Efficient Algorithms for Maximum Regression Depth

Marc van Kreveld
Dept. Comp. Sci.
Utrecht Univ.
marc@cs.uu.nl

Joseph S. B. Mitchell*
Dept. Applied Math. & Stat.
SUNY Stony Brook
jsbm@ams.sunysb.edu

Peter Rousseeuw
Dept. Math. and Comp.
Univ. Instelling Antwerpen
rousse@uia.ua.ac.be

Micha Sharir†
School Math. Sciences
Tel Aviv Univ.
sharir@math.tau.ac.il

Jack Snoeyink‡
Dept. Comp. Sci.
UNC Chapel Hill
snoeyink@cs.unc.edu

Bettina Speckmann§
Inst. Theor. Comp. Sci.
ETH Zürich
speckman@inf.ethz.ch

October 3, 2002

Abstract

We investigate algorithmic questions that arise in the statistical problem of computing lines or hyperplanes of maximum *regression depth* among a set of n points. We work primarily with a dual representation and find points of maximum *undirected depth* in an arrangement of lines or hyperplanes. An $O(n^d)$ time and space algorithm computes directed depth of all points in d dimensions. Properties of undirected depth lead to an $O(n \log^2 n)$ time and $O(n)$ space algorithm for computing a point of maximum depth in two dimensions, which has been improved to an $O(n \log n)$ time algorithm by Langerman and Steiger [17]. Furthermore, we describe the structure of depth in the plane and higher dimensions and also give approximation algorithms for hyperplane arrangements and degenerate line arrangements.

1 Introduction

Motivated by the study of *robust regression* in statistics [13, 20, 24, 21, 22, 25, 23, 28, 27], Peter Rousseeuw posed the question of computing maximum regression depth in his invited talk at the 14th ACM Symposium on Computational Geometry: Given n points P, the *regression depth* of a line is the minimum number of points that must be removed from P to allow the line to rotate to vertical about a pivot point on the line to a vertical position without ever containing a point of P.[1] (This definition is given more generally in the next section.)

*Research largely conducted while the author was a Fulbright Research Scholar at Tel Aviv University. The author is partially supported by NSF (CCR-9504192, CCR-9732220), Boeing, Bridgeport Machines, Sandia, Seagull Technology, and Sun Microsystems.

†Supported by NSF Grants CCR-97-32101 and CCR-94-24398, by grants from the U.S.-Israeli Binational Science Foundation, the G.I.F., the German-Israeli Foundation for Scientific Research and Development, and the ESPRIT IV LTR project No. 21957 (CGAL), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

‡Supported in part by grants from NSERC, the Killam Foundation, and CIES while at the University of British Columbia.

§Supported by the Berlin-Zürich Graduate Program "Combinatorics, Geometry, and Computation", financed by ETH Zürich and the German Science Foundation (DFG).

[1] Rousseeuw also posed a combinatorial question, recently resolved by Amenta et al. [1], who show that for any set of n points in R^d, there exists a hyperplane with regression depth at least $\lceil n/(d+1) \rceil$.

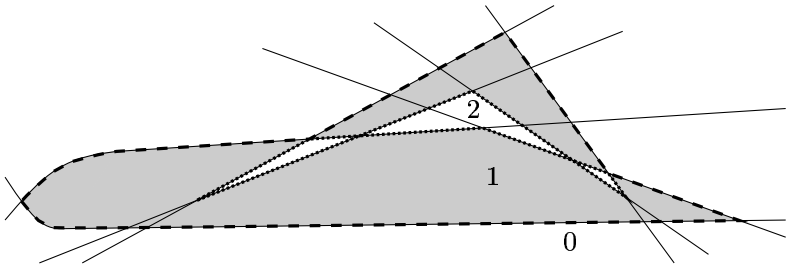

Figure 1: Arrangement with cells of depth 0, 1 (shaded), and 2; maximum depth of 3 occurs at 8 vertices and two edges. (Some lines are curved to fit all intersections on the page)

A line (or hyperplane) of maximum depth has statistical properties that are desirable as a robust regression estimator [29, 30]. The experimental investigation of these properties has been hampered by the inefficiency of the straightforward algorithms for computing maximum depth. These required $\Theta(n^3)$ time in the plane [20] and $\Theta(n^{2d-1} \log n)$ time in dimensions $d \geq 3$ [21, 23].

In the next section, we define an equivalent dual problem, computing *undirected depth* in an arrangement of lines or hyperplanes. The properties of undirected depth will lead to an $O(n^d)$ algorithm for computing regression depth for all dimensions. In Section 3, we focus on arrangements in the plane where additional properties give us an algorithm to compute one line of maximum regression depth in $O(n \log^2 n)$ time. In Section 4, we study the combinatorial complexity of the set of all lines (or hyperplanes) with maximum regression depth and its relationship to k-sets. In Section 5, we comment on algorithms for computing depth in higher dimensions.

2 Duality and undirected depth in arrangements

Although regression depth is defined for a line or hyperplane among n points, it is easier to work with a duality transformation that maps points to hyperplanes and vice versa. We use the duality from Edelsbrunner's book [8]: an inversion about the unit paraboloid $x_d = x_1^2 + x_2^2 + \cdots + x_{d-1}^2$ that maps a point $(p_1, p_2, \ldots, p_d)$ to the hyperplane $x_d = 2p_1x_1 + 2p_2x_2 + \cdots + 2p_{d-1}x_{d-1} - p_d$ and maps a hyperplane $x_d = a_1x_1 + a_2x_2 + \cdots + a_{d-1}x_{d-1} + b$ to the point $(a_1/2, a_2/2, \cdots, a_{d-1}/2, -b)$. This duality preserves point/line incidence and above/below relationships. Note that the duality mapping will neither accept nor produce vertical hyperplanes, which have equations that do not involve the variable x_d.

All rotations of a hyperplane h can be generated as follows. Choose a set of d points Q that define h; that is, each point in Q satisfies the hyperplane equation of h and together they determine the coefficients of this plane equation. (Equivalently, h is the affine hull of Q.) Move one of the points $q_0 \in Q$ by increasing its last coordinate toward infinity. If the points Q are still taken to define h, then h rotates toward the vertical about the $(d-1)$-flat defined by points of $Q \setminus \{q_0\}$.

