On Algorithms for Simplicial Depth

Andrew Y. Cheng
Airport and Aircraft Safety R&D Division
FAA William J. Hughes Technical Center
Atlantic City, New Jersey 08405
Ming Ouyang
Environmental and Occupational Health Sciences Institute
UMDNJ – Robert Wood Johnson Medical School and
Rutgers, The State University of New Jersey
Piscataway, New Jersey 08854

Abstract
Simplicial depth measures how deep a point is among a set of points. Efficient algorithms to compute it are important to its usefulness in applications, such as multivariate analysis in environmental health and bioinformatics. When the points are in E^d, a straightforward method takes $O(n^{d+1})$ time. We discuss an algorithm that takes $O(n^2)$ time when the points are in E^3, and we generalize it to E^4 with a time complexity of $O(n^4)$.

1 Introduction

In one-dimensional space, let p be a real number, and let P be a finite set of real numbers. The *depth* of p with respect to P is the minimum of the following two numbers: the number of elements in P that are less than p and the number of elements in P that are greater than p. There are different definitions of data depth in multi-dimensional space, including halfspace depth [11], simplicial depth [7], peel depth [5], and regression depth [9]. Data depth has applications in multivariate analysis in statistics [2, 5, 7, 9]. In the present work, we focus on algorithms that compute simplicial depths.

In d-dimensional space, a straightforward method to compute simplicial depth takes $O(n^{d+1})$ time. However, in one-dimensional space, it can be computed in $O(n)$ time; in two-dimensional space, it can be computed in $O(n \lg n)$ time [5, 6, 10]. There have been two attempts at algorithms for simplicial depth in three-dimensional space. First, Rousseeuw and Ruts briefly described the outline of an algorithm for three and higher dimensional space (page 523, [10]); however, we find it difficult to furnish the details. Second, Gil, Steiger, and Wigderson described an algorithm for three-dimensional space, with a time complexity of $O(n^2)$ [5]; however, we find a flaw in their algorithm.

Section 2 defines simplicial depth, discusses some interesting properties, and reviews known algorithms. Section 3 discusses algorithms for three-dimensional space; in particular, we point out the difficulty we have with the algorithm by Rousseeuw and Ruts, and we discuss a remedial modification of the algorithm by Gil, Steiger, and Wigderson. Section 4 discusses the algorithm for four-dimensional space, with a time complexity of $O(n^4)$. For spaces higher than four-dimensional, there are no known algorithms faster than the straightforward method.

2 Simplicial Depth

A *simplex* in d-dimensional Euclidean space E^d is the set of points that are convex combinations of $d+1$ affinely independent points; if the points are $p_1, p_2, \ldots, p_{d+1}$, they bring about the simplex $\{p : p = a_1p_1 + a_2p_2 + \ldots + a_{d+1}p_{d+1},\ a_i \geq 0,\ \Sigma a_i = 1\}$. A simplex is a line segment in E^1, a triangle in E^2, and a tetrahedron in E^3.

It is customary to say that a set of points in E^d are in *general position* if any $d+1$ points in the set are affinely independent. Let P be a set of n points in general position in E^d; take every $d+1$ distinct points from P and they uniquely identify a simplex in E^d; let S_P be the set of all such simplices; then, $|S_P| = \binom{n}{d+1}$. Let $\sigma(p, P)$ denote the *simplicial depth* of a point p with respect to a set P, defined by $\sigma(p, P) = |\{s \in S_P : p \in s\}|$. For example, if p is outside the convex hull of P, then $\sigma(p, P) = 0$, and if p is a vertex of the convex hull of P, then $\sigma(p, P) = \binom{n-1}{d}$. When p is a point in P, the following identity holds: $\sigma(p, P) = \sigma(p, P - \{p\}) + \binom{|P|-1}{d}$. Boros and Füredi [1] have shown that for any set P of n points in E^2, there exists a point p such that $\sigma(p, P) = n^3/27 + O(n^2)$; there is a set Q of n points such that for all points p in the plane, $\sigma(p, Q) < n^3/27 + n^2$. Liu [7] observes that simplicial depth is invariant under nonsingular affine transformations.

A straightforward method for the simplicial depth of one query point in E^d is to generate all the simplices and then to count the number of containments; since there are $O(n^{d+1})$ simplices, it takes $O(n^{d+1})$ time in the real RAM model of computation [8]. However, in E^1, the depth of one query point can be computed in $O(n)$ time by a partition algorithm, and the depths of n query points can be computed in $O(n \lg n)$ by first sorting the points and then finding the depths in linear time. There are two approaches to compute simplicial depth in E^2: counting the triangles that contain the query point [5, 6], and counting the triangles that do not contain the query point [10]. Both approaches start by computing the angular ordering of points in P around the query point p, taking $O(n \lg n)$ time. Then the counting can be done in linear time in both approaches.

