH}(\mathsf{P})$. For each such pair, compute the minimum volume bounding box that has e and e' as flush.

Consider the normal $\vec{n}$ of the face of a bounding box that contains e. The normal $\vec{n}$ lie on a great circle on the sphere of directions, which are all the directions that are orthogonal to e. Let us parameterize $\vec{n}$ by a point on this normal. Next, consider the normal $\vec{n'}$ to the face that is flush to e'. Clearly, $\vec{n'}$ is orthogonal both to e' and $\vec{n}$. As such, we can compute this normal in constant time. Similarly, we can compute the third direction of the bounding box using vector product in const time. Thus, if e and e' are fixed, there is one dimensional family of bounding boxes of P that have e and e' flush on them, and comply with all the requirements to be a minimum volume bounding box.

It is now easy to verify that we can compute the representation of this family of bounding boxes, by tracking what vertices of the convex-hull the bounding boxes touches (i.e., this is similar to the rotating calipers algorithm, but one has to be more careful about the details). This can be done in linear time, and as such, one con compute the minimum volume bounding box in this family in linear time. Doing this for all pair of edges, results in $O(n^3)$ time algorithm, where $n = |\mathsf{P}|$.

Theorem 18.4.2 *Let P be a set of n points in $\mathbb{R}^3$. One can compute the minimum volume bounding box of P in $O(n^3)$ time.*

18.5 Approximating the Minimum Volume Bounding Box in Three Dimensions

Let P be a set of n points in $\mathbb{R}^3$, and let B_{opt} denote the minimum volume bounding box of P. We remind the reader, that for two sets A and B in $\mathbb{R}^3$. The *Minkowski sum* of A and B is the set $A \oplus B = \left\{ a + b \;\middle|\; a \in A, b \in B \right\}$.

Let $B = B(\mathsf{P})$ be the bounding box of P computed by Lemma 18.3.1, and let B_ε be a translated copy of $\frac{\varepsilon}{c}B$ centered at the origin, where c is an appropriate constant to be determined shortly. In addition, define $\mathcal{Q} = \mathcal{CH}(\mathsf{P}) \oplus B_\varepsilon$ and $\mathcal{G} = \mathsf{G}(\frac{1}{2}B_\varepsilon)$ denote the grid covering space, where every grid cell is a translated copy of $B_\varepsilon/2$. We approximate P on $\mathcal{G}$. For each point $p \in \mathsf{P}$ let $\mathcal{G}(p)$

be the set of eight vertices of the cell of $\mathcal{G}$ that contains p, and let $S_{\mathcal{G}} = \cup_{p \in S} \mathcal{G}(p)$. Define $\mathcal{P} = \mathcal{CH}(S_{\mathcal{G}})$. Clearly, $\mathcal{CH}(\mathsf{P}) \subseteq P \subseteq Q$. Moreover, one can compute $\mathcal{P}$ in $O(n + (1/\varepsilon^2)\log(1/\varepsilon))$ time. On the other hand, $\mathcal{P} \subseteq B \oplus B_\varepsilon$. The latter term is a box which contains at most $k = 2c/\varepsilon + 1$ grid points along each of the directions set by B, so k is also an upper bound for the number of grid points contained by $\mathcal{P}$ in each direction. As such, the convex hull of $\mathcal{CH}(P)$ is $O(k^2)$, as every grid line can contribute at most two vertices to the convex hull. Let R the set of vertices of $\mathcal{P}$. We next apply the exact algorithm of Theorem 18.4.2 to R. Let $\widehat{B}$ denote the resulting bounding box.

It remains to show that $\widehat{B}$ is a $(1+\varepsilon)$-approximation of $B_{\text{opt}}(\mathsf{P})$. Let $B^{\varepsilon}_{\text{opt}}$ be a translation of $\frac{\varepsilon}{4} B_{\text{opt}}(\mathsf{P})$ that contains B_ε. (The existence of $B^{\varepsilon}_{\text{opt}}$ is guaranteed by Lemma 18.3.1, if we take $c = 160$.) Thus, $\mathsf{R} \subseteq \mathcal{CH}(\mathsf{P}) \oplus B_\varepsilon \subseteq \mathcal{CH}(\mathsf{P}) \oplus B^{\varepsilon}_{\text{opt}} \subseteq B_{\text{opt}}(\mathsf{P}) \oplus B^{\varepsilon}_{\text{opt}}$. Since $B_{\text{opt}}(\mathsf{P}) \oplus B^{\varepsilon}_{\text{opt}}$ is a box, it is a bounding box of P and therefore also of $\mathcal{CH}(\mathsf{P})$. Its volume is

$$\text{Vol}(B_{\text{opt}}(\mathsf{P}) \oplus B^{\varepsilon}_{\text{opt}}) = \left(1 + \frac{\varepsilon}{4}\right)^3 \text{Vol}(B_{\text{opt}}(\mathsf{P})) < (1+\varepsilon)\,\text{Vol}(B_{\text{opt}}(\mathsf{P})),$$

as desired. (The last inequality is the only place where we use the assumption $\varepsilon \leq 1$.)

To recap, the algorithm consists of the four following steps:

1. Compute the box $B(\mathsf{P})$ (see Lemma 18.3.1) in $O(n)$ time.
2. Compute the point set $S_{\mathcal{G}}$ in $O(n)$ time.
3. Compute $P = \mathcal{CH}(S_{\mathcal{G}})$ in $O(n + (1/\varepsilon^2)\log(1/\varepsilon))$ time. This is done by computing the convex hull of all the extreme points of $S_{\mathcal{G}}$ along vertical lines of $\mathcal{G}$. We have $O(1/\varepsilon^2)$ such points, thus computing their convex hull takes $O((1/\varepsilon^2)\log(1/\varepsilon))$ time. Let R be the set of vertices of P.
4. Compute $B_{\text{opt}}(\mathsf{R})$ by the algorithm of Theorem 18.4.2. This step requires $O((1/\varepsilon^2)^3) = O(1/\varepsilon^6)$ time.

Theorem 18.5.1 *Let P be a set of n points in $\mathbb{R}^3$, and let $0 < \varepsilon \leq 1$ be a parameter. One can compute in $O(n + 1/\varepsilon^6)$ time a bounding box $B(\mathsf{P})$ with $\text{Vol}(B(\mathsf{P})) \leq (1+\varepsilon)\,\text{Vol}(B_{\text{opt}}(\mathsf{P}))$.*

Note that the box $B(S)$ computed by the above algorithm is most likely not minimal along its directions. The minimum bounding box of P homothet of $B(S)$ can be computed in additional $O(n)$ time.

18.6 Bibliographical notes

Our exposition follows roughly the work of Barequet and Har-Peled [BH01]. However, the basic idea, of finding the diameter, projecting along it and recursively finding a good bounding box on the projected input, is much older, and can be traced back to the work of Macbeath [Mac50].

For approximating the diameter, one can find in linear time a $(1/\sqrt{3})$-approximation of the diameter in *any* dimension; see [EK89].

The rotating calipers algorithm (Section 18.4.1) is due to Toussaint [Tou83]. The elegant extension of this algorithm to the computation of the exact minimum volume bounding box algorithm is due to O'Rourke [O'R85].

Lemma 18.3.1 is (essentially) from [BH01].

The current constants in Lemma 18.3.1 are unreasonable, but there is no reason to believe they are tight.

Conjecture 18.6.1 *The constants in Lemma 18.3.1 can be improved to be polynomial in the dimension.*

Coresets. One alternative approach to the algorithm of Theorem 18.5.1 is to construct G using $B_\varepsilon/2$ as before, and picking from each non-empty cell of G, one point of P as a representative point. This results in a set $\mathcal{S}$ of $O(1/\varepsilon^2)$ points. Compute the minimum volume bounding box $\mathcal{S}$ using the exact algorithm. Let B denote the resulting bounding box. It is easy to verify that $(1+\varepsilon)B$ contains P, and that it is a $(1+\varepsilon)$-approximation to the optimal bounding box of P. The running time of the new algorithm is identical. The interesting property is that we are running the exact algorithm on on a subset of the input.

This is a powerful technique for approximation algorithms. You first extract a small subset from the input, and run an exact algorithm on this input, making sure that the result provides the required approximation. The subset $\mathcal{S}$ is referred to as *coreset* of B as it preserves a geometric property of P (in our case, the minimum volume bounding box). We will see more about this notion in the following lectures.

Chapter 19

Approximating the Directional Width of a Shape

> "From the days of John the Baptist until now, the kingdom of heaven suffereth violence, and the violent bear it away."
>
> – – Matthew 11:12

19.1 Coreset for Directional Width

Let P be a set of points in $\mathbb{R}^{\mathsf{d}}$. For a vector $v \in \mathbb{R}^{\mathsf{d}}$, such that $v \neq 0$, let

$$\overline{\omega}(v, \mathsf{P}) = \max_{p \in \mathsf{P}} \langle v, p \rangle - \min_{p \in \mathsf{P}} \langle v, p \rangle ,$$

denote the *directional width* of P in the direction of v.

A set $Q \subseteq P$ is a *ε-coreset for directional width*, if

$$\forall v \in \mathbb{S}^{(\mathsf{d}-1)} \quad \overline{\omega}(v, Q) \geq (1 - \varepsilon)\overline{\omega}(v, P).$$

Namely, the coreset Q provides a concise approximation to the directional width of P. The usefulness of such a coreset might become clearer in the light of the following claim.

Claim 19.1.1 *Let P be a set of points in $\mathbb{R}^{\mathsf{d}}$, $0 < \varepsilon \leq 1$ a parameter and let Q be a δ-coreset of P for directional width, for $\delta = \varepsilon/(8\mathsf{d})$. Let $\mathcal{B}_{\text{opt}}(Q)$ denote the minimum volume bounding box of Q. Let B' be the rescaling of $\mathcal{B}_{\text{opt}}(Q)$ around its center by a factor of $(1 + 3\delta)$.*

Then, $\mathsf{P} \subset B'$, and in particular, $\text{Vol}(B') \leq (1 + \varepsilon)\mathcal{B}_{\text{opt}}(\mathsf{P})$, where $\mathcal{B}_{\text{opt}}(\mathsf{P})$ denotes the minimum volume bounding box of P.

Proof: Let v be a direction parallel to one of the edges of $\mathcal{B}_{\text{opt}}(Q)$, and let ℓ be a line through the origin with the direction of v. Let I and I' be the projection of $\mathcal{B}_{\text{opt}}(Q)$ and B', respectively, into ℓ. Let I_P be the interval formed by the projection of $\mathcal{CH}(\mathsf{P})$ into ℓ. We have that $I \subseteq I_P$ and $|I| \geq (1 - \delta)\,|I_P|$. The interval I' is the result of expanding I around its center point c by a factor of $1 + 3\delta$. In particular, the distance between c and the furthest endpoint of I_p is $\leq (1 - (1 - \delta)/2)\left|I_p\right| = (1 + \delta)\left|I_p\right|/2$. Thus, we need to verify that after the expansion of I it contains this endpoint. Namely,

$$(1 + 3\delta)\frac{1 - \delta}{2}\,|I_P| \geq \frac{1 + 2\delta - 3\delta^2}{2}\,|I_P| \geq \frac{1 + \delta}{2}\,|I_P| ,$$

for $\delta \leq 1/3$. Thus, $I_P \subseteq I'$ and $\mathsf{P} \subseteq B'$.

Observe that $\text{Vol}(B') \leq (1 + 3\delta)^{\mathsf{d}}\,\text{Vol}(\mathcal{B}_{\text{opt}}(Q)) \leq \exp(3\delta\mathsf{d})\,\text{Vol}(\mathcal{B}_{\text{opt}}(Q)) \leq (1 + \varepsilon)\mathcal{B}_{\text{opt}}(Q) \leq (1 + \varepsilon)\mathcal{B}_{\text{opt}}(\mathsf{P})$. ■

It is easy to verify, that a coreset for directional width, also preserves (approximately) the diameter and width of the point set. Namely, it captures "well" the geometry of P. Claim 19.1.1 hints on the connection between coreset for directional width and the minimum volume bounding box. In particular, if we have a good bounding box, we can compute a small coreset for directional width.

Lemma 19.1.2 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$, and let B be a bounding box of P, such that $v + c_d B \subseteq \mathcal{CH}(\mathsf{P})$, where v is a vector in $\mathbb{R}^{\mathsf{d}}$, $c_d = (4\mathsf{d} + 1)\mathsf{d}$ and $c_d B$ denote the rescaling of B by a factor of c_d around its center point.*

Then, one can compute a ε-coreset $\mathcal{S}$ for directional width of P. The size of the coreset is $O(1/\varepsilon^{\mathsf{d}-1})$, and construction time is $O\Big(n + \min(n, 1/\varepsilon^{\mathsf{d}-1})\Big)$.

Proof: We partition B into a grid, by breaking each edge of B into $M = \lceil 4/(\varepsilon c_d)\rceil$ equal length intervals (namely, we tile B with M^{d} copies of B/M). A cell in this grid is uniquely defined by a d-tuple $(i_1, \ldots, i_{\mathsf{d}})$. In particular, for a point $p \in \mathsf{P}$, lets $I(p)$ denote the ID of this point. Clearly, $I(p)$ can be computed in constant time.

Given a $(\mathsf{d}-1)$-tuple $I = (i_1, \ldots, i_{\mathsf{d}-1})$ its *pillar* is the set of grid cells that have $(i_1, \ldots, i_{\mathsf{d}-1})$ as the first $\mathsf{d}-1$ coordinates of their ID. Scan the points of P, and for each pillar record the highest and lowest point encountered. Here highest/lowest refer to their value in the dth direction.

We claim that the resulting set $\mathcal{S}$ is the required coreset. Indeed, consider a direction $v \in \mathbb{S}^{(\mathsf{d}-1)}$, and a point $p \in \mathsf{P}$. Let q, q' be the highest and lowest points in P which are inside the pillar of p, and are thus in $\mathcal{S}$. Let $B_q, B_{q'}$ be the two grid cells containing q and q'. Clearly, the projection of $\mathcal{CH}(B_q \cup B_{q'})$ into the direction of v contains the projection of p into the direction of v. Thus, for a vertex u of B_q it is sufficient to show that

$$\overline{\omega}(v, \{u, q\}) \leq \overline{\omega}(v, B/M) \leq (\varepsilon/2)\overline{\omega}(v, \mathsf{P}),$$

since this implies that $\overline{\omega}(v, \mathcal{S}) \geq \overline{\omega}(v, \mathsf{P}) - 2\overline{\omega}(v, B/M) \geq (1-\varepsilon)\overline{\omega}(v, \mathsf{P})$. Indeed,

$$\overline{\omega}(v, B/M) \leq \overline{\omega}(v, B)/M \leq \overline{\omega}(v, \mathsf{P})/(c_d M) \leq \frac{\overline{\omega}(v, \mathsf{P})}{4/\varepsilon} \leq \frac{\varepsilon}{4}\overline{\omega}(v, \mathsf{P}).$$

As for the preprocessing time, it requires a somewhat careful implementation. We construct a hash-table (of size $O(n)$) and store for every pillar the top and bottom points encountered. When handling a point this hash table can be updated in constant time. Once the coreset was computed, the coreset can be extracted from the hash-table in linear time. ■

Theorem 19.1.3 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$, and let $0 < \varepsilon < 1$ be a parameter. One can compute a ε-coreset $\mathcal{S}$ for directional width of P. The size of the coreset is $O(1/\varepsilon^{\mathsf{d}-1})$, and construction time is $O\Big(n + \min(n, 1/\varepsilon^{\mathsf{d}-1})\Big)$.*

Proof: Compute a good bounding box of P using Lemma 18.3.1. Then apply Lemma 19.1.2 to P. ■

19.2 Smaller coreset for directional width

We call $\mathsf{P} \subseteq \mathbb{R}^{\mathsf{d}}$ *α-fat*, for $\alpha \leq 1$, if there exists a point $p \in \mathbb{R}^{\mathsf{d}}$ and a hypercube $\overline{\mathcal{H}}$ centered at the origin so that

$$p + \alpha\overline{\mathcal{H}} \subset \mathcal{CH}(P) \subset p + \overline{\mathcal{H}}.$$

19.2.1 Transforming a set into a fat set

Lemma 19.2.1 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$ such that the volume of $\mathcal{CH}(\mathsf{P})$ is non-zero, and let $\mathcal{H} = [-1, 1]^{\mathsf{d}}$. One can compute in $O(n)$ time an affine transform τ so that $\tau(\mathsf{P})$ is an α-fat point set satisfying $\alpha\mathcal{H} \subset \mathcal{CH}(\tau(\mathsf{P})) \subset \mathcal{H}$, where α is a positive constant depending on d, and so that a subset $\mathcal{S} \subseteq \mathsf{P}$ is an ε-coreset of P for directional width if and only if $\tau(Q)$ is an ε-coreset of $\tau(\mathsf{P})$ for directional width.*

Proof: Using the algorithm of Lemma 18.3.1 compute, in $O(n)$ time, a bounding B of P, such that there exists a vector $\vec{w}$ such that $\vec{w} + \frac{1}{\mathsf{d}(4\mathsf{d}+1)}B \subseteq \mathcal{CH}(\mathsf{P}) \subseteq B$.

Let T_1 be the linear transformation that translates the center of T to the origin. Clearly, $\mathcal{S} \subseteq \mathsf{P}$ is a ε-coreset for direction width of P, if and only if $\mathcal{S}_1 = T_1(\mathcal{S})$ is an ε-coreset for directional width of $\mathsf{P}_1 = T_1(\mathsf{P})$. Next, let T_2 be a rotation that rotates $B_1 = T_1(B)$ such that its sides are parallel to the axises. Again, $\mathcal{S}_2 = T_2(\mathcal{S}_1)$ is a ε-coreset for $\mathsf{P}_2 = T_2(\mathsf{P}_1)$ if and only if $\mathcal{S}_1$ is a coreset for P_1. Finally, let T_3 be a scaling of the axises such that $B_2 = T_2(B_1)$ is mapped to the hypercube $\mathcal{H}$.

Note, that T_3 is just a diagonal matrix. As such, for any $p \in \mathbb{R}^{\mathsf{d}}$, and a vector $v \in \mathbb{S}^{(d-1)}$ we have

$$\langle v, T_3 p\rangle = v^T T_3 p = \left(T_3^T v\right)^T p == \left\langle T_3^T v, p\right\rangle = \langle T_3 v, p\rangle .$$

Let $\mathcal{S}_3 = T_3(\mathcal{S}_2)$ and let $\mathsf{P}_3 = T_3(\mathsf{P}_2)$. Clearly, for $v \in \mathbb{R}^{\mathsf{d}}$, $v \neq 0$ we have

$$\begin{aligned}\overline{\omega}(v, \mathsf{P}_3) &= \max_{p\in\mathsf{P}_3}\langle v, p\rangle - \min_{p\in\mathsf{P}_3}\langle v, p\rangle = \max_{p\in\mathsf{P}_2}\langle v, T_3 p\rangle - \min_{p\in\mathsf{P}_2}\langle v, T_3 p\rangle = \max_{p\in\mathsf{P}_2}\langle T_3 v, p\rangle - \min_{p\in\mathsf{P}_2}\langle T_3 v, p\rangle \\ &= \overline{\omega}(T_3 v, \mathsf{P}_2).\end{aligned}$$

Similarly, $\overline{\omega}(v, \mathcal{S}_3) = \overline{\omega}(T_3 v, \mathcal{S}_2)$.

By definition, $\mathcal{S}_2$ is a ε-coreset for P_2 iff for any non zero $v \in \mathbb{R}^{\mathsf{d}}$, we have $\overline{\omega}(v, \mathsf{P}_2) \geq (1-\varepsilon)\overline{\omega}(v, \mathcal{S}_2)$. Since T_3 non singular, this implies that for any non-zero v, we have $\overline{\omega}(T_3 v, \mathcal{S}_2) \geq (1-\varepsilon)\overline{\omega}(T_3 v, \mathsf{P}_2)$, which holds iff $\overline{\omega}(v, \mathcal{S}_3) = \overline{\omega}(v, T_3(\mathcal{S}_2)) \geq (1-\varepsilon)\overline{\omega}(v, T_3(\mathsf{P}_2)) = (1-\varepsilon)\overline{\omega}(v, \mathsf{P}_3)$. Thus $\mathcal{S}_3$ is a ε-coreset for P_3. Clearly, the other direction holds by a similar argumentation.

Set $T = T_3 T_2 T_1$, and observe that, by the above argumentation, $\mathcal{S}$ is a ε-coreset for P if and only if $T(\mathcal{S})$ is a ε-coreset for $T(\mathsf{P})$. However, note that $T(B) = \mathcal{H}$, and $T(\vec{w} + \frac{1}{\mathsf{d}(4\mathsf{d}+1)}B) \subseteq \mathcal{CH}(T(\mathsf{P}))$. Namely, there exists a vector $\vec{w'}$ such that $\vec{w'} + \frac{1}{\mathsf{d}(4\mathsf{d}+1)}\mathcal{H} \subseteq \mathcal{CH}(T(\mathsf{P})) \subseteq \mathcal{H}$. Namely, the point set $T(\mathsf{P})$ is $\alpha = \frac{1}{\mathsf{d}(4\mathsf{d}+1)}$-fat. ■

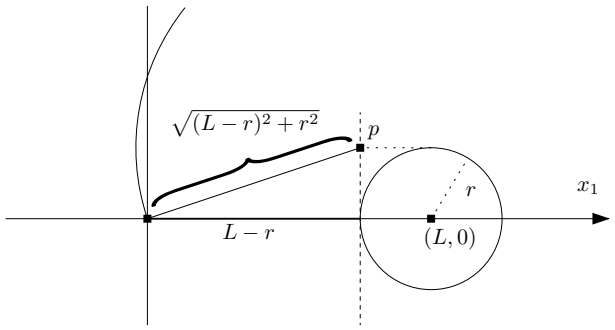


Figure 19.1: Illustration of the proof of Lemma 19.2.3.

19.2.2 Computing a smaller coreset

Observation 19.2.2 *If A is a δ-coreset for directional width of B, and B is a ε-coreset for directional width of C, then A is a $(\delta+\varepsilon)$-coreset of C.*

Proof: For any vector v, we have $\overline{\omega}(v,A) \geq (1-\delta)\overline{\omega}(v,B) \geq (1-\delta)(1-\varepsilon)\overline{\omega}(v,C) \geq (1-\delta-\varepsilon)\overline{\omega}(v,C)$. ■

Thus, given a point-set P, we can first extract from it a $\varepsilon/2$-coreset of size $O(1/\varepsilon^{\mathsf{d}-1})$, using Lemma 19.1.2. Let Q denote the resulting set. We will compute a $\varepsilon/2$-coreset for Q, which would be by the above observation a ε-coreset for directional width of P.

We need the following technical lemma.

Lemma 19.2.3 *Let $\mathbf{b}$ be a ball of radius r centered at $(L,0,\ldots,0) \in \mathbb{R}^{\mathsf{d}}$, where $L \geq 2r$. Let p be an arbitrary point in $\mathbf{b}$, and let $\mathbf{b}'$ be the largest ball centered at p and touching the origin. Then, we have that for $\mu(p) = \min_{(x_1,x_2,\ldots,x_{\mathsf{d}})\in\mathbf{b}'} x_1$ we have $\mu(p) \geq -r^2/L$.*

Proof: Clearly, if we move p in parallel to the x_1-axis by decreasing the value of x_1, we are decreasing the value of $\mu(p)$. Thus, in the worst case $x_1(p) = L - r$. Similarly, the farther away p is from the x_1-axis the smaller $\mu(p)$ is. Thus, by symmetry, the worst case is when $p = (L-r, r, 0, \ldots, 0)$. See Figure 19.1. The distance between p and the origin is $\sqrt{(L-r)^2+r^2}$, and

$$\mu(p) = (L-r) - \sqrt{(L-r)^2+r^2} = \frac{(L-r)^2 - (L-r)^2 - r^2}{(L-r) + \sqrt{(L-r)^2+r^2}} \geq -\frac{r^2}{2(L-r)} \geq -\frac{r^2}{L},$$

since $L \geq 2r$. ■

Lemma 19.2.4 *Let Q be a set of m points. Then one can compute a $\varepsilon/2$-coreset for Q of size $O(1/\varepsilon^{(d-1)/2})$, in time $O(m/\varepsilon^{(d-1)/2})$.*

Proof: Note, that byLemma 19.2.1, we can assume that Q is α-fat for some constant α, and $v + \alpha[-1,1]^{\mathsf{d}} \subseteq \mathcal{CH}(\mathsf{Q}) \subseteq [-1,1]^{\mathsf{d}}$, where $v \in \mathbb{R}^d$. In particular, for any direction $u \in \mathbb{S}^{(\mathsf{d}-1)}$, we have $\overline{\omega}(u,\mathsf{Q}) \geq 2\alpha$.