The dual of a rotation is easy to interpret. The points of Q map to hyperplanes through a common point h^D. Hyperplane ${q_0}^D$ moves parallel to itself up the x_d axis, so the point common to all hyperplanes moves from h^D toward infinity along a ray that is contained in the duals of the stationary points.

Given n primal points P, the number that must be removed to allow a particular rotation are the number that are passed over by the rotation, plus the number that are on the final vertical plane (which our rotation never reaches). This number can be counted in the dual as the number

of hyperplanes dual to points in P that are crossed by the ray corresponding to the rotation, plus the number of hyperplanes parallel to the ray. Therefore, for an arrangement of n hyperplanes $\mathcal{A}$, we define the *undirected depth*, or just *depth*, of a point p to be the minimum number of hyperplanes intersected by some ray from p, counting parallel hyperplanes as intersecting at infinity. Hyperplanes containing p are counted for all rays. For the rest of this paper we focus on computing depth of a point in an arrangement of n lines or hyperplanes.

Since all points in the same cell C of an arrangement have the same depth, we can use the notation depth(p) or depth(C) for the value of undirected depth. (In this paper, unless otherwise stated, we use the word *cell* to refer to a full-dimensional cell in an arrangement.) Figure 1 shows a two dimensional example with labels for some cells of depth 0, 1, and 2; the maximum depth of 3 occurs at 8 vertices and two edges.

The *directions for a cell* C are the directions of rays that intersect depth(C) lines or hyperplanes of the arrangement. We can call such rays *witnesses* that the cell has a certain depth. We next observe three simple lemmas about depth by translating witness rays in the arrangement of hyperplanes in R^d: 1) depth of lower dimensional features in the arrangement can be determined from depth of d-dimensional cells, 2) directions are disjoint for adjacent cells of the same depth, and 3) directions determining depth are inherited from adjacent cells of lower depth.

Lemma 1 *In an arrangement of hyperplanes, let p be a point on k hyperplanes, and let i be the minimum of the depths of cells whose closure contains p. Then depth(p) $= i + k$.*

Proof: First we can observe that depth(p) $\leq i + k$: a ray that starts in the cell and crosses i hyperplanes can be translated to start at p at the cost of crossing all hyperplanes through p that it did not cross before. Second, if we take a ray not contained in a hyperplane incident on p that witnesses depth(p) and translate its starting point infinitesimally into the first cell entered by the ray, we can observe that there is an adjacent cell with depth depth(p) $- k$, which is therefore the minimum cell depth i. ∎

Lemma 2 *In an arrangement of hyperplanes, let h be a hyperplane that separates a cell B of depth i from a cell A of depth at least i. No witness ray for B crosses h.*

Proof: Let ρ be a ray from B that crosses h, and let ρ' be a translation of this ray that begins in A. Translated ray ρ' intersects the same hyperplanes as ρ, except for h. But since ρ' intersects at least i hyperplanes, ρ intersects at least $i + 1$ hyperplanes and is not a witness ray for B. ∎

Lemma 3 (Inheritance lemma) *The directions for a cell of depth i are the union of the directions for the adjacent cells of depth $i - 1$.*

Proof: We prove that the directions for a cell with depth(A) $= i$ contain the union. For any adjacent cell B of depth $i - 1$, let ray ρ be a witness for B. By Lemma 2, translating ρ to start in A adds at most one (and, therefore, exactly one) intersection, and provides a witness that A inherits the direction of ρ.

To prove the other inclusion, take a witness ρ' that depth(A) $= i$. We can choose the start point of ρ' so that ρ' does not pass through any vertex of the arrangement. By clipping ρ' to start in an adjacent cell B, we obtain a witness that depth(B) $\leq i - 1$. But the depth of B cannot be less than $i - 1$, since depth(A) $= i$ and we already know that A inherits all directions for B with only one more intersection. Thus, the directions for A are contained in the union. ∎

As a corollary of Lemma 3, the depth of all points with respect to a set of hyperplanes can be computed by constructing the arrangement of hyperplanes [9, 10] and labeling cells in a breadth-first search. The unbounded cells are labeled with their depth zero. Then, for $i = 1, 2, \ldots$, all cells with label $i-1$ cause their adjacent, unlabeled cells to be labeled i. Finally, lower-dimensional cells can be labeled according to Lemma 1.

Corollary 4 *For n hyperplanes in R^d, the depth of each cell can be computed in $O(n^d)$ time by building the arrangement and traversing the graph of adjacent cells.*

3 An algorithm for maximum depth cells in the plane

Undirected depth in two dimensions satisfies some additional properties that allow an efficient algorithm to compute a 2-dimensional cell of maximum depth. Recently Langerman and Steiger [17] have built upon the results presented in this section to give an optimal algorithm that finds a maximum depth cell in $O(n \log n)$ time and linear space.

Suppose that we are given a set L of n lines in the plane, which we may assume are not vertical. For the moment, let us also assume that they are in general position—we will relax this assumption in Subsection 4.5. Our goal is to find, among all the points of the plane that do not lie on lines of L, a point p whose depth is maximum. Note that vertices of the arrangement $\mathcal{A}(L)$ may attain greater depth than p—we return to these in Subsection 4.1.

We will use a binary search on x-coordinates of vertices of the arrangement $\mathcal{A}(L)$, with a test for which side of a vertical line contains a maximum depth cell. Subsection 3.1 establishes properties that allow a sidedness test; Subsection 3.2 describes a tournament data structure needed to implement the sidedness test.

3.1 A sidedness test

In the plane, we use two concepts to determine which side of a vertical test line can have cells of maximum depth: a "wedge lemma" and the notion of "top directions."

Lemma 5 (Wedge lemma) *Let p be a point, possibly on one line of L, and let u and v be directions of rays from p that each cross at most i other lines. No cell intersecting the convex wedge (cone) defined by these rays from p has depth greater than i.*

Proof: Consider the lines that intersect the union of rays from p in directions u and v. There are at most $2i+1$ intersections, if we count the line containing p only once. If we translate this union within the wedge, although we may lose intersections with lines that intersect both rays, we will not gain intersections. Thus, if the apex is inside a cell of the line arrangement, one of the translated rays will witness that the depth is at most i. ∎

The wedge lemma can be helpful for identifying maximum depth cells, as in the following corollary.

