Algorithms for Data Depth

By

Dan Chen

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy
in
Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

April 2013

Your file Votre référence
ISBN: 978-0-494-94526-1

Our file Notre référence
ISBN: 978-0-494-94526-1

Abstract

The concept of data depth gives a tool for multivariate data analysis. It measures the centrality of a data point with respect to a data set. Many different notions of data depth have been introduced. In this thesis we explore efficient algorithms for computing several notions of data depth, including exact and approximation algorithms for Tukey (halfspace) depth [57, 93], Oja depth [75], and majority depth [61, 90].

Contents

List of Figures

Chapter 1

Introduction

1.1 Data Depth

The term *data depth* comes from non-parametric multivariate descriptive statistics. A *descriptive statistic* is used to summarize a collection of data, for example by estimating the center of the data set. In non-parametric statistics, the probability distribution of the population is not considered, and the test statistics are usually based on the rank of the data. In multivariate data analysis, every data item consists of several elements. The concept of data depth gives a foundation for multivariate data analysis. Broadly speaking, it is a method of generalizing the notions of rank and median to multivariate data. Data depth measures the centrality of a data item with respect to a data set, and gives a center-outward ordering of points in Euclidean space of any dimension. Methods based on data depth are also developed for robust statistics, whose motivation is to develop methods that are less influenced by abnormal observations. Since the multivariate data can be represented as points in Euclidean space $\mathbb{R}^d$, they are often called points in this thesis.

The depth of a point measures the centrality of this point with respect to a given set, S, of points in high dimensional space. Usually, the element of S with the largest depth is called the *median* of S, and a point in $\mathbb{R}^d$ with the largest depth is called the *center* of the data set. The median and center of S are often called *location estimators*, since they summarize the location of S as a single point.

In $\mathbb{R}^1$, the median holds the properties of high *breakdown point*, *affine invariance*, and *monotonicity*. Informally, the *breakdown point* of a measure is the fraction of the input that must be moved to infinity before the center moves to infinity [41, 63]. In $\mathbb{R}^1$, the median has a breakdown point of $\frac{1}{2}$ [3]. After an affine transformation on the data set, the median is subject to the same affine transformation. Therefore, it is *affine invariant*. If we move a point to one side, the median tends to move to that side, never moving to the opposite side; this property is called *monotonicity* [10]. For a good measure of data depth, the resulting "median" should also have these properties, ideally, to the same degree as the one-dimensional median does.

Depending on the particular application, the properties of high breakdown point, affine invariance, and monotonicity may be more or less important. Consider a medical study that has patient data including height, weight, and age. In this case the axes, height, weight, and age, are meaningful, and affine invariance is probably not a critical requirement. However, the results of the study should not change depending on whether weight is measured in pounds or kilograms, so the data depth measure used in this study should be invariant under *non-uniform* scaling of the axes.

In contrast, research that studies the distribution of stars in a galaxy should use affine invariant depth measures, since the coordinate system is essentially chosen arbitrarily rather than a property of the data. The importance of high breakdown point depends on how noisy the data acquisition process is. In a medical study run in a hospital setting, each patient's height and weight can be measured and recorded by a medical professional. Large measurement errors or data entry errors are unlikely, hence the data depth measure used need not have high breakdown point. In contrast, a similar study performed by using volunteers who enter their own information in a web-form should use a data depth measure with a higher breakdown.

Generalizing rank to higher dimensions is not straightforward. Many different notions of data depth have been introduced, such as *Tukey depth* [57, 93], *convex hull peeling depth* [9, 87], *Oja depth* [75], *simplicial depth* [60], *majority depth* [61, 90], *regression depth* [82], and so on. Each notion has different properties and requires different time complexity for computing. We can use the notion with the properties we need. The surveys [3, 51, 62, 77] give detailed introductions to these notions. In

this thesis we will explore algorithms for three depth measures: Tukey depth and Oja depth, and majority depth described in the next section. We focus on these three notions because they are among the most widely cited in the literature. Tukey's original paper [93] have over 500 citations and Oja's paper [75] has over 300 citations. Majority depth [61, 90] is less influential, but its definition makes it interesting from the point of view of combinatorial and computational geometry.

Data depth has been used to develop statistical tests and other multivariate data analysis [57, 60, 62]. It is also used to measure economic inequality and industrial concentration [71].

1.2 Depth Notions

In this section, we give the definitions of the depth notions we are interested in.

1.2.1 Tukey Depth

The Tukey depth [93] is also called *halfspace depth* or *location depth*. Given a set S of points and a point p in $\mathbb{R}^d$, the Tukey depth of p is defined as the minimum number of points of S contained in any closed halfspace that contains p.

$$\mathrm{dep}_{\mathrm{T}}(p, S) = \min\{|h \cap S| : h \text{ is a closed halfspace that contains } p.\}$$

The point with the largest depth is called *Tukey median* or *halfspace median*. In Figure 1.1, the Tukey depth of p is 3, because at least three points are contained in any closed halfspace with p on its boundary and there exists one halfspace containing only 3 points. And the depth of point p in Figure 1.2 is 0.

A data set is said to be in *general position* if it has no $d+1$ points of S lie on a common hyperplane. If the data set is in general position, computing the Tukey depth of a point is equivalent to the *open hemisphere problem* introduced by Johnson and Preparata. Given a set of n points on the unit sphere $\mathbb{S}^{d-1}$ in $\mathbb{R}^d$, the open hemisphere problem is to find an open hemisphere of $\mathbb{S}^{d-1}$ that contains as many

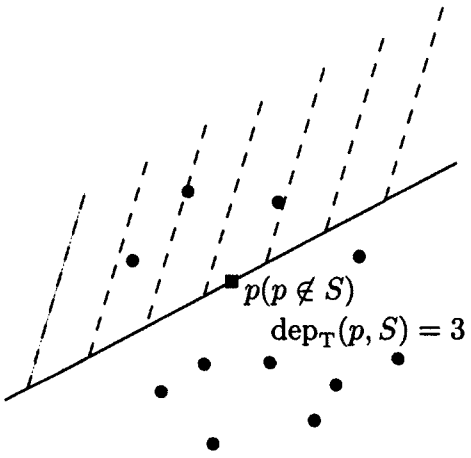

Figure 1.1: An example of Tukey depth in $\mathbb{R}^2$

points as possible. This problem is NP-complete if both n and d are parts of the input [58].

1.2.2 Oja Depth

Given a set S of n points in $\mathbb{R}^d$, the *Oja depth* [75] of a point $p \in \mathbb{R}^d$ is

$$\mathrm{dep}_{\mathrm{O}}(p, S) = \sum_{y_1,\ldots,y_d \in \binom{S}{d}} \mathrm{vol}(p, y_1, \ldots, y_d) \ ,$$

where $\mathrm{vol}(p_1, \ldots, p_{d+1})$ denotes the volume of the simplex whose vertices are $p_1 \ldots p_{d+1}$.[1] A point in $\mathbb{R}^d$ with the minimum Oja depth is called an *Oja center.* In Figure 1.3, the Oja depth of p is the sum of the area of all the triangles whose vertex set consists of p and 2 points of S.

1.2.3 Majority Depth

Let S be a set of points in $\mathbb{R}^d$. If the points in S are in general position, any d points in S define a unique hyperplane $\hbar$. With $\hbar$ as the common boundary, two closed

[1] In Oja's original definition, the sum is normalized by dividing by $\binom{|S|}{d}$. We omit this here since it changes none of our results and clutters our formulas.

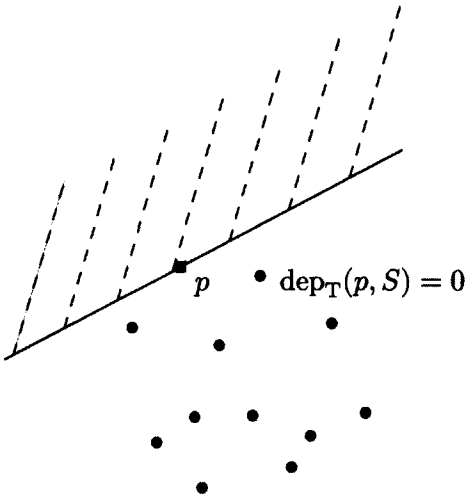

Figure 1.2: Another example of Tukey depth in $\mathbb{R}^2$

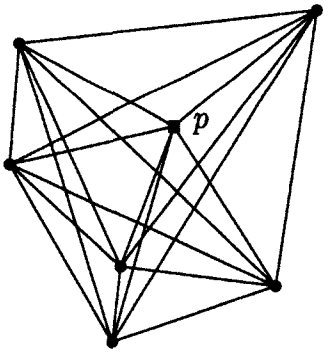

Figure 1.3: An example of Oja depth in $\mathbb{R}^2$

half-spaces are obtained. The one containing more than or equal to $\frac{n+d}{2}$ points is called the major side of $\hbar$ (see Figure 1.4). Note that halving hyperplanes have two major sides. Given a finite set S of n points and a point p in $\mathbb{R}^d$, the majority depth of p is the number of major sides it is in [61, 90]. For example, the majority depth of p in Figure 1.5 is 11.

1.3 Overview of This thesis

In this chapter we have reviewed the definitions of the Tukey depth, Oja depth, and majority depth. In Chapter 2 we explore the properties of these depth notions. The

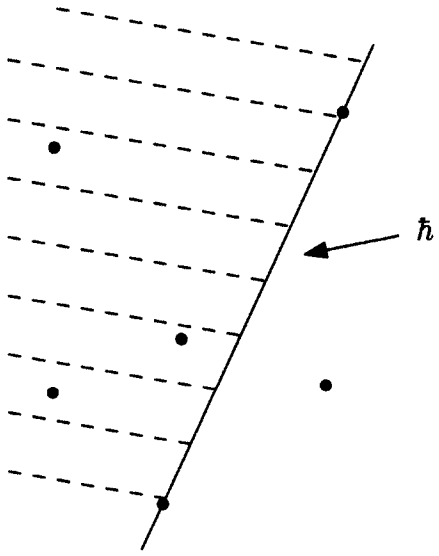

Figure 1.4: An example of major side in $\mathbb{R}^2$

original contributions of this thesis are as follows.

- In Chapter 3 we give necessary background for some techniques used by the algorithms in this thesis. This includes some new combinatorial geometry results on the sizes of pseudoballs in arrangements.

- In Chapter 4 we propose two algorithms for computing Tukey depth. One is a Fixed Parameter Tractable algorithm, which is fast when the depth is small. The other is an approximation algorithm, which is simple and fast.

- In Chapter 5 we discuss the relationships between the centers of mass of certain sets and Oja depth. We develop a centerpoint theorem for Oja depth, and prove that the Oja depth of the center of mass of the point set is an approximation of the minimum Oja depth.

- In Chapter 6, we propose three algorithms for majority depth in $\mathbb{R}^2$. One is an exact algorithm, and the other two are approximation algorithms.

Some of the results in this thesis have been published at the following conferences and journals.

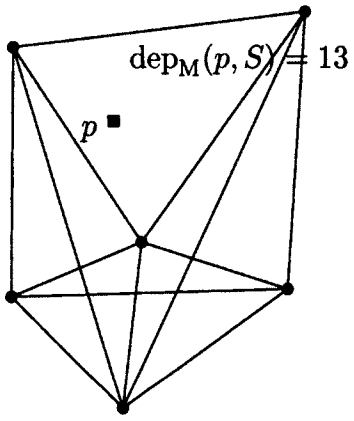

Figure 1.5: An example of majority depth in $\mathbb{R}^2$

- D. Bremner, D. Chen, J. Iacono, S. Langerman, and P. Morin. Output-sensitive algorithms for Tukey depth and related problems. *Statistics and Computing*, 18:259–266, 2008.

- D. Chen, P. Morin, and U. Wagner. Absolute approximation of Tukey depth: Theory and experiments. *Computational Geometry: Theory and Applications*, 46(5):566–573, 2013. Special Issue on Geometry and Optimization.

- D. Chen, O. Devillers, J. Iacono, S. Langerman, and P. Morin. Oja centers and centers of gravity. *Computational Geometry: Theory and Applications*, 46(2): 140–147, 2013. Special Issue on the Canadian Conference on Computational Geometry (CCCG 2010).

- D. Chen and P. Morin. Algorithms for bivariate majority depth. In *Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2011)*, pages 425–430, 2011.

- D. Chen and P. Morin. Approximating majority depth. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012)*, pages 237–242, 2012. Invited to special issue of Computational Geometry: Theory and Applications for CCCG 2012.

Chapter 2

Depth Functions and Properties

In this chapter, we formally defines the properties of data depth, and demonstrate the properties each depth notion has.

2.1 Properties

Given a set $S = \{S_1, \ldots, S_n\}$ of points and a point p in $\mathbb{R}^d$, we let $f_{\mathrm{d}}(p, S)$ denote the depth of point p with respect to S, and $\mathrm{T}(S)$ denote a location estimator of S. We have the following desirable properties for data depth:

1. Generalizing the one-dimensional median: Let u be a unit vector, and $u^t S$ be the one-dimensional projection of S.

$$\max_i f_{\mathrm{d}}(u^t S_i, u^t S) = f_{\mathrm{d}}(\mathrm{med}(u^t S), u^t S), \tag{2.1}$$

   for any unit vector u.

2. High breakdown point [41, 63]: The (finite-sample replacement version of the) breakdown point of a location estimator T at a set S is defined as

$$\min_{1 \le m \le n} \left\{ \frac{m}{n} : \sup_{X_m} \| \mathrm{T}(S) - \mathrm{T}(X_m) \| = \infty \right\} \tag{2.2}$$

   where X_m is a set of points such that $|X_m| = n$, and $|X_m \cap S| = n - m$.

3. Affine equivariant [40]:

$$\mathrm{T}(\{AS + b\}) = A\,\mathrm{T}(S) + b \tag{2.3}$$

for non-singular $d \times d$ matrix A and $b \in \mathbb{R}^d$. If we place appropriate restriction A and b, we obtain the weaker notion of affine equivariant, such as, *translation equivariant*, *rigid transformation equivariant*, *uniform scaling equivariant*, and *non-uniform scaling equivariant*.

4. Affine invariant [40]:

$$f_{\mathrm{d}}(Ap + b, \{AS_i + b\}) = f_{\mathrm{d}}(p, S). \tag{2.4}$$

With an appropriate restriction A, we can obtain rotation invariant, which is weaker than affine invariant.

5. Monotonicity [10]: Defined by Bassett [10] for 1-dimensional estimators. If we move a point $q \in S$ along a vector $\vec{v}$, $(\mathrm{T}(S) - \mathrm{T}(S')) \cdot \vec{v} \geq 0$. This is the same as the *non-decreasing in q* defined by Lopuhaä and Rousseeuw [63].

6. Continuous [44]: Given $\epsilon > 0$, function $f : S \to \mathbb{R}^2$ is an ϵ-perturbation on S if for all $q \in S$, $\|q - f(q)\| \leq \epsilon$. Let F_ϵ^S denote the set of all ϵ-perturbation on S. A median (depth) function is continuous if for all S in $\mathbb{R}^2$ and all $\delta > 0$ there exists an $\epsilon > 0$ such that for all $f \in F_\epsilon^S$,

$$\|\,\mathrm{T}(S) - \mathrm{T}(f(S))\| < \delta. \tag{2.5}$$

7. Decreasing along rays [61]: If x is a center point (with maximum depth value), $f_{\mathrm{d}}(p, S) \leq f_{\mathrm{d}}(x + \alpha(p - x), S)$, for all $p \in \mathbb{R}^d$ and all $\alpha \in [0, 1]$.

8. Level-convex: The boundary of the set of points in $\mathbb{R}^d$ with depth at least k is called the *contour* of depth k [40]. If all the contours f_{d} are convex, we say f_{d} is level-convex.

9. Convex: f_{d} is a convex function.

10. Vanishing at infinity [60, 99]: $f_{\mathrm{d}}(p, S)$ approaches zero, as $\|p\|$ approaches infinity.

11. Unique maximum point.

This is a long list of desirable properties and no depth notion satisfies all these requirements. We summarize the properties of some popular depth notions in Table 2.1.

	Rot. Inv.[1]	Aff. Inv.[2]	R. Tran. Equi.[3]	Tran. Equi.[4]
Tukey	yes	yes [40]	yes	yes
Oja	yes	no	yes	yes
Majority	yes	yes	yes	yes
	U. Scal. Equi.[5]	N. Scal. Equi.[6]	Monotone	Continuous
Tukey	yes	yes	unknown	no
Oja	yes	yes	unknown	unknown
Majority	yes	yes	no	no
	Decr.[7]	Level-convex	Convex	Van. inf.[8]
Tukey	yes	yes [40]	no	yes
Oja	yes	yes	yes	no
Majority	no	no	no	no
	Uni. Max.[9]	Gen. Med.[10]	B.D.P.[11] $\geq \frac{1}{d+1}$	B.D.P. $= \frac{1}{2}$
Tukey	no	yes	yes [40]	no [40]
Oja	no	yes	no [73]	no [73]
Majority	no	yes	unknown	unknown

[1] Rotation invariable.
[2] Affine invariant.
[3] Rigid translation equivariant.
[4] Translation equivariant.
[5] Uniform scaling equivariant.
[6] Non-uniform scaling equivariant.
[7] Decreasing along rays.
[8] Vanishing at infinity.
[9] Unique maximum point.
[10] Generalizing the one-dimensional median.
[11] Breakdown point.

Table 2.1: Depth notions and properties

2.1.1 Properties of Tukey Depth

In $\mathbb{R}^d$, the depth of the Tukey center is in the range of $\lceil \frac{n}{d+1} \rceil - 1$ and $\lceil \frac{n}{2} \rceil - 1$ [40]. The lower bound of $\lceil \frac{n}{d+1} \rceil - 1$ is known as a *Centerpoint Theorem:* For any point set in $\mathbb{R}^d$, there exists a point in $\mathbb{R}^d$ with depth $\lceil \frac{n}{d+1} \rceil - 1$. The breakdown point of Tukey depth is at least $\frac{1}{d+1}$ and can be as high as $\frac{1}{3}$ when $d \geq 2$ [40, 42]. It is affine invariant as affine transformations have no effect on the depth value. The contours of Tukey depth are all convex (see Figure 2.1). The contours are also nested, as the contour of depth $k+1$ is completely contained by the contour of depth k [40].