Consider the situation that, in E^2, for each point p in P, we need to compute its depth with respect to $P - \{p\}$. If we apply an $O(n \lg n)$ time algorithm to each point, it takes $O(n^2 \lg n)$ time. However, we can use the point-line duality to reduce the overall time to $O(n^2)$ [5, 6]. The dual of the point (x, y) is the line $v = xu + y$, and the dual of the line $y = mx + b$ is the point $(-m, b)$. The duality preserves incidence and the above/below relationships [3]. The arrangement of n

lines in a plane can be constructed in $O(n^2)$ time [4]. After the arrangement of the n dual lines is constructed, we can traverse the dual line of p to obtain the ordering of the intersections with the other $n-1$ dual lines in $O(n)$ time. This ordering corresponds to the angular ordering of the $n-1$ primal points around p. We can then proceed to count the containments or the non-containments in $O(n)$ time. Thus, the depths of n query points in E^2 can be answered in $O(n^2)$ time.

3 Computing Simplicial Depth in E^3

In Section 3.1, we explain the difficulty we have encountered in counting the tetrahedra that do not contain the query point. In Section 3.2, we present an algorithm that counts the tetrahedra that contain the query point.

3.1 Counting Tetrahedra Not Containing p

Rousseeuw and Ruts briefly described the outline of an algorithm that computes the depth of one query point in spaces higher than two-dimensional (page 523, [10]). We find it difficult to furnish the details. The idea is as follows. For each p_i in P,

- let Π_i be the plane that passes through p and is orthogonal to $(p_i - p)$;
- project P to Π_i, and let $\Pi_i(q)$ denote the image of q on Π_i (note that $\Pi_i(p_i) = \Pi_i(p)$).

The argument is that if a triangle $\triangle\Pi_i(p_j)\Pi_i(p_k)\Pi_i(p_l)$ does not contain $\Pi_i(p)$, then the tetrahedron $\triangle p_i p_j p_k p_l$ does not contain p. We can use the algorithm for simplicial depth in E^2 in [10] to count the triangles in Π_i, $i = 1, 2, \ldots, n$, that do not contain $\Pi_i(p)$, and the sum of these counts corresponds to the number of tetrahedra that do not contain p.

We need some terminology to explain the difficulty we have encountered. Assume the tetrahedron $\triangle p_i p_j p_k p_l$ does not contain p; there are four planes in E^3 defined by any three vertices of the tetrahedron; these planes divide the exterior of the tetrahedron into 14 cells:

- four cells adjacent to the faces of the tetrahedron,
- six cells adjacent to the edges of the tetrahedron, and
- four cells adjacent to the vertices of the tetrahedron.

If p is in a cell adjacent to a face of the tetrahedron, for example, the triangle $\triangle p_i p_j p_k$, then p_l is "invisible" to p (blocked by the tetrahedron). When the tetrahedron is projected to the plane Π_l, the triangle $\triangle\Pi_l(p_i)\Pi_l(p_j)\Pi_l(p_k)$ in fact *does* contain p. However, when the tetrahedron is projected to the other three planes, Π_i, Π_j, and Π_k, the corresponding triangles do *not* contain p. Therefore, the tetrahedron is counted three times by the above method.

Similarly, if p is in a cell adjacent to a vertex of the tetrahedron (all four vertices of the tetrahedron are visible to p), the tetrahedron is counted three times.

However, if p is in a cell adjacent to an edge of the tetrahedron (all four vertices of the tetrahedron are visible to p), the tetrahedron will be counted *four* times.

Therefore, some tetrahedra are counted three times, and others are counted four times. We can not find a way to resolve this discrepancy.

3.2 Counting Tetrahedra Containing p

Gil, Steiger, and Wigderson [5] described an algorithm that computes simplicial depth of one query point in E^3; however, there is a flaw. We describe a correction. Let P be a set of n points in general position in E^3, and let p be the query point, which is in general position with P. Lemmas 1 and 2 are from [5].