Corollary 6 *Suppose that a cell C has three directions u, v, and w that span the plane by positive linear combinations and witness the value of depth(C). Then C is a deepest cell.*

Proof: Apply the wedge lemma to the three wedges defined by pairs of directions. ∎

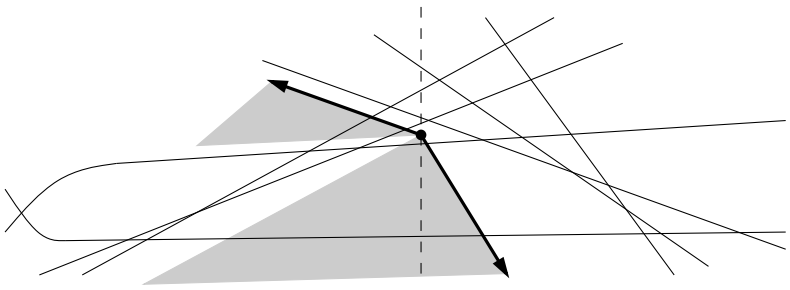

Figure 2: Directions (shaded) and top directions

We can order the witness rays for a cell C by increasing slope to the right of C and decreasing slope to the left. We call the two extreme directions for witness rays the *top directions* for the cell C. There will be a single *top direction* when one side of the line has no witness rays, or when the ray upward is a witness. Figure 2 illustrates a cell with two top directions.

Suppose that we can determine the top directions for all cells along a vertical line ℓ. Then next lemma shows that we can then determine whether a maximum depth cell occurs to the left or right of ℓ. We give an algorithmic proof, since this becomes part of our procedure for computing maximum depth.

Lemma 7 *Given a vertical test line ℓ that does not pass through any vertex in an arrangement of n lines in the plane, and given a top direction for each cell intersected by ℓ, one can determine one side of ℓ that intersects a maximum depth cell.*

Proof: Let i denote the maximum depth of the cells intersected by ℓ. We will be able to sweep up the line ℓ and, on one of the sides of ℓ, maintain a region R that does not intersect a cell of depth greater than i. Region R is, in fact, a wedge from the vertical downward direction, v, to a top direction, u, as illustrated in Figure 3.

Initially, we choose a point $p \in \ell$ in the lowest cell, which will have two top directions, to the right and left of the vertical downward direction v. We chose a top direction as u and form the wedge R. Note that R is contained in this lowest cell, which has depth $0 \leq i$.

Now, move the point p up the line ℓ. As long as p remains in its cell, the top direction does not change; the region R is enlarged by this motion, but cannot intersect a cell of depth greater than i.

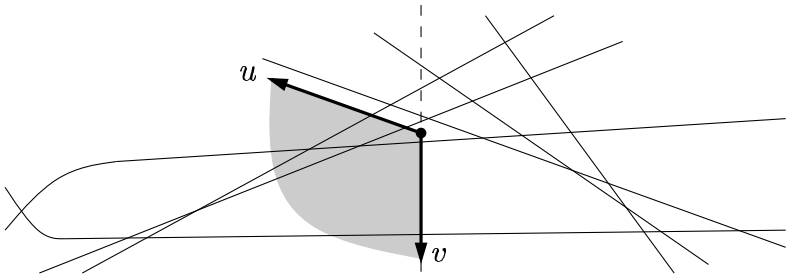

Figure 3: Region R

When p crosses a line of the arrangement, we may obtain a new top direction u'. Let W denote the wedge with apex p and directions u and u'. Applying the wedge lemma to W, we see that no cell of depth greater than i lies in W.

If this new wedge W contains the vertically downward direction v, then we take the new region R from v to u', which is contained in W. Otherwise, we take the new R to be the union of R with W. In either case, R does not intersect a cell of depth greater than i. Finally, if W contains the upward direction $-v$, then the new R contains one of the halfplanes defined by ℓ and we may stop the algorithm.

Since the upward direction is the top direction for the uppermost cell, the algorithm must terminate. ∎

As an aside, one can use a similar argument along a curved path to show that the maximum depth cells are connected.

Corollary 8 *In an arrangement of lines in the plane, the closure of the cells of depth at least i is simply connected.*

Proof: Consider a connected component of the union of the closures of cells of depth $\geq i$, and draw a path in the neighboring cells (which have depths $i-1$ and $i-2$). Applying the wedge lemma as one traverses the path will show that no cell of depth $\geq i$ lies outside the path, so there can be only one component. Note that this component must be simply connected, since every point has a witness ray for depth along which the depth decreases monotonically. ∎

3.2 Computing top directions

In this section we describe a data structure that can determine the top directions for a sequence of adjacent cells in an arrangement of n lines using logarithmic time per cell, after $O(n \log n)$ preprocessing. Preprocessing takes linear time if the lines of the arrangement are sorted by slope.

Let us continue to assume that no line is vertical and let $l_1, l_2, \ldots, l_n$ be the lines ordered by increasing slope. We can identify a cell C in the arrangement with its *bit string* $b(C) = b_1 \ldots b_n$, where bit $b_i = 1$ if line l_i is above the cell C, and $b_i = 0$ otherwise.

Notice that the number of 1 bits in $b(C)$ is exactly the number of lines crossed by a ray ρ from C in the downward direction. Consider rotating the ray ρ from C counter-clockwise. The set of lines crossed by ρ does not change until ray ρ reaches the direction of the line l_1—then bit b_1 is complemented, since ρ will begin to intersect or cease to intersect l_1.

We therefore consider an *extended bit string* $B(C) = b(C)\overline{b(C)}b(C)$, which is the bit string for C, followed by its complement, and the bit string again. The extended string $B(C)$ has $2n+1$ subsequences of length n; we drop the last, since it equals the first. The counts of the number of 1 bits in these $2n$ subsequences give the number of lines intersected by a ray from C to the unbounded cells of the arrangement in the corresponding $2n$ directions. The minimum of these counts is the value depth(C).