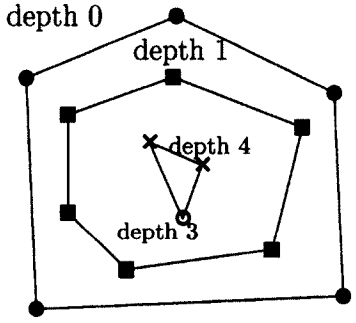

Figure 2.1: Tukey depth contours

2.1.2 Properties of Oja Depth

Oja depth is not affine invariant according to the definition by Donoho and Huber [40]. However, the relative order of the depth values of any two points will not be changed by affine transformations on the data set. In particular, for an affine transformation A that takes point p onto p' and the set of points S onto S', $\text{dep}_\text{O}(p, S) = c\,\text{dep}_\text{O}(p', S)$, where c depends only on A. It is unknown whether Oja depth satisfies any reasonable definition of monotonicity. It is known that it does not have high breakdown point [73].

2.1.3 Properties of Majority Depth

Majority depth is affine invariant [61], i.e. the depth value does not change after an affine transformation of S. Other properties are currently unknown.

Chapter 3

Background

This chapter gives the background for some of the algorithms in later chapters. Results in Section 3.1 are taken from Appendix A in Ref. [15]. Results in Section 3.2 are new and are taken from Ref. [27]. Results in Section 3.3 are taken from Ref. [25].

3.1 Computing a Basic Infeasible Subsystem

Throughout this thesis we assume that the reader is familiar with linear programming. For linear programming, we refer to the books by Chvátal [31], Schrijver [85], and Hillier and Lieberman [56]. This section explains how, given an infeasible linear program, to find a basic infeasible subsystem of that linear program. This routine is required as part of the algorithm for solving MAXIMUMFEASIBLESUBSYSTEM described in Section 4.1.

For any matrix M, let M_J denote the set of rows indexed by J. Given a system of linear inequalities $Mx \geq b$, $M \in \mathbb{R}^{m \times d}$, a *basic infeasible subsystem* is a subset of $\{1, \ldots, m\}$ such that the system $M_I x \geq b_I$ is infeasible , and $|I| \leq d + 1$. Similar to the standard first stage simplex method (see e.g. [31, p. 39]), let e denote the m-vector of all ones, c the length $d + 1$ binary vector with exactly one coordinate equal to one in the last position and let $A = [Me]$. we can write the first stage LP

for our system as

$$\begin{aligned} &\min c^T x = x_{d+1} \\ &\text{subject to } Ax \geq b. \end{aligned} \tag{P}$$

In the case of an infeasible system, the optimal value of this LP will be strictly positive. The dual LP of (P) is

$$\begin{aligned} &\max \quad b^T y \\ &\text{subject to } yA = c \\ &\qquad\qquad y \geq 0. \end{aligned} \tag{D}$$

In the following, we generally follow the notation of [70], except that we interchange the definitions of the primal and dual LPs. Define a *basic partition* (or just *basis*) (β, η) as a partition of the row indices of A such that A_β is nonsingular. For each basic partition, we define a *primal basic solution*

$$x^* = A_\beta^{-1} b_\beta$$

and a *dual basic solution*

$$y^* = c A_\beta^{-1}$$

We say that a basis is *primal feasible* (respectively *dual feasible*) if x^* is feasible for (P) (respectively y^* is feasible for (D)). It is a standard result of linear programming duality that a basis which is both primal and dual feasible defines a pair (x^*, y^*) of optimal solutions to the primal and dual LPs; such a partition is called an *optimal basis partition.*

In general LP algorithms (either directly in the case of simplex type algorithms, or via postprocessing using e.g. [70, 95, 11]) provide an optimal partition (β, η). Consider

the relaxed LP

$$\begin{aligned} &\min c^T x \\ &\text{subject to } A_\beta x \geq b_\beta. \end{aligned} \tag{R}$$

It is easy to verify that an optimal basis partition for (P) is also primal and dual feasible for (R). This implies that the system $M_\beta x \geq b_\beta$ is infeasible, and provides a basic infeasible system. Using interior point algorithms (see [53, 78, 94, 80, 81]), a solution to the first stage LP can be found in $O(m^3 L)$ time, where L is the number of bits in the input. Using the algorithm of Beling and Megiddo [11], an optimal basis partition can be computed in $O(m^{1.594} d)$ time (where the bound is based on the best known methods for fast matrix multiplication).

3.2 Arrangements of Hyperplanes

In this section we study the combinatorics of arrangements of hyperplanes, which gives the tools to prove the efficiency of the algorithms in Section 4.2.

Let H be a set of ℓ hyperplanes in $\mathbb{R}^d$. We say that H is in general position, if every subset of d hyperplanes intersect in one point, and no $d+1$ hyperplanes intersect in one point. We say a hyperplane is *vertical* if it contains a line parallel to the x_d-axis. Without loss of generality, we assume that no hyperplane in H is vertical.

Arrangements The *arrangement* $\mathcal{A}(H)$ of H is the partitioning of $\mathbb{R}^d$ induced by H into *vertices* (intersections of any d hyperplanes in H), *faces* (each flat in $\mathcal{A}(H)$ is divided into pieces by the hyperplanes in H that do not contain the flat, a k-face is a piece in a k-flat which is the intersection of $n-k$ hyperplanes [54, p. 7]), and *regions* (connected components in $\mathbb{R}^d$ separated by hyperplanes in H). We call $\mathcal{A}(H)$ a simple arrangement if H is in general position.

In an arrangement, we say a point p is at the j-level [2, 46, 55], if there are j hyperplanes in H lying vertically below p. (Above and below are with respect to the x_d coordinate.) The j-level of $\mathcal{A}(H)$ is the closure of all the points of H at level j.

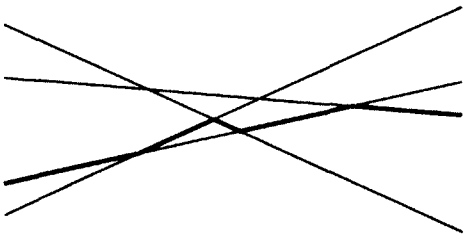

Figure 3.1: The 1-level of an arrangement in $\mathbb{R}^2$

Let m be the number of vertices of the j-level. Tight bounds for m are still open problems. In $\mathbb{R}^2$ the best known upper bound of m is $O(\ell j^{1/3})$ [38], and the best known lower bound for m is $\ell 2^{\Omega(\sqrt{\log j})}$ [91]. In $\mathbb{R}^2$, constructing the j-level takes $O((\ell + m)\log \ell)$ time using Edelsbrunner and Welzl's algorithm [48] with the data structure in [16], and it takes $O(\ell \log \ell + \ell j^{1/3})$ expected time with Chan's randomized algorithm [18] which is output insensitive. In the word RAM model, the construction takes $O\left((\ell + m)\frac{\log \ell}{\log\log \ell}\right)$ time [37].

Pseudo-distance Following Welzl [97], we use δ_H to denote the *pseudo-distance* for pairs of points (relative to H), where $\delta_H(p, q)$ is defined as the number of hyperplanes in H which have p and q on opposite sides. For a point p and an integer σ, we define the pseudo-ball $D_H(p, \sigma)$ as the set of vertices q in $\mathcal{A}(H)$ with $\delta_H(p, q) \leq \sigma$.

Our goal in this section is to show that arrangements have big pseudo-balls. In particular, we will prove

Lemma 3.1. *If H is a set of ℓ hyperplanes in general position in $\mathbb{R}^d$, and σ is an integer, $0 \leq \sigma \leq \ell - d$, then $|D_H(p, \sigma)| \geq \binom{\sigma+d}{d}$ for all vertices p disjoint from H.*

To prove this lemma, we need to use a result, due to Clarkson [33], on the number of *i-bases* in an arrangement. With this result we can prove a lower bound on the size of $D_H(p, \sigma)$. The following is a review of Clarkson's Theorem (with some modifications) on the number of i-bases, which is the main tool used to prove Theorem 4.2. The difference between this proof and the original is in the definition of i-basis.

Let $\mathcal{P}(H)$ be the convex polytope given by $\mathcal{P}(H) = \cap_{h \in H}(h \cup h^+)$, where h^+ is the open halfspace bounded by h and containing point $(\infty, 0, \ldots, 0)$. Let $G \subset H$,

$|G| \geq d$. Then we define $x^*(G)$ as the vertex of $\mathcal{P}(G)$ with lexicographically smallest coordinates. Note that $x^*(G)$ is well defined since $|G| \geq d$ and the hyperplanes in H are in general position. Also note that there exists one subset $B \subset G$ with $|B| = d$ and such that $x^*(B) = x^*(G)$. We call B the *basis* $b(G)$ of G. For any $B \in \binom{H}{d}$, let

$$I_B \equiv \{h \in H \mid b(B \cup \{h\}) \neq B\},$$

be the set of hyperplanes that *violate* $b(B)$. If $|I_B| = i$, B is called an i-basis.

Since any uniform random sample $R \in \binom{H}{r}$, where $d \leq r \leq \ell$, has exactly one basis, we have

$$1 = \sum_{B \in \binom{H}{d}} \Pr\{B = b(R)\} \quad \forall d \leq r \leq \ell. \tag{3.1}$$

Any $B \in \binom{H}{d}$ is the basis of R if and only if $B \subseteq R$ and R does not contain any element of I_B. If B is an i-basis, the probability that B is the basis of R is $\frac{\binom{\ell-i-d}{r-d}}{\binom{\ell}{r}}$. Let $g_i'(H)$ denote the number of i-bases in the arrangement. Equation (3.1) can be rewritten as

$$1 = \sum_{0 \leq i \leq \ell - d} \frac{\binom{\ell-i-d}{r-d}}{\binom{\ell}{r}} g_i'(H) \quad \forall d \leq r \leq \ell. \tag{3.2}$$

Equation (3.2) gives a system of $l - d + 1$ linear equations in $l - d + 1$ variables. Solving this system gives

$$g_i'(H) = \binom{i+d-1}{d-1}. \tag{3.3}$$

For more details see Clarkson [33]. Mulmuley also proves this result with a different method [72].

With this background, we are now ready to prove Lemma 3.1:

Proof (of Lemma 3.1). By a standard projective transformation, we can assume that all hyperplanes in H are below p [46]. An i-basis defines a vertex with distance to p no more than i. We know that the number of i-bases is $\binom{i+d-1}{d-1}$ in $\mathcal{A}(H)$. The number

of vertices with distance to p no more than σ is therefore at least

$$\sum_{i=0}^{\sigma} \binom{i+d-1}{d-1} = \binom{\sigma+d}{d}.$$

□

The bound in Lemma 3.1 is a generalization of the second result of Welzl [97] for the case $d = 2$. It also strengthens the bounds of Chazelle and Welzl [23] for $d \geq 3$. This bound is a lower bound on the number of $\leq k$-*sets* [89].

Lemma 3.1 counts the number of vertices close to p. Next we present a generalization of Lemma 3.1 that counts the number of k-flats that are close to p. We define the distance from p to an k-flat f as

$$\delta_H^k(p, f) = \min_{q \in f} \delta_H(p, q).$$

For a point p and an integer σ, we let $D_H^k(p, \sigma)$ denote the set of k-flats f in the arrangement of H with $\delta_H^k(p, f) \leq \sigma$. Notice that $D_H(p, \sigma) = D_H^0(p, \sigma)$.

Propositon 3.2. *For any point p disjoint from H in $\mathbb{R}^d$,*

$$|D_H^{d-1}(p, \sigma)| \geq 2(\sigma + 1) \quad \forall \sigma \in \left\{0, 1, \ldots, \left\lfloor \frac{\ell}{2} \right\rfloor - 1\right\}.$$

Proof. Welzl's proof [97] for $\mathbb{R}^2$ is also valid for $\mathbb{R}^d$. We can always find a line through p that intersects $\lfloor \frac{\ell}{2} \rfloor$ hyperplanes of H on each side of p, and the first $\sigma+1$ hyperplanes intersected by this line in each direction have distance at most σ to p. □

Lemma 3.3. *If H is a set of ℓ hyperplanes in general position in $\mathbb{R}^d$, and σ is an integer, $0 \leq \sigma \leq \lfloor \frac{\ell}{2} \rfloor - 1$, then $|D_H^k(p, \sigma)| \geq \frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}$ for all vertices p disjoint from H and $0 \leq k \leq d-1$.*

Proof. We are going to prove this theorem by induction on d. The proof is inspired by the proof by Welzl in [97]. In $\mathbb{R}^{k+1}$, we have, by Proposition 3.2,

$$|D_H^k(p, \sigma)| \geq 2(\sigma + 1) = \frac{2^{k+1-k}}{(k+1-k)!}\binom{\sigma+k+1-k}{k+1-k}.$$

Assume that $|D_H^k(p,\sigma)| \geq \frac{2^{t-k}}{(t-k)!}\binom{\sigma+t-k}{t-k}$ in $\mathbb{R}^t$, where $t \geq k+1$. In $\mathbb{R}^{t+1}$, we have at least $2(\sigma+1)$ t-flats with distance to p no more than σ according to Proposition 3.2. Let h_j be a t-flat with distance of j to p. We know that there are at least two such t-flats according to Proposition 3.2. We also know that there is a point q_j in h_j with $\delta_H(p,q_j) \leq j$. Then any vertices in h_j with distance to q_j no more than $\sigma - j$ have distance to p no more than σ. Since h_j is a space of dimension t, there are at least $\frac{2^{t-k}}{(t-k)!}\binom{\sigma-j+t-k}{t-k}$ such vertices. Since a k-flat is the intersection of $t+1-k$ hyperplanes, a vertex can be counted at most $t+1-k$ times. Therefore, in $\mathbb{R}^{t+1}$, we have

$$\begin{aligned} |D_H^k(p,\sigma)| &\geq \frac{2}{t+1-k}\sum_{j=0}^{\sigma}\frac{2^{t-k}}{(t-k)!}\binom{\sigma-j+t-k}{t-k} \\ &= \frac{2^{t+1-k}}{(t+1-k)!}\binom{\sigma+t+1-k}{t+1-k} \end{aligned}$$

Hence, in $\mathbb{R}^d$,

$$|D_H^k(p,\sigma)| \geq \frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}.$$

□

Lemma 3.1 improves previous results of Chazelle and Welzl, and Lemma 3.3 generalizes these results. These results have been published in Ref. [27]. If the hyperplanes in H are not in general position, we can assume that all parallel hyperplanes intersect at the infinity, and every d define a unique point. Then our lemmas of the size of the pseudo-balls still hold.

3.3 Approximate Range Counting

In this section, we consider the problem of approximate range counting in $\mathbb{R}^2$. That is, we study algorithms to preprocess a set of points S so that, given a closed halfplane h and an integer $i \geq 0$, we can quickly return an integer $r_i(h,S)$ such that

$$||h \cap S| - r_i(h,S)| \leq i.$$

This data structure is such that queries are faster when the allowable error, i, is larger.

There are no new results in this section. Rather it is a review of two relevant results on range searching and ϵ-approximations that are closely related, but separated for nearly 20 years. The reason we do this is that, without a guide, it can take some effort to gather and assemble the pieces; some of the proofs are existential. some are stated in terms of discrepancy theory, and some are stated in terms of VC-dimension.

The first tools we need come from a recent result of Chan on optimal partition trees and their application to exact halfspace range counting [19, Theorems 3.2 and 5.3, with help from Theorem 5.2]:

Theorem 3.4. *Let S be a set of n points in $\mathbb{R}^2$ and let $N \geq n$ be an integer. There exists a data structure that can preprocess S in $O(n \log N)$ expected time so that, with probability at least $1 - 1/N$, for any query halfplane, h, the data structure can return $|h \cap S|$ in $O(n^{1/2})$ time.*

We say that a halfplane, h, crosses a set, X, of points if h neither contains X nor is disjoint from X. The partition tree associated with the data structure of Theorem 3.4 is actually a *binary space partition tree.* Each internal node, u, is a subset of $\mathbb{R}^2$ and the two children of a node are the subsets of u obtained by cutting u with a line. Each leaf, w, in this tree has $|w \cap S| \leq 1$. The $O(n^{1/2})$ query time is obtained by designing this tree so that, with probability at least $1 - 1/N$, there are only $O(n^{1/2})$ nodes crossed by any halfplane.

For a geometric graph $G = (S, E)$, the *crossing number* of G is the maximum, over all halfplanes, h, of the number of edges $uw \in E$ such that h crosses $\{u, w\}$. From Theorem 3.4 it is easy to derive a spanning tree of S with crossing number $O(n^{1/2})$ using a bottom-up algorithm: Perform a post-order traversal of the partition tree. When processing a node u with children v and w, add an edge to the tree that joins an arbitrary point in $v \cap S$ to an arbitrary point in $w \cap S$. Since a halfplane cannot cross any edge unless it also crosses the node at which the edge was created, this yields the following result [19, Corollary 7.1]:

Theorem 3.5. *For any n point set, S, and any $N \geq n$, it is possible to compute, in $O(n \log N)$ expected time, a spanning tree, T, of S that, with probability at least $1 - 1/N$, has crossing number $O(n^{1/2})$.*

A spanning tree is not quite what is needed for what follows. Rather, we require a matching of size $\lfloor n/2 \rfloor$. To obtain this, we first convert the tree, T, from Theorem 3.5 into a path by creating a path, P, that contains the vertices of T in the order they are discovered by a depth-first traversal. It is easy to verify that the crossing number of P is at most twice the crossing number of T. Next, we take every second edge of P to obtain a matching:

Corollary 3.6. *For any n point set, S, and any $N \geq n$, it is possible to compute, in $O(n \log N)$ expected time, a matching, M, of size $\lfloor n/2 \rfloor$ that, with probability at least $1 - 1/N$, has crossing number $O(n^{1/2})$.*

The following argument is due to Matoušek, Welzl and Wernsich [67, Lemma 2.5]. Assume, for simplicity, that n is even and let $S' \subset S$ be obtained by taking exactly one endpoint from each edge in the matching M obtained by Corollary 3.6. Consider some halfplane h, and let M_h^I be the subset of the edges of M contained in h and let M_h^C be the subset of edges crossed by h. Then

$$|h \cap S| = 2|M_h^I| + |M_h^C|.$$

In particular,

$$|h \cap S| - |M_h^C| \leq 2|h \cap S'| \leq |h \cap S| + |M_h^C|.$$

Since $|M_h^C| \in O(n^{1/2})$, this is good news in terms of approximate range counting; the set S' is half the size of S, but $2|h \cap S'|$ gives an estimate of $|h \cap S|$ that is off by only $O(n^{1/2})$. Next we show that this can be improved considerably with almost no effort.