Let $\mathcal{S}$ be the sphere of radius $\sqrt{d}+1$ centered at the origin. Set $\delta = \sqrt{\varepsilon\alpha/4} \leq 1/4$. One can construct a set $\mathcal{I}$ of $O(1/\delta^{\mathsf{d}-1}) = O(1/\varepsilon^{(\mathsf{d}-1)/2})$ points on the sphere $\mathcal{S}$ so that for any point x on $\mathcal{S}$, there exists a point $y \in \mathcal{I}$ such that $\|x-y\| \leq \delta$. We process Q into a data structure that can answer ε-approximate nearest-neighbor queries. For a query point q, let $\phi(q)$ be the point of Q returned by this data structure. For each point $y \in \mathcal{I}$, we compute $\phi(y)$ using this data structure. We return the set $\mathcal{S} = \{\phi(y) \mid y \in \mathcal{I}\}$; see Figure 19.2 (ii).

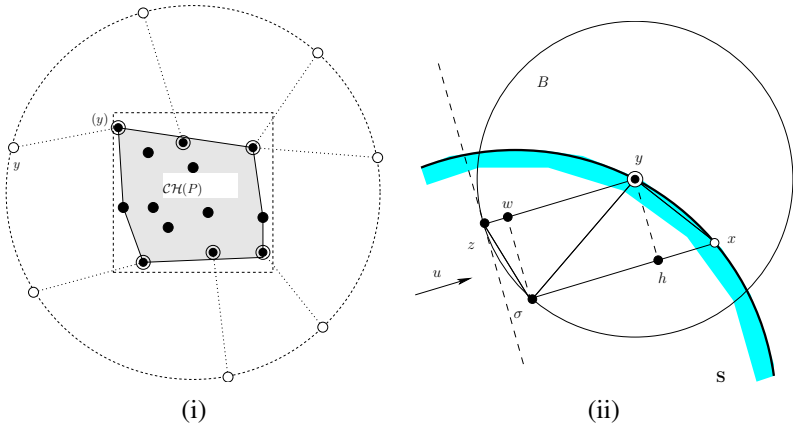
We now show that $\mathcal{S}$ is an $(\varepsilon/2)$-coreset of Q. For simplicity, we prove the claim under the assumption that $\phi(y)$ is the *exact* nearest-neighbor of y in Q. Fix a direction $u \in \mathbb{S}^{(\mathsf{d}-1)}$. Let $\sigma \in \mathsf{Q}$ be the point that maximizes $\langle u, p\rangle$ over all $p \in \mathsf{Q}$. Suppose the ray emanating from σ in direction u hits $\mathcal{S}$ at a point x. We know that there exists a point $y \in \mathcal{I}$ such that $\|x-y\| \leq \delta$. If $\phi(y) = \sigma$, then $\sigma \in \mathcal{S}$ and

$$\max_{p\in\mathsf{Q}} \langle u, p\rangle - \max_{q\in\mathcal{S}} \langle u, q\rangle = 0.$$

Now suppose $\phi(y) \neq \sigma$. Rotate and translate space, such that σ is at the origin, and u is the positive x_1 axis. Setting $L = \|ox\|$ and $r = \delta$, we have that $\langle u, y\rangle \geq -r^2/L \geq -\delta^2/1 = -\delta^2 = -\varepsilon\alpha/4$, by Lemma 19.2.3. We conclude that $\overline{\omega}(u,\mathcal{S}) \geq \overline{\omega}(u,\mathsf{Q}) - 2(\varepsilon\alpha/4) = \overline{\omega}(u,\mathsf{Q}) - \varepsilon\alpha/2$. On the other hand, since $\overline{\omega}(u,\mathsf{Q}) \geq 2\alpha$, it follows tat $\overline{\omega}(u,\mathcal{S}) \geq (1-\varepsilon/2)\overline{\omega}(u,\mathsf{Q})$.

As for the running time, we just perform the scan in the most naive way to find $\phi(y)$ for each $y \in \mathcal{I}$. Thus, the running time is as stated. ■

Theorem 19.2.5 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$. One can compute a ε-coreset for directional width of P in $O(n + 1/\varepsilon^{3(d-1)/2})$ time. The coreset size is $O(\varepsilon^{(d-1)/2})$.*



(i) (ii)

Figure 19.2: (i) An improved algorithm. (ii) Correctness of the improved algorithm.

Proof: We use the algorithm of Lemma 19.2.1 and Lemma 19.1.2 on the resulting set. This computes a $\varepsilon/2$-coreset Q of P of size $O(1/\varepsilon^{\mathsf{d}-1})$. Next, we apply Lemma 19.2.4 and compute a $\varepsilon/2$-coreset δ of Q. This is a ε-coreset of P. ■

19.3 Exercises

Exercise 19.3.1 [5 Points]
Prove that in the worst case, a ε-coreset for directional width has to be of size $\Omega(\varepsilon^{-(d-1)/2})$.

19.4 Bibliographical notes

Section 19.1 and Section 19.2.1 is from [AHV04]. The result of Section 19.2.2 was observed independently by Chan [Cha06] and Yu *et al.* [YAPV04]. It is a simplification of an algorithm of Agarwal *et al.* [AHV04] which in turn is an adaptation of a method of Dudley [Dud74].

The running time of Theorem 19.2.5 can be improved to $O(n + 1/\varepsilon^{\mathsf{d}-1})$ by using special nearest neighbor algorithm on a grid. Trying to use some of the other ANN data-strictures will not work since it would not improve the running time over the naive algorithm. The interested reader, can see the paper by Chan [Cha06].

Chapter 20

Approximating the Extent of Lines, Hyperplanes and Moving Points

> Once I sat on the steps by a gate of David's Tower, I placed my two heavy baskets at my side. A group of tourists was standing around their guide and I became their target marker. "You see that man with the baskets? Just right of his head there's an arch from the Roman period. Just right of his head."
>
> "But he's moving, he's moving!"
>
> I said to myself: redemption will come only if their guide tells them, "You see that arch from the Roman period? It's not important: but next to it, left and down a bit, there sits a man who's bought fruit and vegetables for his family."
>
> – –Yehuda Amichai, Tourists

20.1 Preliminaries

Definition 20.1.1 Given a set of hyperplanes $\mathcal{H}$ in $\mathbb{R}^{\mathsf{d}}$, the minimization diagram of $\mathcal{H}$, known as the ***lower envelope*** of $\mathcal{H}$, is the function $\mathcal{L}_{\mathcal{H}} : \mathbb{R}^{\mathsf{d}-1} \to \mathbb{R}$, where we have $\mathcal{L}(\mathbf{x}) = \min_{h\in\mathcal{H}} h(\mathbf{x})$, for $\mathbf{x} \in \mathbb{R}^{\mathsf{d}-1}$.

Similarly, the ***upper envelope*** of $\mathcal{H}$ is the function $\mathcal{U}(\mathbf{x}) = \max_{h\in\mathcal{H}} h(\mathbf{x})$, for $\mathbf{x} \in \mathbb{R}^{\mathsf{d}-1}$.

The ***extent*** of $\mathcal{H}$ and $\mathbf{x} \in \mathbb{R}^{\mathsf{d}-1}$ is the vertical distance between the upper and lower envelope at $\mathbf{x}$; namely, $\mathcal{E}_{\mathcal{H}}(\mathbf{x}) = \mathcal{U}(\mathbf{x}) - \mathcal{L}(\mathbf{x})$.

20.2 Motivation - Maintaining the Bounding Box of Moving Points

Let $P = \{p_1, \ldots, p_n\}$ be a set of n points moving in $\mathbb{R}^{\mathsf{d}}$. For a given time t, let $p_i(t) = (x_i^1(t), \ldots, x_i^{\mathsf{d}}(t))$ denote the position of p_i at time t. We will use $P(t)$ denote the set P at time t. We say that the motion of P has *degree* k if every $x_i^j(t)$ is a polynomial of degree at most k. We call a motion of degree 1 *linear*. Namely, $p_i(t) = \eth_i + {}_i t$, where $\eth_i, {}_i \in \mathbb{R}^{\mathsf{d}}$. The values $\eth_i, {}_i$ are fixed.

Our purpose is to develop efficient approaches for maintaining various descriptors of the extent of P, including the smallest enclosing orthogonal rectangle of P. This measure indicates how spread out the point set P is. As the points move continuously, the extent measure of interest changes continuously as well, though its combinatorial realization changes only at certain discrete times. For example, the smallest orthogonal rectangle containing P can be represented by a sequence of $2d$ points, each lying on one of the facets of the rectangle. As the points move, the rectangle also changes continuously. At certain discrete times, the points lying on the boundary of the rectangle change, and we have to update the sequence of points defining the rectangle. Similarly, Our approach is to focus on these discrete changes (or *events*) and track through time the combinatorial description of the extent measure of interest.

Since we are computing the axis parallel bounding box of the moving points, we can solve the problem in each dimension separately. Thus, consider the points as points moving linearly in one dimension. The ***extent*** $B(t)$ of $P(t)$ is the smallest interval containing $P(t)$.

It will be convenient to work in a parametric xt-plane in which a moving point $p(t) \in \mathbb{R}$ at time t is mapped to the point $(t, p(t))$. For $1 \le i \le n$, we map the point $p_i \in P$ to the line $\ell_i = \bigcup_t (t, p_i(t))$, for $i = 1, \ldots, n$. Let $L = \{\ell_1, \ldots, \ell_n\}$ be the resulting set of lines, and let $\mathcal{A}(L)$ be their arrangement. Clearly, the extent $B(t_0)$ of $P(t_0)$ is the vertical interval $I(t_0)$ in the arrangement $\mathcal{A}(L)$ connecting the upper and lower envelopes of L at $t = t_0$. See Figure 20.1(i). The combinatorial structure of $I(t)$ changes at the vertices of the two envelopes of L, and all the different combinatorial structures of $I(t)$ can be computed in $O(n \log n)$ time by computing the upper and lower envelopes of L.

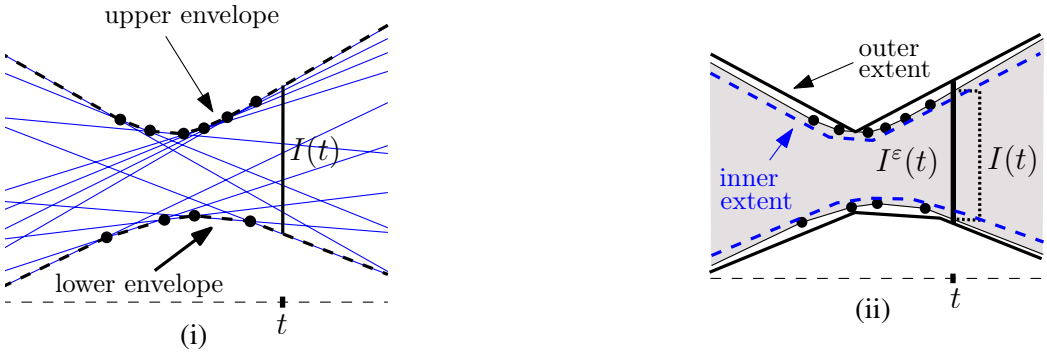


Figure 20.1: (i) The extent of the moving points, is no more than the vertical segment connecting the lower envelope to the upper envelope. The black dots mark where the movement description of $I(t)$ changes. (ii) The approximate extent.

We want to maintain a vertical interval $I^+_\varepsilon(t)$ so that $I(t) \subseteq I^+_\varepsilon(t)$ and $\left|I^+_\varepsilon(t)\right| \leq (1+\varepsilon)\,|I(t)|$ for all t, so that the endpoints of $I^+_\varepsilon(t)$ follow piecewise-linear trajectories, and so that the number of combinatorial changes in $I^\varepsilon(t)$ is small. Alternatively, we want to maintain a vertical interval $I^-_\varepsilon(t) \subseteq I(t)$ such that $\left|I^-_\varepsilon(t)\right| \geq (1-\varepsilon)\,|I(t)|$. Clearly, having one approximation would imply the other by appropriate rescaling.

Geometrically, this has the following interpretation: We want to simplify the upper and lower envelopes of $\mathcal{A}(L)$ by convex and concave polygonal chains, respectively, so that the simplified upper (resp. lower) envelope lies above (resp. below) the original upper (resp. lower) envelope and so that for any t, the vertical segment connecting the simplified envelopes is contained in $(1+\varepsilon)I(t)$. See Figure 20.1 (ii).

In the following, we will use duality, see Lemma 23.2.1 for the required properties we will need.

Definition 20.2.1 For a set of hyperplanes $\mathcal{H}$, a subset $\mathcal{S} \subset \mathcal{H}$ is a *ε-coreset* of $\mathcal{H}$ for the extent measure, if for any $\mathbf{x} \in \mathbb{R}^{d-1}$ we have $\mathcal{E}_{\mathcal{S}} \geq (1-\varepsilon)\mathcal{E}_{\mathcal{H}}$.

Similarly, for a point-set $\mathsf{P} \subseteq \mathbb{R}^d$, a set $\mathcal{S} \subseteq \mathsf{P}$ is a *ε-coreset for vertical extent* of P, if, for any direction $v \in \mathbb{S}^{(d-1)}$, we have that $\mu_v(\mathcal{S}) \geq (1-\varepsilon)\mu_v(\mathsf{P})$, where $\mu_v(\mathsf{P})$ is the vertical distance between the two supporting hyperplanes of P which are perpendicular to v.

Thus, to compute a coreset for a set of hyperplanes, it is by duality and Lemma 23.2.1 enough to find a coreset for the vertical extent of a point-set.

Lemma 20.2.2 *The set $\mathcal{S}$ is a ε-coreset of the point set $\mathsf{P} \subseteq \mathbb{R}^d$ for vertical extent if and only if $\mathcal{S}$ is a ε-coreset for directional width.*

Proof: Consider any direction $v \in \mathbb{S}^{(d-1)}$, and let α be its (smaller) angle with with the x_d axis. Clearly, $\overline{\omega}(v, \mathcal{S}) = \mu_v(\mathcal{S})\cos\alpha$ and $\overline{\omega}(v, \mathsf{P}) = \mu_v(PntSet)\cos\alpha$. Thus, if $\overline{\omega}(v, \mathcal{S}) \geq (1-\varepsilon)\overline{\omega}(v, \mathsf{P})$ then $\mu_v(\mathcal{S}) \geq (1-\varepsilon)\mu_v(\mathsf{P})$, and vice versa. ■

Theorem 20.2.3 *Let $\mathcal{H}$ be a set of n hyperplanes in $\mathbb{R}^d$. One can compute a ε-coreset of $\mathcal{H}$ of, size $O(1/\varepsilon^{d-1})$, in $O(n + \min(n, 1/\varepsilon^{d-1}))$ time. Alternatively, one can compute a ε-coreset of size $O(1/\varepsilon^{(d-1)/2})$, in time $O(n + 1/\varepsilon^{3(d-1)/2})$.*

Proof: By Lemma 20.2.2, the coreset computation is equivalent to computing coreset for directional width. However, this can be done in the stated bounds, by Theorem 19.1.3 and Theorem 19.2.5. ■

Going back to our motivation, we have the following result:

Lemma 20.2.4 *Let $P(t)$ be a set of n points with linear motion in $\mathbb{R}^d$. We can compute an axis parallel moving bounding box $b(t)$ for $P(t)$ that changes $O(d/\sqrt{\varepsilon})$ times (in other times, the bounding box moves with linear motion). The time to compute this bounding box is $O(d(n + 1/\varepsilon^{3/2}))$.*

Furthermore, we have that $\mathrm{Box}(P(t)) \subseteq b(t) \subseteq (1+\varepsilon)\mathrm{Box}(P(t))$, where $\mathrm{Box}(t)$ is the minimum axis parallel bounding box of P.

Proof: We compute the solution for each dimension separately. In each dimension, we compute a coreset of the resulting set of lines in two dimensions, and compute the upper and lower envelope of the coreset. Finally, we expand the upper and lower envelopes appropriately so that the include the original upper and lower envelopes. The bounds on the running time follows from Theorem 20.2.3. ■

20.3 Coresets

At this point, our discussion exposes a very powerful technique for approximate geometric algorithms: (i) extract small subset that represents that data well (i.e., coreset), and (ii) run some other algorithm on the coreset. To this end, we need a more unified definition of coresets.

Definition 20.3.1 (Coresets) Given a set P of points (or geometric objects) in $\mathbb{R}^{\mathsf{d}}$, and an objective function $f : 2^{\mathbb{R}^{\mathsf{d}}} \to \mathbb{R}$ (say, $f(\mathsf{P})$ is the width of P), a ***ε-coreset*** is a subset $\mathcal{S}$ of the points of P such that

$$f(\mathcal{S}) \geq (1-\varepsilon) f(\mathsf{P}).$$

We will state this fact, by saying that $\mathcal{S}$ is a ε-coreset of P for $f(\cdot)$.

If the function $f(\cdot)$ is parameterized, namely $f(Q, v)$, then $\mathcal{S} \subseteq \mathsf{P}$ is a coreset if

$$\forall v \quad f(\mathcal{S}, v) \geq (1-\varepsilon) f(\mathsf{P}, v).$$

As a concrete example, for v a unit vector, consider the function $\overline{\omega}(v, \mathsf{P})$ which is the directional width of P; namely, it is the length of the projection of $\mathcal{CH}(\mathsf{P})$ into the direction of v.

Coresets are of interest when they can be computed quickly, and have small size, hopefully of size independent of n, the size of the input set P. Interestingly, our current techniques are almost sufficient to show the existence of coresets for a large family of problems.

20.4 Extent of Polynomials

Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be a family of d-variate polynomials and let $u_1, \ldots, u_{\mathsf{d}}$ be the variables over which the functions of $\mathcal{F}$ are defined. Each f_i corresponds to a surface in $\mathbb{R}^{\mathsf{d}+1}$: For example, any d-variate linear function can be considered as a hyperplane in $\mathbb{R}^{\mathsf{d}+1}$ (and vice versa). The upper/lower envelopes and extent of $\mathcal{F}$ can now be defined similar to the hyperplane case.

Each monomial over $u_1, \ldots, u_{\mathsf{d}}$ appearing in $\mathcal{F}$ can be mapped to a distinct variable x_i. Let $x_1, \ldots, x_s$ be the resulting variables. As such $\mathcal{F}$ can be linearized into a set $\mathcal{H} = \{h_1, \ldots, h_n\}$ of linear functions over $\mathbb{R}^s$. In particular, $\mathcal{H}$ is a set of n hyperplanes in $\mathbb{R}^{s+1}$. Note that the surface induced by f_i in $\mathbb{R}^{\mathsf{d}+1}$ corresponds only to a subset of the surface of h_i in $\mathbb{R}^{s+1}$. This technique is called ***linearization***.

For example, consider a family of polynomials $\mathcal{F} = \{f_1, \ldots, f_n\}$, where $f_i(x, y) = a_i(x^2+y^2) + b_i x + c_i y + d_i$, and $a_i, b_i, c_i, d_i \in \mathbb{R}$, for $i = 1, \ldots, n$. This family of polynomials defined over $\mathbb{R}^2$, can be linearized to a family of linear functions defined over $\mathbb{R}^3$, by $h_i(x, y, z) = a_i z + b_i x + c_i y + d_i$, and setting $\mathcal{H} = \{h_1, \ldots, h_n\}$. Clearly, $\mathcal{H}$ is a set of hyperplanes in $\mathbb{R}^4$, and $f_i(x, y) = h_i(x, y, x^2+y^2)$. Thus, for any point $(x, y) \in \mathbb{R}^2$, instead of evaluating $\mathcal{F}$ on (x, y), we can evaluate $\mathcal{H}$ on $\eta(x, y) = (x, y, x^2+y^2)$, where $\eta(x, y)$ is the *linearization image* of (x, y). The advantage of this linearization is that $\mathcal{H}$, being a family of linear functions, is now easier to handle than $\mathcal{F}$.

Observe, that $X = \eta(\mathbb{R}^2)$ is a *subset* of $\mathbb{R}^3$ (this is the "standard" paraboloid), and we are interested in the value of $\mathcal{H}$ only on points belonging to X. In particular, the set X is not necessarily convex. The set X resulting from the linearization is a semi-algebraic set of constant complexity, and as such basic manipulation operations of X can be performed in constant time.

Note that for each $1 \leq i \leq n$, $f_i(p) = h_i(\eta(p))$ for $p \in \mathbb{R}^d$. As such, if $\mathcal{H}' \subseteq \mathcal{H}$ is a ε-coreset of $\mathcal{H}$ for the extent, then clearly the corresponding subset in $\mathcal{F}$ is a ε-coreset of $\mathcal{F}$ for the extent measure. The following theorem is a restatement of Theorem 20.2.3 in this settings.

Theorem 20.4.1 *Given a family of d-variate polynomials $\mathcal{F} = \{f_1, \ldots, f_n\}$, and parameter ε, one can compute, in $O(n + 1/\varepsilon^s)$ time, a subset $\mathcal{F}' \subseteq \mathcal{F}$ of $O(1/\varepsilon^s)$ polynomials, such that $\mathcal{F}'$ is a ε-coreset of $\mathcal{F}$ for the extent measure. Here s is the number of different monomials present in the polynomials of $\mathcal{F}$.*

Alternatively, one can compute a ε-coreset, of size $O(1/\varepsilon^{s/2})$, in tiem $O(n + 1/\varepsilon^{3s/2})$.

20.5 Roots of Polynomials

We now consider the problem of approximating the extent a family of square-roots of polynomials. Note, that this is considerably harder than handling polynomials because square-roots of polynomials can not be directly linearized. It turns out, however, that it is enough to $O(\varepsilon^2)$-approximate the extent of the functions inside the roots, and take the root of the resulting approximation.

Theorem 20.5.1 *Let $\mathcal{F} = \left\{(f_1)^{1/2}, \ldots, (f_n)^{1/2}\right\}$ be a family of k-variate functions (over $\mathsf{p} = (x_1, \ldots, x_k) \in \mathbb{R}^k$), where each f_i is a polynomial that is non-negative for every $\mathsf{p} \in \mathbb{R}^k$. Given any $\varepsilon > 0$, we can compute, in $O(n + 1/\varepsilon^{2k'})$ time, a ε-coreset $\mathcal{G} \subseteq \mathcal{F}$ of size $O(1/\varepsilon^{2k'})$, for the measure of the extent. Here k' is the number of different monomials present in the polynomials in $f_1, \ldots, f_n$.*

Alternatively, one can compute a set $\mathcal{G}' \subseteq \mathcal{F}$, in $O(n + 1/\varepsilon^{3k'})$ time, that ε-approximates $\mathcal{F}$, so that $|\mathcal{G}'| = O(1/\varepsilon^{k'})$.

Proof: Let $\mathcal{F}^2$ denote the family $\{f_1, \ldots, f_n\}$. Using the algorithm of Theorem 20.4.1, we compute a δ'-coreset $\mathcal{G}^2 \subseteq \mathcal{F}^2$ of $\mathcal{F}^2$, where $\delta' = \varepsilon^2/64$. Let $\mathcal{G} \subseteq \mathcal{F}$ denote the family $\left\{(f_i)^{1/2} | f_i \in \mathcal{G}^2\right\}$.

Consider any point $\mathbf{x} \in \mathbb{R}^k$. We have that $\mathcal{E}_{\mathcal{G}^2}(\mathbf{x}) \geq (1-\delta')\mathcal{E}_{\mathcal{F}^2}(\mathbf{x})$, and let $a = \mathcal{L}_{\mathcal{F}^2}(\mathbf{x})$, $A = \mathcal{L}_{\mathcal{G}^2}(\mathbf{x})$, $B = \mathcal{U}_{\mathcal{G}^2}(\mathbf{x})$, and $b = \mathcal{U}_{\mathcal{F}^2}(\mathbf{x})$. Clearly, we have $0 \leq a \leq A \leq B \leq b$ and $B - A \geq (1-\delta')(b-a)$. Since $(1+2\delta')(1-\delta') \geq 0$, we have that $(1+2\delta')(B-A) \geq b - a$.

By Lemma 20.5.2 below, we have that Then, $\sqrt{A} - \sqrt{a} \leq (\varepsilon/2)U$, and $\sqrt{b} - \sqrt{B} \leq (\varepsilon/2)U$, where $U = \sqrt{B} - \sqrt{A}$. Namely, $\sqrt{B} - \sqrt{A} \geq (1-\varepsilon)(\sqrt{b} - \sqrt{a})$. Namely, $\mathcal{G}$ is a ε-coreset for the extent of $\mathcal{F}$.