With a relatively simple tournament we can maintain the minimum of the counts and information about directions in which the minimum occurs. We use a static, balanced, binary tree that stores in the leaves the sequence of $2n$ counts. The leftmost leaf stores the count for the upward direction. Each internal node stores three integers: the size of its subtree, the minimum count of the leaves in its subtree, and a correction value.

The correction value is a positive or negative integer that should be added to the counts of all leaves in the subtree. It is processed as follows: before the count of a node is inspected, the

correction value is added to the count and to the correction values of the two children nodes, then set to zero. Since tree operations will process nodes from root to leaf, the value of inspected nodes will always be properly corrected.

The tree supports two operations: a query and an update. The query asks for the leaf with minimum count; in case of equal counts we want both the leftmost leaf and the rightmost leaf with these counts—these give the top directions for the cell C. Since each internal node stores the minimum count in its subtree, such a query is easy to perform in $O(\log n)$ time by following two paths in the tree.

The update operation corresponds to moving from a cell C to a cell C' by crossing some line l_i. This means that the bit string of $b(C')$ differs from $b(C)$ in the i-th bit. In the extended string $B(C')$, three bits change to their complements. Since the $2n$ counts for a cell are obtained by adding n consecutive bits, every count changes—if b_i changes from 0 to 1, then the first i counts increase by one, the next n counts decrease by one, and the final $n - i$ counts increase by one. Thus, we should not update the counts in the leaves explicitly, since this would take linear time; instead we update correction values.

We follow the two paths in the tree to the i-th leaf and the $(i+n)$-th leaf using the size-of-subtree integers stored at the internal nodes. The paths partition the tree into three parts. For all highest nodes left of the search path to the i-th leaf we increment the correction value (or decrement, if b_i changes from 1 to 0). This is done too for the highest nodes right of the search path to the $(i + n)$-th leaf. For the highest nodes between the search paths we decrement (or increment) the correction value. Since there can be at most $O(\log n)$ highest nodes left (or right) of any path in the tree, only $O(\log n)$ correction values are updated.

Because the structure of the tree is static, we implemented it by indexing into a fixed array, and subtree sizes were calculated rather than stored.

Lemma 9 *Using the data structure described above, one can determine the top directions for a sequence of adjacent cells in an arrangement of n lines using logarithmic time per cell, after $O(n \log n)$ preprocessing.*

3.3 Binary search for a maximum depth cell

It is probably no surprise that we use the sidedness test in a binary search on x-coordinates of vertices of the arrangement $\mathcal{A}(L)$. A Java prototype can be seen at `http://www.inf.ethz.ch/~speckman/demos/maxdepth`.

Standard results on slope selection [2, 5, 14, 16] allow us to consider the portion of the arrangement $\mathcal{A}(L)$ that lies between two vertical lines, and to generate the vertex of median x coordinate in $O(n \log n)$ time. We based our implementation on a randomized algorithm of Dillencourt, Mount, and Netanyahu [7].

At a vertical test line ℓ through this median vertex, we sort the intersections with the lines of L and use the tournament described in Subsection 3.2 to compute the depth of each point on the test line ℓ and the top directions in $O(n \log n)$ time. Lemma 7 then allows us to discard one side of the line ℓ, and to continue the search on the other side. The search terminates when there are no intersection points remaining, which occurs after at most $\log(n^2) = 2 \log n$ steps. Thus, we claim the following result.

Theorem 10 *A cell of maximum undirected depth in an arrangement of n lines can be computed in $O(n \log^2 n)$ time and $O(n)$ space.*

4 The structure of depth

Although our binary search identifies a deepest cell, we know from Lemma 1 that the maximum depth in an arrangement will always occur at a vertex. In statistical analysis, we may also wish to know the set of all lines with maximum regression depth, which corresponds to the set of all points at maximum depth. In this section, we characterize the set of points at maximum depth in non-degenerate arrangements in the plane. We also establish relationships with k-sets in all dimensions and show how to efficiently approximate a maximum depth point in degenerate arrangements.

4.1 Finding a deepest vertex in a non-degenerate arrangement

Figure 1 showed an example in which edges and isolated vertices attain the maximum depth, but no cell does. Once we have found a point in a cell of maximum depth, we still must determine whether there is a vertex with greater depth. For arrangements of lines in general position, this is not difficult to do. When the maximum depth of a cell is i, then the maximum depth of a vertex is i, $i+1$, or $i+2$, as illustrated in Figure 4. These cases can be detected by postprocessing after computing a maximum depth cell.

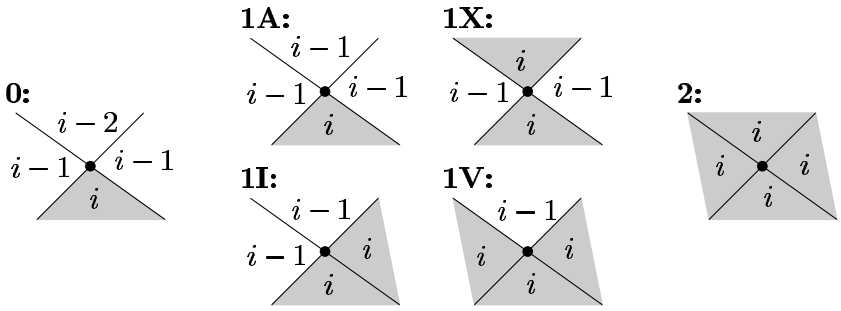

Figure 4: Cases for maximum vertex depth

When the maximum depth vertex v has depth $i+2$ in a non-degenerate arrangement, then the two lines crossing at v form four quadrants containing incident cells at depth i. Lemma 2 says that the directions for these cells are contained in the respective quadrants. During the binary search, test lines to the right of the vertex will eliminate their right side and those to the left will eliminate their left side. Thus, there is at most one such vertex and the binary search will find it.

The maximum depth vertex could instead have depth i—equal to the depth of the maximum depth cell. In this case, every vertex incident on a cell of depth i must also be incident on two cells of depth $i-1$ and one of depth $i-2$, otherwise the vertex depth would be greater than i. This, together with the fact that cells are convex and the maximum depth is connected, implies that there can be only one cell that attains the maximum, which will be found by our binary search.