Rather than choose an arbitrary endpoint of each edge in M to take part in S', we choose each one of the two endpoints at random (and independently of the other $n/2 - 1$ choices). Then, each edge in M_h^C has probability $1/2$ of contributing a point

to $h \cap S'$ and each edge in M_h^I contributes exactly one point to $h \cap S'$. Therefore,

$$\mathrm{E}[2|h \cap S'|] = 2\left(1|M_h^I| + \frac{1}{2}|M_h^C|\right) = |h \cap S|.$$

That is, $2|h \cap S'|$ is an unbiased estimator of $|h \cap S|$. Even better: the error of this estimator is (2 times) a binomial $(|M_h^C|, 1/2)$ random variable, with $|M_h^C| \in O(n^{1/2})$. Using standard results on the concentration of binomial random variables (i.e., Chernoff Bounds [28]), we immediately obtain:

$$\Pr\{|2|h \cap S'| - |h \cap S|| \geq cn^{1/4}(\log N)^{1/2}\} \leq 1/N,$$

for some constant $c > 0$. That is, with probability $1 - 1/N$, $2|h \cap S'|$ estimates $|h \cap S|$ to within an error of $O(n^{1/4}(\log N)^{1/2})$. Putting everything together, we obtain:

Lemma 3.7. *For any n point set, S, and any $N \geq n$, it is possible to compute, in $O(n \log N)$ expected time, a subset S' of S of size $\lfloor n/2 \rfloor$ that, with probability at least $1 - 1/N$, for every halfplane h,*

$$|2|h \cap S'| - |h \cap S|| \in O(n^{1/4}(\log N)^{1/2}).$$

What follows is another argument by Matoušek, Welzl and Wernsich [67, Lemma 2.2]. By repeatedly applying Lemma 3.7, we obtain a sequence of $O(\log n)$ sets $S_0 \supset S_1 \cdots \supset S_r$, $S_0 = S$ and $|S_j| = \lceil n/2^j \rceil$. For $j \geq 1$, the set S_j can be computed from S_{j-1} in $O(2^{-j} n \log N)$ time and has the property that, with probability at least $1 - 1/N$, for every halfplane h,

$$|2^j|h \cap S_j| - |h \cap S|| \in O(2^{3j/4} n^{1/4}(\log N)^{1/2}). \tag{3.4}$$

At this point, we have come full circle. We store each of the sets $S_0, \ldots, S_r$ in an optimal partition tree (Theorem 3.4) so that we can do range counting queries on each set S_i in $O(|S_i|^{1/2})$ time. This (finally) gives the result we need on approximate range counting:

Lemma 3.8. *Given any set S of n points in in $\mathbb{R}^2$ and any $N \geq n$, there exists a data structure that can be constructed in $O(n \log N)$ expected time and, with probability at least $1 - 1/N$, can, for any query halfplane h and any integer $i \in \{0, \ldots, n\}$, return a number $r_i(h, S)$ such that*

$$||h \cap S| - r_i(h, S)| \leq i.$$

Such a query takes $O(\min\{n^{1/2}, (n/i)^{2/3}(\log N)^{1/3}\})$ expected time.

Proof. The data structure is a sequence of optimal partition trees on the sets $S_0, \ldots, S_r$. All of these structures can be computed in $O(n \log N)$ time, since $|S_0| = n$ and the size of each subsequent set decreases by a factor of 2.

To answer a query, (h, i), we proceed as follows: If $i \leq n^{1/4}$, then we perform exact range counting on the set $S_0 = S$ in $O(n^{1/2})$ time to return the value $|h \cap S|$. Otherwise, we perform range counting on the set S_j where j is the largest value that satisfies

$$C 2^{3j/4} n^{1/4} (\log N)^{1/2} \leq i,$$

where the constant C depends on the constant in the big-Oh notation in (3.4). This means $|S_j| = O((n/i)^{4/3}(\log N)^{2/3})$ and the query takes expected time

$$O(|S_j|^{1/2}) = O((n/i)^{2/3}(\log N)^{1/3}),$$

as required. □

Our main application of Lemma 3.8 is a test that checks whether a halfplane, h, contains $n/2$ or more points of S.

Lemma 3.9. *Given any set S of n points in in $\mathbb{R}^2$ and any $N \geq n$, there exists a data structure that can be constructed in $O(n \log N)$ expected time and, with probability at least $1 - 1/N$, can, for any query halfplane h determine if $|h \cap S| \leq n/2$. Such a query takes expected time*

$$Q(i) = \begin{cases} O(n^{1/2}) & \text{for } 0 \leq i \leq n^{1/4} \\ O((n/i)^{2/3}(\log N)^{1/3}) & \text{otherwise,} \end{cases}$$

where $i = ||h \cap S| - n/2|$.

Proof. As preprocessing, we construct the data structure of Lemma 3.8. To perform a query, we perform a sequence of queries (h, i_j), for $j = 0, 1, 2, \ldots$, where $i_j = n/2^j$. The jth such query takes $O(2^{2j/3}(\log N)^{1/3})$ time and the question, "is $|h \cap S| \geq n/2$?" is resolved once $n/2^j < i/2$. Since the cost of successive queries is exponentially increasing, this final query takes time $O(\min\{n^{1/2}, (n/i)^{2/3}(\log N)^{1/3}\})$ and dominates the total query time. □

Chapter 4

Algorithms for Tukey Depth

Many different algorithms have been developed to compute the Tukey depth of a point, or to find a point that maximizes the Tukey depth [14, 21, 59, 65, 83]. There are also many algorithms for approximating Tukey depth in low dimensions [1, 36, 83, 98]. The Tukey depth problem is equivalent to the MAXIMUMFEASIBLESUBSYSTEM problem which is a long-standing problem and has been extensively studied [30, Chapter 7].

In this chapter we give a fixed parameter tractable algorithm and a Monte Carlo approximation for Tukey depth in $\mathbb{R}^d$. The first algorithm is useful for applications like outlier removal where we want to identify low depth points. The second algorithm is better-suited to high depth points where an additive error is small compared to the true depth. The first part of this chapter appears in Ref. [15], and the second part appears in Ref. [27].

4.1 A Fixed Parameter Tractable Algorithm

In this section we give a fixed parameter tractable algorithm [43]. First we show that a Tukey depth problem can be reduced to a MAXIMUMFEASIBLESUBSYSTEM (MAX FS) problem. If a linear system has no solution, we say this system is *infeasible*. Given an infeasible linear system, the MAX FS problem is to find a maximum cardinality feasible subsystem. This problem is NP-hard [17, 84], and it is also hard

to approximate within factor of 2 [5] for the systems that consist of only $\leq$ type of inequalities.

4.1.1 Tukey Depth and MAX FS

Let $\langle a, b\rangle$ denote the dot product of vector a and b, Then the halfspace depth of point p can also be described as:

$$\min_{x \in \mathbb{R}^d \setminus 0} |\{q \in S | \langle x, q\rangle \leq \langle x, p\rangle\}| \tag{4.1}$$

where x is the outward normal vector of the closed halfspace having p on its bound-

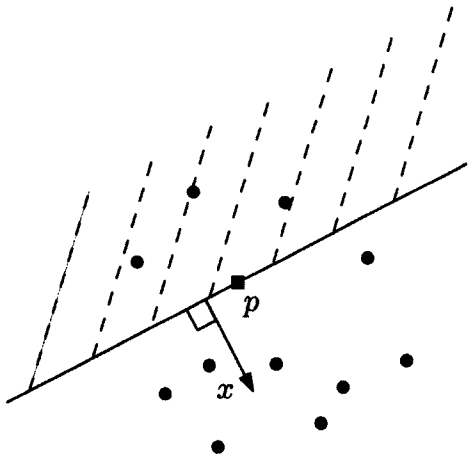

Figure 4.1: Tukey depth in $\mathbb{R}^2$ with vector x

ary. As shown in Figure 4.1, if a point q is contained in the closed halfspace, the corresponding inequality $\langle x, q\rangle \leq \langle x, p\rangle$ will be satisfied. Minimizing the number of the points contained in the halfspace is equivalent to maximizing the number of points excluded from the halfspace. Then, the definition of halfspace depth can also be described as:

$$|S| - \max_{x \in \mathbb{R}^d} |\{q \in S | \langle x, q\rangle > \langle x, p\rangle\}| \tag{4.2}$$

When p is contained in the convex hull of S, and p is on the boundary of a closed halfspace, as shown in Figure 4.1, there must be some data contained by the halfspace.

Then the set of inequalities

$$\langle x, q \rangle > \langle x, p \rangle \qquad \forall q \in S \tag{4.3}$$

or

$$\langle x, q - p \rangle > 0 \qquad \forall q \in S \tag{4.4}$$

in (4.2) can not be satisfied at the same time, in other words, (4.4) is an infeasible linear system. To compute the halfspace depth of point p is then to find the maximum number of inequalities in (4.4) that can be satisfied at the same time, or say to find the maximum feasible subsystem of (4.4). Of course, if p is outside of the convex hull of S, (4.4) will be feasible, and the depth for p will be 0. For convenience of the following discussion, we need to change the strict inequalities to non-strict ones. By the method in [24], we can change (4.4) to the following inequalities:

$$\langle x, q - p \rangle \geq 1 \qquad \forall q \in S \tag{4.5}$$

The two infeasible systems (4.4) and (4.5) have exactly the same MAX FS.

We can also transform a Tukey depth problem in $\mathbb{R}^d$ to two MAX FS in $\mathbb{R}^{d-1}$. First we transform the Tukey depth problem into a MAX FS as (4.4). Then we pick two parallel hyperplanes which are not parallel to the boundary of any halfspaces in (4.4), and the two hyperplanes are separated by p. In each of the two hyperplanes we then get a MAX FS in $\mathbb{R}^{d-1}$ with strict inequalities. The larger solution to the two MAX FS is the solution for the original MAX FS in $\mathbb{R}^d$. If S is in general position, we treat the inequality in (4.4) as non-strict ones. If S is not in general position we can add a constant to right side of the inequalities to transform them into non-strict ones similar to (4.5). Now we will have an arrangement around p. Then we need to pick two parallel hyperplanes that sandwich all the vertices in the arrangement.

Now we show that we can also transform a MAX FS T of size n in $\mathbb{R}^d$ to a Tukey depth problem of size $2n$ in $\mathbb{R}^{d+1}$. In the Euclidean space one dimension higher, the halfspaces in T are all contained in a hyperplane. We pick an arbitrary point p that is not on that hyperplane. For each halfspace in T, we expand it to a halfspace $\hbar$ one

dimension higher and make p on its boundary. We then find a vector $v_{\hbar}$ orthogonal to the boundary of $\hbar$ and contained in $\hbar$. Finally we get a point $p+v_{\hbar}$ for point set S. Repeat this procedure for all other hyperplanes, we can get a set S of n points. We now find a vector v_h that is orthogonal to the hyperplane containing T, and points from p to the hyperplane. If the boundaries of the halfspaces in T are in general position, we can add n copies of point $p + v_h$ to S, then we can get the solution to T by finding the Tukey depth of p with respect to S. Since the decision type of the MAX FS in $\mathbb{R}^2$ is 3 sum hard [52], a decision type Tukey depth problem in $\mathbb{R}^2$ is 3 sum hard too.

4.1.2 The algorithm

Now we give a fixed parameter tractable [43] algorithm for MAX FS problem that, for any fixed value of the output, k, is polynomial in the dimension d. The algorithm uses linear programming as a subroutine in the following way: Given a collection K of closed halfspaces in $\mathbb{R}^d$, an algorithm for linear programming can be used to either:

1. Determine a point $p \in \cap K$ if such a point exists or,

2. report a subset $B \subseteq K, |B| \leq d+1$, such that $\cap B = \emptyset$.

The set B reported in the later case is called a *basic infeasible subsystem.* Standard combinatorial algorithms for linear programming, including algorithms for linear programming in small dimensions [32, 45, 68, 69, 86, 88] as well as the simplex method (cf. [31]), can generate a basic infeasible subsystem given a infeasible linear program. The details of finding a basic infeasible subsystem using linear programming are discussed in Section 3.1.

Let BIS(K) denote a routine that outputs a basic infeasible subsystem of K if K is infeasible, and that outputs the empty set otherwise. Algorithm 1 solves the MAX FS decision problem. The correctness of Algorithm 1 is easily established by induction on the value of k. The running time of the algorithm is given by the recurrence

$$T(n,k) \leq \mathrm{LP}(n,d) + (d+1)T(n-1,k-1),$$

Algorithm 1 MAXIMUMFEASIBLESUBSYSTEM(K, k)

Input: A collection of halfspaces K, and an integer k.
Output: Determine if there exists $K' \subseteq K, |K'| \leq k$, such that $\cap(K \backslash K') \neq \emptyset$.

```
1: B ← BIS(K)
2: if B = ∅ then
3:    return true
4: end if
5: if k = 0 then
6:    return false
7: end if
8: for each h ∈ B do
9:    if MFS(K\{h}, k − 1) = true then
10:      return true
11:   end if
12: end for
13: return false
```

where $\mathrm{LP}(n, d)$ denotes the running time of an algorithm for finding a basic infeasible subsystem in linear program with n constraints and d variables. This recurrence readily resolves to $O((d+1)^k \mathrm{LP}(n, d))$. Using this as a subroutine for Tukey depth computation we obtain the following result:

Theorem 4.1. *The Tukey depth of a point p with respect to a set S of n points in $\mathbb{R}^d$ can be computed in $O((d+1)^k LP(n, d))$ time, where k is the value of the output and $LP(n, d)$ is the time to solve a linear program with n constraints and d variables.*

Remark. The algorithm described above is closely related to Matoušek's algorithm for MAX FS which, in our setting, has a running time of $O(k^{d+1}\mathrm{LP}(n, d))$ [66]. In the language of fixed-parameter tractability, the primary difference between the two algorithms is that Matoušek's algorithm explores the search tree in breadth-first order and uses a dictionary to ensure that identical subtrees are not explored, whereas the current algorithm explores the search tree in depth-first order. The two algorithms can, of course, be combined to obtain an algorithm with running time $O(\min\{k^{d+1}, (d+1)^k\} \times \mathrm{LP}(n, d))$, by interleaving the execution of the two algorithms and stopping when the first one terminates.

4.2 A Monte Carlo Approximation

Due to the hardness of the Tukey depth problem, algorithms for approximating Tukey depth in low dimensions are of interest. Rousseeuw and Struyf [83] proposed four approximation algorithms. The basic idea is to randomly choose m of four categories of lines: (1) all lines connecting p and a point in S, (2) all lines connecting two points in S, (3) all lines perpendicular to the hyperplanes determined by p and $d-1$ points in S, (4) all lines perpendicular to the hyperplanes determined by d points in S, then project all points onto the lines to solve one-dimensional Tukey depth problems, and take the best result as the approximation. Rousseeuw and Struyf claim that the fourth idea worked best.

Wilcox [98] proposed two approximation algorithms. The strategies are similar to those of Rousseeuw and Struyf. The difference is that, in the first approximation, the points are orthogonally projected onto the lines connecting an affine equivariant measure of location e and a point in S; in the second approximation the points are projected onto all the lines determined by two points in S.

Cuesta-Albertos and Nieto-Reyes [36] proposed a notion of *random Tukey depth* where they project all points onto m randomly chosen vectors, and take the best one-dimensional Tukey depth. They claim this depth is a reasonable approximation of Tukey depth. All the above approximations have no theoretical guarantee of the performance.

Afshani and Chan [1] gave a data structure for Tukey depth queries in 3D. For any constant $\epsilon > 0$, their data structure can preprocess a 3D point set in $O(n \log n)$ expected time into a data structure of size $O(n)$ such that the Tukey depth of any query point q can be approximated in $O(\log n \log\log n)$ time. Here, the approximation is relative; their data structure returns a value y such that

$$(1-\epsilon)\,\mathrm{dep_T}(q,S) \le y \le (1-\epsilon)^{-1}\,\mathrm{dep_T}(q,S)\ .$$

In this section we will give a Monte Carlo approximation to the Tukey depth.

4.2.1 Introduction

Suppose points in S are in general position, an upper bound on the Tukey depth of p can be obtained by selecting any non-trivial vector $v \in \mathbb{R}^d$ and computing the Tukey depth of $\langle p, v\rangle$ in the one-dimensional point set

$$\langle S, v\rangle = \{\langle x, v\rangle : x \in S\}. \qquad (*)$$

If v is the inner-normal of the boundary of the halfspace $\hbar$ that defines the depth value of p, then

$$\mathrm{dep_T}\,(p, S) = \mathrm{dep_T}\,(\langle p, v\rangle, \langle S, v\rangle). \qquad (4.6)$$

In $\mathbb{R}^1$, we rank the points $S \cup \{p\}$ starting with 0 from both ends to the median, then the depth of p is its rank. More generally, given any k-flat f orthogonal to the boundary of $\hbar$, we have

$$\mathrm{dep_T}\,(p, S) = \mathrm{dep_T}\,(\langle p, f\rangle, \langle S, f\rangle), \qquad (4.7)$$

where $\langle p, f\rangle$ is the orthogonal projection of p onto f, and $\langle S, f\rangle$ is the orthogonal projection of S onto f.