Lemma 1. *Let p'_i be any point on the ray from p through p_i; the tetrahedron $\triangle p_i p_j p_k p_l$ contains p if and only if the tetrahedron $\triangle p'_i p_j p_k p_l$ contains p.*

Let θ_i be the point where the ray from the origin with the direction $(p_i - p)$ intersects the unit sphere about the origin; let $\hat{\theta}_i$ be the point where the ray from the origin with the direction $(p - p_i)$ intersects the unit sphere about the origin; that is, $\hat{\theta}_i$ is *antipodal* to θ_i. By Lemma 1, the tetrahedron $\triangle p_i p_j p_k p_l$ contains p if and only if the tetrahedron $\triangle\theta_i\theta_j\theta_k\theta_l$ contains the origin. Assuming θ_i, θ_j, and θ_k are not antipodal to one another, the *spherical triangle* $\triangle_s\theta_i\theta_j\theta_k$ is the area on the surface of the unit sphere bounded by the short arcs of the great circles passing any two points in $\{\theta_i, \theta_j, \theta_k\}$.

Lemma 2. *The tetrahedron $\triangle\theta_i\theta_j\theta_k\theta_l$ contains the origin if and only if the spherical triangle $\triangle_s\theta_i\theta_j\theta_k$ contains $\hat{\theta}_l$.*

Let P' be $\{\theta_1, \theta_2, \ldots, \theta_n\}$. By Lemmas 1 and 2, we can count, for $i = 1, \ldots, n$, the number of spherical triangles with vertices in $P' - \{\theta_i\}$ that contain $\hat{\theta}_i$, and $\sigma(p, P)$ is equal to the sum of these counts. Later we will describe a relabeling of the points in P'. With the new labeling in effect, let Π_l be a plane that contains the origin, O, and that separates θ_l from θ_i, θ_j, and θ_k; let Π'_l be a plane that is parallel to Π_l, but is at some distance from the origin. Let $\Pi'_l(q)$ denote the *radial projection* of a point q ($q \neq O$) onto Π'_l: the intersection of Π'_l and the line $\overline{Oq}$. Note that, first, the image of a radially projected spherical triangle is a triangle in the plane; second, the images of a point and its antipodal point coincide, $\Pi'_l(q) = \Pi'_l(\hat{q})$. Trivially,

Lemma 3. *The spherical triangle $\triangle_s\theta_i\theta_j\theta_k$ contains $\hat{\theta}_l$ if and only if, in the plane Π'_l, the triangle $\triangle\Pi'_l(\theta_i)\Pi'_l(\theta_j)\Pi'_l(\theta_k)$ contains $\Pi'_l(\hat{\theta}_l)$.*

By Lemmas 1, 2, and 3, computing the depth of one point in E^3 becomes computing the depths of points in E^2. Let x be a point in general position with $P' \cup \{O\}$; the line $\overline{Ox}$ is the *axis of rotation*. Let Λ be the plane that contains O and is orthogonal to the axis of rotation. Project P' to Λ, and let $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ be the polar angles of the projected images in ascending order, $0 \leq \alpha_i < 2\pi$. Relabel the points in P' so that the indices start at 0 ($P' = \{\theta_0, \theta_1, \ldots, \theta_{n-1}\}$), and θ_i corresponds to α_i, $i = 0, 1, \ldots, n-1$. In what follows, we will assume "wrapping around" of indices (indices modulo n) and "wrapping around" of angles (adding 2π to negative angles, and subtracting 2π from angles that are equal to or greater than 2π).

On the plane Λ, let $\hat{\alpha}_i$ denote the point on the unit circle with the polar angle $\alpha_i + \pi$; that is, $\hat{\alpha}_i$ is *antipodal* to α_i. Suppose $\hat{\alpha}_i$ is angularly between α_j and α_{j+1}; let ϵ_i be the minimum of $(\alpha_{i+1} - \alpha_i)$ and $(\alpha_{j+1} - \hat{\alpha}_i)$. The value ϵ_i is the maximal amount we can rotate the line $\overline{O\alpha_i}$ counterclockwise before crossing another point in $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$. Define the *dividing plane* Π_i to be the plane that contains both

- the axis of rotation, and
- the line on Λ that goes through the origin and has the polar angle $\alpha_i + \epsilon_i/2$.

The choice of $\epsilon_i/2$ is meant to put θ_i and $\hat{\theta}_i$ away from the dividing plane as much as possible. Each dividing plane Π_i, $i = 0, 1, \ldots, n-1$, divides the unit sphere into two hemispheres, and the hemisphere containing $\hat{\theta}_i$ is designated the upper hemisphere.

The algorithm moves from one dividing plane to the next. At each dividing plane Π_i, we need to count the spherical triangular containments of antipodal points in the upper hemisphere. However, once a spherical triangular containment rises in the upper hemisphere, it can remain present for some subsequent upper hemispheres. Thus, to count every spherical triangular containment exactly once, we will count one when it first emerges in the upper hemisphere in entirety. By Lemma 3, we can instead count the triangular containments in the plane Π'_i that are not present in the plane Π'_{i-1}.