The bounds on the size of $\mathcal{G}$ and the running time are easily verified. ■

Lemma 20.5.2 *Let $0 \leq a \leq A \leq B \leq b$, and $0 < \varepsilon \leq 1$ be given parameters, so that $b - a \leq (1+\delta)(B-A)$, where $\delta = \varepsilon^2/16$. Then, $\sqrt{A} - \sqrt{a} \leq (\varepsilon/2)U$, and $\sqrt{b} - \sqrt{B} \leq (\varepsilon/2)U$, where $U = \sqrt{B} - \sqrt{A}$.*

Proof: Clearly,

$$\sqrt{A} + \sqrt{B} \quad \leq \quad \sqrt{a} + \sqrt{A-a} + \sqrt{b} \leq \sqrt{a} + \sqrt{\delta b} + \sqrt{b} \leq (1+\sqrt{\delta})(\sqrt{a} + \sqrt{b}).$$

Namely, $\frac{\sqrt{A}+\sqrt{B}}{1+\sqrt{\delta}} \leq \sqrt{a} + \sqrt{b}$. On the other hand,

$$\begin{aligned}\sqrt{b} - \sqrt{a} &= \frac{b-a}{\sqrt{b} + \sqrt{a}} \leq \frac{(1+\delta)(B-A)}{\sqrt{b} + \sqrt{a}} \leq (1+\delta)(1+\sqrt{\delta})\frac{B-A}{\sqrt{B} + \sqrt{A}} \\ &= \text{`}(1+\varepsilon^2/16)(1+\varepsilon/4)(\sqrt{B} - \sqrt{A}) \leq (1+\varepsilon/2)(\sqrt{B} - \sqrt{A}).\end{aligned}$$

■

20.6 Applications

20.6.1 Minimum Width Annulus

Let $\mathsf{P} = \{p_1, \ldots, p_n\}$ be a set of n points in the plane. Let $f_i(q)$ denote the distance of the ith point from the point q. It is easy to verify that $f_i(q) = \sqrt{\left(x_q - x_{p_i}\right)^2 + \left(y_q - y_{p_i}\right)^2}$. Let $\mathcal{F} = \{f_1, \ldots, f_n\}$. It is easy to verify that for a center point $\mathbf{x} \in \mathbb{R}^2$, the width of the minimum width annulus containing P which is centered at $\mathbf{x}$ has width $\mathcal{E}_{\mathcal{F}}(\mathbf{x})$. Thus, we would like to compute a ε-coreset for $\mathcal{F}$.

Consider the set of functions $\mathcal{F}^2$. Clearly, $f_i^2(x,y) = \left(x - x_{p_i}\right)^2 + \left(y - y_{p_i}\right)^2 = x^2 - 2x_{p_i}x + x_{p_i}^2 + y^2 - 2y_{p_i}y + y_{p_i}^2$. Clearly, all the functions of $\mathcal{F}^2$ have this (additive) common factor of $x^2 + y^2$. Since we only care about the vertical extent, we have $\mathcal{H} = \left\{-2x_{p_i}x + x_{p_i}^2 - 2y_{p_i}y + y_{p_i}^2 \;\middle|\; i = 1, \ldots, n\right\}$ has the same extent as $\mathcal{F}^2$; formally, for any $\mathbf{x} \in \mathbb{R}^2$, we have $\mathcal{E}_{\mathcal{F}^2}(\mathbf{x}) = \mathcal{E}_{\mathcal{H}}(\mathbf{x})$. Now, $\mathcal{H}$ is just a family of hyperplanes in $\mathbb{R}^3$, and it has a $\varepsilon^2/64$-coreset $\mathcal{S}_{\mathcal{H}}$ for the extent of size $1/\varepsilon$ which can be computed in $O(n + 1/\varepsilon^3)$ time. This corresponds to a $\varepsilon^2/64$-coreset $\mathcal{S}_{\mathcal{F}^2}$ of $\mathcal{F}^2$. By Theorem 20.5.1, this corresponds to a ε-coreset $\mathcal{S}_{\mathcal{F}}$ of $\mathcal{F}$. Finally, this corresponds to coreset $\mathcal{S} \subseteq \mathsf{P}$ of size $O(1/\varepsilon)$, such that the minimum width annulus of $\mathcal{S}$, if we expand it by $(1+2\varepsilon)$, it contains all the points of P. Thus, we can just find the minimum width annulus of $\mathcal{S}$. This can be done in $O(1/\varepsilon^2)$ time using an exact algorithm. Putting everything together, we get:

Theorem 20.6.1 *Let P be a set of n points in the plane, and let $0 \leq \varepsilon \leq 1$ be a parameter. One can compute a $(1+\varepsilon)$-approximate minimum width annulus to P in $O(n + 1/\varepsilon^3)$ time.*

20.7 Exercises

20.8 Bibliographical notes

Linearization was widely used in fields such as machine learning [CS00] and computational geometry [AM94].

There is a general technique for finding the best possible linearization (i.e., a mapping η with the target dimension as small as possible), see [AM94] for details.

Chapter 21

Approximating the Extent of Lines, Hyperplanes and Moving Points II

> Drug misuse is not a disease, it is a decision, like the decision to step out in front of a moving car. You would call that not a disease but an error in judgment. When a bunch of people begin to do it, it is a social error, a life-style. In this particular life-style the motto is "be happy now because tomorrow you are dying," but the dying begins almost at once, and the happiness is a memory. ... If there was any "sin," it was that these people wanted to keep on having a good time forever, and were punished for that, but, as I say, I feel that, if so, the punishment was far too great, and I prefer to think of it only in a Greek or morally neutral way, as mere science, as deterministic impartial cause-and-effect.
>
> – A Scanner Darkly, Philip K. Dick

21.1 More Coresets

21.1.1 Maintaining certain measures of moving points

We show that our techniques can be extended to handle other measure of moving points (width, diameter, etc). Let $\mathsf{P} = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^{\mathsf{d}}$, each moving independently. Let $p_i(t) = (p_{i1}(t), \ldots, p_{id}(t))$ denote the position of point p_i at time t. Set $\mathsf{P}(t) = \{p_i(t) \mid 1 \leq i \leq n\}$. If each p_{ij} is a polynomial of degree at most r, we say that the motion of P has *degree* r. We call the motion of P *linear* if $r = 1$ and *algebraic* if r is bounded by a constant.

Given a parameter $\varepsilon > 0$, we call a subset $Q \subseteq P$ an ε-coreset of P for directional width if for any direction $u \in \mathbb{S}^{(\mathsf{d}-1)}$, we have

$$(1 - \varepsilon)\overline{\omega}(u, P(t)) \leq \overline{\omega}(u, Q(t)) \qquad \text{for all } t \in \mathbb{R}.$$

21.1.1.1 Computing an ε-coreset for directional width.

First let us assume that the motion of P is linear, i.e., $p_i(t) = a_i + b_i t$, for $1 \leq i \leq n$, where $a_i, b_i \in \mathbb{R}^{\mathsf{d}}$. For a direction $u = (u_1, \ldots, u_{\mathsf{d}}) \in \mathbb{S}^{(\mathsf{d}-1)}$, we define a $(\mathsf{d}+1)$-variate polynomial

$$f_i(u, t) \quad = \quad \langle p_i(t), u \rangle = \langle a_i + b_i t, u \rangle = \sum_{j=1}^{\mathsf{d}} a_{ij} u_j + \sum_{j=1}^{\mathsf{d}} b_{ij} \cdot (t u_j).$$

Set $\mathcal{F} = \{f_1, \ldots, f_n\}$. Then

$$\overline{\omega}(u, P(t)) = \max_i \langle p_i(t), u \rangle - \min_i \langle p_i(t), u \rangle = \max_i f_i(u, t) - \min_i f_i(u, t) = \mathcal{E}_{\mathcal{F}}(u, t).$$

Since $\mathcal{F}$ is a family of $(\mathsf{d}+1)$-variate polynomials, which admits a linearization of dimension $2\mathsf{d}$ (there are $2\mathsf{d}$ monomials), using Theorem 20.4.1, we conclude the following.

Theorem 21.1.1 *Given a set P of n points in $\mathbb{R}^{\mathsf{d}}$, each moving linearly, and a parameter $\varepsilon > 0$, we can compute an ε-coreset of P for directional width of size $O(1/\varepsilon^{2\mathsf{d}})$, in $O(n + 1/\varepsilon^{2\mathsf{d}})$ time, or an ε-coreset of size $O(1/\varepsilon^{\mathsf{d}})$ in $O(n + 1/\varepsilon^{3(\mathsf{d})})$ time.*

If the degree of motion of P is $r > 1$, we can write the d-variate polynomial $f_i(u,t)$ as:

$$f_i(u,t) \quad = \quad \langle p_i(t), u\rangle = \left\langle \sum_{j=0}^{r} a_{ij}t^j, u \right\rangle = \sum_{j=0}^{r} \left\langle a_{ij}t^j, u\right\rangle$$

where $a_{ij} \in \mathbb{R}^{\mathsf{d}}$. A straightforward extension of the above argument shows that f_i's admit a linearization of dimension $(r+1)\mathsf{d}$. Using Theorem 20.4.1, we obtain the following.

Theorem 21.1.2 *Given a set P of n moving points in $\mathbb{R}^{\mathsf{d}}$ whose motion has degree $r > 1$ and a parameter $\varepsilon > 0$, we can compute an ε-coreset for directional width of P of size $O(1/\varepsilon^{(r+1)\mathsf{d}})$ in $O(n + 1/\varepsilon^{(r+1)\mathsf{d}})$ time, or of size $O(1/\varepsilon^{(r+1)\mathsf{d}}/2)$ in $O(n + 1/\varepsilon^{3(r+1)\mathsf{d}/2})$ time.*

21.1.2 Minimum-width cylindrical shell

Let $\mathsf{P} = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^d$, and a parameter $\varepsilon > 0$. Let $w^* = w^*(P)$ denote the width of the thinnest cylindrical shell, the region lying between two co-axial cylinders, containing P. Let $d(\ell, p)$ denote the distance between a point $p \in \mathbb{R}^d$ and a line $\ell \subset \mathbb{R}^d$. If we fix a line ℓ, then the width of the thinnest cylindrical shell with axis ℓ and containing P is $w(\ell, P) = \max_{p\in P} d(\ell, p) - \min_{p\in P} d(\ell, p)$. A line $\ell \in \mathbb{R}^d$ not parallel to the hyperplane $x_d = 0$ can be represented by a $(2d-2)$-tuple $(x_1, \ldots, x_{2d-2}) \in \mathbb{R}^{2d-2}$:

$$\ell = \{p + tq \mid t \in \mathbb{R}\},$$

where $p = (x_1, \ldots, x_{d-1}, 0)$ is the intersection point of ℓ with the hyperplane $x_d = 0$ and $q = (x_d, \ldots, x_{2d-2}, 1)$ is the orientation of ℓ (i.e., q is the intersection point of the hyperplane $x_d = 1$ with the line parallel to ℓ and passing through the origin). (The lines parallel to the hyperplane $x_d = 0$ can be handled separately by a simpler algorithm, so let us assume this case does not happend.) The distance between ℓ and a point p is the same as the distance of the line $\ell' = \{(p - \mathsf{p}) + tq \mid t \in \mathbb{R}\}$ from the origin; see Figure 21.1. The point y on ℓ' closest to the origin satisfies $y = (p - \mathsf{p}) + tq$ for some t, and at the same time $\langle y, q\rangle = \langle (p - \mathsf{p}) + tq, q\rangle = 0$, which implies that $t = -\langle (p - \mathsf{p}), q\rangle / \|q\|^2$. Thus,

$$d(\ell, \mathsf{p}) = \|y\| = \|(p - \mathsf{p}) + tq\| = \left\| (p - \mathsf{p}) - \frac{[\langle p - \mathsf{p}, q\rangle]q}{\|q\|^2} \right\|,$$

Define $f_i(\ell) = f_i(p, q) = d(\ell, p_i)$, and set $\mathcal{F} = \{f_i \mid p_i \in P\}$. Then $w^* = \min_{x\in\mathbb{R}^{2d-2}} \mathcal{E}_{\mathcal{F}}(x)$. (We assume for simplicity that the axis of the optimal shell is not parallel to the hyperplane $x_d = 0$.) Let $f_i'(p, q) = \|q\|^2 \cdot f_i(p, q) = \left\| \|q\|^2 (p - p_i) - \langle p - p_i, q\rangle q \right\|$, and set $\mathcal{F}' = \left\{f_1', \ldots, f_n'\right\}$.

Define $g_i(p, q) = \left(f_i'(p, q)\right)^2$, and let $\mathcal{G} = \{g_1 \ldots, g_n\}$. Then g_i is a $(2d-2)$-variate polynomial and has $O(d^2)$ monomials. Therefore $\mathcal{G}$ admits a linearization of dimension $O(d^2)$. By Theorem 20.4.1, we compute a $O(\varepsilon^2)$-coreset of $\mathcal{G}$ of size $O\left(1/\varepsilon^{d^2}\right)$ in $O\left(n + 1/\varepsilon^{O(d^2)}\right)$. This in turn corresponds to a ε-coreset of $\mathcal{F}'$, by Theorem 20.5.1. It is now easy to verify that this corresponds to a ε-coreset (for the extent) for $\mathcal{F}$. Finally, this corresponds to a subset $\mathcal{S} \subseteq \mathsf{P}$, such that $\mathcal{S}$ is a coreset of P for $w(\ell, \mathsf{P})$. Formally, a subset $\mathcal{S} \subseteq \mathsf{P}$ is a *ε-coreset for cylindrical shell width* , if

$$w(\ell, \mathcal{S}) \geq (1 - \varepsilon)w(\ell, \mathsf{P}), \quad \text{for all } v \in \mathbb{S}^{(\mathsf{d}-1)}.$$

Thus, we can compute in $O(n + 1/\varepsilon^{O(d^2)})$ time a set $Q \subseteq P$ of $1/\varepsilon^{O(d^2)}$ points so that for any line ℓ, $w(\ell, P) \geq w(\ell, Q) \geq (1 - \varepsilon)w(\ell, P)$ as well as a cylindrical shell of width at most $(1 + \varepsilon)w^*(P)$ that contains P. Hence, we conclude the following.

Theorem 21.1.3 *Given a set P of n points in $\mathbb{R}^d$ and a parameter $\varepsilon > 0$, we can compute in $O(n + 1/\varepsilon^{O(d^2)})$ time a subset $\mathcal{S} \subseteq P$ of size $O(1/\varepsilon^{O(d^2)})$ so that for any line ℓ in $\mathbb{R}^d$, we have $w(\ell, \mathcal{S}) \geq (1 - \varepsilon)w(\ell, P)$.*

Note, that Theorem 21.1.3 does not compute the optimal cylinder, it just computes a small coreset for this problem. Clearly, we can now run any brute force algorithm on this coreset. This would result in running time $O(n + 1/\varepsilon^{O(d^4)})$, which would output a cylinder which if expanded by factor $1 + \varepsilon$, will cover all the points of P. In fact, the running time can be further improved.

21.2 Exercises

21.3 Bibliographical notes

Section 21.1.1 is from Agarwal *et al.* [AHV04], and the results can be (very slightly) improved by treating the direction as $(\mathsf{d} - 1)$-dimensional entity, see [AHV04] for details.

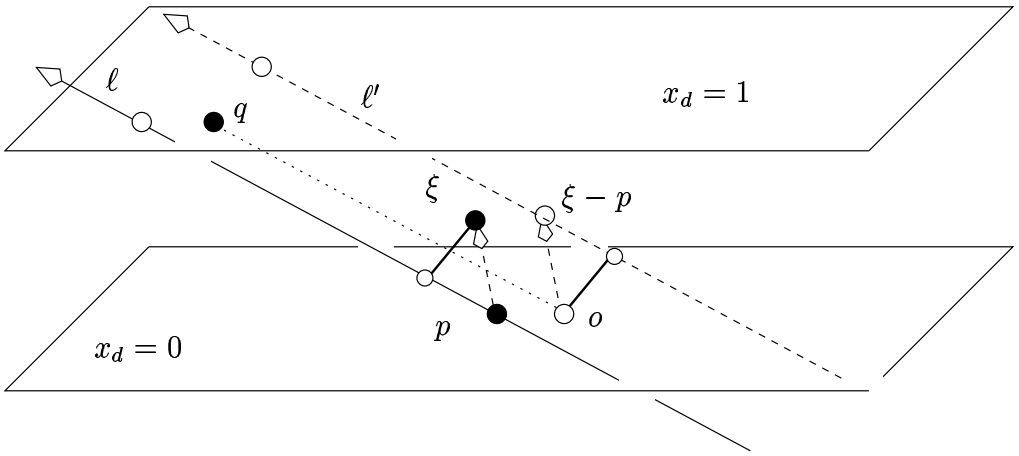


Figure 21.1: Parameterization of a line ℓ in $\mathbb{R}^3$ and its distance from a point p; the small hollow circle on ℓ is the point closest to p.

Theorem 21.1.3 is also from [AHV04]. The improved running time to compute the approximate cylinder mentioned in the text, follows by a more involved algorithm, which together with the construction of the coreset, also compute a compact representation of the extent of the coreset. The technical details are not trivial, and we skip them. In particular, the resulting running time for computing the approximate cylindrical shell is $O(n + 1/\varepsilon^{O(d^2)})$. See [AHV04] for more details.

Computing a compact representation of the extent of hyperplanes. FILL IN

Chapter 22

Approximation Using Shell Sets

> "And so ended Svejk's Budejovice anabasis. It is certain that if Svejk had been granted liberty of movement he would have got to Budejovice on his own. However much the authorities may boast that it was they who brought Svejk to his place of duty, this is nothing but a mistake. With Svejk energy and irresistible desire to fight, the authorities action was like throwing a spanner into the works."
>
> – – The good soldier Svejk, Jaroslav Hasek

22.1 Covering problems, expansion and shell sets

Consider a set P of n points in $\mathbb{R}^{\mathsf{d}}$, that we are interested in covering by the best shape in a family of shapes $\mathcal{F}$. For example, $\mathcal{F}$ might be the set of all balls in $\mathbb{R}^{\mathsf{d}}$, and we are looking for the minimum enclosing ball of P. A ε-coreset $\mathcal{S} \subseteq \mathsf{P}$ would guarantee that *any* ball that covers $\mathcal{S}$ will cover the whole point set if we expand it by $(1+\varepsilon)$.

However, sometimes, computing the coreset is computationally expensive, the coreset does not exist at all, or its size is prohibitively large. It is still natural to look for a small subset $\mathfrak{S}$ of the points, such that finding the optimal solution for $\mathfrak{S}$ generates (after appropriate expansion) an approximate solution to the original problem.

Definition 22.1.1 (Shell sets) Given a set P of points (or geometric objects) in $\mathbb{R}^{\mathsf{d}}$, and $\mathcal{F}$ be a family of shapes in $\mathbb{R}^{\mathsf{d}}$. Let $f : \mathcal{F} \to \mathbb{R}$ be a target optimization function, and assume that there is a natural expansion operation defined over $\mathcal{F}$. Namely, given a set $\mathbf{r} \in \mathcal{F}$, one can compute a set $(1+\varepsilon)\mathbf{r}$ which is the expansion of $\mathbf{r}$ by a factor of $1+\varepsilon$. In particular, we would require that $f((1+\varepsilon)\mathbf{r}) \leq (1+\varepsilon) f(\mathbf{r})$.

Let $f_{\mathrm{opt}}(\mathsf{P}) = \min_{\mathbf{r}\in\mathcal{F}, \mathsf{P}\subseteq\mathbf{r}} f(\mathbf{r})$ be the shape in $\mathcal{F}$ that bests fits P.

Furthermore, assume that $f_{\mathrm{opt}}(\cdot)$ is a monotone function, that is for $A \subseteq B \subseteq \mathsf{P}$ we have $f_{\mathrm{opt}}(A) \leq f_{\mathrm{opt}}(B)$.

A subset $\mathfrak{S} \subseteq \mathsf{P}$ is a ***ε-shell set*** for P, if `SlowAlg` on a set B that contains $\mathfrak{S}$, if the range $\mathbf{r}$ returned by `SlowAlg`$(\mathfrak{S})$ covers $\mathfrak{S}$, $(1+\varepsilon)\mathbf{r}$ covers P, and $f(\mathbf{r}) \leq (1+\varepsilon) f_{\mathrm{opt}}(\mathfrak{S})$. Namely, the range $(1+\varepsilon)\mathbf{r}$ is an $(1+\varepsilon)$-approximation to the optimal range of $\mathcal{F}$ covering P.

A shell set $\mathfrak{S}$ is a ***monotone ε-shell set*** if for any subset B containing $\mathfrak{S}$, if we apply `SlowAlg`(B) and get the range $\mathbf{r}$, then P is contained inside $(1+\varepsilon)\mathbf{r}$ and $\mathbf{r}$ covers B.

Note, that ε-shell sets are considerably more restricted and weaker than coresets. Of course, a ε-coreset is automatically a (monotone) ε-shell set. Note also, that if a problem has a monotone shell set, then to approximate it efficiently, all we need to do is to find some set, hopefully small, that contains the shell set.

22.2 The Setting

Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$, and let $\mathcal{F}$ be a family of shapes in $\mathbb{R}^{\mathsf{d}}$. Furthermore, let us assume that the range space $X = (\mathbb{R}^{\mathsf{d}}, \mathcal{F})$ has low VC dimension $\dim_{\mathsf{VC}}$. Finally, assume that we want to compute the best shape in $\mathcal{F}$ that covers P under a target function $f(\cdot)$. Namely, we would like to compute $f_{\mathrm{opt}}(P) = \min_{\mathbf{r}\in\mathcal{F}, P\subseteq B} f(\mathbf{r})$.

Assume that $f_{\mathrm{opt}}(\cdot)$ is a *monotone target function*. Namely, if $S \subseteq T \subseteq \mathsf{P}$ then $f_{\mathrm{opt}}(S) \leq f_{\mathrm{opt}}(T)$. This monotonicity property holds (almost) always for problems with small coresets.

Next, assume that we only have a slow algorithm `SlowAlg`that can solve (maybe approximately) the given optimization problem. Namely, given a subset $S \subseteq \mathsf{P}$, it computes a range $\mathbf{r} \in \mathcal{F}$ such that $f(\mathbf{r}) \leq (1+\varepsilon) f_{\mathrm{opt}}(S)$, and the running time of `SlowAlg` is $T_{\texttt{SlowAlg}}(|S|)$.

`ComputeShellSet(P)`
We initialize all the points of P to have weight 1, and we repeatedly do the following:

- Pick a random sample R from P of size $r = O((\dim_{\mathsf{VC}}/\delta)\log(\dim_{\mathsf{VC}}/\delta))$, where $\delta = 1/(4k_{\text{opt}})$. With constant probability R is a δ-net for P by Theorem 5.3.4.
- Compute, using `SlowAlg`(R) the range $\mathbf{r}$ in $\mathcal{F}$, such that $(1+\varepsilon)\mathbf{r}$ covers R and realizes (maybe approximately) $f_{\text{opt}}(R)$.
- Compute the set S of all the points of P outside $(1+\varepsilon)\mathbf{r}$. If the total weight of those points exceeds $\delta w(P)$ then the random sample is bad, and return to the first step.
- If the set S is empty then return R as the required shell set, and $\mathbf{r}$ as the approximation.
- Otherwise, double the weight of the points of S.

When done, return $\mathbf{r}$ and the set R.

Figure 22.1: The algorithm for approximating optimal cover and computing a small shell set.

Finally, assume that we know that a small *monotone* shell set of size k_{opt} exists for P, but unfortunately we have no way of computing it explicitly (because, for example, we only have a constructive proof of the existence of such a shell set).

A natural question is how to compute this small shell set quickly, or alternatively compute an approximate shell set which is not much bigger. Clearly, once we have such a small shell set, we can approximate the optimal cover for P in $\mathcal{F}$.

Example. We start with a toy example, a more interesting example is given below. Let $\mathcal{F}$ be the set of all balls in $\mathbb{R}^{\mathrm{d}}$, and let $f(\mathbf{r})$ be the radius of the ball $\mathbf{r} \in \mathcal{F}$. It is known that there is a ε-shell set for the minimum radius ball of size $O(1/\varepsilon)$ (we will prove this fact later in the course). The expansion here is the natural enlargement of a ball radius.

22.3 The Algorithm for Computing the Shell Set

Assume, that a kind oracle, told us that the there exist a monotone ε-shell set for P of size k_{opt}, and that $\mathcal{F}$ is of VC dimension $\dim_{\mathsf{VC}}$. The algorithm to approximate the optimal cover of P and extract a small shell set is depicted in Figure 22.1. Note, that if we do not have a kind oracle at our possession, we can just perform a binary search for the right value of k_{opt}.