In this section, we analyze the following 2 heuristics for this problem:

1 Randomly select a set Q of $d-1$ points from S. Let π be the unique hyperplane containing $Q \cup \{p\}$, and let v be a vector orthogonal to π. Apply (4.6) to get an upper bound on $\mathrm{dep_T}\,(p, S)$.

2 Randomly select a set Q of $d-k$ points from S. Let π be the unique $(d-k)$-flat containing $Q \cup \{p\}$, and let f be a k-flat orthogonal to π. Apply (4.7) to get an upper bound on $\mathrm{dep_T}\,(p, S)$.

The first algorithm described above is the third method proposed by Rousseeuw and Struyf. The second algorithm is a generalization of this method. Notice that when we project the points in S to the vector or k-flat, those points in Q do not contribute to the depth of p.

The first algorithm reduces the original problem to a one-dimensional Tukey depth problem, but the second reduces to a k-dimensional Tukey depth problem. The projection of S to a vector takes $O(dn)$ time. Then the first heuristic requires $O(dn)$ time. In the second heuristic, the projection of S to a k-flat takes $O(kdn)$ time, and the k-dimensional Tukey depth problem has the following time complexity:

For $k = 1$, the Tukey depth is easily computed in $O(n)$ time by counting the number of points less than p and the number of points greater than p, and taking the minimum of those 2 quantities.

For $k = 2$, the Tukey depth of p can be computed in $O(n \log n)$ time by sorting the points of S radially about p and scanning this sorted list using two pointers [83].

For $k \geq 3$, the algorithms already become significantly more complicated. When $k = 3$, a brute-force algorithm runs in $O(n^3)$ time, and an algorithm of Chan [22] runs in $O((n + t^2) \log n)$ time, where t is the depth of p.

The output of each of these heuristics is an upper bound on the depth of p. In the remainder of this section we analyze how good these upper bounds can be with the following two theorems, which bound the probability that the approximated depth exceeds the true depth by more than σ.

Theorem 4.2. *Let S be a set of n points in general position in $\mathbb{R}^d$, S' be a subset of $d-1$ elements chosen at random and without replacement from S, v be the vector perpendicular to the plane containing S' and another point p, σ be an integer such that $0 \leq \sigma \leq \lfloor \frac{n}{d} \rfloor - 1$. Then*

$$\Pr\{\mathrm{dep_T}(\langle p, v\rangle, \langle S, v\rangle) \leq \mathrm{dep_T}(p, S) + \sigma\} \geq \frac{\binom{\sigma+d-1}{d-1}}{\binom{n}{d-1}}.$$

Theorem 4.3. *Let S be a set of n points in general position in $\mathbb{R}^d$, S' be a subset of $d-k$ elements chosen at random and without replacement from S, f be the k-flat orthogonal to the $(d-k)$-flat containing S' and another point p, σ be an integer such that $0 \leq \sigma \leq \lfloor \frac{n}{2} \rfloor - 1$. Then*

$$\Pr\{\mathrm{dep_T}(\langle p, f\rangle, \langle S, f\rangle) \leq \mathrm{dep_T}(p, S) + \sigma\} \geq \frac{2^{d-k}\binom{\sigma+d-k}{d-k}}{(d-k)!\binom{n}{d-k}}.$$

Here is a sketch of the proof of Theorem 4.2: Under point/hyperplane duality, the selection of v is equivalent to selecting a random vertex in an arrangement of hyperplanes in $d-1$ dimensions. This selection of v approximates $\mathrm{dep_T}(p, S)$ to within σ provided that the vertex is contained in a particular pseudo-ball of radius σ. Therefore the proof boils down to showing that the number of vertices of an arrangement in a pseudo-ball of radius σ is sufficiently large. In particular, Lemma 3.1 shows that the number of vertices in such a pseudo-ball is at least $\binom{\sigma+d-1}{d-1}$.

The proof of Theorem 4.3 is similar, except that we lower-bound the number of k flats that intersect a pseudo-ball of radius σ.

4.2.2 The Algorithm

In order to relate the hyperplane arrangements studied in Section 3.2 to the approximation algorithms for Tukey depth, we need to introduce a duality [46].

Point/hyperplane duality. For a point $a = (a_1, a_2, \ldots, a_d)$ in S, its dual image, denoted by a^*, is a hyperplane in T with equation $x_d = a_1x_1 + a_2x_2 + \ldots + a_{d-1}x_{d-1} - a_d$, and for a hyperplane b with equation $x_d = b_1x_1 + b_2x_2 + \ldots + b_{d-1}x_{d-1} - b_d$, its dual image, denoted by b^*, is the point $(b_1, b_2, \ldots, b_d)$. Duality preserves incidences between points and hyperplanes and reverses the above/below relationship. The point a lies on the hyperplane b if and only if b^* lies on a^*; a lies above b if and only if a^* is below b^*. All the hyperplanes through point p in the primal dualize to all the points on the hyperplane p^* in the dual.

The dual arrangement. Given a set S of n points in $\mathbb{R}^d$, we define the *dual arrangement* $\mathcal{A}(T)$ of S as a set of n hyperplanes, T, that are the duals of the points in S. In the dual arrangement, we say a hyperplane is *vertical* if it contains a line parallel to the x_d-axis.

Duality and Tukey depth. Finding the Tukey depth of p is equivalent to finding a hyperplane h (with inner-normal v) through p with the fewest points either above

or below. In the dual, this is the same as finding a point h_v^* on they hyperplane p^* with the fewest hyperplanes of T either below or above.

The hyperplanes in T divide p^* into cells. Within a cell, the number of hyperplanes above or below any two points is the same. Suppose cell c in T contains the optimal points (h_v^* is a point inside c). For any vertex b^* in $\mathcal{A}(T)$ with $\delta_T(h_v^*, b^*) = \sigma$, the normal vector v_b of its primal image b gives a depth value at most σ more than the optimal depth value (Heuristic 1 in page 30). Similarly, for any k-flat y^* in $\mathcal{A}(T)$ with $\delta_T^k(h_v^*, y^*) = \sigma$, the $(k+1)$-flat f_y orthogonal to its primal image y gives a depth value at most σ more than optimal depth value (Heuristic 2 in page 30).

Analysis of First Heuristic

Now let us analyze how well the first heuristic works. Sampling $d-1$ points from S is the same as sampling $d-1$ hyperplanes in T which will define a vertex on p^*. Then we only need to consider the $d-1$ dimensional arrangement $\mathcal{A}(T_{p^*})$ restricted to p^*. According to Lemma 3.1, $|D_{T_{p^*}}(h_v^*, \sigma)| \geq \binom{\sigma+d-1}{d-1}$. Since there are $\binom{n}{d-1}$ vertices on p^*, by one sampling, the probability that we get a depth value with an error no more than σ is at least

$$\frac{\binom{\sigma+d-1}{d-1}}{\binom{n}{d-1}} = \frac{(\sigma+d-1)!(n-d+1)!}{\sigma! n!}. \tag{4.8}$$

Let $P_\sigma = \frac{(\sigma+d-1)!(n-d+1)!}{\sigma! n!}$. We can repeat this heuristic s times and use the smallest result as an approximation. The probability that the smallest depth value that has an error more than σ is at most $(1-P_\sigma)^s$. Hence, the probability that we get a depth value with an error no more than σ is at least

$$1-(1-P_\sigma)^s \geq 1-\frac{1}{e} \text{ for } s = \frac{\sigma! n!}{(\sigma+d-1)!(n-d+1)!} \leq \left(\frac{n}{\sigma}\right)^{d-1}. \tag{4.9}$$

If we set σ to ϵn, where ϵ is a fixed constant, this approximation runs in $O(\epsilon^{1-d} dn)$ time.

Analysis of Second Heuristic

In the second heuristic, sampling $d-k$ points from S is the same as sampling $d-k$ hyperplanes in T which will define a $(k-1)$-flat on p^*. According to Lemma 3.3, we have $|D^{k-1}_{T_{p^*}}(h^*_v, \sigma)| \geq \frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}$. Since there are $\binom{n}{d-k}$ $(k-1)$-flats on p^*, by one sampling, the probability that we get a depth value with an error no more than σ is at least

$$\frac{\frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}}{\binom{n}{d-k}} = \frac{2^{d-k}(\sigma+d-k)!(n-d+k)!}{(d-k)!\sigma!n!}. \tag{4.10}$$

Similar to the above analysis, we let $P'_\sigma = \frac{2^{d-k}(\sigma+d-k)!(n-d+k)!}{(d-k)!\sigma!n!}$. Running this heuristic s times, the probability that we get a depth value with an error no more than σ is at least

$$\begin{aligned} &1-(1-P'_\sigma)^s \geq 1-\frac{1}{e} \\ &\text{for } s = \frac{(d-k)!\sigma!n!}{2^{d-k}(\sigma+d-k)!(n-d+k)!} \leq \left(\frac{d-k}{2}\right)^{d-k} \cdot \left(\frac{n}{\sigma}\right)^{d-k}. \end{aligned} \tag{4.11}$$

This approximation needs less samples when d is small, but we need to solve s Tukey depth problems in $\mathbb{R}^k$. For $k=2$, if we set σ to ϵn, this approximation runs in $O\left(\left(\frac{d-2}{2}\right)^{d-2}\epsilon^{2-d}n\log n\right)$ time.

Our approximation algorithms are comparable to the following simple approximation. For a fixed constant ϵ and a large enough constant c, we make a random sample R of S, where each element of S is selected with probability $\frac{c\log n}{\epsilon n} < 1$. We then compute $\mathrm{dep_T}(p, R)$ with brute-force. With high probability, $\mathrm{dep_T}(p,R)\cdot\frac{\epsilon n}{c\log n}$ is an approximation of $\mathrm{dep_T}(p,S)$ with error no more than ϵn. This approximation runs in $O\left((\epsilon^{-1}c\log n)^d\right)$ time. While this is asymptotically faster than our algorithms, $\log^d n$ can be significantly larger than n in many cases. This is the case, for example, with all the data sets considered in the next section.

4.2.3 Experimental Results

We tested the two approximation algorithms on a Dell Precision 490 workstation with a 2.80 GHz Intel Xeon CPU. For the second approximation, we tested the case of

$k = 2$, and the 2-dimensional problems are solved with a scan and sort algorithm [83]. The two algorithms are run s times (as indicated in (4.9) and (4.11)) and tested with the 9 data sets listed in Table 4.1:

Name	Item #	Attrib #	Source
Iris	150	4	UCI MLR.
Wine4d	178	4	UCI MLR. 4 attributes of the Wine data set
Wine5d	178	5	UCI MLR. 5 attributes of the Wine data set
Auto4d	392	4	UCI MLR. 4 attributes of the Auto MPG data set
Auto5d	392	5	UCI MLR. 5 attributes of the Auto MPG data set
Rand4d	500	4	Randomly generated
Forest4d	517	4	UCI MLR. 4 attributes of the Forest Fires data set [35]
Forest5d	517	5	UCI MLR. 5 attributes of the Forest Fires data set [35]
Pima4d	768	4	UCI MLR. 4 attributes of the Pima Indians Diabetes data set
Pima5d	768	5	UCI MLR. 5 attributes of the Pima Indians Diabetes data set
Yeast4d	1484	4	UCI MLR. 4 attributes of the Yeast data set

Table 4.1: The data sets

The Rand4d data set is randomly generated, and the data items are uniformly distributed in a unit ball. All other data sets are extracted from some data sets in the University of California, Irvine (UCI) Machine Learning Repository (MLR) [7]. The data points in the data sets extracted from UCI MLR are not in general position. Even worse, there are duplicate data points in some data sets. There are no duplicates in Wine4d, Wine5d, Pima4d, Pima5d, and Rand4d. There are a few duplicates in Iris, Auto4d, and Auto5d. There are many duplicates in Yeast, Forest4d, and Forest5d.

The running time of the algorithms on different data sets is given in Table 4.2. The second approximation runs faster, but it is more sensitive to rounding error. In order to generate a 2d problem in the second approximation, we first find 2 perpendicular vectors in the 2-flat orthogonal to the $(d-2)$-flat containing the $d-2$ sampling points and p, then project all points in the data set onto the 2 vectors. The values are used

as the coordinates of points in the 2-dimensional space. This projection and the sorting of the 2-dimensional points bring rounding errors. To overcome this problem, exact arithmetic is applied on the Iris data set with GMP (GNU Multiple Precision Library). GMP slows down the algorithm dramatically, hence it is not practical to use it on larger data sets.

Data Set	σ value	Algorithm	Running time	Max error[1]	Average error[1]
Iris	2	approx 1	50s(GMP)	2	0.309
		approx 2	7s(GMP)	2	0.258
Wine4d	2	approx 1	2s	2	0.372
		approx 2	1s	2	0.326
Wine5d	2	approx 1	70s	3	0.223
		approx 2	8s	3	0.086
Auto4d	2	approx 1	31s	1	0.213
		approx 2	2s	2	0.213
Auto5d	2	approx 1	2400s	1	0.164
		approx 2	187s	1	0.071
Rand4d	2	approx 1	77s	1	0.186
		approx 2	3s	2	0.169
Forest4d	2	approx 1	87s	2	0.309
		approx 2	4s	2	0.136
Forest5d	3	approx 1	3880s	2	0.319[2]
		approx 2	287s	1	0.088[3]
Pima4d	2	approx 1	387s	2	0.299
		approx 2	12s	1	-1.031[4]
Pima5d	4	approx 1	12350s	2	0.708
		approx 2	815s	2	-0.333[5]
Yeast4d	3	approx 1	2400s	1	0.571
		approx 2	56s	1	0.429

[1] Some points are not tested because we do not know the real depth of them.
[2] 2 depth values are smaller than the real ones (due to rounding errors).
[3] 7 depth values are smaller than the real ones (due to rounding errors).
[4] 23 depth values are smaller than the real ones (due to rounding errors).
[5] 14 depth values are smaller than the real ones (due to rounding errors).

Table 4.2: The performance of the algorithms

The true depth values are computed with the binary search idea in [24] which requires solving a series of mixed integer program. It takes a long time and a large

amount of memory to solve the integer programs. Many instances can not be solved due to time and memory limitations. The time required to solve integer programs is output-sensitive, so that problems with larger depth values take longer. For example, we need a few hours to solve a problem with depth 10 in Pima5d. On the other hand, the approximation algorithms do not have this sensitivity. They take roughly the same time to solve all the problems in the same data set.

For smaller data sets, the tests were run with the absolute error σ set to 2. However, in the vast majority of cases (at least those in which the true depth can be computed exactly), both approximation algorithms computed the depth correctly with no error. In a small number of cases the error is 1 or 2. The second approximation gave less average error.

4.3 Conclusion

In this chapter we proposed several algorithms [15, 27] for the Tukey depth problem. The fixed parameter tractable algorithm runs quickly when depth of p is small, and is therefore well-suited to outlier removal, where the goal is to identify low-depth points. The Monte-Carlo approximation algorithms are simple, easy to implement, and our results show that, as well as having theoretical guarantees, they work well in practice. In addition to the algorithms themselves, the analysis of the the Monte-Carlo algorithm requires sharper and generalized bounds on the size of pseudoballs in $\mathbb{R}^d$. These results presented as Lemma 3.1 and Lemma 3.3 in Chapter 3 are of independent interest in the field of combinatorial geometry.

Chapter 5

Algorithms for Oja Depth

In this chapter we consider relationships between centers of mass of certain sets and Oja depth. The results in chapter appears in Ref. [26]. The *center of mass* of a finite point set $S \subset \mathbb{R}^d$ is the average of those points,

$$\mathrm{com}(S) = |S|^{-1} \sum_{x \in S} x \ .$$

If $P \subset \mathbb{R}^d$ is a bounded object of non-zero volume, the center of mass of P is

$$\mathrm{com}(P) = \frac{\int_{x \in P} x \, \mathrm{d}x}{\mathrm{vol}(P)} \ .$$

In this chapter, we prove the following results about the Oja depth of an n point set S, whose convex hull A has unit volume and that has an Oja center x:

$$\mathrm{dep_O}(\mathrm{com}(A), S) \leq \binom{n}{d}/(d+1) \ , \tag{5.1}$$

$$\mathrm{dep_O}(\mathrm{com}(S), S) \leq (d+1)\,\mathrm{dep_O}(x, S) \ . \tag{5.2}$$

The bound in (5.1) is not known to be tight. The bound in (5.2) is tight, up to a lower-order term, for some point sets S.

Our first result, (5.1), is a form of *Centerpoint Theorem* that upper-bounds the Oja depth of $\mathrm{com}(A)$, and hence also the Oja depth of x, in terms of the volume of

the convex hull of S. Previously, centerpoint theorems were known for other depth functions such as Tukey depth [64, 76, 92] and simplicial depth [8, 13, 60]. To the best of our knowledge, this is the first such result for Oja depth.

Our next result, (5.2), can be viewed in two ways:

1. The first is a linear-time algorithm to find a point whose depth is a constant factor approximation of the depth of the Oja center. In 1-d, Oja depth is minimized by the median, which can be found in $O(n)$ time. However, in 2-d, the best known algorithm for minimizing Oja depth exactly takes $O(n \log^3 n)$ time [4], and is relatively complicated. Approximation algorithms for minimizing Oja depth, based on uniform grids and sampling from $\binom{S}{d}$, are given by Ronkainen, Oja, and Orponen [79]; this algorithm and several others are implemented in the R-package, OjaNP [50]. However, in pathological cases, their approximation algorithm is not guaranteed (or even likely) to find a point that closely approximates the Oja center, either in terms of distance or in terms of its Oja depth.[1]

2. Another view of (5.2) is that it gives insight into the Oja depth function and the Oja center. In some sense, it tells us that the Oja center is not terribly different from the center of mass of S, since the center of mass of S minimizes, to within a constant factor, the Oja depth function.