For $i = 0, 1, \ldots, n-1$, the algorithm performs the following for Π'_i. Suppose $\theta_i, \theta_{i+1}, \ldots, \theta_j$ are in the upper hemisphere of Π_{i-1}. During the transition from Π_{i-1} to Π_i,

- $\hat{\theta}_i$ moves to the upper hemisphere, and it is called the new antipodal point;
- some points, $\theta_{j+1}, \theta_{j+2}, \ldots, \theta_{j+k}$, move to the upper hemisphere, and they are called the new primary points;
- $\hat{\theta}_{j+k+1}, \hat{\theta}_{j+k+2}, \ldots, \hat{\theta}_{i-1}$ are called the old antipodal points;
- $\theta_{i+1}, \theta_{i+2}, \ldots, \theta_j$ are called the old primary points.

The algorithm then counts, in the plane Π'_i,

1. for each old antipodal point, the number of triangles containing it that have one or more of their vertices among the new primary points, while the rest of the vertices are among the old primary points; and
2. for the new antipodal point, the number of triangles containing it that are formed by every three primary points (old or new).

Lemma 4. *Each tetrahedron containing the origin will be counted twice by the algorithm.*

Proof. Since the point x is in general position with $P' \cup \{O\}$, for each tetrahedron containing the origin, the axis of rotation $\overline{Ox}$ intersects with two of its faces. For each of these two faces, its three vertices can not be separated from the other vertex of the tetrahedron by any of the dividing planes, and therefore, their projected images do not contain the antipodal of the other vertex of the tetrahedron. For each of the other two faces of the tetrahedron, it will be counted exactly once, as soon as

1. the other vertex of the tetrahedron is in the lower hemisphere, and one or more of its three vertices have just moved to the upper hemisphere so that all of them are in the upper hemisphere, or
2. the other vertex of the tetrahedron has just moved to the lower hemisphere, while the three vertices are all in the upper hemisphere. $\diamond$

The above algorithm counts a spherical triangle containing an antipodal point when the configuration makes the transition to the upper hemisphere. The flaw of the algorithm in [5] is in that it treats Π_0 differently: it counts *all* containments in Π_0, not just the *new* ones as if rotating from Π_{n-1} to Π_0. Some of the containments in Π_0 will move out and reappear later, and then they are counted again. Therefore, some tetrahedra are counted three times.

We now go into the details of the operations performed at each dividing plane. The initial configuration of the algorithm is at Π_{n-1}:

0. *Initialization:* Radially project points in P' to Π'_{n-1}, compute the lines that are dual to the projected points, and construct the arrangement of the dual lines. This step takes $O(n^2)$.

For $i = 0, 1, \ldots, n-1$:

1. *Rotate to the dividing plane* Π_i *and the corresponding arrangement of the duals.* We do not need to construct the arrangement of the duals from scratch. The reason is if two points are present in both Π'_{i-1} and Π'_i, then the left/right and the above/below relationships between them are preserved from Π'_{i-1} to Π'_i. Thus the relationships are preserved in the arrangement of the duals. We only need to remove old points that have moved to the lower hemisphere, and to add new points that have moved to the upper hemisphere. The amount of time we need for this step varies from iteration to iteration, but the total time is $O(n^2)$, because each point appears once and disappears once, and it takes $O(n)$ time for each update.

2. *Count the new triangular containments of old antipodal points.* During the execution of Step 1, after we add the dual line of a new primary point, q, to the arrangement, we can count the new triangular containments of old antipodal points as follows. Let p be an old antipodal point. The duals of p and q intersect at some point r; let x and y be the numbers of dual lines

of old primary points that are above and below r, respectively. By the properties of the duality, xy is the number of new triangular containments of p where q is a vertex. Thus, we can traverse the dual line of q in the arrangement, and we keep track of x and y, the numbers of dual lines of old primary points that are above and below where we are, respectively. Whenever we encounter an intersection with the dual line of an old antipodal point, we add xy to our count. After we finish traversing the dual line of q, we treat q as an old primary point as we move on to the remaining new primary points. This step takes $O(n)$ time for each new primary point, and $O(n^2)$ time in total.

3. *Count triangular containments of the new antipodal point.* This can be done in $O(n)$ time by the algorithms mentioned in Section 2. The total time for this step is $O(n^2)$.

After all dividing planes are processed, we divide the count by two. In total, the time to compute the depth of one query point in E^3 is $O(n^2)$.