There are several non-trivial technicalities in implementing this algorithm The first one is that Theorem 5.3.4 is for unweighted sets, but by replicating a point p of w_p times (conceptually), where w_p is the weight of p, it follows that it still holds in this weighted settings.

Random sampling from a weighted set. Another technicality is that the weights might be quite large. To overcome this, we will store the weight of an element by storing an index i, such that the weight of the element is 2^i, We still need to do m independent draws from this weighted set. The easiest way to do that, is to compute the element e in of P in maximum weight, and observing that all elements of weight $\leq w_e/n^{10}$ have weight which is so tiny, so that it can be ignored, where w_p is the weight of e. Thus, normalize all the weights of by dividing them by $2^{\lfloor \lg w_e/n^{10} \rfloor}$, and remove all elements with weights smaller than 1. For a point p, let $\widehat{\omega}(p)$ denote its normalized weight. Clearly, all the normalized weights are integers in the range $1, \ldots, 2n^{10}$. Thus, we now have to pick points for a set with (small) integer weights. Place the elements in an array, and compute the prefix sum array of their weights. That is $a_k = \sum_{i=1}^{k} \widehat{\omega}(p_i)$, for $i = 1, \ldots, n$. Next, pick a random number γ uniformly in the range $[0, a_n]$, and using a binary search, find the j, such that $a_{j-1} \leq \gamma < a_j$. This picks the points p_j to be in the random sample. This requires $O(n)$ preprocessing, but a single random sample can now be done in $O(\log n)$ time. We need to perform r independent samples. Thus, this takes $O(n + r\log n)$.

22.3.1 Correctness

Lemma 22.3.1 *The algorithm described above computes a ε-shell set for P of size $O(r) = O(k_{\text{opt}}\dim_{\mathsf{VC}}\log(k_{\text{opt}}\dim_{\mathsf{VC}}))$. The algorithm performs $O(4k_{\text{opt}}\ln n)$ iterations.*

Proof: We only need to prove that the algorithm terminates in the claimed number of iterations. Observe, that with constant probability (say ≥ 0.9), the sample R_i, in the ith iteration, is an δ-net for P_{i-1} (the weighted version of P in the end of the $(i-$

1)th iteration), in relation to the ranges of $\mathcal{F}$. Observe, that this also implies that R_i is a δ-net for the complement family $\overline{\mathcal{F}} = \left\{ \mathbb{R}^{\mathsf{d}} \setminus \mathbf{r} \;\middle|\; \mathbf{r} \in \mathcal{F} \right\}$, with constant probability (since $(\mathbb{R}^{\mathsf{d}}, \mathcal{F})$ and $(\mathbb{R}^{\mathsf{d}}, \overline{\mathcal{F}})$ have the same VC dimension).

If R_i is such a δ-net, then we know that range $\mathbf{r}$ we compute completely covers the set R_i, and as such, for any range $\mathbf{r}' \in \overline{\mathcal{F}}$ that avoids R_i we have $w(\mathbf{r}') \leq \delta w(\mathsf{P}_i)$. In particular, this implies that $\omega(S_i) \leq \delta w(\mathsf{P}_{i-1})$. If not, than R_i is not a δ-net, and we resample. The probability for that is ≤ 0.1. As such, we expect to repeat this $O(1)$ times in each iteration, till we have $w(S_i) \leq \delta w(\mathsf{P}_{i-1})$.

Thus, in each iteration, the algorithm doubles the weight of at most a δ-fraction of the total point set. Thus $w(\mathsf{P}_i) \leq (1+\delta) w(\mathsf{P}_{i-1}) = n(1+\delta)^i$.

On the other hand, consider the smallest shell $\mathfrak{S}$ of P, which is of size k_{opt}. If all the elements of $\mathfrak{S}$ are in R_i, then the algorithm would have terminated, since $\mathfrak{S}$ is a monotone shell set1. Thus, if we continue to the next iteration, it must be that $|\mathfrak{S} \cap S_i| \geq 1$. In particular, we are doubling the weight of at least one element of the shell set. We conclude that the weight of P_i in the ith iteration, is at least

$$k_{\mathrm{opt}} 2^{i/k_{\mathrm{opt}}},$$

since in every iteration at least one element of $\mathfrak{S}$ gets its weight redoubled. Thus, we have

$$\exp\left(\frac{i}{2k_{\mathrm{opt}}}\right) \leq 2^{i/k_{\mathrm{opt}}} \leq k_{\mathrm{opt}} 2^{i/k_{\mathrm{opt}}} \leq (1+\delta)^i n \leq n \cdot \exp(\delta i) = n \cdot \exp\left(\frac{i}{4k_{\mathrm{opt}}}\right).$$

Namely, $\exp\left(\frac{i}{4k_{\mathrm{opt}}}\right) \leq n$. Implying that $i \leq 4k_{\mathrm{opt}} \ln n$. Namely, after $4k_{\mathrm{opt}} \ln n$ iterations the algorithm terminates, and thus returns the required shell set and approximation. ■

Theorem 22.3.2 *Under the settings of Section 22.2, one can compute a monotone ε-shell set for P of size $O(k_{\mathrm{opt}} \dim_{\mathsf{VC}} \log(k_{\mathrm{opt}} \dim_{\mathsf{VC}}))$. The running time of the resulting algorithm is $O\Big((n + T(k_{\mathrm{opt}} \dim_{\mathsf{VC}} \log(k_{\mathrm{opt}} \dim_{\mathsf{VC}}))) k_{\mathrm{opt}} \ln n\Big)$, with high probability, for $k_{\mathrm{opt}} \leq n/\log^3 n$. Furthermore, one can compute an ε-approximation to $f_{\mathrm{opt}}(P)$ in the same time bounds.*

Proof: The algorithm is described above. The bounds on the running time follows from the bounds on the number of iterations from Lemma 22.3.1. The only problem we need to address, is that the resampling would repeatedly fail, and the algorithm would spend exuberant amount of time on resampling. However, the probability of failure in sampling is ≤ 0.1. Furthermore, we need at most $4k_{\mathrm{opt}} \log n$ good samples before the algorithm succeeds. It is now straightforward to show using Chernoff inequality, that with high probability, we will perform at most $8k_{\mathrm{opt}} \log n$ samplings before achieving the required number of good samples. ■

22.3.2 Set Covering in Geometric Settings

Interestingly, the algorithm we discussed, can be used to get an improved approximation algorithm for the set covering problem in geometric settings. We remind the reader that set covering is the following problem.

Problem: Set Covering

Instance: $(S, \mathcal{F})$
S - a set of n elements
$\mathcal{F}$ - a family of subsets of S, s.t. $\bigcup_{X \in \mathcal{F}} X = S$.
Question: What is the set $\mathcal{X} \subseteq \mathcal{F}$ such that $\mathcal{X}$ contains as few sets as possible, and $\mathcal{X}$ covers S?

The natural algorithm for this problem is the greedy algorithm that repeatedly pick the set in the family $\mathcal{F}$ that covers the largest number of uncovered elements in S. It is not hard to show that this provides a $O(|S|)$ approximation. In fact, it is known that set covering can be better approximated unless $P = NP$.

Assume, however, that we know that the VC dimension of the set system $(S, \mathcal{F})$ has VC dimension $\dim_{\mathsf{VC}}$. In fact, we need a stronger fact, that the dual family

$$\mathsf{S} = \left(\mathcal{F}, \left\{ U(s, \mathcal{F}) \;\middle|\; s \in S \right\}\right),$$

is of low VC dimension $\dim_{\mathsf{VC}}$, where $U(s, \mathcal{F}) = \left\{ X \;\middle|\; s \in X, X \in \mathcal{F} \right\}$.

It turns out that the algorithm of Figure 22.1 also works in this setting. Indeed, we set the weight of the sets to 1, we pick a random sample of sets. IF they cover the universe S, we are done. Otherwise, there must be a point p which is not covered. Arguing as above, we know that the random sample is a δ-net of (the weighted) S, and as such all the sets containing p have total weight $\leq \delta(\mathsf{S})$. As such, double the weight of all the sets covering p, and repeat. Arguing as above, one can show that the algorithm terminates after $O(k_{\mathrm{opt}} \log m)$ iterations, where m is the number of sets, where k_{opt} is the number of sets in the optimal cover of S. Furthermore, the size of the cover generated is $O(k_{\mathrm{opt}} \dim_{\mathsf{VC}} \log(k_{\mathrm{opt}} \dim_{\mathsf{VC}}))$.

Theorem 22.3.3 *Let $(S, \mathcal{F})$ be a range space, such that the dual range space S has VC dimension $\dim_{\mathsf{VC}}$. Then, one can compute a set covering for S using sets of $\mathcal{F}$ using $O(k_{\text{opt}}\dim_{\mathsf{VC}} \log(k_{\text{opt}}\dim_{\mathsf{VC}}))$ sets. This requires $O(k_{\text{opt}} \log n)$ iterations, and takes polynomial time.*

Note, that we did not provide in Theorem 22.3.3 exact running time bounds. Usually in geometric settings, one can get improved running time using the underlying geometry. Interestingly, the property that the dual system has low VC dimension "buys" one a lot, as it implies that one can do $O(\log k_{\text{opt}})$ approximation, instead of $O(\log n)$ in the general case.

22.4 Application - Covering Points by Cylinders

22.5 Clustering and Coresets

We would like to cover a set P of n points $\mathbb{R}^{\mathsf{d}}$ by k balls, such that the radius of maximum radius ball is minimized. This is known as the *k-center clustering* problem (or just *k-center*). The *price function*, in this case, $\mathrm{rd}_k(P)$ is the radius of the maximum radius ball in the optimal solution.

Definition 22.5.1 Let P be a point set in $\mathbb{R}^{\mathsf{d}}$, $1/2 > \varepsilon > 0$ a parameter.

For a cluster c, let $c(\delta)$ denote the cluster resulting form expanding c by δ. Thus, if c is a ball of radius r, then $c(\delta)$ is a ball of radius $r + \delta$. For a set C of clusters, let

$$C(\delta) = \left\{ c(\delta) \;\middle|\; c \in C \right\},$$

be the *additive expansion operator*; that is, $C(\delta)$ is a set of clusters resulting form expanding each cluster of C by δ.

Similarly,

$$(1+\varepsilon)C = \left\{ (1+\varepsilon)c \;\middle|\; c \in C \right\},$$

is the *multiplicative expansion operator*, where $(1+\varepsilon)c$ is the cluster resulting from expanding c by a factor of $(1+\varepsilon)$. Namely, if C is a set of balls, then $(1+\varepsilon)C$ is a set of balls, where a ball $c \in C$, corresponds to a ball radius $(1+\varepsilon)\,\mathrm{radius}(c)$ in $(1+\varepsilon)C$.

A set $\mathcal{S} \subseteq \mathsf{P}$ is an (additive) *ε-coreset* of P, in relation to a price function radius, if for any clustering C of $\mathcal{S}$, we have that P is covered by $C(\varepsilon\,\mathrm{radius}(C))$, where $\mathrm{radius}(C) = \max_{c \in C} \mathrm{radius}(c)$. Namely, we expand every cluster in the clustering by an ε-fraction of the size of the *largest* cluster in the clustering. Thus, if C is a set of k balls, then $C(\varepsilon f(C))$ is just the set of balls resulting from expanding each ball by εr, where r is the radius of the largest ball.

A set $\mathcal{S} \subseteq \mathsf{P}$ is a *multiplicative ε-coreset* of P, if for any clustering C of $\mathcal{S}$, we have that P is covered by $(1+\varepsilon)C$.

Lemma 22.5.2 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$, and $\varepsilon > 0$ a parameter. There exists an additive ε-coreset for the k-center problem, and this coreset has $O(k/\varepsilon^{\mathsf{d}})$ points.*

Proof: Let C denote the optimal clustering of P. Cover each ball of C by a grid of side length $\varepsilon r_{\text{opt}}/d$, where r_{opt} is the radius of the optimal k-center clustering of P. From each such grid cell, pick one points of P. Clearly, the resulting point set $\mathcal{S}$ is of size $O(k/\varepsilon^{\mathsf{d}})$ and it is an additive coreset of P. ■

The following is a minor extension of an argument used in [APV02].

Lemma 22.5.3 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$, and $\varepsilon > 0$ a parameter. There exists a multiplicative ε-coreset for the k-center problem, and this coreset has $O\left(k!/\varepsilon^{dk}\right)$ points.*

Proof: For $k = 1$, the additive coreset of P is also a multiplicative coreset, and it is of size $O(1/\varepsilon^{\mathsf{d}})$.

As in the proof of Lemma 22.5.2, we cover the point set by a grid of radius $\varepsilon r_{\text{opt}}/(5d)$, let SQ the set of cells (i.e., cubes) of this grid which contains points of P. Clearly, $|\mathrm{SQ}| = O(k/\varepsilon^{\mathsf{d}})$.

Let $\mathcal{S}$ be the additive ε-coreset of P. Let C be any k-center clustering of $\mathcal{S}$, and let Δ be any cell of SQ.

If Δ intersects all the k balls of C, then one of them must be of radius at least $(1-\varepsilon/2)\mathrm{rd}(P,k)$. Let c be this ball. Clearly, when we expand c by a factor of $(1+\varepsilon)$ it would completely cover Δ, and as such it would also cover all the points of $\Delta \cap \mathsf{P}$.

Thus, we can assume that Δ intersects at most $k-1$ balls of C. As such, we can inductively compute an ε-multiplicative coreset of $P \cap \Delta$, for $k-1$ balls. Let Q_Δ be this set, and let $\mathcal{Q} = \mathcal{S} \cup \bigcup_{\Delta \in \mathrm{SQ}} Q_\Delta$.

Note that $|\mathcal{Q}| = T(k,\varepsilon) = O(k/\varepsilon^d)T(k-1,\varepsilon) + O(k/\varepsilon^{\mathsf{d}}) = O\left(k!/\varepsilon^{dk}\right)$. The set $\mathcal{Q}$ is the required multiplicative coreset by the above argumentation. ■

22.6 Union of Cylinders

Let assume we want to cover P by k cylinders of minimum maximum radius (i.e., fit the points to k lines). Formally, consider $\mathcal{G}$ to be the set of all cylinders in $\mathbb{R}^{\mathsf{d}}$, and let $\mathcal{F} = \left\{ c_1 \cup c_2 \cup \ldots \cup c_k \;\middle|\; c_1, \ldots, c_k \in \mathcal{F} \right\}$ be the set, which its members are union of k cylinders. For $C \in \mathcal{F}$, let $f(C) = \max_{c \in C} \text{radius}(c)$. Let $f_{\text{opt}}(P) = \min_{C \in \mathcal{F}, P \subseteq C} f(C)$.

One can compute the optimal cover of P by k cylinders in $O(n^{(2\mathsf{d}-1)k+1})$ time, see below for details. Furthermore, $(\mathbb{R}^{\mathsf{d}}, \mathcal{F})$ has VC dimension $\dim_{\mathsf{VC}} = O(dk \log(dk))$. Finally, one can show that this set of cylinders has ε-coreset of small size ???. Thus, we would like to compute a small ε-coreset, and compute an approximation quickly.

22.6.0.1 Covering by Cylinders - A Slow Algorithm

It is easy to verify (but tedious) that the VC dimension of $(\mathbb{R}^{\mathsf{d}}, \mathcal{F}_k)$ is bounded by $\dim_{\mathsf{VC}} = O(dk \log(dk))$. Furthermore, it has a small coreset Furthermore, given a set P of n points, and consider its minimum radius enclosing cylinder c. The cylinder c has (at most) $2\mathsf{d} - 2$ points of P on its boundary which if we compute their minimum enclosing cylinder, it is c. Note, that c might contain even more points on its boundary, we are only claiming that there is a defining subset of size $2\mathsf{d} - 1$. This is one of those "easy to see" but very tedious to verify facts. Let us quickly outline an intuitive explanation (but not a proof!) of this fact. Consider the set of lines $\mathcal{L}^{\mathsf{d}}$ of lines in $\mathbb{R}^{\mathsf{d}}$. Every member of $\ell \in \mathcal{L}^{\mathsf{d}}$ can be parameterized by the closest point p on ℓ to the origin, and consider the hyperplane that passes through p and is orthogonal to op, where o is the origin. The line ℓ now can be parameterized by its orientation in h. This requires specifying a point on the $\mathsf{d} - 2$ dimensional unit hypersphere $\mathbb{S}^{(\mathsf{d}-2)}$. Thus, we can specify ℓ using $2\mathsf{d} - 2$ real numbers. Next, define for each point $p_i \in \mathsf{P}$, its distance $g_i(\ell)$ from $\ell \in \mathcal{L}^{\mathsf{d}}$. This is a messy but a nice algebraic function defined over $2\mathsf{d} - 2$ variables. In particular, $g_i(\ell)$ induces a surface in $2\mathsf{d} - 1$ dimensions (i.e., $\cup_\ell(\ell, g_i(\ell))$. Consider the arrangement $\mathcal{A}$ of those surfaces. Clearly, the minimum volume cylinder lies on a feature of this arrangement, thus to specify the minimum radius cylinder, we just need to specify the feature (i.e., vertex, edge, etc) of the arrangement that contains this point. However, every feature in an arrangement of well behaved surfaces in $2\mathsf{d} - 1$ dimensions, can be specified by $2\mathsf{d} - 1$ surfaces. (This is intuitively clear but requires a proof - an intersection of k surfaces, is going to be $d - k$ dimensional, where d is the dimension of the surfaces. If we add a surface to the intersection and it does not reduce the dimension of the intersection, we can reject it, and take the next surface passing through the feature we care about.). Every such surface corresponds to a original point.

Thus, if we want to specify a minimum radius cylinder induced by a subset of P, all we need to specify are $2\mathsf{d} - 1$ points. To specify k such cylinders, we need to specify $M = (2\mathsf{d} - 1)k$ points. This immediately implies that we can find the optimal cover of P by k cylinders in $O(n^{(2\mathsf{d}-1)k+1})$ time, but just enumerating all such subsets of M points, and computing for each subset its optimal cover (note, that the O notation hides a constant that depends on k and d).

Thus, we have a slow algorithm that can compute the optimal cover of P by k cylinders.

22.6.1 Existence of a Small Coreset

Since the coreset in this case is either multiplicative or additive, it is first important to define the expansion operation carefully. In particular, if C is a set of k cylinders, the $(1 + \varepsilon)$-expanded set of cylinders would be $C(\varepsilon \,\text{radius}(C))$, where $\text{radius}(C)$ is the radius of the largest cylinder in C.

Let P be the given set of n points in $\mathbb{R}^{\mathsf{d}}$. Let C_{opt} be the optimal cover of P by k cylinders. For each cylinder of C_{opt} place $O(1/\varepsilon^{\mathsf{d}-1})$ parallel lines inside it, so that for any point inside the union of the cylinders, there is a line in this family in distance $\leq (\varepsilon/10) r_{\text{opt}}$ from it. Let L denote this set of lines.

Let Q be the point set resulting from snapping each point of P to its closest point on L. We claim that Q is a $(\varepsilon/10)$-coreset for P, as can be easily verified. Indeed, if a set C of k cylinders cover Q, then the largest cylinder must be of radius $r \geq (1 - \varepsilon/10)\text{rd}(\mathsf{P}, k)$, where $\text{rd}(\mathsf{P}, k)$ is the radius of the optimal coverage of P by k cylinders. Otherwise, $\text{rd}(\mathsf{P}, k) \leq r + (\varepsilon/10)\text{rd}(\mathsf{P}, k) < \text{rd}(\mathsf{P}, k)$.

The set Q lies on $O(1/\varepsilon^{\mathsf{d}-1})$-lines. Let $\ell \in \mathsf{L}$ be such a line, and consider the point-set Q_ℓ. Assume for a second that there was a multiplicative ε-coreset $\mathcal{T}_\ell$ on this line. If $\mathcal{T}_\ell$ is covered by k cylinders, each cylinder intersect ℓ along an interval. Expanding each such cylinder by a factor of $(1 + \varepsilon)$ is equivalent to expanding each such intersecting interval by a factor of $1 + \varepsilon$. However, by Lemma 22.5.3, we know that such a multiplicative $(\varepsilon/10)$-coreset exists, of size $O(1/\varepsilon^k)$. Thus, let $\mathcal{T}_\ell$ be the multiplicative $(\varepsilon/10)$-coreset for Q_ℓ for k intervals on the line. Let $\mathcal{T} = \cup_{\ell \in \mathsf{L}} \mathcal{T}_\ell$. We claim that $\mathcal{T}$ is a (additive) $(\varepsilon/10)$-coreset for Q. This is trivial, since being a multiplicative coreset for each line implies that the union is a multiplicative coreset, and a δ-multiplicative coreset is also a δ-additive coreset. Thus, $\mathcal{T}$ is a $((1 + \varepsilon/10)^2 - 1)$-coreset for P. The only problem is that the points of $\mathcal{T}$ are not points in P. However, they corresponds to points in P which are in distance at most $(\varepsilon/10)\text{rd}(\mathsf{P}, k)$ from them. Let $\mathcal{S}$ be the corresponding set of points of P. It is now easy to verify that $\mathcal{S}$ is indeed a ε-coreset for P, since $((1 + \varepsilon/10)^2 - 1) + \varepsilon/10 \leq \varepsilon$. We summarize:

Theorem 22.6.1 *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$. There exists a (additive) ε-coreset for P of size $O(k/\varepsilon^{\mathsf{d}-1+k})$ for covering the points by k-cylinders of minimum radius.*

22.7 Bibliographical notes

Section 22.3.2 is due to Clarkson [Cla93]. This technique was used to approximate terrains [AD97], and covering polytopes [Cla93].

The observation that this argument can be used to speedup approximation algorithms is due to Agarwal *et al.* [APV02]. The discussion of shell sets is implicit in the work of Bădoiu *et al.* [BHI02].

Chapter 23

Duality

> I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.
>
> – – Jerome K. Jerome, Three men in a boat

Duality is a transformation that maps lines and points into points and lines, respectively, while preserving some properties in the process. Despite its relative simplicity, it is a powerful tool that can dualize what seems like "hard" problems into easy dual problems.

23.1 Duality of lines and points

Consider a line $\ell \equiv y = ax + b$ in two dimensions. It is being parameterized by two constants a and b, which we can interpret, paired together, as a point in the parametric space of the lines. Naturally, this also gives us a way of interpreting a point as defining coefficients of a line. Thus, conceptually, points are lines and lines are points.

Formally, the ***dual point*** to the line $\ell \equiv y = ax + b$ is the point $\ell^\star = (a, -b)$. Similarly, for a point $p = (c, d)$ its ***dual line*** is $p^\star = cx - d$. Namely,

$$\begin{aligned} \mathsf{p} = (a, b) \quad &\Rightarrow \quad \mathsf{p}^\star : y = ax - b \\ \ell : y = cx + d \quad &\Rightarrow \quad \ell^\star = (c, -d). \end{aligned}$$

We will consider a line $\ell \equiv y = cx + d$ to be a linear function in one dimension, and let $\ell(x) = cx + d$.

A point $\mathsf{p} = (a, b)$ lies ***above*** a line $\ell \equiv y = cx + d$ if p lies vertically above ℓ. Formally, we have that $b > \ell(a) = ca + d$. We will denote this fact by $\mathsf{p} > \ell$. Similarly, the point p lies ***below*** ℓ if $b < \ell(a) = ca + d$, denoted by $\mathsf{p} < \ell$.

A line ℓ ***supports*** a convex set $S \subseteq \mathbb{R}^2$ if it intersects S but the interior of S lies completely on one side of ℓ.

Basic properties. For a point $\mathsf{p} = (a, b)$ and a line $\ell \equiv y = cx + d$, we have:

(P1) $\mathsf{p}^{\star\star} = (\mathsf{p}^\star)^\star = \mathsf{p}$.

> Indeed, $\mathsf{p}^\star \equiv y = ax - b$ and $(\mathsf{p}^\star)^\star = (a, -(-b)) = \mathsf{p}$.

(P2) The point p lies above (resp. below, on) the line ℓ, if and only if the point $\ell^\star$ lies above (resp. below, on) the line $\mathsf{p}^\star$. (Namely, a point and a line change their vertical ordering in the dual.)

> Indeed, $\mathsf{p} > \ell(a)$ if and only if $b > ca + d$. Similarly, $(c, -d) = \ell^\star > \mathsf{p}^\star \equiv y = ax - b$ if and only if $-d > ac - b$, which is equivalent to the above condition.

(P3) The vertical distance between p and ℓ is the same as that between $\mathsf{p}^\star$ and $\ell^\star$.