Finally if the maximum depth vertex has depth $i+1$ then every cell of depth i has to have at least one incident vertex of depth $i+1$. Since our binary search finds a cell of depth i a traversal of its boundary will yield a vertex of depth $i+1$.

Theorem 11 *After computing a deepest cell, one can compute a deepest vertex in $O(n \log n)$ additional time.*

Proof: Once we have computed some cell of maximum depth i, we must determine whether a maximum depth vertex has depth i, $i+1$, or $i+2$. This is most easily done by constructing the cell as the intersection of the n halfplanes that are defined by lines of the arrangement and that contain the cell. Intersection is equivalent to convex hull computation, and takes $O(n \log n)$ time. Then we can use the tournament to check the depth of all vertices, also in $O(n \log n)$ time. By the above discussion, we either find that all vertices are of depth i, or there is a unique vertex of depth $i+2$, or some vertex is of depth $i+1$. ∎

4.2 Deepest points in non-degenerate arrangements

It is natural to ask for the set of all points with maximum undirected depth, which corresponds to the set of all lines that have maximum regression depth. This appears to be a more difficult question than asking for a single vertex.

We can expand on the discussion of the previous subsection to characterize the maximum depth points in non-degenerate arrangements:

Lemma 12 *If the maximum cell depth is i, then the maximum depth points form either*

1. *a single point of depth $i+2$,*
2. *a convex polygon whose vertices, edges, and interior all have depth i, or*
3. *a single chain of $O(n)$ segments and some isolated points of depth $i+1$.*

Proof: The first and second cases are discussed in the previous subsection; we establish the structure of the third by considering the configurations of Figure 4 that give vertices and edges of depth $i+1$.

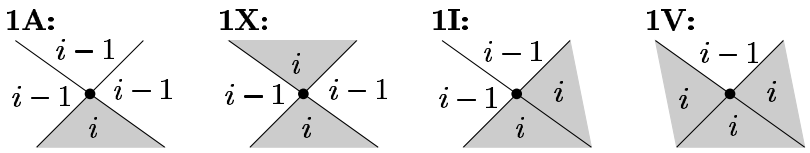

Figure 5: Wedge lemma applied

If we consider the witness directions for cells of depth $i-1$ in these cases, and apply the wedge lemma, we can make the following observations:

In 1I there is a wedge defined by directions for the two cells of depth $i-1$ that includes a ray on the line separating these two cells.

In 1A, there are two such wedges. The wedge lemma implies that cells in these wedges are of depth at most $i-1$. This immediately implies that all edges in the wedge have depth at most i. In fact, vertices in the wedge also have depth at most i, since the only way for a vertex to have depth $i+1$ would be to have four incident cells of depth $i-1$, but then $i-1$ would be the maximum depth cell in the arrangement.

In 1X there is a wedge that contains one of the two incident cells of depth i. We can extend the wedge lemma to observe that cells and edges in this wedge have depth at most i. (The argument bounding the depth of edges is that there are at most $2i$ lines that can be intersected by translates of the two rays of the wedge and there is a bonus of $+1$ for starting the rays on

the edge. Therefore, one of the rays intersects at most $i + 1/2$ lines, showing that the depth of the edge is at most i.)

It is clear that configurations 1A and 1X give isolated vertices of depth $i + 1$, that 1I gives the end of a chain, and that 1V gives the middle of a chain. We need to show that there is at most one chain.

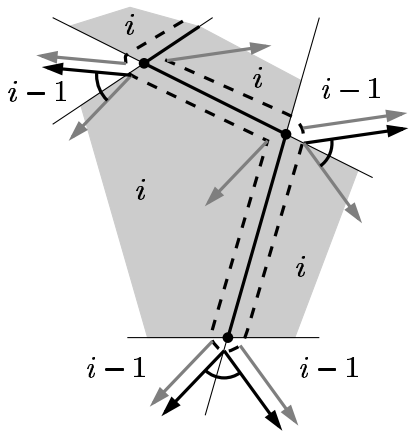

Figure 6: A chain of maximum depth

Consider walking on a chain starting from a 1L configuration (see Fig. 6). For each segment on the chain there is a wedge of witness rays that certify—by extending the wedge lemma as described above—that no edge of depth $i+1$ can be found earlier on the same segment. In the 1L case this wedge is formed by the two witness rays mentioned above, in the 1V case the wedge is formed by a witness ray for the cell of depth $i - 1$ and a translated copy of a witness ray from a previous segment (see Lemma 3). This implies that at most one segment of every line of the arrangement can be part of a chain.

Now let us construct a path surrounding the chain by infinitesimally translating a copy of each segment into its adjacent cells end connecting the endpoints (see Fig. 6). Every point on this cycle has at least one witness ray of depth at most i and the union of these rays cover the whole plane except the original chain. This certifies—by again extending the wedge lemma—that no edge of depth $i + 1$ exists that is not part of the original chain. ∎

Unfortunately, there are close connections between points with given undirected depth and k-sets that imply superlinear bounds on the number of isolated points if the maximum depth points consist of a single chain and some isolated points (the third case of Lemma 12).

4.3 Connections with k-sets

In this subsection we observe the connections between the complexity of points with given undirected depth and the concept of k-sets in a configuration of points. There has been considerable attention in computational geometry devoted to k-sets, and the dual concept of k-levels in an arrangement of lines or hyperplanes; see, e.g., [4, 6, 8, 19, 26].

The *k-level* of an arrangement $\mathcal{A}$ for a particular direction θ consists of all points p such that a ray from p in direction θ intersects exactly k hyperplanes. (Usually, hyperplanes containing p are not counted.) In the dual, the k intersected hyperplanes become a *k-set*: k points that can be separated from the configuration by an open halfspace bounded by a hyperplane, namely p^D. Note that point p has undirected depth at most k (assuming that p does not lie on any hyperplane) and that the hyperplane p^D has regression depth at most k as shown by rotation about any line outside the convex hull of the dual points. The combinatorial complexity of k-levels and algorithms to compute them have been intensively studied, although many open problems remain.