4 Computing Simplicial Depth in E^4

The algorithm in Section 3.2 can be adapted to find the depth of one query point in E^4. Lemmas 1, 2, and 3 can be generalized to E^4. Two points, x_1 and x_2, in general position with $P' \cup \{O\}$ are needed; the plane defined by O, x_1, and x_2 is the *axis of rotation*. Let Λ be the plane that contains O and is orthogonal to the axis of rotation. The *dividing hyperplane* Π_i is similarly defined: it contains O, x_1, x_2, and the points in Λ with the polar angle $\alpha_i + \epsilon_i/2$. The algorithm then proceeds in the same way except that it counts tetrahedra in hyperplanes, instead of triangles in planes, that contain antipodal points.

Lemma 5. *Each four-dimensional simplex containing the origin will be counted twice by the algorithm.*

Proof. For each simplex containing the origin, the axis of rotation intersects with three of its faces, and thus each of these faces will not be separated from the other vertex of the simplex by any of the dividing hyperplanes. For each of the other two faces of the simplex, it will be counted exactly once. $\diamond$

At each dividing hyperplane, there are $O(n)$ old antipodal points, and one new antipodal point; in both cases, there are $O(n)$ primary points. We use the algorithm of Section 3.2 for each of the antipodal points in a dividing hyperplane, spending $O(n^2)$ time for each of the points. Therefore, we need $O(n^3)$ time for one dividing hyperplane, and $O(n^4)$ time in total. The straightforward method takes $O(n^5)$ time.

The same idea can be used to compute simplicial depth in E^d, $d > 4$, where the problem is similarly turned into computing $O(n^2)$ simplicial depths in E^{d-1}. The resulting algorithm takes $O(n^{2d-4})$ time; however, the straightforward algorithm is at least as good.

5 Summary

A straightforward method to compute the depth of one point in E^d takes $O(n^{d+1})$ time, and it takes $O(n^{d+2})$ time when there are n query points. Previously, there have been efficient algorithms for one and two-dimensional spaces. We discussed algorithms for E^3 and E^4. For spaces higher than four-dimensional, there are no known algorithms faster than the straightforward method. The time complexities are briefly summarized in the following table.

Time Complexity	1 query point	n query points
one-dimensional	$O(n)$	$O(n \lg n)$
two-dimensional	$O(n \lg n)$	$O(n^2)$
three-dimensional	$O(n^2)$	$O(n^3)$
four-dimensional	$O(n^4)$	$O(n^5)$

Acknowledgment

We wish to thank Regina Liu for introducing the subject to us. We wish to thank Vašek Chvátal, William Steiger, and an anonymous referee: their comments on an earlier version helped us improve the presentation. William Steiger's encouragement at the beginning and his guidance throughout this work helped us tremendously. Ming Ouyang is partially supported by the USEPA funded Center for Exposure and Risk Modeling (CR827033).

References

[1] E. Boros and Z. Füredi, "Triangles Covering the Centre of an n-set", *Geometriae Dedicata* **17** (1984), 69-77.

[2] A. Y. Cheng, "Statistical Process Control for Complex Settings and the Establishment of Threshold Systems", Ph.D. dissertation, Rutgers University, January 1999.

[3] H. Edelsbrunner, "Algorithms in Combinatorial Geometry", Springer-Verlag, Berlin, 1987.

[4] H. Edelsbrunner, J. O'Rourke, and R. Seidel, "Constructing Arrangements of Lines and Hyperplanes with Applications", *SIAM J. Comput.* **15** (1986), 341-363.

[5] J. Gil, W. Steiger, and A. Wigderson, "Geometric Medians", *Discrete Mathematics* **108** (1992), 37-51.

[6] S. Khuller and J. S. B. Mitchell, "On a Triangle Counting Problem", *Information Processing Letters* **33** (1989), 319-321.

[7] R. Y. Liu, "On a Notion of Data Depth Based on Random Simplices", *The Annals of Statistics* **18** (1990), 405-414.

[8] F. P. Preparata and M. I. Shamos, "Computational Geometry", Springer-Verlag, New York, 1985.

[9] P. J. Rousseeuw and M. Hubert "Regression Depth", *Journal of the American Statistical Association* **94** (1999), 388-402.

[10] P. J. Rousseeuw and I. Ruts, "Bivariate Location Depth", *Applied Statistics* **45** (1996), 516-526.

[11] J. W. Tukey, "Mathematics and the Picturing of Data", *Proceedings of the International Congress of Mathematicians*, Vancouver, 2, 523-531.