5.1 Oja Center and Mass Center of A

In this section, we relate the Oja depth of the center of mass of the convex hull of S to the volume of the convex hull of S. Throughout this section, A denotes the convex hull of S and we assume, without loss of generality, that $\text{vol}(A) = 1$.

Our upper-bound is based on the following *central identity*: For any disjoint sets

[1]This follows from the fact that the value of the Oja depth function and the location of the Oja center can be arbitrarily different for two sets S_1 and S_2 that differ in only d points [74].

$X, Y \subseteq \mathbb{R}^d$ with $\text{vol}(X \cup Y) > 0$,

$$\text{com}(X \cup Y) = \frac{\text{vol}(X)\,\text{com}(X) + \text{vol}(Y)\,\text{com}(Y)}{\text{vol}(X \cup Y)}.$$

We first give an inductive proof of our result for point sets in $\mathbb{R}^2$, and then give a proof for point sets in $\mathbb{R}^d$ that uses tools from convex geometry.

5.1.1 An Upper Bound in $\mathbb{R}^2$

The following result shows that, for a convex polygon E (e.g., $E = A$), it is not possible to form an overly-large triangle that has $\text{com}(E)$ as one of its vertices:

Lemma 5.1. *Let A be a convex polygon and let p_1 and p_2 be any two points in A. Then* $\text{vol}(p_1, p_2, \text{com}(A)) \leq \text{vol}(A)/3$.

Proof. Assume, without loss of generality, that p_1p_2 is horizontal and that $\text{com}(A)$ is above p_1p_2. We may assume that p_1p_2 is edge of A since, otherwise, we can remove the part of A that is below p_1p_2. This decreases $\text{vol}(A)$ and moves $\text{com}(A)$ further away from the segment p_1p_2, which increases $\text{vol}(\text{com}(A), p_1, p_2)$.

The proof is by induction on the number of vertices of A. If A is a triangle then one can easily verify the result. Therefore, assume A has $n \geq 4$ vertices. Consider an edge ab of A where $a \neq p_2$ is adjacent to p_1 and let $c \neq a$ be adjacent to b (see Figure 5.1). Since A has 4 or more vertices, we may assume that the y-coordinate of b is not smaller than the y-coordinate of a. Otherwise we can reverse the roles of p_1 and p_2 and redefine a and b with respect to the new p_1.

Draw a ray r whose origin is at p_1 and such that the triangle t_1 supported by p_1a, ab and r and the triangle t_2 supported by r, ab, and the line through bc have the same area. Such a ray is guaranteed to exist by a standard continuity argument that starts with r containing a and rotates about p_1 until r contains b.

Now, convert A into a polygon A' by removing t_1 and adding t_2. This does not change the area of A. Furthermore, since t_1 and t_2 are separated by a horizontal line, with t_2 above t_1, this implies that $\text{com}(A')$ has a larger y-coordinate than $\text{com}(A)$, so

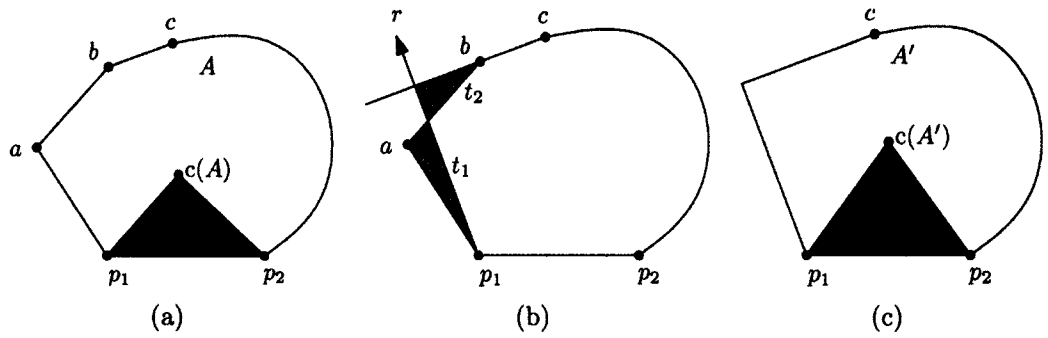

Figure 5.1: The proof of Lemma 5.1.

$\text{vol}(\text{com}(A), p_1, p_2)) \leq \text{vol}(\text{com}(A'), p_1, p_2)$. Note, also, that A' has $n-1$ vertices so, by induction,

$$\text{vol}(\text{com}(A), p_1, p_2) \leq \text{vol}(\text{com}(A'), p_1, p_2) \leq \text{vol}(A')/3 = \text{vol}(A)/3 \ ,$$

completing the proof. □

This bound is tight when A is a triangle, and p_1 and p_2 are two of the vertices of the triangle.

Theorem 5.2. *Let S be a set of points in $\mathbb{R}^2$ whose convex hull, A, has unit area. Then* $\text{dep}_{\text{O}}(\text{com}(A), S) \leq n^2/6$.

Proof. Applying Lemma 5.1, we immediately get

$$\text{dep}_{\text{O}}(\text{com}(A), S) = \sum_{p_1, p_2 \in \binom{S}{2}} \text{vol}(p_1, p_2, \text{com}(A)) \leq \binom{n}{2}/3 < \frac{n^2}{6} \ ,$$

as required. □

5.1.2 An Upper Bound in $\mathbb{R}^d$

In this section, we extend our bound to $\mathbb{R}^d$. However, it seems difficult to apply induction to polytopes in $\mathbb{R}^d$, so we resort to some tools from convex geometry.

Let A be a convex body in $\mathbb{R}^d$, where $d \geq 2$. Suppose A lies between parallel hyperplanes $x_1 = a$ and $x_1 = b$, where $a < b$. For each x with $a \leq x \leq b$, let A_x be

the intersection of A with the hyperplane $x_1 = x$, and define r_x by the equation

$$\omega_{d-1} r_x^{d-1} = \mathrm{vol}_{d-1}(A_x) \ ,$$

where $\mathrm{vol}_{d-1}(X)$ denotes the $(d-1)$-dimensional volume of X and ω_{d-1} is the $(d-1)$-dimensional volume of the unit $(d-1)$-ball. In this way, r_x is the radius of a $(d-1)$-ball whose $(d-1)$-dimensional volume is the same as that of A_x. For each $a \leq x \leq b$, let C_x be defined by the equation

$$C_x = \{(x, x_2, \ldots, x_d) : x_2^2 + \cdots + x_d^2 \leq r_x^2\}.$$

Then the set

$$C = \cup\{C_x : a \leq x \leq b\}$$

is called the *Schwarz rotation-symmetral* of A in the x_1-axis. For example, in $\mathbb{R}^3$, C is a stack of disks perpendicular to, and centered on, the x-axis. Each disk has the same area as the corresponding slice of A.

Theorem 5.3 (Webster [96]). *Let A be a convex body in $\mathbb{R}^d$ $(d \geq 2)$ whose Schwarz rotation-symmetral in the x_1-axis is C. Then C is a convex body having the same volume as A.*

Lemma 5.4. *Let P be a convex polytope in $\mathbb{R}^d$ and let $p_1, \ldots, p_d$ be any d points in P. Then* $\mathrm{vol}(p_1, \ldots, p_d, \mathrm{com}(P)) \leq \mathrm{vol}(P)/(d+1)$.

Proof. Our proof this lemma is based on the fact that the center of mass of a cone or pyramid with height ℓ in $\mathbb{R}^d$ is $\frac{\ell}{d+1}$. To see this, we assume that base of the cone of pyramid contains the origin and is perpendicular to the x_1-axis. According to central identity, the x_1-value of the center of mass is

$$\frac{\int_0^\ell x_1 a(\ell - x_1)^{d-1} \mathrm{d}x_1}{\frac{a\ell^{d-1}\ell}{d}},$$

where a is the constant involved in the computing the $(d-1)$-dimensional volume of the base. Solving this integration we get $\frac{\ell}{d+1}$ [49].

Let h be a hyperplane that contains $p_1, \ldots, p_d$. If $\text{com}(P)$ is in h, or $p_1, \ldots, p_d$ are not in general position, $\text{vol}(p_1, \ldots, p_d, \text{com}(P)) = 0$ and there is nothing to prove. Otherwise, rotate P to make h perpendicular to the x_1-axis with $\text{com}(P)$ above h. If there is any volume of P below h, we can cut that part off from P to obtain a new polytope P'. The volume of P' will be less than that of P, and its center of mass $\text{com}(P')$ will be above $\text{com}(P)$. In this way, if $(P, p_1, \ldots, p_d)$ is a counterexample to the lemma, then so is $(P', p_1, \ldots, p_d)$.

The face, B, of P' in h contains $p_1, \ldots, p_d$. If P' is a pyramid, we have

$$\text{vol}(p_1, \ldots, p_d, \text{com}(P')) \leq \text{vol}(B, \text{com}(P')) = \text{vol}(P')/(d+1) \leq \text{vol}(P)/(d+1) \ .$$

If P' is not a pyramid, let q be a point on the x_1 axis, above h, and such that the pyramid D with B as base and q as apex has the same volume as P'. Let C be the Schwarz rotation-symmetral of P' in the x_1-axis, and R be that of D (see Figure 5.2). Note that R is a conic pyramid. By Theorem 5.3, C is convex and $\text{vol}(C) = \text{vol}(P') = \text{vol}(R)$. Let c be the intersection of the surfaces of C and R above B. Note that the surface of R is bounded by a collection of lines that pass through q. By convexity, each of these lines intersects C in at most 2 points. One of these points has the same x_1-coordinate as B and the other points lie on the boundary of a $(d-1)$-ball c.

Let x_c be the x_1-coordinate of c. Since C is convex and a rotation symmetral, the surface of C below $x_1 = x_c$ is outside R. By the definition of Schwarz rotation-symmetral, the volume of C that is outside of R is below x_c, and the volume of R outside of C is above x_c. According to central identity, we have

$$\text{com}(R) = \frac{\text{vol}(R \cap C)\,\text{com}(R \cap C) + \text{vol}(R \setminus C)\,\text{com}(R \setminus C)}{\text{vol}(R)},$$

and

$$\text{com}(C) = \frac{\text{vol}(R \cap C)\,\text{com}(R \cap C) + \text{vol}(C \setminus R)\,\text{com}(C \setminus R)}{\text{vol}(C)}.$$

Since $\text{com}(R \setminus C)$ is above $\text{com}(C \setminus R)$, $\text{com}(R)$ is above $\text{com}(C)$.

The centers of mass of P' and C have the same x_1 value because, in the Schwarz

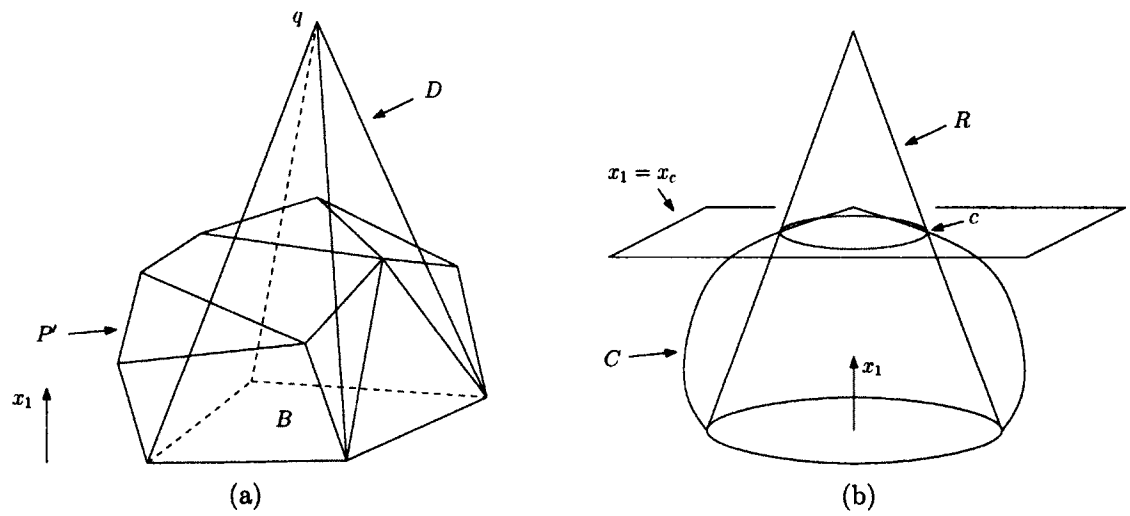

Figure 5.2: The Schwarz rotation-symmetral of P' and D

rotation-symmetral, C_x has the same x_1 value as A_x. So do the centers of mass of D and R. Since D is a pyramid, the convex hull of $p_1, \ldots, p_d$ is contained in B, $\mathrm{com}(P)$ is below $\mathrm{com}(P')$, and $\mathrm{com}(P')$ is below $\mathrm{com}(D)$, we have

$$\mathrm{vol}(p_1, \ldots, p_d, \mathrm{com}(P')) \leq \mathrm{vol}(P')/(d+1) \leq \mathrm{vol}(P)/(d+1).$$

To see why the first inequality is true, observe that

$$\begin{aligned}\mathrm{vol}(p_1, \ldots, p_d, \mathrm{com}(P')) &\leq \mathrm{vol}(B, \mathrm{com}(P')) = \mathrm{vol}(B, \mathrm{com}(C)) \\ &\leq \mathrm{vol}(B, \mathrm{com}(D)) = \mathrm{vol}(R)/(d+1) = \mathrm{vol}(P')/(d+1) \ .\end{aligned}$$

This completes the proof. □

The bound in Lemma 5.4 is tight, for example, when S consists of the $d+1$ vertices of a simplex. Next we show how this relates to Oja depth:

Theorem 5.5. *Let S be a set of points in $\mathbb{R}^d$ whose convex hull, A, has unit volume. Then* $\mathrm{dep_O}(\mathrm{com}(A), S) \leq \binom{n}{d}/(d+1)$.

Proof.

$$\begin{aligned}\mathrm{dep_O}(\mathrm{com}(A), S) &= \sum_{y_1,\ldots,y_d \in \binom{S}{d}} \mathrm{vol}(\mathrm{com}(A), y_1, \ldots, y_d) \\ &\leq \binom{n}{d}/(d+1) \ ,\end{aligned}$$

where the inequality is an application of Lemma 5.4. □

5.2 Oja Center and Mass Center of S

In this section, we show that the center of mass of S provides a constant-factor approximation of the minimum Oja depth. We begin by proving the result for point sets in 1 dimension:

Lemma 5.6. *For any finite set $S \subset \mathbb{R}$, and any $x \in \mathbb{R}$, $\mathrm{dep_O}(\mathrm{com}(S), S) \leq 2\,\mathrm{dep_O}(x, S)$.*

Proof. Denote the elements of S by $p_1, \ldots, p_n$ in any order. Let the multiset S_i contain $p_1, \ldots, p_i$ as well as $n - i$ copies of x. Let $c_i = \mathrm{com}(S_i)$. We will show, by induction on i, that $\mathrm{dep_O}(c_i, S_i) \leq 2\,\mathrm{dep_O}(x, S_i)$ for all $i \in \{0, \ldots, n\}$. This is sufficient, since $S_n = S$.

For the base case S_0 consists of n copies of x, so $c_0 = x$ and $\mathrm{dep_O}(c_0, S_0) = 0 = 2\,\mathrm{dep_O}(x, S_0)$. Next, we assume that $\mathrm{dep_O}(c_i, S_i) \leq 2\,\mathrm{dep_O}(x, S_i)$ and prove that $\mathrm{dep_O}(c_{i+1}, S_{i+1}) \leq 2\,\mathrm{dep_O}(x, S_{i+1})$. Note that

$$\mathrm{dep_O}(x, S_{i+1}) = \mathrm{dep_O}(x, S_i) + |p_{i+1} - x| \ .$$

Furthermore,

$$c_{i+1} = c_i + (p_{i+1} - x)/n \ ,$$

so

$$\begin{aligned}
&\operatorname{dep}_{\mathrm{O}}(c_{i+1}, S_{i+1}) \\
&= \operatorname{dep}_{\mathrm{O}}(c_i, S_i) + \sum_{q \in S_i} (|c_{i+1} - q| - |c_i - q|) + (|c_{i+1} - p_{i+1}| - |c_{i+1} - x|) \\
&\leq \operatorname{dep}_{\mathrm{O}}(c_i, S_i) + n|p_{i+1} - x|/n + (|c_{i+1} - p_{i+1}| - |c_{i+1} - x|) \\
&\leq \operatorname{dep}_{\mathrm{O}}(c_i, S_i) + 2|p_{i+1} - x| \\
&\leq 2 \operatorname{dep}_{\mathrm{O}}(x, S_i) + 2|p_{i+1} - x| \\
&= 2 \operatorname{dep}_{\mathrm{O}}(x, S_{i+1}) \ ,
\end{aligned}$$

as required. □

We remark that the above proof uses little more than triangle inequality. In particular, the same proof shows that the center of mass gives a 2-approximation for the Fermat-Weber center in any dimension,[2] which has also been proved by Bereg et al. with a different technique [12]. Unfortunately, in higher dimensions, Oja depth does not enjoy this nice property.

Theorem 5.7. *For any finite set* $S \subseteq \mathbb{R}^d$, $\operatorname{dep}_{\mathrm{O}}(\operatorname{com}(S), S) \leq (d+1) \operatorname{dep}_{\mathrm{O}}(x, S)$ *for any* $x \in \mathbb{R}^d$.

Proof. The case $d = 1$ is covered by Lemma 5.6, so we assume $d \geq 2$. In this proof, we will make use of the fact that, for any d-simplex T with vertex set V_T and a point $q \in \mathbb{R}^d$,

$$\operatorname{vol}(T) \leq \sum_{p_1, \ldots, p_d \in \binom{V_T}{d}} \operatorname{vol}(p_1, \ldots, p_d, q) \ , \tag{5.3}$$

since T is contained in the union of the simplices on the right hand side. Equality occurs if q is inside T.