> Indeed, the vertical distance between p and ℓ is $|b - \ell(a)| = |b - (ca + d)|$. The vertical distance between $\ell^\star = (c, -d)$ and $\mathsf{p}^\star \equiv y = ax - b$ is $|(-d) - \mathsf{p}^\star(c)| = |-d - (ac - b)| = |b - (ca + d)|$.

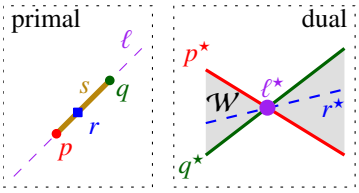
(P4) The vertical distance $\delta(\ell, \hbar)$ between two parallel lines ℓ and $\hbar \equiv y = ax + e$ is the same as the length of the vertical segment $\ell^\star \hbar^\star$.

> The vertical distance between ℓ and $\hbar$ is $|b - e|$. Similarly, since $\ell^\star = (a, -b)$ and $\hbar^\star = (a, -e)$ we have that the vertical distance between them is $|(-b) - (-e)| = |b - e|$.

The missing lines. Consider the vertical line $\ell \equiv x = 0$. Clearly, ℓ does not have a dual point (specifically, its hypothetical dual point has an x coordinate with infinite value). In particular, our duality can not handle vertical lines. To visualize the problem, consider a sequence of non-vertical lines ℓ_i that converges to a vertical line ℓ. The sequence of dual points $\ell_i^\star$ is a sequence of points that diverges to infinity.

23.1.1 Examples

23.1.1.1 Segments and Wedges

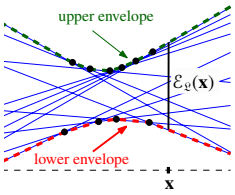


Consider a segment $s = \mathsf{pq}$ that lies on a line ℓ. Observe, that the dual of a point $\mathsf{r} \in \ell$ is a line $\mathsf{r}^\star$ that passes through the point $\ell^\star$. In fact, the two lines $\mathsf{p}^\star$ and $\mathsf{q}^\star$ define two double wedges. Let $\mathcal{W}$ be the double wedge that does not contain the vertical line that passes through $\ell^\star$.

Consider now the point r as it moves along s. When it is equal to p then its dual line $\mathsf{r}^\star$ is the line $\mathsf{p}^\star$. Now, as r moves along the segment s the dual line $\mathsf{r}^\star$ rotates around $\ell^\star$, till it arrives to $\mathsf{q}^\star$ (and then r reaches q).

What about the other wedge? It represents the two rays forming $\ell \setminus s$. The vertical line through $\ell^\star$ represents the singularity point in infinity where the two rays are "connected" together. Thus, as r travels along one of the rays (say starting at q) of $\ell \setminus s$, the dual line $\mathsf{r}^\star$ becomes steeper and steeper, till it becomes vertical. Now, the point r "jumps" from the "infinite endpoint" of the ray, to the "infinite endpoint" of the other ray. Simultaneously, the line $\mathsf{r}^\star$ is continuing to rotate from its current vertical position, sweeping over the whole wedge, till r travels back to p. (The reader that feels uncomfortable with notions line "infinite endpoint" can rest assured that the author feels the same way. As such, this should be taken as an intuitive description of whats going on and not a formally correct one.)

23.1.1.2 Convex hull and upper/lower envelopes



Consider a set of lines L in the plane. The minimization diagram of L, known as the ***lower envelope*** of L, is the function $\mathcal{L}_\mathsf{L} : \mathbb{R} \to \mathbb{R}$, where we have $\mathcal{L}(x) = \min_{\ell \in \mathsf{L}} \ell(x)$, for $x \in \mathbb{R}$. Similarly, the ***upper envelope*** of L is the function $\mathcal{U}(x) = \max_{\ell \in \mathsf{L}} \ell(x)$, for $x \in \mathbb{R}$. The ***extent*** of L at $x \in \mathbb{R}$ is the vertical distance between the upper and lower envelope at x; namely, $\mathcal{E}_\mathsf{L}(x) = \mathcal{U}(x) - \mathcal{L}(x)$.

Computing the lower and/or upper envelopes can be useful. A line might represent a linear constraint, where the feasible solution must lie above this line. Thus, the feasible region is the region of points that lie above all the given lines. Namely, the region of the feasible solution is defined by the upper envelope of the lines. The upper envelope is just a polygonal chain made out of two infinite rays and a sequence of segments, where each segment/ray lies on one of the given lines. As such, the upper envelop can be described as the sequence of lines appearing on it, and the vertices where they change.

Developing an efficient algorithm for computing the upper envelope of a set of lines is a tedious but doable task. However, it becomes trivial if one uses duality.

Lemma 23.1.1 *Let L be a set of lines in the plane. Let $\alpha \in \mathbb{R}$ be an any number, $\beta^- = \mathcal{L}_\mathsf{L}(\alpha)$ and $\beta^+ = \mathcal{U}_\mathsf{L}(\alpha)$. Let $\mathsf{p} = (\alpha, \beta^-)$ and $\mathsf{q} = (\alpha, \beta^+)$. Then:*

(i) the dual lines $\mathsf{p}^\star$ and $\mathsf{q}^\star$ are parallel, and they are both perpendicular to the direction $(\alpha, -1)$.

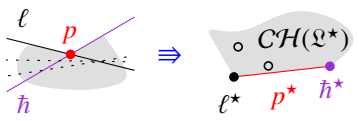
(ii) The lines $\mathsf{p}^\star$ and $\mathsf{q}^\star$ support $\mathcal{CH}(\mathsf{L}^\star)$.

(iii) The extent $\mathcal{E}_\mathsf{L}(\alpha)$ is the vertical distance between the lines $\mathsf{p}^\star$ and $\mathsf{q}^\star$.

Proof: (i) We have $\mathsf{p}^\star \equiv y = \alpha x - \beta^-$ and $\mathsf{q}^\star \equiv y = \alpha x - \beta^+$. These two lines are parallel since they have the same slope. In particular, they are parallel to the direction $(1, \alpha)$. But this direction is perpendicular to the direction $(\alpha, -1)$.

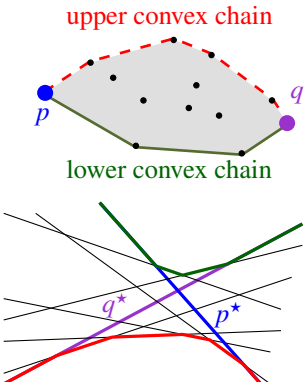
(ii) By property (P2), we have that all the points of $\mathsf{L}^\star$ are below (or on) the line $\mathsf{p}^\star$. Furthermore, since p is on the lower envelope of L it follows that $\mathsf{p}^\star$ must pass through one of the points $\mathsf{L}^\star$. Namely, $\mathsf{p}^\star$ supports $\mathcal{CH}(\mathsf{L}^\star)$ and it lies above it. Similar argument applies to $\mathsf{q}^\star$.

(iii) We have that $\mathcal{E}_\mathsf{L}(\alpha) = \beta^+ - \beta^-$. The vertical distance between the two parallel lines $\mathsf{p}^\star$ and $\mathsf{q}^\star$ is $\mathsf{q}^\star(0) - \mathsf{p}^\star(0) = -\beta^+ - (-\beta^-) = \beta^+ - \beta^-$, as required. ■



Thus, consider a vertex p of the upper envelope of the set of lines L. The point p is the intersection point of two lines ℓ and $\hbar$ of L. Consider the dual set of points $\mathsf{L}^\star$ and the dual line $\mathsf{p}^\star$. Since p lies above (or on) all the lines of L, by the above discussion, it must be that the line $\mathsf{p}^\star$ lies below (or on) all the points of $\mathsf{L}^\star$. On the other hand, the line $\mathsf{p}^\star$ passes through the two points $\ell^\star$ and $\hbar^\star$. Namely, $\mathsf{p}^\star$ is a line that supports the convex hull of $\mathsf{L}^\star$ and it passes through two of its vertices.

The convex hull of $\mathsf{L}^\star$ is a convex polygon $\mathcal{P}$, which can be broken into two convex chains by breaking it at the two extreme points in the x direction. We will refer to the upper polygonal chain of the convex hull as ***upper convex chain*** and to the other one as ***lower convex chain***. In particular, two consecutive segments of the upper envelope corresponds to two consecutive vertices on the lower chain of the convex hull of $\mathsf{L}^\star$. Thus, the convex-hull of $\mathsf{L}^\star$ can be decomposed into two chains. The lower chain corresponds to the upper envelope of L, and the upper chain corresponds to the lower envelope of L. Of special interest are the two x extreme points p and q of the convex hull. They are the dual of the two lines with the highest/smallest slope in L (we are assuming here that the slopes of lines in L are distinct). These two lines appear on both the upper and lower envelope of the lines and they contain the four infinite rays of these envelopes.



Lemma 23.1.2 *Given a set L of n lines in the plane, one can compute its lower and upper envelopes in $O(n \log n)$ time.*

Proof: One can compute the convex hull of n points in the plane in $O(n \log n)$ time. Thus, computing the convex hull of $\mathsf{L}^\star$ and dualizing the upper and lower chains of $\mathcal{CH}(\mathsf{L}^\star)$ results in the required envelopes. ■

23.2 Higher Dimensions

The above discussion can be easily extended to higher dimensions. We provide the basic properties without further proof, since they are easy extension of the two dimensional case. A hyperplane $h : x_\mathsf{d} = b_1 x_1 + \cdots + b_{\mathsf{d}-1} x_{\mathsf{d}-1} - b_\mathsf{d}$ in $\mathbb{R}^\mathsf{d}$ can be interpreted as a function from $\mathbb{R}^{\mathsf{d}-1}$ to $\mathbb{R}$. Given a point $\mathsf{p} = (p_1, \ldots, p_\mathsf{d})$ let $h(\mathsf{p}) = b_1 p_1 + \cdots + b_{\mathsf{d}-1} p_{\mathsf{d}-1} - b_\mathsf{d}$. In particular, a point p lies *above* the hyperplane h if $p_\mathsf{d} > h(\mathsf{p})$. Similarly, p is *below* the hyperplane h if $p_\mathsf{d} < h(\mathsf{p})$. Finally, a point is on the hyperplane if $h(\mathsf{p}) = p_\mathsf{d}$.

The ***dual*** of a point $\mathsf{p} = (p_1, \ldots, p_\mathsf{d}) \in \mathbb{R}^\mathsf{d}$ is a hyperplane $\mathsf{p}^\star \equiv x_\mathsf{d} = p_1 x_1 + \cdots p_{\mathsf{d}-1} x_{\mathsf{d}-1} - p_\mathsf{d}$, and the *dual* of a hyperplane $h \equiv x_\mathsf{d} = a_1 x_1 + a_2 x_2 + \cdots + a_{\mathsf{d}-1} x_{\mathsf{d}-1} + a_\mathsf{d}$ is the point $h^\star = (a_1, \ldots, a_{\mathsf{d}-1}, -a_\mathsf{d})$. There are several alternative definitions of duality, but they are essentially similar. Summarizing:

$$\begin{aligned} \mathsf{p} = (p_1, \ldots, p_\mathsf{d}) \quad &\Rightarrow \quad \mathsf{p}^\star \equiv x_\mathsf{d} = p_1 x_1 + \cdots p_{\mathsf{d}-1} x_{\mathsf{d}-1} - p_\mathsf{d} \\ h \equiv x_\mathsf{d} = a_1 x_1 + a_2 x_2 + \cdots + a_{\mathsf{d}-1} x_{\mathsf{d}-1} + a_\mathsf{d} \quad &\Rightarrow \quad h^\star = (a_1, \ldots, a_{\mathsf{d}-1}, -a_\mathsf{d}). \end{aligned}$$

In the following we would slightly abuse notations, and for a point $\mathsf{p} \in \mathbb{R}^\mathsf{d}$ we will refer to $(p_1, \ldots, p_{\mathsf{d}-1}, \mathcal{L}_\mathcal{H}(\mathsf{p}))$ as the point $\mathcal{L}_\mathcal{H}(\mathsf{p})$. Similarly, $\mathcal{U}_\mathcal{H}(\mathsf{p})$ would denote the corresponding point on the upper envelope of $\mathcal{H}$.

The proof of the following lemma is an easy extension of the 2d case.

Lemma 23.2.1 *For a point $\mathsf{p} = (b_1, \ldots, b_\mathsf{d})$, we have:*

(i) $\mathsf{p}^{\star\star} = \mathsf{p}$.

(ii) *The point p lies above (resp. below, on) the hyperplane h, if and only if the point $h^\star$ lies above (resp. below, on) the hyperplane $h^\star$.*

(iii) *The vertical distance between p and h is the same as that between $\mathsf{p}^\star$ and $h^\star$.*

(iv) *The vertical distance $\delta(h, g)$ between two parallel hyperplanes h and g is the same as the length of the vertical segment $h^\star g^\star$.*

(v) *Let $\mathcal{H}$ be the set of hyperplanes in $\mathbb{R}^d$. For any $x \in \mathbb{R}^{\mathsf{d}-1}$, the hyperplanes h and g dual to the points $\mathcal{L}_\mathcal{H}(\mathsf{p})$ and $\mathcal{U}_\mathcal{H}(\mathsf{p})$, respectively, are parallel, normal to the vector $(\mathsf{p}, -1) \in \mathbb{R}^\mathsf{d}$, and supports the set $\mathcal{CH}(\mathcal{H}^\star)$. Furthermore, the points of $\mathcal{H}^\star$ lies below (resp., above) the hyperplane h (resp., above g).*
Also, $\mathcal{E}_\mathcal{H}(\mathsf{p})$ is the vertical distance between h and g.

(vi) *Computing the lower and upper envelope of $\mathcal{H}$ is equivalent to computing the convex hull of the dual set of points $\mathcal{H}^\star$.*

23.3 Exercises

Exercise 23.3.1 Prove Lemma 23.2.1

Exercise 23.3.2 Show a counter example proving that no duality can preserve (exactly) orthogonal distances between points and lines.

23.4 Bibliographical notes

The duality discussed here should not be confused with linear programming duality [Van97]. Although the two topics seems to be connected somehow, the author is unaware of a natural and easy connection.

A natural question is whether one can find a duality that preserves the orthogonal distances between lines and points. The surprising answer is no, as Exercise 23.3.2 testifies. In fact, it is not too hard to show using topological arguments that any duality must distort such distances arbitrarily bad [FH06].

Open Problem 23.4.1 *Given a set* P *of n points in the plane, and a set* L *of n lines in the plane, consider the best possible duality (i.e., the one that minimizes the distortion of orthogonal distances) for* P *and* L*. What is the best distortion possible, as a function of n?*

Here, we define the distortion of the duality as

$$\max_{\mathsf{p}\in\mathsf{P},\ell\in\mathsf{L}} \left(\frac{\mathbf{d}(\mathsf{p},\ell)}{\mathbf{d}(\mathsf{p}^\star,\ell^\star)}, \frac{\mathbf{d}(\mathsf{p}^\star,\ell^\star)}{\mathbf{d}(\mathsf{p},\ell)} \right).$$

A striking (negative) example of the power of duality is the work of Overmars and van Leeuwen [OvL81] on the dynamic maintenance of convex hull in 2d, and the maintenance of the lower/upper envelope of lines in the plane. Clearly, by duality, the two problems are identical. However, the authors (smart people indeed) did not observe it, and the paper is twice longer than it should be solving the two problems separately.

Duality is heavily used throughout computational geometry, and it is hard to imagine managing without it. Results and techniques that use duality include bounds on k-sets/k-levels [Dey98], partition trees [Mat92], and coresets for extent measure [AHV04] (this is a random short list of relevant results and it is by no means exhaustive).

23.4.1 Projective geometry and duality

The "missing lines phenomena" encountered above is inherent to all dualities, since the space of a lines in the plane has the topology of an open Möbius strip which is not homeomorphic to the plane. There are a lot of other possible dualities, and the one presented here is the one most useful for our purposes.

One way to overcome this, is to add an extra coordinate. Now a point is represented by a triplet (w, x, y) represents the planar point $(x/w, y/w)$ (thus a point no longer has a unique representation). Thus, the triplets $(1, 1, 1)$ and $(2, 2, 2)$ represent the same point. In fact, a point is essentially a line in three dimensions. Similarly, a line is now represented by a triplet $\langle A, B, C\rangle$ which is the plane $Aw + Bx + Cy = 0$ that passes through the origin. Duality is now defined in a very natural way. Geometrically, we can interpret a plane as a point on the sphere, and a point by its representative point on the sphere. Note, that two antipodal points are considered to be the same. Now, we get a classical projective geometry, where any two distinct lines intersect in a single intersection point and any two distinct points define a single line.

This homogeneous representation has the beauty that all lines are no represented, and duality is universal. Expressions for intersection point of two lines no longer involve division, which makes life much easier if one wants to implement geometric algorithms using exact arithmetic. For example, this representation is currently used by the CGAL project [FGK+00] (a software library implementing basic geometric algorithms). Another advantage is that now that any theorem in the primal, has an immediate dual theorem in the dual. This is mathematically very elegant.

Geometrically, the points and lines are now triplets in three dimensions (w, x, y). The two dimensional plane, is the plane having $w = 1$. The plane $\langle 1, 0, 0\rangle$, which is parallel to the plane $w = 0$ represents the line at infinity.

In fact, this duality is still not perfect, since now there natural definition for what is a segment connecting two points disappears (i.e., there are two portions of a great circle connecting a pair of points, which one is the segment pq). There is an extension of this notion to add orientation. Thus $\langle 1, 1, 1\rangle$ and $\langle -1, -1, -1\rangle$ represent different lines. Intuitively, one of them represents one half plane bounded by this line, and the other represents the other half-plane. Now, if one goes through the details carefully everything falls into place and you can speak about segments (or precisely oriented segments), and so on.

This whole topic is presented quite nicely in the book by Stolfi [Sto91].

23.4.2 Duality, Voronoi Diagrams and Delaunay Triangulations.

Given a set P of points in $\mathbb{R}^{\mathsf{d}}$ its ***Voronoi diagram*** Voronoi Diagram is a partition of space into cells, where each cell is the region closest to one of the points of P. The Delaunay triangulation of a point-set is a planar graph (for $\mathsf{d} = 2$) where two points are connected by a straight segment if there is a ball that touches both points and its interior is empty. It is easy to verify that these two structures are dual to each other in the sense of graph duality. Maybe more interestingly, this duality has also an easy geometric interpretation.

Indeed, given P, its Voronoi diagram boils down to the computation of the lower envelope of cones. This set of cones can be linearized and then, the computation of the Voronoi diagram boils down to computing the lower envelope of hyperplanes, one hyperplane for each point of P. Similarly, the computation of the Delaunay triangulation of P can be reduced, after lifting the points

to the hyperboloid, to the computation of the convex hull of the points. In fact, the projection down of the lower part of the convex hull is the required triangulation. Thus, the two structures are dual to each other also lifting/linearization and direct duality. The interested reader should check out [dBvKOS00].

Chapter 24

Finite Metric Spaces and Partitions

24.1 Finite Metric Spaces

Definition 24.1.1 A ***metric space*** is a pair $(\mathcal{X}, \mathbf{d})$ where $\mathcal{X}$ is a set and $\mathbf{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric***, satisfying the following axioms: (i) $\mathbf{d}(x, y) = 0$ iff $x = y$, (ii) $\mathbf{d}(x, y) = \mathbf{d}(y, x)$, and (iii) $\mathbf{d}(x, y) + \mathbf{d}(y, z) \geq \mathbf{d}(x, z)$ (triangle inequality).

For example, $\mathbb{R}^2$ with the regular Euclidean distance is a metric space.

It is usually of interest to consider the finite case, where $\mathcal{X}$ is an a set of n points. Then, the function $\mathbf{d}$ can be specified by $\binom{n}{2}$ real numbers; that is, the distance between every pair of points of $\mathcal{X}$. Alternatively, one can think about $(\mathcal{X}, \mathbf{d})$ is a weighted complete graph, where we specify positive weights on the edges, and the resulting weights on the edges comply with the triangle inequality.

In fact, finite metric spaces rise naturally from (sparser) graphs. Indeed, let $G = (\mathcal{X}, E)$ be an undirected weighted graph defined over $\mathcal{X}$, and let $\mathbf{d}_G(x, y)$ be the length of the shortest path between x and y in G. It is easy to verify that $(\mathcal{X}, \mathbf{d}_G)$ is a finite metric space. As such if the graph G is sparse, it provides a compact representation to the finite space $(\mathcal{X}, \mathbf{d}_G)$.

Definition 24.1.2 Let $(\mathcal{X}, d)$ be an n-point metric space. We denote the *open ball* of radius r about $x \in \mathcal{X}$, by $\mathbf{b}(x, r) = \left\{ y \in \mathcal{X} \;\middle|\; \mathbf{d}(x, y) < r \right\}$.

Underling our discussion of metric spaces are algorithmic applications. The hardness of various computational problems depends heavily on the structure of the finite metric space. Thus, given a finite metric space, and a computational task, it is natural to try to map the given metric space into a new metric where the task at hand becomes easy.

Example 24.1.3 Consider the problem of computing the diameter, while it is not trivial in two dimensions, it is easy in one dimension. Thus, if we could map points in two dimensions into points in one dimension, such that the diameter is preserved, then computing the diameter becomes easy. In fact, this approach yields an efficient approximation algorithm, see Exercise 24.7.3 below.

Of course, this mapping from one metric space to another is going to introduce error. We would be interested in minimizing the error introduced by such a mapping.

Definition 24.1.4 Let $(\mathcal{X}, \mathbf{d}_\mathcal{X})$ and $(Y, \mathbf{d}_Y)$ be metric spaces. A mapping $f : \mathcal{X} \to Y$ is called an ***embedding***, and is ***C-Lipschitz*** if $\mathbf{d}_Y(f(x), f(y)) \leq C \cdot \mathbf{d}_\mathcal{X}(x, y)$ for all $x, y \in \mathcal{X}$. The mapping f is called ***K-bi-Lipschitz*** if there exists a $C > 0$ such that

$$CK^{-1} \cdot \mathbf{d}_\mathcal{X}(x, y) \leq \mathbf{d}_Y(f(x), f(y)) \leq C \cdot \mathbf{d}_\mathcal{X}(x, y),$$

for all $x, y \in \mathcal{X}$.

The least K for which f is K-bi-Lipschitz is called the ***distortion*** of f, and is denoted dist(f). The least distortion with which $\mathcal{X}$ may be embedded in Y is denoted $c_Y(\mathcal{X})$.

There are several powerful results in this vain, that show the existence of embeddings with low distortion. These include:

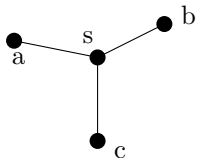
(A) **Probabilistic trees.** Every finite metric can be randomly embedded into a tree such that the "expected" distortion for a specific pair of points is $O(\log n)$.

(B) **Embedding into Euclidean space.** Any n-point metric space can be embedded into (finite dimensional) Euclidean space with $O(\log n)$ distortion.

(C) **Dimension reduction.** Any n-point set in Euclidean space with the regular Euclidean distance can be embedded into $\mathbb{R}^k$ with distortion $(1 + \varepsilon)$, where $k = O(\varepsilon^{-2} \log n)$.

24.2 Examples

What is distortion? When considering a mapping $f : \mathcal{X} \to \mathbb{R}^d$ of a metric space $(\mathcal{X}, \mathbf{d})$ to $\mathbb{R}^d$, it would useful to observe that since $\mathbb{R}^d$ can be scaled, we can consider f to be an ***expansive mapping*** (i.e., no distances shrink). Furthermore, we can in fact assume that there is at least one pair of points $x, y \in \mathcal{X}$, such that $\mathbf{d}(x, y) = \|x - y\|$. As such, we have $\mathrm{dist}(f) = \max_{x,y} \frac{\|x-y\|}{\mathbf{d}(x,y)}$.



Why distortion is necessary? Consider the a graph $G = (V, E)$ with one vertex s connected to three other vertices a, b, c, where the weights on the edges are all one (i.e., G is the star graph with three leaves). We claim that G can not be embedded into Euclidean space with distortion $\leq \sqrt{2}$. Indeed, consider the associated metric space $(V, \mathbf{d}_G)$ and an (expansive) embedding $f : V \to \mathbb{R}^d$.

Let $\triangle$ denote the triangle formed by $a'b'c'$, where $a' = f(a), b' = f(b)$ and $c' = f(c)$. Next, consider the following quantity $\max(\|a' - s'\|, \|b' - s'\|, \|c' - s'\|)$ which lower bounds the distortion of f. This quantity is minimized when $r = \|a' - s'\| = \|b' - s'\| = \|c' - s'\|$. Namely, s' is the center of the smallest enclosing circle of $\triangle$. However, r is minimized when all the edges of $\triangle$ are of equal length, and are in fact of length $\mathbf{d}_G(a, b) = 2$. It follows that $\mathrm{dist}(f) \geq r = 2/\sqrt{3}$.