In a similar manner, we define the *k-envelope* in an arrangement $\mathcal{A}$ to be the union of all points with undirected depth k. Examples can be seen back in Figure 1. There have been some results on 1-envelopes of lines [11, 15], but we know of no deeper results.

We show that the worst-case combinatorial complexity of k-envelopes is asymptotically the same as the worst-case complexity of a k-level in any fixed dimension. The exact asymptotic worst-case complexity of a k-level is still unknown [6, 8]. In the plane, it known to be between $\Omega(n \log n)$ and $O(n^{4/3})$.

We begin with the lower bounds that show that the complexity of a k-envelope is at least as great as that of a k-level.

Lemma 13 *The worst-case complexity of the k-envelope of an arrangement of n hyperplanes is at least as large as the worst-case complexity of a k-level in an arrangement of $n - dk$ hyperplanes, for $k < n/d$.*

Proof: Consider the k-level in an arrangement of $n - kd > 0$ hyperplanes, none of which are parallel to the x_d axis. There is a unique unbounded cell in this arrangement that contains the vertically-downward direction, θ. In this cell we can construct a simplex Δ with one horizontal face such that all rays through the horizontal face from the opposite vertex remain inside the cell. Scale and translate Δ until Δ contains the full complexity of the k-level. Then add to the arrangement k perturbed copies of the hyperplanes through each of the d non-horizontal faces of Δ.

For points on the k-level, rays in the downward direction intersect k old hyperplanes and none of the new ones. Rays in directions outside the cell of the downward direction intersect at least k of the new hyperplanes. Thus, the k-level appears on the k-envelope. ∎

The construction above says nothing about the complexity of the points with maximum depth of $k \approx n/d$. With another construction, illustrated in Figure 7, we can show that the complexity of the points with maximum depth in the plane is lower bounded by the complexity of a median level.

Lemma 14 *The worst-case complexity of the set of points with maximum undirected depth in an arrangement of n lines is at least as large as the worst-case complexity of the median level in an arrangement of $n/3$ lines.*

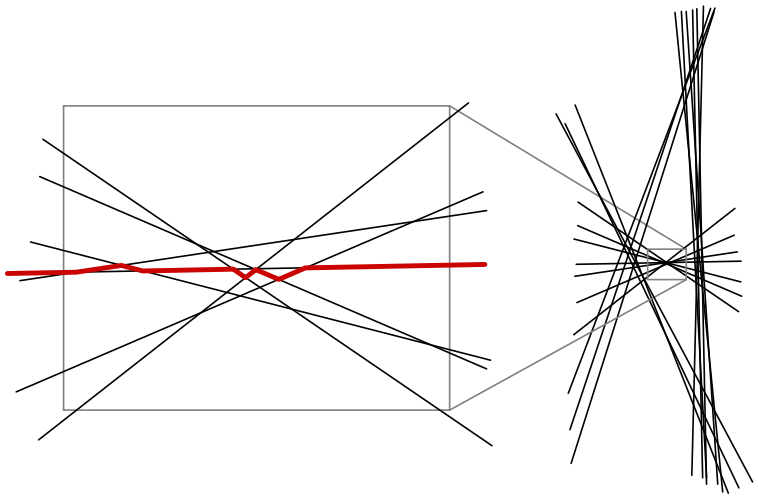

Figure 7: Median level to maximum depth

Proof: Consider any arrangement with $2m$ lines, none of which is parallel to the vertical y axis, and enclose it in a triangle with a vertical longest side, and two other nearly-vertical sides. Add $2m$ lines through the longest side and m through each of the others, then perturb the new lines to be in general position.

Unbounded cells in the original arrangement now have undirected depth at most $2m$ by crossing only new lines. Bounded cells in the original arrangement also have undirected depth at most $2m$ by crossing m old lines and m new with a near-vertical ray. The former median level has undirected depth of exactly $2m$, and thus contributes points of maximum depth. ∎

The proofs of complexity for k-levels can be adapted to prove upper bounds for k-envelopes. For example, we can prove the following in the plane.

Lemma 15 *In the plane, the worst-case complexity of the k-envelope is at most $O(n^{4/3})$.*

The idea here is to adapt Dey's proof [6] for the complexity of a k-level.

4.4 Output-sensitive construction for maximum depth in non-degenerate planar arrangements

The Overmars/van Leeuwen [18] algorithm for dynamic convex hulls, when applied to the duals of the lines, allows us to maintain a description of the current cell as we walk from cell to cell in the arrangement. With the characterization of the points of maximum depth from Subsection 4.2, this allows us to compute a description of the maximum depth points in an output-sensitive manner.

Theorem 16 *The set of all points at maximum depth in an arrangement of lines in general position can be computed at the cost of $O(\log^2 n)$ per feature.*

Proof: (Sketch) The key observation is that there is only one candidate for the next isolated point in the cell contained in the wedge of a 1X configuration—namely, the point with tangent parallel to the tangent of the wedge. Thus, isolated points occur in strings of 1X configurations that end with a 1A configuration, and we can use a binary search in the Overmars/van Leeuwen data structure [18] to find the next candidate and enter the next cell. ∎

4.5 Depth of vertices in degenerate arrangements

Efficiently finding a deepest vertex in a degenerate arrangement of lines appears to be difficult. However, we can efficiently find a vertex whose depth is within a factor of $(1 - o(1))$ from the maximum depth vertex.

Lemma 17 *A point whose depth is at least $(1 - \frac{\log(\log n)}{\log n})$ times the maximum can be found in $O(n \log n)$ time.*

Proof: First, compute the cell of maximum depth in the arrangement. Then, using an algorithm of Guibas et al. [12], find all vertices V that are contained in at least $n\frac{3\log(\log n)}{\log n}$ lines in $O(n \log n)$ time. There are at most $O(\frac{\log n}{\log(\log n)})$ of these vertices, and their depth can be tested in $O(n)$ time each once the lines are sorted by slope.