Define S_i as in the proof of Lemma 5.6. Let S' be S_{i+1} with one occurrence of p_{i+1} removed. The induction and base case are the same as in Lemma 5.6. First, we

[2]The Fermat-Weber center of a point set S in $\mathbb{R}^d$ is the point x that minimizes $\sum_{y \in S} \|x - y\|$.

have

$$\begin{aligned} \mathrm{dep}_{\mathrm{O}}(x, S_{i+1}) &= \mathrm{dep}_{\mathrm{O}}(x, S_i) \\ &\quad + \sum_{Q \in \binom{S_i}{d-1}} \mathrm{vol}(x, p_{i+1}, Q). \end{aligned} \tag{5.4}$$

and

$$\begin{aligned} \mathrm{dep}_{\mathrm{O}}(c_{i+1}, S_{i+1}) = \mathrm{dep}_{\mathrm{O}}(c_i, S_i) \\ + \sum_{P \in \binom{S_i}{d}} (\mathrm{vol}(c_{i+1}, P) - \mathrm{vol}(c_i, P)) \end{aligned} \tag{5.5}$$

$$+ \sum_{Q \in \binom{S'}{d-1}} (\mathrm{vol}(c_{i+1}, p_{i+1}, Q) - \mathrm{vol}(c_{i+1}, x, Q)). \tag{5.6}$$

We denote by $y^{\perp}$ the projection of a point y on a line perpendicular to the $d-1$

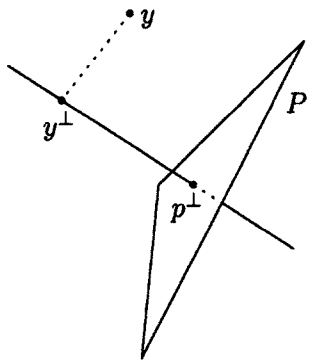

Figure 5.3: The projection of y and P

dimensional simplex P (see Figure 5.3), and by $\|pq\|$ the Euclidean distance between points p and q. Let p be any point on P, then

$$\begin{aligned} \mathrm{vol}\,|(c_{i+1}, P) - \mathrm{vol}(c_i, P)| &= (1/d)\,\mathrm{vol}_{d-1}(P) \;\big|\; \|c_i^{\perp} p^{\perp}\| - \|c_{i+1}^{\perp} p^{\perp}\| \;\big| \\ &\leq (1/d)\,\mathrm{vol}_{d-1}(P) \|c_i^{\perp} c_{i+1}^{\perp}\| \\ &\leq (1/d)\,\mathrm{vol}_{d-1}(P) \|(1/n) x^{\perp} p_{i+1}^{\perp}\| \;. \end{aligned}$$

Then if $x^\perp$ and $p_{i+1}^\perp$ are on the same side of the hyperplane supporting P, we have

$$\begin{aligned}(1/d)\,\mathrm{vol}_{d-1}(P)\|(1/n)x^\perp p_{i+1}^\perp\| &\le (1/nd)\,\mathrm{vol}_{d-1}(P)\ \big|\ \|x^\perp p^\perp\| - \|p_{i+1}^\perp p^\perp\|\ \big| \\ &\le (1/n)|\,\mathrm{vol}(p_{i+1},P) - \mathrm{vol}(x,P)| \\ &\le (1/n)\sum_{Q\in\binom{P}{d-1}} \mathrm{vol}(x,p_{i+1},Q)\end{aligned}$$

Otherwise if $x^\perp$ and $p_{i+1}^\perp$ are on different sides of the hyperplane supporting P, we have $\|x^\perp p_{i+1}^\perp\| = \|x^\perp p^\perp\| + \|p_{i+1}^\perp p^\perp\|$. In this case the two simplices Px and Pp_{i+1} are disjoints and the convex hull of Pxp_{i+1} is covered by the union of the simplices Qxp_{i+1} for $Q \in \binom{P}{d-1}$ (see Figure 5.4), thus

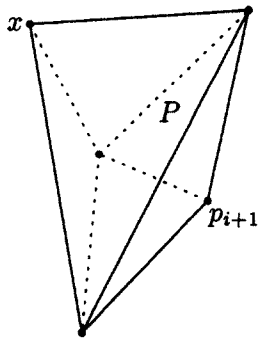

Figure 5.4: Point x and p_{i+1} are different sides of P

$$\begin{aligned}(1/d)\,\mathrm{vol}_{d-1}(P)\|(1/n)x^\perp p_{i+1}^\perp\| &\le (1/nd)\,\mathrm{vol}_{d-1}(P)(\|x^\perp p^\perp\| + \|p_{i+1}^\perp p^\perp\|) \\ &\le (1/n)(\mathrm{vol}(p_{i+1},P) + \mathrm{vol}(x,P)) \\ &\le (1/n)\sum_{Q\in\binom{P}{d-1}} \mathrm{vol}(x,p_{i+1},Q)\end{aligned}$$

We can now prove that (5.5) $\leq$ (5.4) as follows:

$$\begin{aligned}\sum_{P\in\binom{S_i}{d}} (\mathrm{vol}(c_{i+1},P)-\mathrm{vol}(c_i,P)) &\leq \sum_{P\in\binom{S_i}{d}} |\mathrm{vol}(c_{i+1},P)-\mathrm{vol}(c_i,P)| \\ &\leq (1/n)\sum_{P\in\binom{S_i}{d}}\sum_{K\in\binom{P}{d-1}} \mathrm{vol}(x,p_{i+1},K) \\ &\leq ((n-(d-1))/n)\sum_{K\in\binom{S_i}{d-1}} \mathrm{vol}(x,p_{i+1},K)\ .\end{aligned}$$

Next, we show that (5.6) $\leq d \times$ (5.4). Applying (5.3),

$$\begin{aligned}&\sum_{Q\in\binom{S'}{d-1}} (\mathrm{vol}(c_{i+1},p_{i+1},Q)-\mathrm{vol}(c_{i+1},x,Q)) \\ &\leq \sum_{Q\in\binom{S'}{d-1}} \left(\mathrm{vol}(x,p_{i+1},Q)+\sum_{R\in\binom{Q}{d-2}} \mathrm{vol}(x,p_{i+1},c_{i+1},R)\right) \\ &\leq \sum_{Q\in\binom{S_i}{d-1}} \mathrm{vol}(x,p_{i+1},Q)+(n-1-(d-2))\sum_{R\in\binom{S_i}{d-2}} \mathrm{vol}(x,p_{i+1},c_{i+1},R)\ .\end{aligned}$$

Let $\bar{\mathrm{vol}}(p_1,\ldots,p_{d+1})$ denote the signed volume of the simplex with vertices $p_1,\ldots,p_{d+1}$. By linearity of the determinant we have

$$\begin{aligned}\bar{\mathrm{vol}}(x,p_{i+1},c_{i+1},R) &= (1/n)\sum_{y\in S_{i+1}} \bar{\mathrm{vol}}(x,p_{i+1},y,R) \\ &= (1/n)\sum_{y\in S_i} \bar{\mathrm{vol}}(x,p_{i+1},y,R)\end{aligned}$$

Since the absolute value of the sum can be bounded by the sum of the absolute values, we get

$$\mathrm{vol}(x,p_{i+1},c_{i+1},R) \leq (1/n)\sum_{y\in S_i} \mathrm{vol}(x,p_{i+1},y,R),$$

and thus

$$\sum_{R \in \binom{S_i}{d-2}} \mathrm{vol}(x, p_{i+1}, c_{i+1}, R) \leq ((d-1)/n) \sum_{Q \in \binom{S_i}{d-1}} \mathrm{vol}(x, p_{i+1}, Q).$$

Thus we get (5.6) $\leq d \times$ (5.4).

Finally, we resubstitute to obtain

$$\begin{aligned} \mathrm{dep}_{\mathrm{O}}(c_{i+1}, S_{i+1}) &\leq \mathrm{dep}_{\mathrm{O}}(c_i, S_i) + (d+1) \sum_{Q \in \binom{S_i}{d-1}} \mathrm{vol}(x, p_{i+1}, Q) \\ &\leq (d+1)\,\mathrm{dep}_{\mathrm{O}}(x, S_i) + (d+1) \sum_{Q \in \binom{S_i}{d-1}} \mathrm{vol}(x, p_{i+1}, Q) \\ &= (d+1)\,\mathrm{dep}_{\mathrm{O}}(x, S_{i+1}) \ . \end{aligned}$$

Thus we have $\mathrm{dep}_{\mathrm{O}}(\mathrm{com}(S), S) \leq (d+1)\,\mathrm{dep}_{\mathrm{O}}(x, S)$, completing the proof. □

We remark that Lemma 5.6 and Theorem 5.7 are essentially the best possible. To see this, take the multiset S that contains $n-d$ copies of the origin o, and each of the remaining d points has one different coordinate 1 and all other coordinates 0. In this case $\mathrm{dep}_{\mathrm{O}}(o, S) = 1/d!$ and $\mathrm{dep}_{\mathrm{O}}(\mathrm{com}(S), S) = (d+1-O(d^2/n)) \times 1/d!$.

5.3 Conclusion

We have given several results relating Oja depth and centers of mass. Our first result is a form of centerpoint theorem for Oja depth. Our second result gives an extremely simple constant factor approximation for the Oja center. There are several directions for future work.

We do not know of any point set that gives a lower-bound matching the upper-bound of Theorem 5.5. The best lower-bound we know is that placing $n/(d+1)$ points at each vertex of any d-simplex of unit volume yields to an Oja depth of $n^d/(d+1)^d$ for any point inside the simplex. For $d = 2$, for example, Theorem 5.5 implies $\mathrm{dep}_{\mathrm{O}}(x, S) \leq n^2/6 - O(n)$ where as the best lower bound (above) has $\mathrm{dep}_{\mathrm{O}}(x, S) \geq n^2/9$. This construction leads us to the following conjecture:

Conjecture 1. *For any point set $S \subset \mathbb{R}^d$ whose convex hull has unit volume, there exists $x \in \mathbb{R}^d$, such that* $\mathrm{dep}_{\mathrm{O}}(x, S) \leq n^d/(d+1)^d$.

Chapter 6

Algorithms for Majority Depth

In this chapter we consider the problem of computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$. We assume that the points in S are in general position. For the algorithms in Section 6.3 and Section 6.4, the tools are given in Section 6.1 and Section 6.2. For the algorithm in Section 6.5, The tools are given in Section 6.5.1, and Section 3.3.

6.1 Majority Depth in the Dual Arrangement

In this section we examine the majority depth in the dual arrangement of S. Let $\mathcal{A}(T)$ be the dual arrangement [2, 46, 55] of S, where T is a set of dual hyperplanes of the points in S. For a hyperplane $\hbar$ determined by d points in S, the major side of $\hbar$ contains at least $\lceil \frac{n-d}{2} \rceil$ points in its interior, and they are either above or below $\hbar$. Let $\hbar^*$ be the dual image of $\hbar$ in $\mathcal{A}(T)$. Then, below or above $\hbar^*$ there are the same number of hyperplanes. We define the major side of $\hbar^*$ as a direction of the x_d-axis along which the ray from $\hbar^*$ intersects at least $\lceil \frac{n-d}{2} \rceil$ hyperplanes. The directions of the major sides of $\hbar$ and $\hbar^*$ are opposite since the relative position between a point and a hyperplane is reversed in the dual space. However, if a point is in the major side of $\hbar$, the dual image of the point (a hyperplane) is on the major side of $\hbar^*$.

In the dual arrangement, we call vertices red if they have major side facing down, blue if the have major side facing up, and purple if they have major side facing both

up and down (both sides are major). Then the majority depth of p is equal to the number of purple vertices plus the number of red vertices above p^* plus the number of blue vertices below p^*.

When n is odd, the vertices with level less than $\lceil \frac{n-d}{2} \rceil$ in $\mathcal{A}(T)$ are blue, and

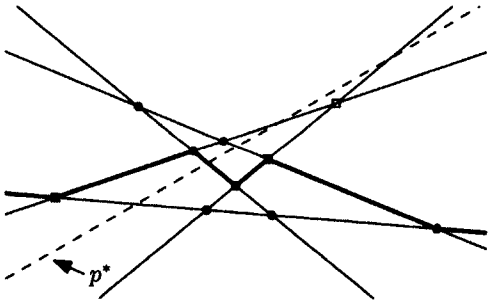

Figure 6.1: The vertices and major sides when n is odd

the ones with level more than that are red (see Figure 6.1). For each vertex on the $\lceil \frac{n-d}{2} \rceil$-level, if the convex angle of its two adjacent segments faces up it is blue, and if it faces down it is red.

When n is even, the situation is a little different. As shown in Figure 6.2, the

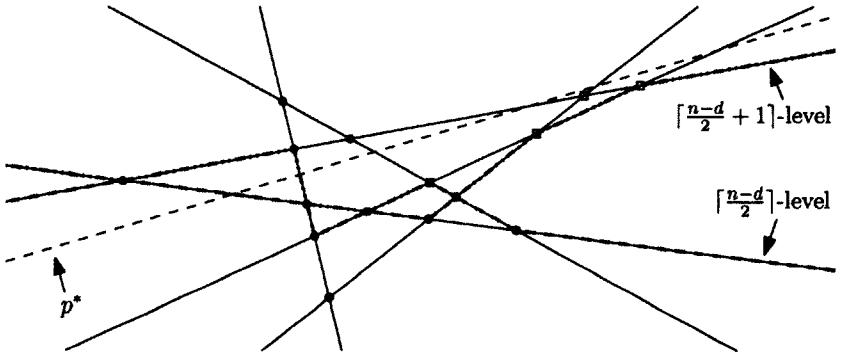

Figure 6.2: The vertices and major sides when n is even

vertices with level less than $\lceil \frac{n-d}{2} + 1 \rceil$ are blue, and the ones with level more than $\lceil \frac{n-d}{2} \rceil$ are red. The ones on both of these two levels are purple.

Computing the majority depth of p with respect to S is to count the number of major sides p is in. Since the total number of vertices in a simple arrangement is $\binom{n}{d}$, to compute the majority depth it is sufficient to count the number of vertices

in $\mathcal{A}(T)$ whose major side does not contain p^*. This problem involves counting the number of vertices of $\mathcal{A}(T)$ that are contained in a set of polygons whose boundary is determined by p^* and the median level of $\mathcal{A}(T)$. We study this problem in the next section.

6.2 Counting Vertices

In this section we discuss how to count the vertices of a 2-dimensional arrangement of n line segments confined by a simple polygon (see Figure 6.3). Since there can be $\Omega(n^2)$ intersections in this arrangement, a sweep line algorithm would take too much time. In the following we discuss a couple of more efficient ways of counting the vertices.

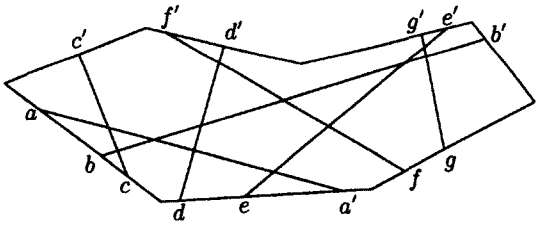

Figure 6.3: An arrangement in a simple polygon

We first transform the arrangement into a structure as shown in Figure 6.4, which makes the pattern of intersections clearer to us. In this structure, the polygon is

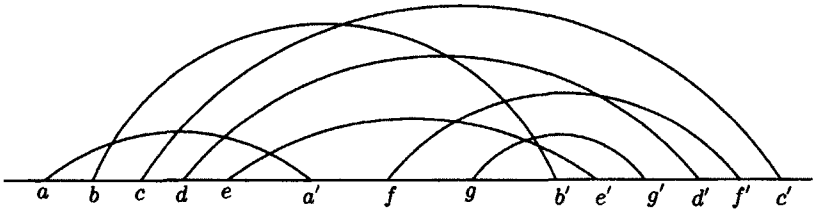

Figure 6.4: The transformed arrangement

cut at some point and laid flat, and all the line segments are bent into arcs, so that no two arcs intersect twice. The number of intersections in the new structure is the same as that in the original one, because, for any two line segments intersecting in the polygon, the corresponding arcs intersect once.

Notice that for an arc a, any other arc that intersects a has an endpoint laying between the two ends of a. To count the intersections in the new structure, we can use a queue. Starting from one end of the new structure, we add the endpoints of the arcs to the queue. Once the other end of an arc is in the queue, we count the number of endpoints between the two endpoints of the arc, which is the number of intersections the arc contributes. We then remove the two ends from the queue. Upon reaching the other end of the structure, the queue will be empty and all the intersections will be counted. If we implement the queue with an augmented binary tree [34, Chapter 14.1], finding the distance between the two ends of an arc and deleting the other end of the arc takes $O(\log n)$ time, so the number of intersections can be counted in $O(n \log n)$ time.

Another way to count the intersections is to use an array A of size $2n$. Starting from one end of the structure, we walk to the other end. Once we come across a starting end of an arc, we append a 1 to A. Once we come across a finishing end of an arc a, we append a 0 to A. Let the index of the starting end of a in A be i, and that of the finishing end be j. We then set $A[i]$ to 0. Let $sum(k)$ denote $\sum_{l \leq k} A[l]$. The number of the intersections that a contributes is the number of endpoints we came across between $A[i]$ and $A[j]$, which is $sum(j) - sum(i)$. We can compute $sum(k)$ in $O\left(\frac{\log n}{\log \log n}\right)$ time with Dietz's algorithm [39] in the word RAM model. Then counting all intersections takes $O\left(n \frac{\log n}{\log \log n}\right)$ time.

6.3 Algorithms for Bivariate Majority Depth

In this section we show how to use the intersection counting structure of the previous section to obtain an efficient algorithm for the majority depth problem in $\mathbb{R}^2$. In the following we will first describe the algorithm when n is odd, then we describe the modifications needed when n is even.