It is known that $\Omega(\log n)$ distortion is necessary in the worst case when embedding a graph into euclidean space. This is shown using expanders [Mat02].

24.2.1 Hierarchical Tree Metrics

The following metric is quite useful in practice, and nicely demonstrate why algorithmically finite metric spaces are useful.

Definition 24.2.1 *Hierarchically well-separated tree* (HST) is a metric space defined on the leaves of a rooted tree T. To each vertex $u \in T$ there is associated a label $\Delta_u \geq 0$. This label is zero for all the leaves of T, and it is a positive number for all the interior nodes. The labels are such that if a vertex u is a child of a vertex v then $\Delta_u \leq \Delta_v$. The distance between two leaves $x, y \in T$ is defined as $\Delta_{\mathrm{lca}(x,y)}$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T.

A HST T is a *k-HST* if for a vertex $v \in T$, we have that $\Delta_v \leq \Delta_{\overline{\mathrm{p}}(v)}/k$, where $\overline{\mathrm{p}}(v)$ is the parent of v in T.

Note that a HST is a very limited metric. For example, consider the cycle $G = C_n$ of n vertices, with weight one on the edges, and consider an expansive embedding f of G into a HST $\mathcal{H}$. It is easy to verify, that there must be two consecutive nodes of the cycle, which are mapped to two different subtrees of the root r of $\mathcal{H}$. Since $\mathcal{H}$ is expansive, it follows that $\Delta_r \geq n/2$. As such, $\mathrm{dist}(f) \geq n/2$. Namely, HSTs fail to faithfully represent even very simple metrics.

24.2.2 Clustering

One natural problem we might want to solve on a graph (i.e., finite metric space) $(\mathcal{X}, \mathbf{d})$ is to partition it into clusters. One such natural clustering is the *k-median clustering*, where we would like to choose a set $C \subseteq \mathcal{X}$ of k centers, such that

$$\nu_C(\mathcal{X}, \mathbf{d}) = \sum_{\mathsf{q} \in \mathcal{X}} \mathbf{d}(\mathsf{q}, C)$$

is minimized, where $\mathbf{d}(\mathsf{q}, C) = \min_{c \in C} \mathbf{d}(\mathsf{q}, c)$ is the distance of q to its closest center in C.

It is known that finding the optimal k-median clustering in a (general weighted) graph is NP-complete. As such, the best we can hope for is an approximation algorithm. However, if the structure of the finite metric space $(\mathcal{X}, \mathbf{d})$ is simple, then the problem can be solved efficiently. For example, if the points of $\mathcal{X}$ are on the real line (and the distance between a and b is just $|a - b|$), then k-median can be solved using dynamic programming.

Another interesting case is when the metric space $(\mathcal{X}, \mathbf{d})$ is a HST. Is not too hard to prove the following lemma. See Exercise 24.7.1.

Lemma 24.2.2 *Let $(\mathcal{X}, \mathbf{d})$ be a HST defined over n points, and let $k > 0$ be an integer. One can compute the optimal k-median clustering of $\mathcal{X}$ in $O(k^2 n)$ time.*

Thus, if we can embed a general graph G into a HST $\mathcal{H}$, with low distortion, then we could approximate the k-median clustering on G by clustering the resulting HST, and "importing" the resulting partition to the original space. The quality of approximation, would be bounded by the distortion of the embedding of G into $\mathcal{H}$.

24.3 Random Partitions

Let $(\mathcal{X}, d)$ be a finite metric space. Given a partition $P = \{C_1, \ldots, C_m\}$ of $\mathcal{X}$, we refer to the sets C_i as *clusters*. We write $\mathcal{P}_{\mathcal{X}}$ for the set of all partitions of $\mathcal{X}$. For $x \in \mathcal{X}$ and a partition $P \in \mathcal{P}_{\mathcal{X}}$ we denote by $P(x)$ the unique cluster of P containing x. Finally, the set of all probability distributions on $\mathcal{P}_{\mathcal{X}}$ is denoted $\mathcal{D}_{\mathcal{X}}$.

What we want, and what we can get. The target is to partition the metric space into clusters, such that each cluster would have diameter at most Δ, for some prespecified parameter Δ.

We would like to have a partition that does not disrupt distances too "badly". Intuitively, that means that a pair of points that is in distance larger than Δ will be separated by the clustering, but points that are closer to each other would be in the same cluster. This is of course impossible, as any clustering must separate points that are close to each other. To see that, consider a set of points densely packed on the interval $[0, 10]$, and let $\Delta < 5$. Clearly, there would always be two close points that would be in two separate clusters.

As such, our strategy would be to use partitions that are constructed randomly, and the best we can hope for, is that the probability of them being separated is a function of their distance t, which would be small if t is small. As an example, for the case of points on the real line, take the natural partition into intervals of length Δ (that is, all the points in the interval $[i\Delta, (i+1)\Delta)$ would belong to the same cluster), and randomly shift it by a random number x picked uniformly in $[0, \Delta)$. Namely, all the points belonging to $[x + i\Delta,\ x + (i+1)\Delta)$ would belong to the same cluster. Now, it is easy to verify that for any two points $\mathsf{p}, \mathsf{q} \in \mathbb{R}$, of distance $t = |\mathsf{p} - \mathsf{q}|$ from each other, the probability that they are in two different intervals is bounded by t/Δ (see Exercise 24.7.4). And intuitively, this is the best one can hope for.

As such, the clustering scheme we seek should separate two points in distance t from each other with probability $(t/\Delta) * \text{noise}$, where noise is hopefully small.

24.3.1 Constructing the partition

Let $\Delta = 2^u$ be a prescribed parameter, which is the required diameter of the resulting clusters. Choose, uniformly at random, a permutation π of $\mathcal{X}$ and a random value $\alpha \in [1/4, 1/2]$. Let $R = \alpha\Delta$, and observe that it is uniformly distributed in the interval $[\Delta/4, \Delta/2]$.

The partition is now defined as follows: A point $x \in \mathcal{X}$ is assigned to the cluster C_y of y, where y is the first point in the permutation in distance $\leq R$ from x. Formally,

$$C_y = \left\{ x \in \mathcal{X} \;\middle|\; x \in \mathbf{b}(y, R) \text{ and } \pi(y) \leq \pi(z) \text{ for all } z \in \mathcal{X} \text{ with } x \in \mathbf{b}(z, R) \right\}.$$

Let $P = \{C_y\}_{y \in \mathcal{X}}$ denote the resulting partition.

Here is a somewhat more intuitive explanation: Once we fix the radius of the clusters R, we start scooping out balls of radius R centered at the points of the random permutation π. At the ith stage, we scoop out only the remaining mass at the ball centered at x_i of radius r, where x_i is the ith point in the random permutation.

24.3.2 Properties

Lemma 24.3.1 *Let $(\mathcal{X}, d)$ be a finite metric space, $\Delta = 2^u$ a prescribed parameter, and let P be the partition of $\mathcal{X}$ generated by the above random partition. Then the following holds:*

(i) For any $C \in P$, we have $\operatorname{diam}(C) \leq \Delta$.

(ii) Let x be any point of $\mathcal{X}$, and t a parameter $\leq \Delta/8$. For $B = \mathbf{b}(x, t)$, we have that

$$\mathbf{Pr}[B \not\subseteq P(x)] \leq \frac{8t}{\Delta} \ln \frac{M}{m},$$

where $M = |\mathbf{b}(x, \Delta/8)|$, $m = |\mathbf{b}(x, \Delta)|$.

Proof: Since $C_y \subseteq \mathbf{b}(y, R)$, we have that $\operatorname{diam}(C_y) \leq \Delta$, and thus the first claim holds.

Let U be the set of points $w \in \mathbf{b}(x, \Delta)$ such that $\mathbf{b}(w, R) \cap B \neq \emptyset$. Arrange the points of U in increasing distance from x, and let $w_1, \ldots, w_{M'}$ denote the resulting order, where $M' = |U|$. For $k = 1, \ldots, M'$, let $I_k = [\mathbf{d}(x, w_k) - t, \mathbf{d}(x, w_k) + t]$ and write $\mathcal{E}_k$ for the event that w_k is the *first* point in π such that $B \cap C_{w_k} \neq \emptyset$, and yet $B \not\subseteq C_{w_k}$. Observe that if $B \not\subseteq P(x)$ then one of he events $\mathcal{E}_1, \ldots, \mathcal{E}_{M'}$ must occur.

Note that if $w_k \in \mathbf{b}(x, \Delta/8)$, then $\mathbf{Pr}[\mathcal{E}_k] = 0$ since $t \leq \Delta/8$ and $B = \mathbf{b}(x, t) \subseteq \mathbf{b}(x, \Delta/8) \subseteq \mathbf{b}(w_k, \Delta/4) \subseteq \mathbf{b}(w_k, R)$. Indeed, when we "scoop" out the cluster C_{w_k}, either B would be fully contained inside C_{w_k}, or alternatively, if B is not fully contained inside C_{w_k}, then some parts of B were already "scooped out" by some other point of U, and as such $\mathcal{E}_k$ does not happen.

In particular, $w_1, \ldots, w_m$ are inside $\mathbf{b}(x, \Delta/8)$ and as such $\mathbf{Pr}[\mathcal{E}_1] = \cdots = \mathbf{Pr}[\mathcal{E}_a] = 0$. Also, note that if $\mathbf{d}(x, w_k) < R - t$ then $\mathbf{b}(w_k, R)$ contains B and as such $\mathcal{E}_k$ can not happen. Similarly, if $\mathbf{d}(x, w_k) > R + t$ then $\mathbf{b}(w_k, R) \cap B = \emptyset$ and $\mathcal{E}_k$ can not happen. As such, if $\mathcal{E}_k$ happen then $R - t \le \mathbf{d}(x, w_k) \le R + t$. Namely, if $\mathcal{E}_k$ happen then $R \in I_k$. We conclude that

$$\mathbf{Pr}[\mathcal{E}_k] = \mathbf{Pr}[\mathcal{E}_k \cap (R \in I_k)] = \mathbf{Pr}[R \in I_k] \cdot \mathbf{Pr}[\mathcal{E}_k \,|\, R \in I_k] .$$

Now, R is uniformly distributed in the interval $[\Delta/4, \Delta/2]$, and I_k is an interval of length $2t$. Thus, $\mathbf{Pr}[R \in I_k] \le 2t/(\Delta/4) = 8t/\Delta$.

Next, to bound $\mathbf{Pr}[\mathcal{E}_k \,|\, R \in I_k]$, we observe that $w_1, \ldots, w_{k-1}$ are closer to x than w_k and their distance to $\mathbf{b}(x, t)$ is smaller than R. Thus, if any of them appear before w_k in π then $\mathcal{E}_k$ does not happen. Thus, $\mathbf{Pr}[\mathcal{E}_k \,|\, R \in I_k]$ is bounded by the probability that w_k is the first to appear in π out of $w_1, \ldots, w_k$. But this probability is $1/k$, and thus $\mathbf{Pr}[\mathcal{E}_k \,|\, R \in I_k] \le 1/k$.

We are now ready for the kill. Indeed,

$$\begin{aligned}
\mathbf{Pr}[B \not\subseteq P(x)] &= \sum_{k=1}^{M'} \mathbf{Pr}[\mathcal{E}_k] = \sum_{k=m+1}^{M'} \mathbf{Pr}[\mathcal{E}_k] = \sum_{k=m+1}^{M'} \Pr[R \in I_k] \cdot \Pr[\mathcal{E}_k \,|\, R \in I_k] \\
&\le \sum_{k=m+1}^{M'} \frac{8t}{\Delta} \cdot \frac{1}{k} \le \frac{8t}{\Delta} \ln \frac{M'}{m} \le \frac{8t}{\Delta} \ln \frac{M}{m},
\end{aligned}$$

since $\sum_{k=m+1}^{M'} \frac{1}{k} \le \int_m^{M'} \frac{dx}{x} = \ln \frac{M'}{m}$ and $M' \le M$. ■

24.4 Probabilistic embedding into trees

In this section, given n-point finite metric $(\mathcal{X}, \mathbf{d})$. we would like to embed it into a HST. As mentioned above, one can verify that for any embedding into HST, the distortion in the worst case is $\Omega(n)$. Thus, we define a randomized algorithm that embed $(\mathcal{X}, d)$ into a tree. Let T be the resulting tree, and consider two points $x, y \in \mathcal{X}$. Consider the *random variable* $\mathbf{d}_T(x, y)$. We constructed the tree T such that distances never shrink; i.e. $\mathbf{d}(x, y) \le \mathbf{d}_T(x, y)$. The ***probabilistic distortion*** of this embedding is $\max_{x,y} \mathbf{E}\left[\frac{\mathbf{d}_T(x,y)}{\mathbf{d}(x,y)}\right]$. Somewhat surprisingly, one can find such an embedding with logarithmic probabilistic distortion.

Theorem 24.4.1 *Given n-point metric $(\mathcal{X}, d)$ one can randomly embed it into a 2-HST with probabilistic distortion $\le 24 \ln n$.*

Proof: The construction is recursive. Let $\operatorname{diam}(P)$, and compute a random partition of $\mathcal{X}$ with cluster diameter $\operatorname{diam}(P)/2$, using the construction of Section 24.3.1. We recursively construct a 2-HST for each cluster, and hang the resulting clusters on the root node v, which is marked by $\Delta_v = \operatorname{diam}(P)$. Clearly, the resulting tree is a 2-HST.

For a node $v \in T$, let $\mathcal{X}(v)$ be the set of points of $\mathcal{X}$ contained in the subtree of v.

For the analysis, assume $\operatorname{diam}(P) = 1$, and consider two points $x, y \in \mathcal{X}$. We consider a node $v \in T$ to be in level i if $\operatorname{level}(v) = \lceil \lg \Delta_v \rceil = i$. The two points x and y correspond to two leaves in T, and let $\widehat{u}$ be the least common ancestor of x and y in t. We have $\mathbf{d}_T(x, y) \le 2^{\operatorname{level}(v)}$. Furthermore, note that along a path the levels are strictly monotonically increasing.

In fact, we are going to be conservative, and let w be the first ancestor of x, such that $\mathbf{b} = \mathbf{b}(x, \mathbf{d}(x, y))$ is not completely contained in $\mathcal{X}(u_1), \ldots, \mathcal{X}(u_m)$, where $u_1, \ldots, u_m$ are the children of w. Clearly, $\operatorname{level}(w) > \operatorname{level}(\widehat{u})$. Thus, $\mathbf{d}_T(x, y) \le 2^{\operatorname{level}(w)}$.

Consider the path σ from the root of T to x, and let $\mathcal{E}_i$ be the event that $\mathbf{b}$ is not fully contained in $\mathcal{X}(v_i)$, where v_i is the node of σ of level i (if such a node exists). Furthermore, let Y_i be the indicator variable which is 1 if $\mathcal{E}_i$ is the first to happened out of the sequence of events $\mathcal{E}_0, \mathcal{E}_{-1}, \ldots$. Clearly, $\mathbf{d}_T(x, y) \le \sum Y_i 2^i$.

Let $t = \mathbf{d}(x, y)$ and $j = \lfloor \lg \mathbf{d}(x, y) \rfloor$, and $n_i = \left|\mathbf{b}(x, 2^i)\right|$ for $i = 0, \ldots, -\infty$. We have

$$\mathbf{E}[\mathbf{d}_T(x, y)] \le \sum_{i=j}^{0} \mathbf{E}[Y_i] \, 2^i \le \sum_{i=j}^{0} 2^i \, \mathbf{Pr}\left[\mathcal{E}_i \cap \overline{\mathcal{E}_{i-1}} \cap \overline{\mathcal{E}_{i-1}} \cdots \overline{\mathcal{E}_0}\right] \le \sum_{i=j}^{0} 2^i \cdot \frac{8t}{2^i} \ln \frac{n_i}{n_{i-3}},$$

by Lemma 24.3.1. Thus,

$$\mathbf{E}[\mathbf{d}_T(x, y)] \le 8t \ln \left(\prod_{i=j}^{0} \frac{n_i}{n_{i-3}} \right) \le 8t \ln(n_0 \cdot n_1 \cdot n_2) \le 24t \ln n.$$

It thus follows, that the expected distortion for x and y is $\le 24 \ln n$. ■

24.4.1 Application: approximation algorithm for k-median clustering

Let $(\mathcal{X}, \mathbf{d})$ be a n-point metric space, and let k be an integer number. We would like to compute the optimal k-median clustering. Number, find a subset $C_{\text{opt}} \subseteq \mathcal{X}$, such that the price of the clustering $\nu_{C_{\text{opt}}}(\mathcal{X}, \mathbf{d})$ is minimized, see Section 24.2.2. To this end, we randomly embed $(\mathcal{X}, \mathbf{d})$ into a HST $\mathcal{H}$ using Theorem 24.4.1. Next, using Lemma 24.2.2, we compute the optimal k-median clustering of $\mathcal{H}$. Let C be the set of centers computed. We return C together with the partition of $\mathcal{X}$ it induces as the required clustering.

Theorem 24.4.2 *Let $(\mathcal{X}, \mathbf{d})$ be a n-point metric space. One can compute in polynomial time a k-median clustering of $\mathcal{X}$ which has expected price $O(\alpha \log n)$, where α is the price of the optimal k-median clustering of $(\mathcal{X}, \mathbf{d})$.*

Proof: The algorithm is described above, and the fact that its running time is polynomial can be easily be verified. To prove the bound on the quality of the clustering, for any point $p \in \mathcal{X}$, let $\overline{c}(p)$ denote the closest point in C_{opt} to p according to $\mathbf{d}$, where C_{opt} is the set of k-medians in the optimal clustering. Let C be the set of k-medians returned by the algorithm, and let $\mathcal{H}$ be the HST used by the algorithm. We have

$$\beta = \nu_C(\mathcal{X}, \mathbf{d}) \leq \nu_C(\mathcal{X}, \mathbf{d}_{\mathcal{H}}) \leq \nu_{C_{\text{opt}}}(\mathcal{X}, \mathbf{d}_{\mathcal{H}}) \leq \sum_{p \in \mathcal{X}} \mathbf{d}_{\mathcal{H}}(p, C_{\text{opt}}) \leq \sum_{p \in \mathcal{X}} \mathbf{d}_{\mathcal{H}}(p, \overline{c}(p)).$$

Thus, in expectation we have

$$\begin{aligned} \mathbf{E}[\beta] &= \mathbf{E}\left[\sum_{p \in \mathcal{X}} \mathbf{d}_{\mathcal{H}}(p, \overline{c}(p))\right] = \sum_{p \in \mathcal{X}} \mathbf{E}[\mathbf{d}_{\mathcal{H}}(p, \overline{c}(p))] = \sum_{p \in \mathcal{X}} O(\mathbf{d}(p, \overline{c}(p)) \log n) \\ &= O\left((\log n) \sum_{p \in \mathcal{X}} \mathbf{d}(p, \overline{c}(p))\right) = O\left(\nu_{C_{\text{opt}}}(\mathcal{X}, \mathbf{d}) \log n\right), \end{aligned}$$

by linearity of expectation and Theorem 24.4.1. ■

24.5 Embedding any metric space into Euclidean space

Lemma 24.5.1 *Let $(\mathcal{X}, \mathbf{d})$ be a metric, and let $Y \subset \mathcal{X}$. Consider the mapping $f : \mathcal{X} \to \mathbb{R}$, where $f(x) = \mathbf{d}(x, Y) = \min_{y \in Y} \mathbf{d}(x, y)$. Then for any $x, y \in \mathcal{X}$, we have $|f(x) - f(y)| \leq \mathbf{d}(x, y)$. Namely f is nonexpansive.*

Proof: Indeed, let x' and y' be the closet points of Y, to x and y, respectively. Observe that $f(x) = \mathbf{d}(x, x') \leq \mathbf{d}(x, y') \leq \mathbf{d}(x, y) + \mathbf{d}(y, y') = \mathbf{d}(x, y) + f(y)$ by the triangle inequality. Thus, $f(x) - f(y) \leq \mathbf{d}(x, y)$. By symmetry, we have $f(y) - f(x) \leq \mathbf{d}(x, y)$. Thus, $|f(x) - f(y)| \leq \mathbf{d}(x, y)$. ■

24.5.1 The bounded spread case

Let $(\mathcal{X}, \mathbf{d})$ be a n-point metric. The ***spread*** of $\mathcal{X}$, denoted by $\Phi(\mathcal{X}) = \frac{\operatorname{diam}(\mathcal{X})}{\min_{x,y \in \mathcal{X}, x \neq y} \mathbf{d}(x,y)}$, is the ratio between the diameter of $\mathcal{X}$ and the distance between the closest pair of points.

Theorem 24.5.2 *Given a n-point metric $\mathcal{Y} = (\mathcal{X}, d)$, with spread Φ, one can embed it into Euclidean space $\mathbb{R}^k$ with distortion $O(\sqrt{\ln \Phi} \ln n)$, where $k = O(\ln \Phi \ln n)$.*

Proof: Assume that $\operatorname{diam}(\mathcal{Y}) = \Phi$ (i.e., the smallest distance in $\mathcal{Y}$ is 1), and let $r_i = 2^{i-2}$, for $i = 1, \ldots, \alpha$, where $\alpha = \lceil \lg \Phi \rceil$. Let $P_{i,j}$ be a random partition of P with diameter r_i, using Theorem 24.4.1, for $i = 1, \ldots, \alpha$ and $j = 1, \ldots, \beta$, where $\beta = \lceil c \log n \rceil$ and c is a large enough constant to be determined shortly.

For each cluster of $P_{i,j}$ randomly toss a coin, and let $V_{i,j}$ be the all the points of $\mathcal{X}$ that belong to clusters in $P_{i,j}$ that got 'T' in their coin toss. For a point $u \in x$, let $f_{i,j}(x) = \mathbf{d}(x, \mathcal{X} \setminus V_{i,j}) = \min_{v \in \mathcal{X} \setminus V_{i,j}} \mathbf{d}(x, v)$, for $i = 0, \ldots, m$ and $j = 1, \ldots, \beta$. Let $F : \mathcal{X} \to \mathbb{R}^{(m+1)\cdot\beta}$ be the embedding, such that $F(x) = (f_{0,1}(x), f_{0,2}(x), \ldots, f_{0,\beta}(x), f_{1,1}(x), f_{0,2}(x), \ldots, f_{1,\beta}(x), \ldots, f_{m,1}(x), f_{m,2}(x), \ldots, f_{m,\beta}(x))$.

Next, consider two points $x, y \in \mathcal{X}$, with distance $\phi = \mathbf{d}(x, y)$. Let k be an integer such that $r_u \leq \phi/2 \leq r_{u+1}$. Clearly, in any partition of $P_{u,1}, \ldots, P_{u,\beta}$ the points x and y belong to different clusters. Furthermore, with probability half $x \in V_{u,j}$ and $y \notin V_{u,j}$ or $x \notin V_{u,j}$ and $y \in V_{u,j}$, for $1 \leq j \leq \beta$.

Let $\mathcal{E}_j$ denote the event that $\mathbf{b}(x, \rho) \subseteq V_{u,j}$ and $y \notin V_{u,j}$, for $j = 1, \ldots, \beta$, where $\rho = \phi/(64 \ln n)$. By Lemma 24.3.1, we have

$$\mathbf{Pr}\left[\mathbf{b}(x, \rho) \not\subseteq P_{u,j}(x)\right] \leq \frac{8\rho}{r_u} \ln n \leq \frac{\phi}{8 r_u} \leq 1/2.$$

Thus,

$$\begin{aligned} \mathbf{Pr}\left[\mathcal{E}_j\right] &= \mathbf{Pr}\left[\left(\mathbf{b}(x, \rho) \subseteq P_{u,j}(x)\right) \cap \left(x \in V_{u,j}\right) \cap \left(y \notin V_{u,j}\right)\right] \\ &= \mathbf{Pr}\left[\mathbf{b}(x, \rho) \subseteq P_{u,j}(x)\right] \cdot \mathbf{Pr}\left[x \in V_{u,j}\right] \cdot \mathbf{Pr}\left[y \notin V_{u,j}\right] \geq 1/8, \end{aligned}$$

since those three events are independent. Notice, that if $\mathcal{E}_j$ happens, than $f_{u,j}(x) \geq \rho$ and $f_{u,j}(y) = 0$.