Either a vertex of V has maximum depth, or, by Lemma 1, a point in the cell of maximum depth is less than $n\frac{3\log(\log n)}{\log n}$ from the true maximum value. Since Amenta et al. proved in [1] that the maximum value is at least $\lceil n/3 \rceil$ we therefore have an approximation factor of at least $(1 - \frac{\log(\log n)}{\log n})$. ∎

One heuristic that involves less programming is to symbolically perturb the lines of the arrangement to simulate general position, and compute the cell of maximum depth. In the original arrangement this cell may correspond to a vertex, in which case we evaluate the depth of this vertex, or to a cell, in which case we construct the cell and evaluate the depth of all of its vertices. From the wedge lemma it can be seen that the actual maximum depth will be at most double the computed depth.

5 Computing Depth in Higher Dimensions

For three and higher dimensions, Corollary 4 says that we can compute max depth in $O(n^d)$ time and space by evaluating depth at all cells and vertices of an arrangement. It is challenging to develop more efficient algorithms.

5.1 The wedge lemma cannot extend to $\Re^3$

The solution for the planar case was based on the wedge lemma, which allowed us to argue that certain regions of the plane could not contain a cell of maximum depth. When thinking about the extension to 3-dimensions, one would first try to generalize the wedge lemma: that for a point p whose depth i is witnessed by three vectors $\vec{u}$, $\vec{v}$ and $\vec{w}$, the cone defined by $\vec{u}$, $\vec{v}$ and $\vec{w}$ does not contain a cell of depth greater than i. The following construction shows that this is not true.

Let point p be the origin of the coordinate system. We construct an arrangement of fifteen planes such that the positive x-axis, the positive y-axis, and the positive z-axis each witness that $\text{depth}(p) = 2$, the point $q = (2, 2, 2)$ will have $\text{depth}(q) = 3$.

There are six planes that intersect the positive octant: Planes $x = 4$, $y = 4$, and $z = 4$ are parallel to the coordinate planes. Planes $5 = -x - y + 5z$, $5 = -x + 5y - z$, and $5 = 5x - y - z$ pass

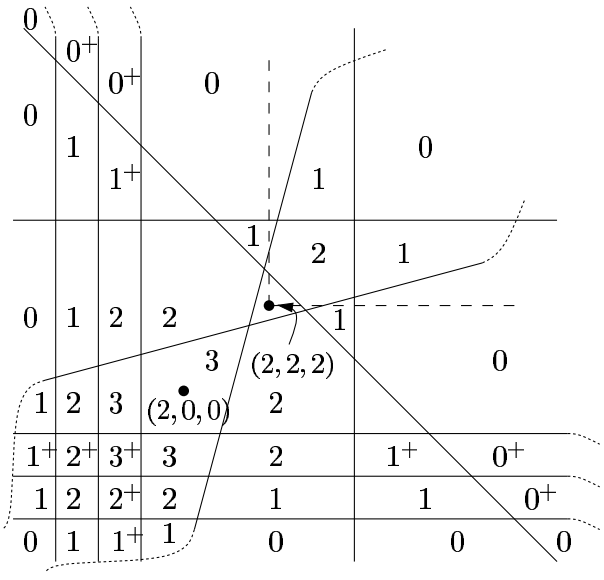

Figure 8: Cross-section of nine planes with the plane $x = 2$, with depth values for directions in this plane. Plusses denote a possible increment by one due to perturbation.

through a common point $(5/3, 5/3, 5/3)$, and each intersect one of the positive coordinate axes. Note that the first intersects the z axis at $(0, 0, 1)$ and the x and y axes at $(-5, 0, 0)$ and $(0, -5, 0)$. Note that if the coordinate frame is translated from the origin to $q = (2, 2, 2)$, then each positive axis intersects *three* of these six planes, which already shows that the argument used to prove the 2-dimensional wedge lemma does not hold in the 3-dimensional case.

The remaining nine planes are chosen to make sure that only directions in or near the positive octant can give depth counts below three for all cells in the positive octant. They are perturbed versions of $x = -1$, $x = -2$, $x = -3$ and similarly, $y, z = -1, -2, -3$. The perturbations are such that none of the planes intersect the positive octant. The common intersection of the half-spaces bounded by these planes and containing the origin can be seen as the perturbed positive octant. These make sure that for any point in the positive octant, and any direction outside the positive octant by a small angle, the depth count in that direction will be at least three.

Let us first consider the number of planes intersected by rays from q inside the positive quadrant of the plane $x = 2$ (which itself is not one of the six planes). By constructing a figure of this cross-section, one can easily verify that all of these directions give rays intersecting three or four planes, see Figure 8. The perturbation of the planes does not influence the depth of the cell containing q.

Finally, when we consider directions from q where x, y, z-contributions are all strictly positive, we simply observe that any such direction intersects each one of $x = 4$, $y = 4$, and $z = 4$. Thus, the depth$(q) = 3$.

This example also shows that the closures of cells of a particular depth need not be connected. The proof of the wedge lemma does imply that the positive quadrants of the three coordinate planes do not intersect cells of depth greater than two—if we translate a pair of positive coordinate axes within the quadrant that they define, we do not gain new intersections. Thus, Corollary 8 cannot be extended beyond the plane.

5.2 Computing depth in higher dimensions with reduced space

We close this paper by showing how to reduce the space requirement by a linear factor using hyperplane cuttings.

Theorem 18 *For $d \geq 3$, one can compute the depth of a set of hyperplanes in $\Re^d$ in time $O(n^d)$, using $O(n^{d-1})$ space.*

Proof: Let H be the original set of n hyperplanes. For parameter $r = n^{1/d}$, a $(1/r)$-cutting of the hyperplanes H results in a set of $O(r^d) = O(n)$ simplices in $\Re^d$, each intersecting at most n/r hyperplanes. This requires $O(nr^{d-1})$ time, using the result of Chazelle [3].

We consider a subproblem for each simplex τ and its intersecting hyperplanes H'. If we considered only the hyperplanes in H', then by Corollary 4 we could solve the subproblem for τ by building the arrangement $\mathcal{A}(H')$ in $\Re^d$. This takes $O(|H'|^d)$ time and space, which is $O(n^{d-1})$ since $|H'| \leq n/r$. We can modify this algorithm to capture the remaining hyperplanes by looking at the unbounded cells of $\mathcal{A}(H')$ on the sphere of directions.