If n is odd, we first compute the median level of the dual arrangement of S and the intersections between it and p^*. If they intersect, p^* splits the levels into sections (as the schematic example shown in Figure 6.5). Each section along with p^* form a simple polygon except the leftmost and rightmost sections, which form unbounded

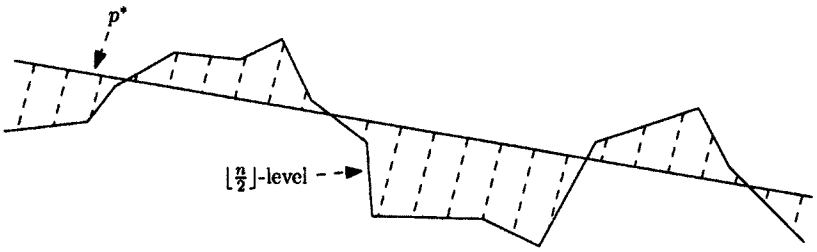

Figure 6.5: The regions when n is odd

regions. In order to count the vertices in the unbounded region with the methods in Section 6.2, we need to find the leftmost and rightmost vertices. Since the extreme points of the set of vertices of $\mathcal{A}(T)$ can be found in $O(n \log n)$ time (in $O(n \log \log n)$ time in the word RAM model [6]) by sorting the lines by slope [29], we can find those two vertices in $O(n \log n)$ time. Then we can add a vertical line to the left of the leftmost vertex, and one to the right of the rightmost vertex to bound the unbounded region (An example is shown in Figure 6.6). Now we can count the vertices in each

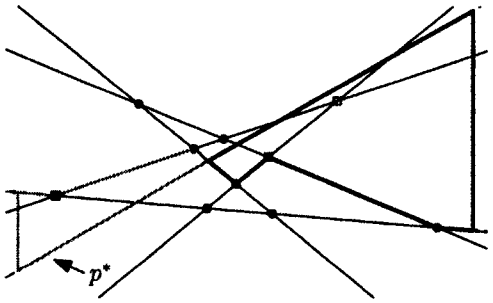

Figure 6.6: The polygons when n is odd

polygon, and the ones on the median level whose major side does not contain p^*.

If n is even we need to compute both the $\frac{n}{2}$-level and $(\frac{n}{2} - 1)$-level. The polygons should be formed by part of p^* and the one of the two median levels which is further away from p^* (see the schematic example in Figure 6.7). In Figure 6.8 is an example where all regions are bounded. Then we need to count all the vertices in the polygons, and count the vertices that on both those levels since they should be counted twice for the depth of p.

The number of lines that intersect with p^* is n, and the number of lines that intersect with the two vertical lines is no more than $2n$. Since, in the polygons, each

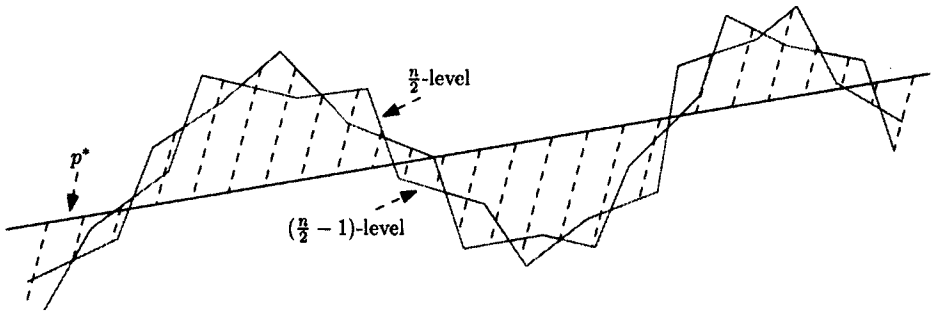

Figure 6.7: The regions when n is even

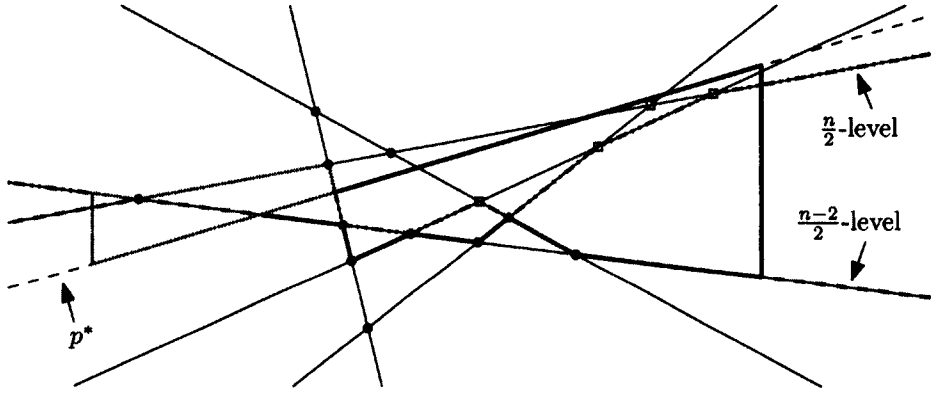

Figure 6.8: The polygons when n is even

line segment that intersects with the median level has a unique extension on the median level, the total number of line segments that intersect with the median level is no more than m, the number of vertices of the median level. Each line segment in the polygons has two ends on the boundaries, therefore, the total number of line segments in all the polygons is no more than $3n + m$.

We obtain two different algorithms for computing majority depth depending on which algorithm we use for computing the median level and counting the vertices in a polygon.

Theorem 6.1. *The majority depth in* $\mathbb{R}^2$ *can be computed in*

1. $O((n+m)\log n)$ *time with Brodal and Jacob's data structure.*
2. $O\left((n+m)\frac{\log n}{\log\log n}\right)$ *time in the word RAM model.*

The complexity of these algorithms is determined by the value of m, which is the number of vertices of the median level.

6.4 A Simple Approximation

According to Dey [38, Theorem 4.2], we have the following result for the number of vertices between two levels.

Lemma 6.2. *Let L be any set of n lines and let s be the number of vertices of $\mathcal{A}(L)$ that are on levels $k, k+1, \ldots, k+j-1$. Then, $s \in O(nk^{1/3}j^{2/3})$.*

For a constant $\epsilon > 0$, the complexity between the $(\frac{n}{2} - \lfloor \epsilon^{3/2} n \rfloor)$-level and $(\frac{n}{2} + \lfloor \epsilon^{3/2} n \rfloor)$-level is $O(\epsilon n^2)$. If we pick a random level from these two levels, the expected complexity of this level is $O(\frac{n}{\epsilon^{1/2}})$. This leads to the following approximation algorithm for majority depth.

We pick a random number $r \in \{0, \ldots, \lfloor \epsilon^{3/2} n \rfloor\}$, and compute $(\frac{n}{2} - r)$-level and $(\frac{n}{2} + r)$-level. We then count the vertices above both p^* and the $(\frac{n}{2} + r)$-level, and the ones below both p^* and the $(\frac{n}{2} - r)$-level with the technique discussed above. This approximation has an additive error $O(\epsilon n^2)$ which is given by the vertices between those two levels, and we can count the error exactly.

Since the expected complexity of each of these two levels is $O(\frac{n}{\epsilon^{1/2}})$, the expected running time is $O(\frac{n}{\epsilon^{1/2}} \log n)$ with Brodal and Jacob's data structure. Further more, the time complexity will not be changed if we count the error exactly.

6.5 A Monte Carlo Approximation

It seems difficult for any algorithm that computes the exact majority depth of a point to avoid (at least implicitly) computing the median level of $\mathcal{A}(T)$. This motivates approximation by random sampling. In particular, one can use the simple technique of sampling vertices of $\mathcal{A}(T)$ and check whether

1. each sample lies above or below p^*; and

2. each sample lies above or below the median level of $\mathcal{A}(T)$.

In the primal, this is equivalent to taking random pairs of points in S and checking, for each such pair, (x, y), if, (1) the closed upper halfplane, h_{xy}, with x and y on its boundary, contains p and (2) if h_{xy} contains $n/2$ or more points of S.

The former test takes constant time, but the later test leads to a data structuring problem: Preprocess the set T so that one can quickly test, for any query point, x, whether x is above or below the median level of $\mathcal{A}(T)$. (Equivalently, does a query halfplane, h, contain $n/2$ or more points of S.) We know of two immediate solutions to this problem. The first solution is to compute the median level explicitly, in $O(\min\{m \log n, n^{4/3}\})$ time, after which any query can be answered in $O(\log n)$ time by binary search on the x-coordinate of x. The second solution is to construct a half-space range counting structure—a partition tree—in $O(n \log n)$ time that can count the number of points of S in h_{xy} in $O(n^{1/2})$ time [19].

The first solution is not terribly good, since the algorithm in Section 6.3 that computing the *exact* majority depth of p can be done in time that is within a logarithmic factor of m, the complexity of the median level. (Though if the goal is to preprocess in order to approximate the majority depth for many different points, then this method may be the right choice.)

In this section, we show that the second solution can be improved considerably, at least when the application is approximating majority depth. In particular, we show that when the query point is a randomly chosen vertex of the arrangement $\mathcal{A}(T)$, a careful combination of partition trees [19] and ϵ-approximations [67] can be used to answer queries in $O((\log n)^{4/3})$ expected time. This faster query time means that we can use more random samples which leads to a more accurate approximation.

6.5.1 Side of Median Level Testing

We now use the results in Section 3.3 to tackle the main problem that comes up in trying to estimate majority depth in $\mathbb{R}^2$ by random sampling: Given a pair of points $x, y \in S$, determine if there are more than $n/2$ points in the upper halfspace, h_{xy}, whose boundary is the line through x and y. In the dual setting the question

becomes: Given a random vertex, x, of $\mathcal{A}(T)$, determine whether x is above or below the median level of $\mathcal{A}(T)$. The data structure in Lemma 3.9 answers these queries in time $O((n/i)^{2/3}(\log N)^{1/3})$ when the vertex x is on the $n/2-i$ or $n/2+i$ level.

To prove our main theorem, we need a special case of Lemma 6.2 where $k = n/2-i$ and $j = 2i+1$:

Corollary 6.3. *Let L be any set of n lines. Then, for any $i \in \{1,\ldots,n/2\}$ the maximum total number of vertices of $\mathcal{A}(L)$ whose level is in $\{n/2-i,\ldots,n/2+i\}$ is $O(n^{4/3}i^{2/3})$.*

Corollary 6.3 is useful because it gives bounds on the distribution of the level of a randomly chosen vertex of $\mathcal{A}(T)$.

Theorem 6.4. *Given any set, L, of n lines and any $c > 0$, there exists a data structure that can test if a point x is above or below the median level of L. For any constant, c, this structure can be made to have the following properties:*

1. *It can be constructed in $O(n\log n)$ expected time and uses $O(n)$ space;*
2. *with probability at least $1-n^{-c}$, it answers correctly for all possible queries; and*
3. *when given a random vertex of $\mathcal{A}(L)$ as a query, the expected query time is $O((\log n)^{4/3})$.*

Proof. The data structure is, of course, the data structure of Lemma 3.8 with $N = n^c$. Let n_i be the number of vertices of $\mathcal{A}(L)$ on levels $n/2-i$ and $n/2+i$. Then the expected query time of this data structure is at most

$$F(n_0,\ldots,n_{n/2}) = \frac{1}{\binom{n}{2}}\sum_{i=0}^{n/2} n_i Q(i), \tag{6.1}$$

where, for sufficiently large n, $Q(i)$ is upper-bounded by

$$Q(i) \le \begin{cases} \beta n^{1/2} & \text{for } 0 \le i \le n^{1/4} \\ \beta(n/i)^{2/3}(\log N)^{1/3} & \text{otherwise,} \end{cases}$$

for some constant $\beta > 0$. Our goal, therefore, is to upper-bound $F(n_0, \ldots, n_{n/2})$ subject to Dey's constraints (Lemma 6.2):

$$\sum_{i=0}^{j} n_i \leq \gamma n^{4/3} j^{2/3}$$

for some constant $\gamma > 0$ and all $j \in \{0, \ldots, n/2\}$.

Working in our favour is that $Q(i) \geq Q(i')$ for all $i \leq i'$. This implies that, to obtain an upper bound on $F(n_0, \ldots, n_{n/2})$, we can set

$$\sum_{i=0}^{j} n_i = \begin{cases} \gamma n^{4/3} & \text{if } j = 0 \\ \gamma n^{4/3} j^{2/3} & \text{otherwise} \end{cases} \tag{6.2}$$

for all $j \in \{0, \ldots, n/2$. To see why this is so, suppose we have a sequence $s = n_0, \ldots, n_{n/2}$ that satisfies Dey's constraints but for which $\sum_{i=0}^{j} n_i < \gamma n^{4/3} j^{2/3}$ for some index j. If $j = n/2$ then we can obviously increase the value of n_j, still satisfy Dey's constraints and increase the value of $F(n_0, \ldots, n_j)$. Otherwise ($j \in \{0, \ldots, n/2 - 1\}$), the sequence

$$s' = n_0, \ldots, n_j + \delta, n_{j+1} - \delta, \ldots, n_{n/2},$$

where $\delta = \gamma n^{4/3} j^{2/3} - \sum_{i=0}^{j} n_i$, also satisfies Dey's constraints. Furthermore,

$$F(s') - F(s) = \delta Q(j) - \delta Q(j+1) \geq 0,$$

so $F(s') \geq F(s)$. Repeatedly applying this type of modification (or using induction) shows that the sequence $s = n_0, \ldots, n_{n/2}$ that satisfies (6.2) is a sequence that maximizes $F(s)$.

Finally, we can bound the sequence that satisfies (6.2) by differentiating $\gamma n^{4/3} j^{2/3}$ with respect to j. This yields $n_i \in O(n^{4/3}/i^{1/3})$ for all $i \in \{1, \ldots, n/2\}$. Plugging this

back into (6.1) yields

$$\begin{aligned} F(n_0,\ldots,n_{n/2}) &\leq \frac{1}{\binom{n}{2}}\left(O(n^{4/3}Q(0)) + \sum_{i=0}^{n/2} O(n^{4/3}Q(i)/i^{1/3})\right) \\ &\leq o(1) \\ &\quad + \frac{1}{\binom{n}{2}} \sum_{i=1}^{n^{1/4}} O(n^{4/3}n^{1/2}/i^{1/3}) \qquad (6.3) \\ &\quad + \frac{1}{\binom{n}{2}} \sum_{i=n^{1/4}+1}^{n/2} O(n^{4/3}(n/i)^{2/3}(\log N)^{1/3}/i^{1/3}) \qquad (6.4) \end{aligned}$$

Recall that $\int_1^n i^{-1/3}\mathrm{d}i = \frac{3}{2}(n^{2/3}-1)$. Using this integral to bound the sum in (6.3) allows us to just squeak by:

$$\begin{aligned} (6.3) &= \frac{1}{\binom{n}{2}} \sum_{i=1}^{n^{1/4}} O(n^{4/3}n^{1/2}/i^{1/3}) \\ &= \frac{1}{\binom{n}{2}} O(n^{4/3}n^{1/2}(n^{1/4})^{2/3}) \quad \text{(bounding by integral)} \\ &= O(1) \end{aligned}$$

We are not so lucky with the sum in (6.4), which ends up being harmonic:

$$\begin{aligned} (6.4) &= \frac{1}{\binom{n}{2}} \sum_{i=n^{1/4}+1}^{n/2} O(n^{4/3}(n/i)^{2/3}(\log N)^{1/3}/i^{1/3}) \\ &= \sum_{i=n^{1/4}+1}^{n/2} O((\log N)^{1/3}/i) \\ &= O((\log n)(\log N)^{1/3}) \quad (\text{since } \sum_{i=1}^{n} \frac{1}{i} = O(\log n)) \\ &= O((\log n)^{4/3}), \end{aligned}$$

since $N = n^c$ and c is a constant. To summarize, the expected running time of the

query algorithm is at most

$$F(n_0, \ldots, n_{n/2}) \leq o(1) + (6.3) + (6.4) = O((\log n)^{4/3}).$$

□

6.5.2 Estimating Majority Depth

Finally, we return to our application, namely estimating majority depth.

Theorem 6.5. *Given a set S of n points in $\mathbb{R}^2$ and any constant $c > 0$, there exists a data structure that can preprocess S in $O(n \log n)$ expected time such that, for any point p, the data structure can compute, in $O(r(\log n)^{4/3})$ expected time, a value $\mathrm{dep_M}'(p, S)$ such that*

$$\Pr\left\{\frac{|\mathrm{dep_M}'(p,S) - \mathrm{dep_M}(p,S)|}{\mathrm{dep_M}(p,S)} \geq \epsilon\right\} \leq \exp(-\Omega(\epsilon^2 r d_n)) + n^{-c},$$

where $\mathrm{dep_M}(p, S)$ is the majority depth of p with respect to S and $d_n = \mathrm{dep_M}(p,S)/\binom{n}{2}$ is the normalized majority depth of p.

Proof. The data structure is the one described in Theorem 6.4. Let $d_n = \mathrm{dep_M}(p,S)/\binom{n}{2}$. Select r random vertices of $\mathcal{A}(T)$ (by taking random pairs of lines in T) and, for each sample, test if it contributes to $\mathrm{dep_M}(p, S)$. This yields a count $r' \leq r$ where

$$\mathrm{E}[r'] = r d_n.$$

We then return the value $\mathrm{dep_M}'(p,S) = (r'/r)\binom{n}{2}$, so that $\mathrm{E}[\mathrm{dep_M}'(p,S)] = \mathrm{dep_M}(p,S)$, as required.