Let X_j be an indicator variable which is 1 if $\mathcal{E}_i$ happens, for $j = 1, \ldots, \beta$. Let $Z = \sum_j X_j$, and we have $\mu = \mathbf{E}[Z] = \mathbf{E}\left[\sum_j X_j\right] \geq \beta/8$. Thus, the probability that only $\beta/16$ of $\mathcal{E}_1, \ldots, \mathcal{E}_\beta$ happens, is $\mathbf{Pr}[Z < (1 - 1/2)\,\mathbf{E}[Z]]$. By the Chernoff inequality, we have $\mathbf{Pr}[Z < (1 - 1/2)\,\mathbf{E}[Z]] \leq \exp\left(-\mu 1/(2 \cdot 2^2)\right) = \exp(-\beta/64) \leq 1/n^{10}$, if we set $c = 640$.

Thus, with high probability

$$\|F(x) - F(y)\| \geq \sqrt{\sum_{j=1}^{\beta} \Big(f_{u,j}(x) - f_{u,j}(y)\Big)^2} \geq \sqrt{\rho^2 \frac{\beta}{16}} = \sqrt{\beta}\frac{\rho}{4} = \phi \cdot \frac{\sqrt{\beta}}{256 \ln n}.$$

On the other hand, $\big|f_{i,j}(x) - f_{i,j}(y)\big| \leq \mathbf{d}(x,y) = \phi \leq 64\rho \ln n$. Thus,

$$\|F(x) - F(y)\| \leq \sqrt{\alpha\beta(64\rho \ln n)^2} \leq 64\sqrt{\alpha\beta}\rho \ln n = \sqrt{\alpha\beta} \cdot \phi.$$

Thus, setting $G(x) = F(x)\frac{256 \ln n}{\sqrt{\beta}}$, we get a mapping that maps two points of distance ϕ from each other to two points with distance in the range $\Big[\phi, \phi \cdot \sqrt{\alpha\beta} \cdot \frac{256 \ln n}{\sqrt{\beta}}\Big]$. Namely, $G(\cdot)$ is an embedding with distortion $O(\sqrt{\alpha} \ln n) = O(\sqrt{\ln \Phi} \ln n)$.

The probability that G fails on one of the pairs, is smaller than $(1/n^{10}) \cdot \binom{n}{2} < 1/n^8$. In particular, we can check the distortion of G for all $\binom{n}{2}$ pairs, and if any of them fail (i.e., the distortion is too big), we restart the process. ■

24.5.2 The unbounded spread case

Our next task, is to extend Theorem 24.5.2 to the case of unbounded spread. Indeed, let $(\mathcal{X}, d)$ be a n-point metric, such that $\operatorname{diam}(\mathcal{X}) \leq 1/2$. Again, we look on the different resolutions $r_1, r_2, \ldots$, where $r_i = 1/2^{i-1}$. For each one of those resolutions r_i, we can embed this resolution into β coordinates, as done for the bounded case. Then we concatenate the coordinates together.

There are two problems with this approach: (i) the number of resulting coordinates is infinite, and (ii) a pair x, y, might be distorted a "lot" because it contributes to all resolutions, not only to its "relevant" resolutions.

Both problems can be overcome with careful tinkering. Indeed, for a resolution r_i, we are going to modify the metric, so that it ignores short distances (i.e., distances $\leq r_i/n^2$). Formally, for each resolution r_i, let $G_i = (\mathcal{X}, \widehat{E_i})$ be the graph where two points x and y are connected if $\mathbf{d}(x,y) \leq r_i/n^2$. Consider a connected component $C \in G_i$. For any two points $x, y \in C$, we have $\mathbf{d}(x,y) \leq n(r_i/n^2) \leq r_i/n$. Let $\mathcal{X}_i$ be the set of connected components of G_i, and define the distances between two connected components $C, C' \in \mathcal{X}_i$, to be $\mathbf{d}_i(C, C') = \mathbf{d}(C, C') = \min_{c \in C, c' \in C'} \mathbf{d}(c, c')$.

It is easy to verify that $(\mathcal{X}_i, \mathbf{d}_i)$ is a metric space (see Exercise 24.7.2). Furthermore, we can naturally embed $(\mathcal{X}, \mathbf{d})$ into $(\mathcal{X}_i, \mathbf{d}_i)$ by mapping a point $x \in \mathcal{X}$ to its connected components in $\mathcal{X}_i$. Essentially $(\mathcal{X}_i, \mathbf{d}_i)$ is a snapped version of the metric $(\mathcal{X}, d)$, with the advantage that $\Phi((\mathcal{X}, \mathbf{d}_i)) = O(n^2)$. We now embed $\mathcal{X}_i$ into $\beta = O(\log n)$ coordinates. Next, for any point of $\mathcal{X}$ we embed it into those β coordinates, by using the embedding of its connected component in $\mathcal{X}_i$. Let E_i be the embedding for resolution r_i. Namely, $E_i(x) = (f_{i,1}(x), f_{i,2}(x), \ldots, f_{i,\beta}(x))$, where $f_{i,j}(x) = \min(\mathbf{d}_i(x, \mathcal{X} \setminus V_{i,j}), 2r_i)$. The resulting embedding is $F(x) = \oplus E_i(x) = (E_1(x), E_2(x), \ldots,)$.

Since we slightly modified the definition of $f_{i,j}(\cdot)$, we have to show that $f_{i,j}(\cdot)$ is nonexpansive. Indeed, consider two points $x, y \in \mathcal{X}_i$, and observe that

$$\big|f_{i,j}(x) - f_{i,j}(y)\big| \leq \big|\mathbf{d}_i(x, V_{i,j}) - \mathbf{d}_i(y, V_{i,j})\big| \leq \mathbf{d}_i(x,y) \leq \mathbf{d}(x,y),$$

as a simple case analysis[④] shows.

For a pair $x, y \in \mathcal{X}$, and let $\phi = \mathbf{d}(x,y)$. To see that $F(\cdot)$ is the required embedding (up to scaling), observe that, by the same argumentation of Theorem 24.5.2, we have that with high probability

$$\|F(x) - F(y)\| \geq \phi \cdot \frac{\sqrt{\beta}}{256 \ln n}.$$

To get an upper bound on this distance, observe that for i such that $r_i > \phi n^2$, we have $E_i(x) = E_i(y)$. Thus,

$$\begin{aligned}
\|F(x) - F(y)\|^2 &= \sum_i \|E_i(x) - E_i(y)\|^2 = \sum_{i, r_i < \phi n^2} \|E_i(x) - E_i(y)\|^2 \\
&= \sum_{i, \phi/n^2 < r_i < \phi n^2} \|E_i(x) - E_i(y)\|^2 + \sum_{i, r_i < \phi/n^2} \|E_i(x) - E_i(y)\|^2 \\
&= \beta\phi^2 \lg\big(n^4\big) + \sum_{i, r_i < \phi/n^2} (2r_i)^2\beta \leq 4\beta\phi^2 \lg n + \frac{4\phi^2\beta}{n^4} \leq 5\beta\phi^2 \lg n.
\end{aligned}$$

Thus, $\|F(x) - F(y)\| \leq \phi\sqrt{5\beta \lg n}$. We conclude, that with high probability, $F(\cdot)$ is an embedding of $\mathcal{X}$ into Euclidean space with distortion $\Big(\phi\sqrt{5\beta \lg n}\Big) \Big/ \Big(\phi \cdot \frac{\sqrt{\beta}}{256 \ln n}\Big) = O(\log^{3/2} n)$.

[④] Indeed, if $f_{i,j}(x) < \mathbf{d}_i(x, V_{i,j})$ and $f_{i,j}(y) < \mathbf{d}_i(x, V_{i,j})$ then $f_{i,j}(x) = 2r_i$ and $f_{i,j}(y) = 2r_i$, which implies the above inequality. If $f_{i,j}(x) = \mathbf{d}_i(x, V_{i,j})$ and $f_{i,j}(y) = \mathbf{d}_i(x, V_{i,j})$ then the inequality trivially holds. The other option is handled in a similar fashion.

We still have to handle the infinite number of coordinates problem. However, the above proof shows that we care about a resolution r_i (i.e., it contributes to the estimates in the above proof) only if there is a pair x and y such that $r_i/n^2 \leq \mathbf{d}(x,y) \leq r_i n^2$. Thus, for every pair of distances there are $O(\log n)$ relevant resolutions. Thus, there are at most $\eta = O(n^2\beta \log n) = O(n^2 \log^2 n)$ relevant coordinates, and we can ignore all the other coordinates. Next, consider the affine subspace h that spans $F(P)$. Clearly, it is $n-1$ dimensional, and consider the projection $G : \mathbb{R}^\eta \to \mathbb{R}^{n-1}$ that projects a point to its closest point in h. Clearly, $G(F(\cdot))$ is an embedding with the same distortion for P, and the target space is of dimension $n-1$.

Note, that all this process succeeds with high probability. If it fails, we try again. We conclude:

Theorem 24.5.3 (Low quality Bourgain theorem.) *Given a n-point metric M, one can embed it into Euclidean space of dimension $n-1$, such that the distortion of the embedding is at most $O(\log^{3/2} n)$.*

Using the Johnson-Lindenstrauss lemma, the dimension can be further reduced to $O(\log n)$. In fact, being more careful in the proof, it is possible to reduce the dimension to $O(\log n)$ directly.

24.6 Bibliographical notes

The partitions we use are due to Calinescu *et al.* [CKR01]. The idea of embedding into spanning trees is due to Alon *et al.* [AKPW95], which showed that one can get a probabilistic distortion of $2^{O\left(\sqrt{\log n \log\log n}\right)}$. Yair Bartal realized that by allowing trees with additional vertices, one can get a considerably better result. In particular, he showed [Bar96] that probabilistic embedding into trees can be done with polylogarithmic average distortion. He later improved the distortion to $O(\log n \log\log n)$ in [Bar98]. Improving this result was an open question, culminating in the work of Fakcharoenphol *et al.* [FRT03] which achieve the optimal $O(\log n)$ distortion.

Interestingly, if one does not care about the optimal distortion, one can get similar result (for embedding into probabilistic trees), by first embedding the metric into Euclidean space, then reduce the dimension by the Johnson-Lindenstrauss lemma, and finally, construct an HST by constructing a quadtree over the points. The "trick" is to randomly translate the quadtree. It is easy to verify that this yields $O(\log^4 n)$ distortion. See the survey by Indyk [Ind01] for more details. This random shifting of quadtrees is a powerful technique that was used in getting several result, and it is a crucial ingredient in Arora [Aro98] approximation algorithm for Euclidean TSP.

Our proof of Lemma 24.3.1 (which is originally from [FRT03]) is taken from [KLMN04]. The proof of Theorem 24.5.3 is by Gupta [Gup00].

A good exposition of metric spaces is available in Matoušek [Mat02].

24.7 Exercises

Exercise 24.7.1 (Clustering for HST.) [4 Points]

Let $(\mathcal{X}, \mathbf{d})$ be a HST defined over n points, and let $k > 0$ be an integer. Provide an algorithm that computes the optimal k-median clustering of $\mathcal{X}$ in $O(k^2 n)$ time.

[**Hint:** Transform the HST into a tree where every node has only two children. Next, run a dynamic programming algorithm on this tree.]

Exercise 24.7.2 (Partition induced metric.) [10 Points]

(A) **[2 Points]** Give a counter example to the following claim: Let $(\mathcal{X}, \mathbf{d})$ be a metric space, and let P be a partition of $\mathcal{X}$. Then, the pair $(P, \mathbf{d}')$ is a metric, where $\mathbf{d}'(C, C') = \mathbf{d}(C, C') = \min_{x \in C, y \in C'} \mathbf{d}(x,y)$ and $C, C' \in P$.

(B) **[8 Points]** Let $(\mathcal{X}, \mathbf{d})$ be a n-point metric space, and consider the set $U = \left\{ i \;\middle|\; 2^i \leq \mathbf{d}(x,y) \leq 2^{i+1}, \text{ for } x, y \in \mathcal{X} \right\}$. Prove that $|U| = O(n)$. Namely, there are only n different resolutions that "matter" for a finite metric space.

Exercise 24.7.3 (Computing the diameter via embeddings.) [7 Points]

(A) **[1 Points]** Let ℓ be a line in the plane, and consider the embedding $f : \mathbb{R}^2 \to \ell$, which is the projection of the plane into ℓ. Prove that f is 1-Lipschitz, but it is not K-bi-Lipschitz for any constant K.

(B) **[3 Points]** Prove that one can find a family of projections $\mathcal{F}$ of size $O(1/\sqrt{\varepsilon})$, such that for any two points $x, y \in \mathbb{R}^2$, for one of the projections $f \in \mathcal{F}$ we have $\mathbf{d}(f(x), f(y)) \geq (1-\varepsilon)\mathbf{d}(x,y)$.

(C) **[1 Points]** Given a set P of n in the plane, given a $O(n/\sqrt{\varepsilon})$ time algorithm that outputs two points $x, y \in P$, such that $\mathbf{d}(x,y) \geq (1-\varepsilon)\mathrm{diam}(P)$, where $\mathrm{diam}(P) = \max_{z,w \in P} \mathbf{d}(z,w)$ is the diameter of P.

(D) **[2 Points]** Given P, show how to extract, in $O(n)$ time, a set $Q \subseteq P$ of size $O(\varepsilon^{-2})$, such that $\mathrm{diam}(Q) \geq (1-\varepsilon/2)\mathrm{diam}(P)$. (Hint: Construct a grid of appropriate resolution.)

In particular, give an $(1-\varepsilon)$-approximation algorithm to the diameter of P that works in $O(n + \varepsilon^{-2.5})$ time. (There are slightly faster approximation algorithms known for approximating the diameter.)

Exercise 24.7.4 (Partitions in Euclidean Space.) [10 Points]

(A) **[1 Points]** For a real number $\Delta > 0$, and a random number $x \in [0, \Delta]$ consider the random partition of the real line into intervals of length Δ, such that all the points falling into the interval $[i\Delta, i\Delta + x)$ are in the same cluster. Prove, that for two points, $\mathsf{p}, \mathsf{q} \in \mathbb{R}$, the probability of p and q to be in two different clusters is at most $|\mathsf{p} - \mathsf{q}| / \Delta$.

(B) **[3 Points]** Consider the d dimensional grid of sidelength Δ, and let p be a random vector in the hypercube $[0, \Delta]^d$. Shift the grid by p, and consider the partition of $\mathbb{R}^d$ induced by this grid. Formally, the space is partitioned into clusters, where all the points inside the cube $\mathsf{p} + [0, \Delta)^d$ are one cluster. Consider any two points $\mathsf{q}, \mathsf{r} \in \mathbb{R}^d$. Prove, that the probability that q and r are in different clusters is bounded by $d \, \|\mathsf{q} - \mathsf{r}\| / \Delta$.

(C) **[6 Points]** Strengthen (B), by showing that the probability is bounded by $\sqrt{d} \, \|\mathsf{q} - \mathsf{r}\| / \Delta$. [**Hint:** Consider the distance $t = \|\mathsf{q} - \mathsf{r}\|$ to be fixed, and figure out what is the worst case for this partition.]

Part (C) implies that we can partition space into clusters with diameter $\Delta' = \sqrt{d}\Delta$ such that the probability of points in distance t from each other to be separated is bounded by dt/Δ'.

Acknowledgments

The presentation in this write-up follows closely the insightful suggestions of Manor Mendel.

Chapter 25

Tail Inequalities

"Wir müssen wissen, wir werden wissen" (We must know, we shall know)
—— David Hilbert

25.1 Markov Inequality

Theorem 25.1.1 (Markov Inequality) *For a non-negative variable X, and $t > 0$, we have:*

$$\mathbf{Pr}[X \geq t] \leq \frac{\mathbf{E}[X]}{t}.$$

Proof: Assume that this is false, and there exists $t_0 > 0$ such that $\mathbf{Pr}[X \geq t_0] > \frac{\mathbf{E}[X]}{t_0}$. However,

$$\begin{aligned}\mathbf{E}[X] &= \sum_x x \cdot \mathbf{Pr}[X = x] = \sum_{x<t_0} x \cdot \mathbf{Pr}[X = x] + \sum_{x \geq t_0} x \cdot \mathbf{Pr}[X = x] \\ &\geq 0 + t_0 \cdot \mathbf{Pr}[X \geq t_0] > 0 + t_0 \cdot \frac{\mathbf{E}[X]}{t_0} = \mathbf{E}[X],\end{aligned}$$

a contradiction. ■

Theorem 25.1.2 (Chebychev inequality) *Let X be a random variable with $\mu_x = \mathbf{E}[X]$ and σ_x be the standard deviation of X. That is $\sigma_X^2 = \mathbf{E}\Big[(X - \mu_x)^2\Big]$. Then, $\mathbf{Pr}[|X - \mu_X| \geq t\sigma_X] \leq \frac{1}{t^2}$.*

Proof: Note that

$$\mathbf{Pr}[|X - \mu_X| \geq t\sigma_X] = \mathbf{Pr}\Big[(X - \mu_X)^2 \geq t^2\sigma_X^2\Big].$$

Set $Y = (X - \mu_X)^2$. Clearly, $E\Big[Y\Big] = \sigma_X^2$. Now, apply Markov inequality to Y. ■

Definition 25.1.3 Variables X, Y are *independent* if for any x, y we have:

$$\mathbf{Pr}[(X = x) \cap (Y = y)] = \mathbf{Pr}[X = x] \cdot \mathbf{Pr}[Y = y].$$

The following is easy to verify:

Claim 25.1.4 *If X and Y are independent, then $\mathbf{E}[XY] = \mathbf{E}[X]\,\mathbf{E}[Y]$.*
If X and Y are independent then $Z = e^X, W = e^Y$ are also independent variables.

25.2 Tail Inequalities

25.2.1 The Chernoff Bound — Special Case

Theorem 25.2.1 *Let $X_1, \ldots, X_n$ be n independent random variables, such that $\mathbf{Pr}[X_i = 1] = \mathbf{Pr}[X_i = -1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^n X_i$. Then, for any $\Delta > 0$, we have*

$$\mathbf{Pr}[Y \geq \Delta] \leq e^{-\Delta^2/2n}.$$

Proof: Clearly, for an arbitrary t, to specified shortly, we have

$$\mathbf{Pr}[Y \geq \Delta] = \mathbf{Pr}[\exp(tY) \geq \exp(t\Delta)] \leq \frac{\mathbf{E}[\exp(tY)]}{\exp(t\Delta)},$$

the first part follows by the fact that $\exp(\cdot)$ preserve ordering, and the second part follows by the Markov inequality.

Observe that

$$\begin{aligned}
\mathbf{E}[\exp(tX_i)] &= \frac{1}{2}e^{t} + \frac{1}{2}e^{-t} = \frac{e^t + e^{-t}}{2} \\
&= \frac{1}{2}\left(1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots\right) \\
&+ \frac{1}{2}\left(1 - \frac{t}{1!} + \frac{t^2}{2!} - \frac{t^3}{3!} + \cdots\right) \\
&= \left(1 + \quad + \frac{t^2}{2!} + \quad + \cdots + \frac{t^{2k}}{(2k)!} + \cdots\right),
\end{aligned}$$

by the Taylor expansion of $\exp(\cdot)$. Note, that $(2k)! \geq (k!)2^k$, and thus

$$\mathbf{E}[\exp(tX_i)] = \sum_{i=0}^{\infty} \frac{t^{2i}}{(2i)!} \leq \sum_{i=0}^{\infty} \frac{t^{2i}}{2^i(i!)} = \sum_{i=0}^{\infty} \frac{1}{i!}\left(\frac{t^2}{2}\right)^i = \exp\left(t^2/2\right),$$

again, by the Taylor expansion of $\exp(\cdot)$. Next, by the independence of the X_is, we have

$$\begin{aligned}
\mathbf{E}[\exp(tY)] &= \mathbf{E}\left[\exp\left(\sum_i tX_i\right)\right] = \mathbf{E}\left[\prod_i \exp(tX_i)\right] = \prod_{i=1}^{n} \mathbf{E}[\exp(tX_i)] \\
&\leq \prod_{i=1}^{n} e^{t^2/2} = e^{nt^2/2}.
\end{aligned}$$

We have

$$\mathbf{Pr}[Y \geq \Delta] \leq \frac{\exp\left(nt^2/2\right)}{\exp(t\Delta)} = \exp\left(nt^2/2 - t\Delta\right).$$

Next, by minimizing the above quantity for t, we set $t = \Delta/n$. We conclude,

$$\mathbf{Pr}[Y \geq \Delta] \leq \exp\left(\frac{n}{2}\left(\frac{\Delta}{n}\right)^2 - \frac{\Delta}{n}\Delta\right) = \exp\left(-\frac{\Delta^2}{2n}\right).$$

■

By the symmetry of Y, we get the following:

Corollary 25.2.2 *Let $X_1, \ldots, X_n$ be n independent random variables, such that $\mathbf{Pr}[X_i = 1] = \mathbf{Pr}[X_i = -1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have*

$$\mathbf{Pr}[|Y| \geq \Delta] \leq 2e^{-\Delta^2/2n}.$$

Corollary 25.2.3 *Let $X_1, \ldots, X_n$ be n independent* coin flips*, such that $\mathbf{Pr}[X_i = 0] = \mathbf{Pr}[X_i = 1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have*

$$\mathbf{Pr}\left[\left|Y - \frac{n}{2}\right| \geq \Delta\right] \leq 2e^{-2\Delta^2/n}.$$

Remark 25.2.4 Before going any further, it is might be instrumental to understand what this inequalities imply. Consider then case where X_i is either zero or one with probability half. In this case $\mu = \mathbf{E}[Y] = n/2$. Set $\delta = t\sqrt{n}$ ($\sqrt{\mu}$ is approximately the standard deviation of X if $p_i = 1/2$). We have by

$$\mathbf{Pr}\left[\left|Y - \frac{n}{2}\right| \geq \Delta\right] \leq 2\exp\left(-2\Delta^2/n\right) = 2\exp\left(-2(t\sqrt{n})^2/n\right) = 2\exp\left(-2t^2\right).$$

Thus, Chernoff inequality implies exponential decay (i.e., $\leq 2^{-t}$) with t standard deviations, instead of just polynomial (like the Chebychev's inequality).

25.2.2 The Chernoff Bound — General Case

Here we present the Chernoff bound in a more general settings.

Question 25.2.5 *Let*

1. $X_1, \ldots, X_n$ - *n independent Bernoulli trials, where*

$$\mathbf{Pr}[X_i = 1] = p_i, \text{ and } \mathbf{Pr}[X_i = 0] = q_i = 1 - p_i.$$

   Each X_i is known as a Poisson trials.

2. $X = \sum_{i=1}^{b} X_i$. $\mu = \mathbf{E}[X] = \sum_i p_i$.

Question*: Probability that $X > (1+\delta)\mu$?*

Theorem 25.2.6 *For any $\delta > 0$, we have* $\mathbf{Pr}[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$.

Or in a more simplified form, for any $\delta \leq 2e - 1$,

$$\mathbf{Pr}[X > (1+\delta)\mu] < \exp\left(-\mu\delta^2/4\right), \tag{25.1}$$

and

$$\mathbf{Pr}[X > (1+\delta)\mu] < 2^{-\mu(1+\delta)}, \tag{25.2}$$

for $\delta \geq 2e - 1$.

Proof: We have $\mathbf{Pr}[X > (1+\delta)\mu] = \mathbf{Pr}\left[e^{tX} > e^{t(1+\delta)\mu}\right]$. By the Markov inequality, we have:

$$\mathbf{Pr}[X > (1+\delta)\mu] < \frac{\mathbf{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}}$$

On the other hand,

$$\mathbf{E}\left[e^{tX}\right] = \mathbf{E}\left[e^{t(X_1+X_2\ldots+X_n)}\right] = \mathbf{E}\left[e^{tX_1}\right]\cdots\mathbf{E}\left[e^{tX_n}\right].$$

Namely,

$$\mathbf{Pr}[X > (1+\delta)\mu] < \frac{\prod_{i=1}^{n}\mathbf{E}\left[e^{tX_i}\right]}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n}\left((1-p_i)e^0 + p_i e^t\right)}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n}(1+p_i(e^t-1))}{e^{t(1+\delta)\mu}}.$$

Let $y = p_i(e^t - 1)$. We know that $1 + y < e^y$ (since $y > 0$). Thus,

$$\begin{aligned}
\mathbf{Pr}[X > (1+\delta)\mu] &< \frac{\prod_{i=1}^{n}\exp(p_i(e^t-1))}{e^{t(1+\delta)\mu}}. = \frac{\exp(\sum_{i=1}^{n} p_i(e^t-1))}{e^{t(1+\delta)\mu}} \\
&= \frac{\exp((e^t-1)\sum_{i=1}^{n} p_i)}{e^{t(1+\delta)\mu}} = \frac{\exp((e^t-1)\mu)}{e^{t(1+\delta)\mu}} = \left(\frac{\exp(e^t-1)}{e^{t(1+\delta)}}\right)^{\mu} \\
&= \left(\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right)^{\mu},
\end{aligned}$$

if we set $t = \log(1+\delta)$.