Each hyperplane in H induces a great circle on the sphere of directions. Their arrangement on the sphere, which we call $\mathcal{S}(H)$, has $O(n^{d-1})$ cells. Choosing one direction out of each cell of $\mathcal{S}(H)$ gives a set $\{\rho_1, \rho_2, \ldots, \rho_k\}$ of $k = O(n^{d-1})$ directions sufficient to witness the maximum depth from any point $p \in \Re^d$.

We return to the subproblem for simplex τ and its hyperplanes $H' \subset H$. For each direction ρ_i, define a weight w_i that equals the number of planes of $H \setminus H'$ intersected by a ray from τ

in direction ρ_i. It does not matter which point of τ is the ray origin for this definition, since planes of $H \setminus H'$ do not intersect τ.

We can compute the arrangement $\mathcal{S}(H')$, and a direction with lowest weight for each cell of $\mathcal{S}(H')$ in two steps. First, compute weights for all directions in $\mathcal{S}(H)$ with respect to hyperplanes of $H \setminus H'$ by choosing a cell with its direction ρ_i, counting the number of hyperplanes intersected by a ray in direction ρ_i, then traversing the remaining cells of $\mathcal{S}(H)$, adding or subtracting unity each time we cross the great circle for a hyperplane of $H \setminus H'$. Second, delete hyperplanes of $H \setminus H'$ from $\mathcal{S}(H)$ to form arrangement $\mathcal{S}(H')$. Whenever two cells merge, keep a direction with lowest weight. The time for each subproblem is proportional to $|\mathcal{S}(H)| = O(n^{d-1})$, for a total of $O(n^d)$ time overall.

Now, form the hyperplane arrangement $\mathcal{A}(H')$, and initially label all bounded cells with depth ∞. Each unbounded cell of $\mathcal{A}(H')$ corresponds to a cell of the spherical arrangement $\mathcal{S}(H')$; label it with the corresponding weight and direction. Conceptually, what we have done is translate hyperplanes of $H \setminus H'$ away from τ until their intersection contains all vertices of $\mathcal{A}(H')$, and assigned the depths for directions that do not cross hyperplanes of H'. (This translation does not change the depth of any ray from τ.) The minimum assigned depth is correct, so we can perform the loop as before: for $i = 1, 2, \ldots, n$, all cells with label $i-1$ cause their adjacent, higher-labeled cells to be relabeled i. This takes $O(n^{d-1})$ time and space for each subproblem, with a total time of $O(n^d)$. ∎

Acknowledgments

J. Mitchell and M. Sharir thank E. Arkin and S. Har-Peled for several helpful discussions and suggestions.

References

[1] N. Amenta, M. Bern, D. Eppstein, and S.-H. Teng. Regression depth and center points. *Discrete Comput. Geom.*, 23:305-323, 2000.

[2] H. Brönnimann and B. Chazelle. Optimal slope selection via cuttings. *Comput. Geom. Theory Appl.*, 10(1):23–29, 1998.

[3] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[4] B. Chazelle and F. P. Preparata. Halfspace range search: An algorithmic application of k-sets. *Discrete Comput. Geom.*, 1:83–93, 1986.

[5] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM J. Comput.*, 18(4):792–810, 1989.

[6] T. K. Dey. Improved bounds on planar k-sets and related problems. *Discrete Comput. Geom.*, 19:373–382, 1998.

[7] M. B. Dillencourt, D. M. Mount, and N. S. Netanyahu. A randomized algorithm for slope selection. *Internat. J. Comput. Geom. Appl.*, 2:1–27, 1992.

[8] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[9] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.

[10] H. Edelsbrunner, R. Seidel, and M. Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993.

[11] D. Eu, E. Guévremont, and G. T. Toussaint. On envelopes of arrangements of lines. *J. Algorithms*, 21:111–148, 1996.

[12] L. J. Guibas, M. H. Overmars, and J.-M. Robert. The exact fitting problem for points. *Comput. Geom. Theory Appl.*, 6:215–230, 1996.

[13] M. Hubert and P. J. Rousseeuw. The catline for deep regression. *J. Multivar. Analysis*, 66:270-296, 1998.

[14] M. J. Katz and M. Sharir. Optimal slope selection via expanders. *Inform. Process. Lett.*, 47:115–122, 1993.

[15] M. Keil. A simple algorithm for determining the envelope of a set of lines. *Inform. Process. Lett.*, 39:121–124, 1991.

[16] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

[17] Stefan Langerman and William Steiger. An optimal algorithm for hyperplane depth in the plane. In *Proc. 11th ACM-SIAM SODA*, pages 54–59, San Francisco, CA, 2000.

[18] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[19] G. W. Peck. On k-sets in the plane. *Discrete Math.*, 56:73–74, 1985.

[20] P. J. Rousseeuw and M. Hubert. Regression depth. *J. Amer. Statist. Assoc.*, 94:388-402, 1999.

[21] P. J. Rousseeuw and M. Hubert. Depth in an arrangement of hyperplanes. *Discrete Comput. Geom.*, 22:167-176, 1999.

[22] P. J. Rousseeuw and I. Ruts. Constructing the bivariate Tukey median. *Statistica Sinica*, 8:827–839, 1998.

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

[24] P. J. Rousseeuw, S. Van Aelst and M. Hubert. Rejoinder to discussion of "Regression Depth". *J. Amer. Statist. Assoc.*, 94:419-433.

[25] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[26] M. Sharir. k-sets and random hulls. *Combinatorica*, 13:483–495, 1993.

[27] A. Struyf and P. J. Rousseeuw. Halfspace depth and regression depth characterize the empirical distribution. *J. Multivar. Analysis*, 69:135-153, 1999.

[28] A. Struyf and P. J. Rousseeuw. High-dimensional computation of the deepest location. *Computational Statistics and Data Analysis*, 34:415-426, 2000.

[29] S. Van Aelst and P. J. Rousseeuw. Robustness of Deepest Regression. *J. Multivar. Analysis*, 73:82-106, 2000.

[30] S. Van Aelst, P. J. Rousseeuw, M. Hubert, and A. Struyf. The deepest regression method. Manuscript, Dept. of Mathematics and Computer Science, University of Antwerp, Belgium, 2000.