To prove the error bound, we use the fact that r' is a binomial (d_n, r) random variable. Applying Chernoff Bounds [28] on r' yields:

$$\Pr\{|r' - r d_n| \geq \epsilon r d_n\} \leq \exp(-\Omega(\epsilon^2 r d_n)).$$

Finally, the algorithm may fail not because of badly chosen samples. but rather,

because the data structure of Theorem 6.4 fails. The probability that this happens is at most n^{-c}. Therefore, the overall result follows from the union bound. □

6.6 Conclusion

We have given an algorithm for computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$. The algorithm's running time is dependent on the size of the median level of the dual arrangement of S. Even without leaving 2 dimensions, this work leaves several open questions:

1. (Depth of a point) Is there an $O(n \log^{O(1)} n)$ time algorithm for computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$?

2. (Deepest point) Given a set S of n points in $\mathbb{R}^2$, how quickly can we compute a point p whose majority depth (with respect to S) is maximum?

3. (Centerpoint) Determine the maximum value $k = f(n)$ for which the following statement is true: For any set S of n points in $\mathbb{R}^2$, there exists a point $p \in \mathbb{R}^2$ whose majority depth, with respect to S, is at least $\binom{n}{2}/2 + k$.

4. (Faster algorithm in the word RAM model) The related problem of counting inversions has recently been solved in $O(n\sqrt{\log n})$ running time [20]. This unfortunately does not improve our algorithm. Can the factor $\frac{\log n}{\log \log n}$ in the running time of our algorithm be replaced by $\sqrt{\log n}$?

An algorithm for the first problem would have to avoid computing the median level. The second problem is easily solved in $O(n^4)$ time and $O(n^2)$ space by traversing the arrangement of lines through all $\binom{n}{2}$ pairs of points in S using the topological sweep algorithm [47].

Although the estimation of majority depth is the original motivation for studying this problem, the underlying question of the tradeoffs involved in preprocessing for testing whether a point is above or below the median level seems a fundamental question that is still far from answered. In particular, we have no good answer to the following question:

5. What is the fastest linear-space data structure for testing if an arbitrary query point is above or below the median level of a set of n lines?

To the best of our knowledge, the current state of the art is partition trees, which can only answer these queries in $O(n^{1/2})$ time.

Chapter 7

Conclusion

Due to the geometric nature of the data depth problems, the development of computational geometry has been contributing greatly to the development of data depth. In the following we summarize our contributions and directions for future work.

7.1 Summary of Contributions

7.1.1 Algorithms for Tukey Depth

This thesis contains three main contributions related to Tukey depth.

- A new fixed parameter tractable algorithm for Tukey depth that is fast when the depth is small (Section 4.1).
- An extremely simple approximation algorithms for Tukey depth (Section 4.2)
  - Theoretical bounds and experimental results that show these algorithms work even better than the theoretical bounds suggest (Section 4.2.3).
- New bounds on sizes of pseudoballs and its generalizations in the arrangement of hyperplanes in $\mathbb{R}^d$ (Section 3.2).

7.1.2 New Results on Oja Depth

This thesis presents two new contributions related to Oja depth.

- A centerpoint Theorem showing that Oja depth of center point can be upper-bounded by a constant fraction of the convex hull. This is devised by an (apparently new) theorem about simplices contained in convex bodies (Section 5.1).
- An approximation algorithm obtained by proving that the center of mass of the point set minimizes Oja depth to within a constant factor (Section 5.2).

7.1.3 Algorithms for Majority Depth

This thesis contains three new contributions related to the majority depth in $\mathbb{R}^2$.

- An exact algorithm by counting the vertices of the dual arrangement of S in a simple polygon (Section 6.3).
- An approximation algorithm, in which the expected running time of computing the "median level" is less the worst case running time of that in the exact algorithm (Section 6.4).
- Another approximation algorithm, in which we use a new range counting approximation data structure (Section 6.5).

7.2 Directions for future work

- For any point set $S \subset \mathbb{R}^d$ whose convex hull has unit volume, can we always find an $x \in \mathbb{R}^d$, such that $\mathrm{dep_O}(x, S) \leq n^d/(d+1)^d$?
- Is there an $O(n \log^{O(1)} n)$ time algorithm for computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$?
- Given a set S of n points in $\mathbb{R}^2$, how quickly can we compute a point p whose majority depth (with respect to S) is maximum?

- Determine the maximum value $k = f(n)$ for which the following statement is true: For any set S of n points in $\mathbb{R}^2$, there exists a point $p \in \mathbb{R}^2$ whose majority depth, with respect to S, is at least $\binom{n}{2}/2 + k$.

- The related problem of counting inversions has recently been solved in $O(n\sqrt{\log n})$ running time [20]. This unfortunately does not improve our algorithm in Section 6.3. Can the factor $\frac{\log n}{\log \log n}$ in the running time of this algorithm be replaced by $\sqrt{\log n}$?

- What is the fastest linear-space data structure for testing if an arbitrary query point is above or below the median level of a set of n lines?

Bibliography

[1] P. Afshani and T. M. Chan. On approximate range counting and depth. In *SCG '07: Proceedings of the twenty-third annual symposium on Computational geometry*, pages 337–343, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-705-6.

[2] P. Agarwal and M. Sharir. Arrangements and their applications. In *Handbook of Computational Geometry*, pages 49–119. Elsevier Science Publishers North-Holland, 1998.

[3] G. Aloupis. Geometric measures of data depth. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2006.

[4] G. Aloupis, S. Langerman, M. Soss, and G. Toussaint. Algorithms for bivariate medians and a Fermat-Toricelli problem for lines. *Computational Geometry: Theory and Applications*, 26(1):69–79, 2003.

[5] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147 (1–2):181–210, 1995.

[6] A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in linear time? *Journal of Computer and System Sciences*, 57(1):74–93, 1998.

[7] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`.

[8] I. Bárány. A generalization of Carathéodory's Theorem. *Discrete Mathematics*, 40:141–150, 1982.

[9] V. Barnett. The ordering of multivariate data. *Journal Of The Royal Statistical Society Series A-Statistics In Society*, 139(3):318–355, 1976.

[10] G. Bassett. Equivariant, monotonic, 50% breakdown estimators. *The American Statistician*, 45(2):135–137, 1991.

[11] P. Beling and N. Megiddo. Using fast matrix multiplication to find basic solutions. *Theoretical Computer Science*, 205(1-2):307–316, 1998.

[12] S. Bereg, B. Bhattacharya, D. Kirkpatrick, and M. Segal. Competitive algorithms for maintaining a mobile center. *Mobile Networks and Applications*, 11(2):177–186, 2006.

[13] E. Boros and Z. Füredi. The maximal number of covers by the triangles of a given vertex set in the plane. *Geometrica Dedicata*, 17:69–77, 1984.

[14] D. Bremner, K. Fukuda, and V. Rosta. Primal dual algorithms for data depth. In *Data Depth: Robust Multivariate Analysis, Computational Geometry, and Applications*, AMS DIMACS Book Series, 2006.

[15] D. Bremner, D. Chen, J. Iacono, S. Langerman, and P. Morin. Output-sensitive algorithms for Tukey depth and related problems. *Statistics and Computing*, 18: 259–266, 2008.

[16] G. Brodal and R. Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 617–626, 2002.

[17] N. Chakravarti. Some results concerning post-infeasibility analysis. *European Journal Of Operational Research*, 73(1):139–143, 1994.

[18] T. Chan. Remarks on k-level algorithms in the plane. Manuscript, 1999.

[19] T. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47: 661–690, 2012.

[20] T. Chan and M. Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the 21st ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 161–173, 2010.

[21] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 430–436, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[22] T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005. ISSN 0097-5397.

[23] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4(5):467–489, 1989. ISSN 0179-5376.

[24] D. Chen. A branch and cut algorithm for the halfspace depth problem. Master's thesis, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, 2007.

[25] D. Chen and P. Morin. Approximating majority depth. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG 2012)*, pages 237–242, 2012. Invited to special issue of Computational Geometry: Theory and Applications for CCCG 2012.

[26] D. Chen, O. Devillers, J. Iacono, S. Langerman, and P. Morin. Oja centers and centers of gravity. *Computational Geometry: Theory and Applications*, 46(2): 140–147, 2013. Special Issue on the Canadian Conference on Computational Geometry (CCCG 2010).

[27] D. Chen, P. Morin, and U. Wagner. Absolute approximation of Tukey depth: Theory and experiments. *Computational Geometry: Theory and Applications*, 46(5):566–573, 2013. Special Issue on Geometry and Optimization.

[28] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[29] Y. Ching and D. Lee. Finding the diameter of a set of lines. *Pattern Recognition*, 18(3-4):249–255, 1985.

[30] J. W. Chinneck. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*, volume 118 of *International Series in Operations Research and Management Sciences.* Springer, New York, USA, 2008. ISBN 978-0387749310.

[31] V. Chvátal. *Linear Programming.* W. H. Freeman and Company, New York, 1983.

[32] K. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM*, 42(2):488–499, 1995.

[33] K. L. Clarkson. A bound on local minima of arrangements that implies the upper bound theorem. *Discrete and Computational Geometry*, 10:427–233, 1993.

[34] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, MA USA, 2nd edition, 2001.

[35] P. Cortez and A. Morais. A data mining approach to predict forest fires using meteorological data. In J. Neves, M. F. Santos, and Machado J., editors, *New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence*, pages 512–523. Springer, 2007.

[36] J. A. Cuesta-Albertos and A. Nieto-Reyes. The random Tukey depth. *Computational Statistics and Data Analysis*, 52(11):4979–4988, 2008. ISSN 0167-9473.

[37] E. Demaine and M. Pătraşcu. Tight bounds for dynamic convex hull queries (again). In *Proceedings of the 23rd annual ACM symposium on Computational geometry*, SoCG '07, pages 354–363, New York, NY, USA, 2007. ACM.

[38] T. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.

[39] P. Dietz. Optimal algorithms for list indexing and subset rank. In F. Dehne, J. Sack, and N. Santoro, editors, *Algorithms and Data Structures*, volume 382 of *Lecture Notes in Computer Science*, pages 39–46. Springer Berlin, 1989.

[40] D. Donoho and M. Gasko. Breakdowm properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20(4): 1803–1827, 1992.

[41] D. Donoho and P. Huber. The notion of breakdown-point. In P. Bickel, K. Doksum, and J. Hodges, editors, *A Festschrift for Erich L. Lehmann*, pages 157–184. Wadsworth, Belmont, CA. USA, 1983.

[42] D. L. Donoho. Breakdown properties of multivariate location estimators. Ph.D. Qualifying Paper, Department of Statistics, Harvard University, 1982.

[43] R. Downey and M. Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer, New York, 1999.

[44] S. Durocher and D. Kirkpatrick. The projection median of a set of points. *Computational Geometry: Theory and Applications*, 42(5):364–375, 2009. Special Issue on the Canadian Conference on Computational Geometry (CCCG 2005 and CCCG 2006).

[45] M. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM Journal on Computing*, 13(1):31–45, 1984.

[46] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Heidelberg, Germany, 1987. ISBN 0-387-13722-X.

[47] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38(1):165–194, 1989.

[48] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM Journal on Computing*, 15(1):271–284, 1986.

[49] C. Edwards and D. Penney. *Calculus with analytic geometry.* Prentice Hall, Upper Saddle River, NJ USA, 5th edition, 1998.

[50] D. Fischer, J. Mttnen, K. Nordhausen, and D. Vogel. *OjaNP: Multivariate Methods Based on the Oja Median and Related Concepts*, 2010. R package version 0.9-4 `http://cran.r-project.org/web/packages/OjaNP/index.html`.

[51] K. Fukuda and V. Rosta. Data depth and optimization. Technical report, [Cited 9 Dec 2006], Available at `http://www.ifor.math.ethz.ch/about_us/press/Leitartikel_Marz_2005.pdf`, 2005.

[52] A. Gajentaan and M. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5(3):165–185, 1995.

[53] C. Gonzaga. An algorithm for solving linear programming programs in $O(n^3l)$ operations. In *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 1–28, New York, 1989. Springer-Verlag.

[54] H. Guggenheimer. *Applicable Geometry: Global and Local Convexity.* Applied Mathematics Series. R. E. Krieger Pub. Co, Huntington, New York, 1977.

[55] D. Halperin. Handbook of discrete and computational geometry. chapter 24, pages 529–562. Chapman and Hall / CRC, Boca Raton, FL, USA, 2nd edition, 2004.

[56] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, New York, 8 edition, 2005.

[57] J. L. Hodges. A bivariate sign test. *The Annals of Mathematical Statistics*, 26 (3):523–527, 1955.

[58] D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6(1):93–107, 1978.

[59] S. Langerman and W. Steiger. Optimization in arrangements. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607, pages 50–61, London, UK, 2003. Springer-Verlag. ISBN 3-540-00623-0.

[60] R. Liu. On a notion of data depth based on random simplices. *Annals of Statistics*, 18(1):405–414, 1990.

[61] R. Liu and K. Singh. A quality index based on data depth and multivariate rank tests. *Journal of the American Statistical Association*, 88(421):252–260, 1993.

[62] R. Liu, J. Parelius, and K. Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference, (with discussion and a rejoinder by Liu and Singh). *Annals of statistics*, 27(3):783–858, 1999.

[63] H. Lopuhaä and P. Rousseeuw. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals of Statistics*, 19 (1):229–248, 1991.

[64] J. Matoušek. *Lectures in Discrete Geometry.* Springer-Verlag, New York, NY, 2002.

[65] J. Matoušek. Computing the center of planar point sets. In J.E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the Special Year*, pages 221–230. AMS, Providence, 1991.

[66] J. Matoušek. On geometric optimization with few violated constraints. *Discrete and Computational Geometry*, 14:365–384, 1995.

[67] J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and approximations for bounded vc-dimension. *Combinatorica*, 13:455–466, 1993.

[68] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[69] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, 1984.

[70] N. Megiddo. On finding primal- and dual-optimal bases. *ORSA Journal on Computing*, 3(1):63–65, 1991.

[71] K. Mosler. *Multivariate Dispersion, Central Regions and Depth: The Lift Zonoid Approach*, volume 165 of *Lecture Notes in Statistics*. Springer, New York, 2002.

[72] K. Mulmuley. Dehn-Sommerville relations, upper bound theorem, and levels in arrangements. In *SCG '93: Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 240–246, New York, NY, USA, 1993. ACM.

[73] A. Niinimaa, H. Oja, and M. Tableman. The finite-sample breakdown point of the Oja bivariate median and of the corresponding half-samples version. *Statistics & Probability Letters*, 10(4):325–328, 1990.

[74] A. Niinimaa, H. Oja, and M. Tableman. The finite-sample breakdown point of the Oja bivariate median and of the corresponding half-samples version. *Statistics & Probability Letters*, 10:325–328, 1990.

[75] H. Oja. Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, 1(6):327–332, 1983.

[76] J. Pach and P. K. Agarwal. *Combinatorial Geometry*. John Wiley & Sons, New York, NY, 1995.

[77] E. Rafalin and D. Souvaine. Computational geometry and statistical depth measures. In M. Hubert, G. Pison, A. Struyf, and S. Van Aelst, editors, *Theory and Applications of Recent Robust Methods*, Statistics for Industry and Technology, pages 283–296. Birkhauser, Basel, 2004.

[78] J. Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.

[79] T. Ronkainen, H. Oja, and P. Orponen. Computation of the multivariate Oja median. In R. Dutter and P. Filzmoser, editors, *International Conference on Robust Statistics (ICORS 2001)*, 2003.

[80] C. Roos. An $O(n^3l)$ approximate center method for linear programming. In Szymon Dolecki, editor, *Optimization*, volume 1405 of *Lecture Notes in Mathematics*, pages 147–158. Springer, Berlin, 1989.

[81] C. Roos and J. -Ph. Vial. A polynomial method of approximate centers for linear programming. *Mathematical Programming*, 54:295–305, 1992.

[82] P. J. Rousseeuw and M. Hubert. Regression depth. *Journal of the American Statistical Association*, 94(446):388–402, 1999.

[83] P. J. Rousseeuw and A. Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8(3):193–203, 1998.

[84] J. K. Sankaran. A note on resolving infeasibility in linear-programs by constraint relaxation. *Operations Research Letters*, 13(1):19–20, 1993.

[85] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, New York, NY, USA, 1986.

[86] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete and Computational Geometry*, 6:423–434, 1991.

[87] M. Shamos. Geometry and statistics: Problems at the interface. In J. Traub, editor, *Algorithms and Complexity: New Direction and Recent Results*, pages 251–180. Academic Press Inc., 1976.

[88] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In A. Finkel and M. Jantzen, editors, *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 567–579. Springer, Berlin, 1992.

[89] M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for k-sets in three dimensions. *Discrete & Computational Geometry*, 26:195–204, 2001.

[90] K. Singh. A notion of majority depth. Technical report, Department of Statistics, Rutgers University, 1991.

[91] G. Tóth. Point sets with many k-sets. *Discrete & Computational Geometry*, 26 (2):187–194, 2001.

[92] J. Tukey. Mathematics and the picturing of data. In *International Conference of Mathematicians*, 1971.

[93] J. W. Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians: Vancouver*, volume 2, pages 523–531, Montreal, 1975. Canadian Mathematical Congress.

[94] P. Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)l)$ arithmetic operations. *Mathematical Programming*, 47:175–201, 1990.

[95] S. Vavasis and Y. Ye. Identifying an optimal basis in linear programming. *Annals of Operations Research*, 62:565–572, 1996.

[96] R. Webster. *Convexity.* Oxford University Press, New York, USA, 1995.

[97] E. Welzl. On spanning trees with low crossing numbers. In B. Monien and T. Ottmann, editors, *Data Structures and Efficient Algorithms: Final Report on the DFG Special Joint Initiative, B. Monien and Th. Ottmann (Eds.), LNCS 594*, Lecture Notes in Computer Science, pages 233–249. Springer, London, 1992.

[98] R. Wilcox. Approximating Tukey's depth. *Communications in Statistics - Simulation and Computation*, 32(4):977–985, 2003.

[99] Y. Zuo and R. Serfling. General notions of statistical depth function. *The Annals of Statistics*, 28(2):461–482, 2000.