For the proof of the simplified form, see Section 25.2.3. ■

Definition 25.2.7 $F^+(\mu,\delta) = \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$.

Example 25.2.8 Arkansas Aardvarks win a game with probability $1/3$. What is their probability to have a winning season with n games. By Chernoff inequality, this probability is smaller than

$$F^+(n/3, 1/2) = \left[\frac{e^{1/2}}{1.5^{1.5}}\right]^{n/3} = (0.89745)^{n/3} = 0.964577^n.$$

For $n = 40$, this probability is smaller than 0.236307. For $n = 100$ this is less than 0.027145. For $n = 1000$, this is smaller than $2.17221 \cdot 10^{-16}$ (which is pretty slim and shady). Namely, as the number of experiments is increases, the distribution converges to its expectation, and this converge is exponential.

Theorem 25.2.9 *Under the same assumptions as Theorem 25.2.6, we have:*

$$\mathbf{Pr}[X < (1-\delta)\mu] < e^{-\mu\delta^2/2}.$$

Values	Probabilities	Inequality	Ref
$-1, +1$	$\mathbf{Pr}[X_i = -1] =$ $\mathbf{Pr}[X_i = 1] = \frac{1}{2}$	$\mathbf{Pr}[Y \geq \Delta] \leq e^{-\Delta^2/2n}$ $\mathbf{Pr}[Y \leq -\Delta] \leq e^{-\Delta^2/2n}$ $\mathbf{Pr}[\lvert Y\rvert \geq \Delta] \leq 2e^{-\Delta^2/2n}$	Theorem 25.2.1 Theorem 25.2.1 Corollary 25.2.2
$0, 1$	$\mathbf{Pr}[X_i = 0] =$ $\mathbf{Pr}[X_i = 1] = \frac{1}{2}$	$\mathbf{Pr}\Big[\Big\lvert Y - \frac{n}{2}\Big\rvert \geq \Delta\Big] \leq 2e^{-2\Delta^2/n}$	Corollary 25.2.3
0,1	$\mathbf{Pr}[X_i = 0] = 1 - p_i$ $\mathbf{Pr}[X_i = 1] = p_i$	$\mathbf{Pr}[Y > (1+\delta)\mu] < \Big(\frac{e^\delta}{(1+\delta)^{1+\delta}}\Big)^\mu$	Theorem 25.2.6
	For $\delta \leq 2e - 1$ $\delta \geq 2e - 1$ For $\delta \geq 0$	$\mathbf{Pr}[Y > (1+\delta)\mu] < \exp\Big(-\mu\delta^2/4\Big)$ $\mathbf{Pr}[Y > (1+\delta)\mu] < 2^{-\mu(1+\delta)}$ $\mathbf{Pr}[Y < (1-\delta)\mu] < \exp\Big(-\mu\delta^2/2\Big)$	Theorem 25.2.6 Theorem 25.2.9

Table 25.1: Summary of Chernoff type inequalities covered. Here we have n variables $X_1, \ldots, X_n$, $Y = \sum_i X_i$ and $\mu = \mathbf{E}[Y]$.

Definition 25.2.10 $F^-(\mu, \delta) = e^{-\mu\delta^2/2}$.

$\Delta^-(\mu, \varepsilon)$ - what should be the value of δ, so that the probability is smaller than ε.

$$\Delta^-(\mu, \varepsilon) = \sqrt{\frac{2 \log 1/\varepsilon}{\mu}}$$

For large δ:

$$\Delta^+(\mu, \varepsilon) < \frac{\log_2(1/\varepsilon)}{\mu} - 1$$

25.2.3 A More Convenient Form

Proof: **(of simplified form of Theorem 25.2.6)** Eq. (25.2) is just Exercise 25.4.1. As for Eq. (25.1), we prove this only for $\delta \leq 1/2$. For details about the case $1/2 \leq \delta \leq 2e - 1$, see [MR95]. By Theorem 25.2.6, we have

$$\mathbf{Pr}[X > (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu = \exp(\mu\delta - \mu(1+\delta)\ln(1+\delta)).$$

The Taylor expansion of $\ln(1+\delta)$ is

$$\delta - \frac{\delta^2}{2} + \frac{\delta^3}{3} - \frac{\delta^4}{4} + \cdot \geq \delta - \frac{\delta^2}{2},$$

for $\delta \leq 1$. Thus,

$$\begin{aligned} \mathbf{Pr}[X > (1+\delta)\mu] &< \exp\Big(\mu\Big(\delta - (1+\delta)\Big(\delta - \delta^2/2\Big)\Big)\Big) = \exp\Big(\mu\Big(\delta - \delta + \delta^2/2 - \delta^2 + \delta^3/2\Big)\Big) \\ &\leq \exp\Big(\mu\Big(-\delta^2/2 + \delta^3/2\Big)\Big) \leq \exp\Big(-\mu\delta^2/4\Big), \end{aligned}$$

for $\delta \leq 1/2$. ■

25.3 Bibliographical notes

The exposition here follows more or less the exposition in [MR95]. The special symmetric case (Theorem 25.2.1) is taken from [Cha01], although the proof is only very slightly simpler than the generalized form, it does yield a slightly better constant, and it would be useful when discussing discrepancy.

An orderly treatment of probability is outside the scope of our discussion. The standard text on the topic is the book by Feller [Fel91]. A more accessible text might be any introductory undergrad text on probability, in particular [MN98] has a nice chapter on the topic.

Exercise 25.4.2 (without the hint) is from [Mat99].

25.4 Exercises

Exercise 25.4.1 (Simpler Tail Inequality.) [1 Points]

[2 Points] Prove that for $\delta > 2e - 1$, we have

$$F^+(\mu, \delta) < \left[\frac{e}{1+\delta}\right]^{(1+\delta)\mu} \leq 2^{-(1+\delta)\mu}.$$

Exercise 25.4.2 (Chernoff inequality is tight.) [10 Points]

Let $S = \sum_{i=1}^{n} S_i$ be a sum of n independent random variables each attaining values $+1$ and -1 with equal probability. Let $P(n, \Delta) = \mathbf{Pr}[S > \Delta]$. Prove that for $\Delta \leq n/C$,

$$P(n, \Delta) \geq \frac{1}{C} \exp\left(-\frac{\Delta^2}{Cn}\right),$$

where C is a suitable constant. That is, the well-known Chernoff bound $P(n, \Delta) \leq \exp(-\Delta^2/2n))$ is close to the truth.

[**Hint:** Use Stirling's formula. There is also an elementary solution, using estimates for the middle binomial coffieicnets [MN98, pages 83–84], but this solution is considerably more involved and yields unfriendly constants.**]**

Exercise 25.4.3 (Tail inequality for geometric variables.) [10 Points]

Let $X_1, \ldots, X_m$ be m independent random variables with geometric distribution with probability p (i.e., $\mathbf{Pr}[X_i = j] = (1-p)^{j-1}p$). Let $Y = \sum_i X_i$, and let $\mu = \mathbf{E}[Y] = m/p$. Prove that

$$\mathbf{Pr}[Y \geq (1+\delta)\mu] \leq \exp\left(-\frac{m\delta^2}{8}\right).$$

Bibliography

[AACS98] P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete Comput. Geom.*, 19:315–331, 1998.

[AB99] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge, 1999.

[AC07] P. Afshani and T. M. Chan. On approximate range counting and depth. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 337–343, 2007.

[Ach01] D. Achlioptas. Database-friendly random projections. In *Proc. 20th ACM Sympos. Principles Database Syst.*, pages 274–281, 2001.

[ACNS82] M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. Crossing-free subgraphs. *Ann. Discrete Math.*, 12:9–12, 1982.

[AD97] P. K. Agarwal and P. K. Desikan. An efficient algorithm for terrain simplification. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 139–147, 1997.

[AEIS99] A. Amir, A. Efrat, P. Indyk, and H. Samet. Efficient algorithms and regular data structures for dilation, location and proximity problems. In *Proc. 40th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 160–170, 1999.

[AG86] N. Alon and E. Győri. The number of small semispaces of a finite set of points in the plane. *J. Combin. Theory Ser. A*, 41:154–157, 1986.

[AGK+01] V. Arya, N. Garg, R. Khandekar, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *Proc. 33rd Annu. ACM Sympos. Theory Comput.*, pages 21–29, 2001.

[AH05] B. Aronov and S. Har-Peled. On approximating the depth and related problems. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, pages 886–894, 2005.

[AHK06] S. Arora, E. Hazan, and S. Kale. Multiplicative weights method: a meta-algorithm and its applications. manuscript. Available from , 2006.

[AHS07] B. Aronov, S. Har-Peled, and M. Sharir. On approximate halfspace range counting and relative epsilon-approximations. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 327–336, 2007.

[AHV04] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51(4):606–635, 2004.

[AHY07] P. Agarwal, S. Har-Peled, and H. Yu. Embeddings of surfaces, curves, and moving points in euclidean space. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 381–389, 2007.

[AKPW95] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, February 1995.

[AM94] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11:393–418, 1994.

[AM98] S. Arya and D. Mount. ANN: library for approximate nearest neighbor searching. `http://www.cs.umd.edu/~mount/ANN/`, 1998.

[AM02] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 147–155, 2002.

[AM04] S. Arya and D. M. Mount. Computational geometry: Proximity and location. In D. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, chapter 63. CRC Press LLC, Boca Raton, FL, 2004. to appear.

[Ame94] N. Amenta. Helly-type theorems and generalized linear programming. *Discrete Comput. Geom.*, 12:241–261, 1994.

[AMM02] S. Arya, T. Malamatos, and D. M. Mount. Space-efficient approximate Voronoi diagrams. In *Proc. 34th Annu. ACM Sympos. Theory Comput.*, pages 721–730, 2002.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. Assoc. Comput. Mach.*, 45(6), 1998.

[AMS94] P. K. Agarwal, J. Matoušek, and O. Schwarzkopf. Computing many faces in arrangements of lines and segments. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 76–84, 1994.

[APV02] P. K. Agarwal, C. M. Procopiuc, and K. R. Varadarajan. Approximation algorithms for k-line center. In *Proc. 10th Annu. European Sympos. Algorithms*, pages 54–63, 2002.

[Aro98] S. Arora. Polynomial time approximation schemes for euclidean TSP and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5):753–782, Sep 1998.

[AS00] N. Alon and J. H. Spencer. *The probabilistic method*. Wiley Inter-Science, 2nd edition, 2000.

[Aur91] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[Bal97] K. Ball. An elementary introduction to modern convex geometry. In *Flavors of geometry*, volume MSRI Publ. 31. Cambridge Univ. Press, 1997. `http://www.msri.org/publications/books/Book31/files/ball.pdf`.

[Bar96] Y. Bartal. Probabilistic approximations of metric space and its algorithmic application. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 183–193, October 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 161–168, 1998.

[Bar02] A. Barvinok. *A course in convexity*, volume 54 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2002.

[BEG94] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *J. Comput. Syst. Sci.*, 48:384–409, 1994.

[BH01] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38:91–109, 2001.

[BHI02] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via coresets. In *Proc. 34th Annu. ACM Sympos. Theory Comput.*, pages 250–257, 2002.

[BM58] G. E.P. Box and M. E. Muller. A note on the generation of random normal deviates. *Annl. Math. Stat.*, 28:610–611, 1958.

[BMP05] P. Brass, W. Moser, and J. Pach. *Research Problems in Discrete Geometry*. Springer, 2005.

[BVZ01] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. Translated by H. Brönnimann.

[Cal95] P. B. Callahan. *Dealing with higher dimensions: the well-separated pair decomposition and its applications*. Ph.D. thesis, Dept. Comput. Sci., Johns Hopkins University, Baltimore, Maryland, 1995.

[Car76] L. Carroll. The hunting of the snark, 1876.

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[Cha96] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 284–290, 1996.

[Cha98] T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comput. Geom.*, 20:359–373, 1998.

[Cha01] B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, New York, 2001.

[Cha02] T. M. Chan. Closest-point problems simplified on the ram. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 472–473. Society for Industrial and Applied Mathematics, 2002.

[Cha05] T. M. Chan. Low-dimensional linear programming with violations. *SIAM J. Comput.*, pages 879–893, 2005.

[Cha06] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1-2):20–35, 2006.

[Che86] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1986.

[Che06] K. Chen. On k-median clustering in high dimensions. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 1177–1185, 2006.

[Che07] K. Chen. A constant factor approximation algorithm for k-median with outliers. manuscript, 2007.

[CK95] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.*, 42:67–90, 1995.

[CKMN01] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 642–651, 2001.

[CKR01] G. Calinescu, H. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 8–16. Society for Industrial and Applied Mathematics, 2001.

[Cla83] K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 226–232, 1983.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[Cla88] K. L. Clarkson. Applications of random sampling in computational geometry, II. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 1–11, 1988.

[Cla93] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3th Workshop Algorithms Data Struct.*, volume 709 of *Lect. Notes in Comp. Sci.*, pages 246–252. Springer-Verlag, 1993.

[Cla94] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 160–164, 1994.

[Cla95] K. L. Clarkson. Las Vegas algorithms for linear and integer programming. *J. Assoc. Comput. Mach.*, 42:488–499, 1995.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press / McGraw-Hill, Cambridge, Mass., 2001.

[CM96] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *J. Algorithms*, 21:579–597, 1996.

[CMS93] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom. Theory Appl.*, 3(4):185–212, 1993.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[CS00] N. Cristianini and J. Shaw-Taylor. *Support Vector Machines*. Cambridge Press, 2000.

[CW89] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4:467–489, 1989.

[dBDS95] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discrete Comput. Geom.*, 14:261–286, 1995.

[dBS95] M. de Berg and O. Schwarzkopf. Cuttings and applications. *Internat. J. Comput. Geom. Appl.*, 5:343–355, 1995.

[dBvKOS00] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[Dey98] T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete Comput. Geom.*, 19(3):373–382, 1998.

[DG03] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Rand. Struct. Alg.*, 22(3):60–65, 2003.

[DK85] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6:381–392, 1985.

[DNIM04] M. Datar, Immorlica N, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 253–262, 2004.

[Dud74] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.

[Dun99] C. A. Duncan. *Balanced Aspect Ratio Trees*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, 1999.

[Dur95] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, August 1995.

[EGS05] D. Eppstein, M. T. Goodrich, and J. Z. Sun. The skip quadtree: a simple dynamic data structure for multidimensional data. In *Proc. 21st Annu. ACM Sympos. Comput. Geom.*, pages 296–305. ACM, June 2005.

[EK89] O. Egecioglu and B. Kalantari. Approximating the diameter of a set of points in the Euclidean space. *Inform. Process. Lett.*, 32:205–211, 1989.

[Ele97] G. Elekes. On the number of sums and products. *ACTA Arithmetica*, pages 365–367, 1997.

[ERvK96] H. Everett, J.-M. Robert, and M. van Kreveld. An optimal algorithm for the ($\leq k$)-levels, with applications to separation and transversal problems. *Internat. J. Comput. Geom. Appl.*, 6:247–261, 1996.

[Fel71] W. Feller. *An Introduction to Probability Theory and its Applications*, volume II. John Wiley & Sons, NY, 1971.

[Fel91] W. Feller. *An Introduction to Probability Theory and its Applications*. John Wiley & Sons, NY, 1991.

[FG88] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 434–444, 1988.

[FGK+00] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the design of CGAL a computational geometry algorithms library. *Softw. – Pract. Exp.*, 30(11):1167–1202, 2000.

[FH05] J. Fischer and S. Har-Peled. Dynamic well-separated pair decomposition made easy. In *CCCG*, pages 235–238, 2005.

[FH06] J. Fischer and S. Har-Peled. On coresets for clustering and related problems. manuscript, 2006.

[For97] S. Fortune. Voronoi diagrams and Delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 20. CRC Press LLC, Boca Raton, FL, 1997.

[FRT03] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. 35th Annu. ACM Sympos. Theory Comput.*, pages 448–455, 2003.

[FS97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[Gar82] I. Gargantini. An effective way to represent quadtrees. *Commun. ACM*, 25(12):905–910, 1982.

[Gar02] R. J. Gardner. The Brunn-Minkowski inequality. *Bull. Amer. Math. Soc.*, 39:355–405, 2002.

[GK92] P. Gritzmann and V. Klee. Inner and outer j-radii of convex bodies in finite-dimensional normed spaces. *Discrete Comput. Geom.*, 7:255–280, 1992.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg, 2nd edition, 1988. 2nd edition 1994.

[Gol95] M. Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACT News*, 26(2):96–104, 1995.

[Gon85] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[GP84] J. E. Goodman and R. Pollack. On the number of k-subsets of a set of n points in the plane. *J. Combin. Theory Ser. A*, 36:101–104, 1984.

[GRSS95] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2:3–27, 1995.

[Grü03] B. Grünbaum. *Convex Polytopes*. Springer, 2nd edition, May 2003. Prepared by Volker Kaibel, Victor Klee, and Günter Ziegler.

[GT00] A. Gupta and E. Tardos. A constant factor approximation algorithm for a class of classification problems. In *Proc. 32nd Annu. ACM Sympos. Theory Comput.*, pages 652–658, 2000.

[Gup00] A. Gupta. *Embeddings of Finite Metrics*. PhD thesis, University of California, Berkeley, 2000.

[Har00a] S. Har-Peled. Constructing planar cuttings in theory and practice. *SIAM J. Comput.*, 29(6):2016–2039, 2000.

[Har00b] S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Comput.*, 30(4):1341–1367, 2000.

[Har01a] S. Har-Peled. A practical approach for computing the diameter of a point-set. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 2001.

[Har01b] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 94–103, 2001.

[HI00] S. Har-Peled and P. Indyk. When crossings count - approximating the minimum spanning tree. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 166–175, 2000.

[HM03] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing disc. In *Proc. 11th Annu. European Sympos. Algorithms*, volume 2832 of *Lect. Notes in Comp. Sci.*, pages 278–288. Springer-Verlag, 2003.

[HM04] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. In *Proc. 36th Annu. ACM Sympos. Theory Comput.*, pages 291–300, 2004.

[HM06] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.

[HS06] S. Har-Peled and M. Sharir. Relative ε-approximations in geometry. Manuscript. Available from `http://www.uiuc.edu/~sariel/papers/06/integrate`, 2006.

[HÜ05] S. Har-Peled and A. Üngör. A time-optimal delaunay refinement algorithm in two dimensions. In *Proc. 21st Annu. ACM Sympos. Comput. Geom.*, pages 228–236, 2005.

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 604–613, 1998.

[Ind99] P. Indyk. Sublinear time algorithms for metric space problems. In *Proc. 31st Annu. ACM Sympos. Theory Comput.*, pages 154–159, 1999.

[Ind01] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 10–31, 2001. Tutorial.

[Ind04] P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 39, pages 877–892. CRC Press LLC, Boca Raton, FL, 2nd edition, 2004.

[Joh48] F. John. Extremum problems with inequalities as subsidary conditions. *Courant Anniversary*, pages 187–204, 1948.

[Kal92] G. Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 475–482, 1992.

[KF93] I. Kamel and C. Faloutsos. On packing r-trees. In *Proc. 2nd Intl. CConf. Info. Knowl. Mang.*, pages 490–499, 1993.

[Kle02] J. Kleinberg. An impossibility theorem for clustering. In *Neural Info. Proc. Sys.*, 2002.

[KLMN04] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metric spaces. In *Proc. 45th Annu. IEEE Sympos. Found. Comput. Sci.*, page to appear, 2004.

[KMN+04] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom. Theory Appl.*, 28:89–112, 2004.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 2(30):457–474, 2000.

[KS06] H. Kaplan and M. Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar voronoi diagrams, and approximate range counting. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 484–493, 2006.

[KT06] J. Kleinberg and E. Tardos. *Algorithm design*. Addison-Wesley, 2006.

[Lei84] F. T. Leighton. New lower bound techniques for VLSI. *Math. Syst. Theory*, 17:47–70, 1984.

[Leo98] S. J. Leon. *Linear Algebra with Applications*. Prentice Hall, 5th edition, 1998.

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318, 1988.

[LLS01] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. *J. Comput. Syst. Sci.*, 62(3):516–527, 2001.

[Mac50] A.M. Macbeath. A compactness theorem for affine equivalence-classes of convex regions. *Canad. J. Math*, 3:54–61, 1950.

[Mag02] A. Magen. Dimensionality reductions that preserve volumes and distance to affine spaces, and their algorithmic applications. In *The 6th Intl. Work. Rand. Appr. Tech. Comp. Sci.*, pages 239–253, 2002.

[Mat90] J. Matoušek. Bi-lipschitz embeddings into low-dimensional euclidean spaces. *Comment. Math. Univ. Carolinae*, 31:589–600, 1990.

[Mat92] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[Mat95a] J. Matoušek. On enclosing k points by a circle. *Inform. Process. Lett.*, 53:217–221, 1995.

[Mat95b] J. Matoušek. On geometric optimization with few violated constraints. *Discrete Comput. Geom.*, 14:365–384, 1995.

[Mat98] J. Matoušek. On constants for cuttings in the plane. *Discrete Comput. Geom.*, 20:427–448, 1998.

[Mat99] J. Matoušek. *Geometric Discrepancy*. Springer, 1999.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[Meg83] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12(4):759–776, 1983.

[Meg84] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31:114–127, 1984.

[Mil04] G. L. Miller. A time efficient Delaunay refinement algorithm. In *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms*, pages 400–409, 2004.

[MN98] J. Matoušek and J. Nešetřil. *Invitation to Discrete Mathematics*. Oxford Univ Pr, 1998.

[MP03] R. R. Mettu and C. G. Plaxton. The online median problem. *SIAM J. Comput.*, 32(3):816–832, 2003.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[Mul94a] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[Mul94b] K. Mulmuley. An efficient algorithm for hidden surface removal, II. *J. Comp. Sys. Sci.*, 49:427–453, 1994.

[O'R85] J. O'Rourke. Finding minimal enclosing boxes. *Internat. J. Comput. Inform. Sci.*, 14:183–199, 1985.

[OvL81] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[Rab76] M. O. Rabin. Probabilistic algorithms. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 21–39. Academic Press, New York, NY, 1976.

[Rup93] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 83–92, 1993.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

[Sag94] H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.

[Sam89] H. Samet. *Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods*. Addison-Wesley, Reading, MA, 1989.

[Sei91] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[Sei93] R. Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 37–68. Springer-Verlag, 1993.

[Sha03] M. Sharir. The Clarkson-Shor technique revisited and extended. *Comb., Prob. & Comput.*, 12(2):191–201, 2003.

[Smi00] M. Smid. Closest-point problems in computational geometry. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier Science Publishers B. V. North-Holland, Amsterdam, 2000.

[SSS02] Y. Sabharwal, N. Sharma, and S. Sen. Improved reductions of nearest neighbors search to plebs with applications to linear-sized approximate voronoi decopositions. In *Proc. 22nd Conf. Found. Soft. Tech. Theoret. Comput. Sci.*, pages 311–323, 2002.

[Sto91] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, New York, NY, 1991.

[SW92] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, volume 577 of *Lect. Notes in Comp. Sci.*, pages 569–579. Springer-Verlag, 1992.

[Szé97] L. A. Székely. Crossing numbers and hard Erdős problems in discrete geometry. *Combinatorics, Probability and Computing*, 6:353–358, 1997.

[Tót01] G. Tóth. Point sets with many k-sets. *Discrete Comput. Geom.*, 26(2):187–194, 2001.

[Tou83] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, pages A10.02/1–4, 1983.

[Üng04] A. Üngör. Off-centers: A new type of steiner points for computing size-optimal quality-guaranteed delaunay triangulations. In *Latin Amer. Theo. Inf. Symp.*, pages 152–161, 2004.

[Vai86] P. M. Vaidya. An optimal algorithm for the all-nearest-neighbors problem. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 117–122, 1986.

[Van97] R. J. Vanderbei. *Linear programming: Foundations and extensions*. Kluwer, 1997.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

[Wel86] E. Welzl. More on k-sets of finite sets in the plane. *Discrete Comput. Geom.*, 1:95–100, 1986.

[Wel92] E. Welzl. On spanning trees with low crossing numbers. In *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative*, volume 594 of *Lect. Notes in Comp. Sci.*, pages 233–249. Springer-Verlag, 1992.

[WVTP97] M. Waldvogel, G. Varghese, J. Turener, and B. Plattner. Scalable high speed ip routing lookups. In *Proc. ACM SIGCOMM 97*, Octeber 1997.

[YAPV04] H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using core sets. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 263–272, 2004.

Index