Algorithms for Bivariate Zonoid Depth

Harish Gopala[*] Pat Morin[†]

Abstract

Zonoid depth is a new notion of data depth proposed by Dyckerhoff et al [DKM96]. We give efficient algorithms for solving several zonoid depth problems for 2-dimensional (bivariate) data.

1 Introduction

Data depth measures how deep or central a given point x in $\mathbb{R}^d$ is relative to a given data cloud or a probability distribution in $\mathbb{R}^d$. Some examples of data depth are halfspace, simplicial, convex hull peeling and regression depths [Raf].

Given a set of points $S = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^2$ in general position,

$$CH(S) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \mid 0 \le \lambda_i \le 1, \sum_{i=1}^{n} \lambda_i = 1 \right\}$$

is called the convex hull of points in S. But

$$Z_k(S) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \mid 0 \le \lambda_i \le \frac{1}{k} < 1, \sum_{i=1}^{n} \lambda_i = 1 \right\}$$

is called the *zonoid of depth k* or *k-zonoid*. Since a zonoid is defined by a set of linear constraints, it forms a convex polygon. Furthermore, for $k_1 > k_2$, the k_1-zonoid is a subset of the k_2-zonoid, hence $\{1, \ldots, n\}$-zonoids are nested. For other properties of zonoids, see Dyckerhoff et al [DKM96] and Mosler [Mos02].

Now, the *zonoid depth* of a point p is defined as the maximum k for which p lies inside $Z_k(S)$. Dyckerhoff et al [DKM96] give an algorithm to compute the depth of a point in a data cloud of fixed dimension d by solving a linear program in the variables $\lambda_1, \ldots, \lambda_n$. To obtain an efficient algorithm, they make use of the fact that most of the constraints on the λ_i's are independent of S. However, the running time of their algorithm is unclear.

In this paper, we exhibit a relationship between zonoids and k-sets. Using this relationship, we develop efficient algorithms for computing a k-zonoid, all k-zonoids for $1 \le k \le n$ and for computing the zonoid depth of a point.

The remainder of this paper is organized as follows: In Section 2, we discuss the relationship between zonoid depth and k-sets. In Section 3, we give algorithms for computing a specific zonoid, computing all zonoids, and computing the zonoid depth of a point.

2 Properties of Zonoids

Given a set S of n points in general position and an integer $0 \le k \le n-2$, a set $S' \subseteq S$ is called a *k-set* if S' has k points in it and these can be separated from the remaining $n-k$ points of S with a straight line.

Consider a set S of n points. If we construct all possible 1-sets of S, we obtain the vertices of the $CH(S)$, represented by the thick points in Figure 1. By joining all such 1-set points, we get the zonoid of depth 1 or 1-zonoid or $Z_1(S)$, and $Z_1(S) = CH(S)$.

Now, on the same set S, construct all possible 2-sets. In each 2-set, take the mean (represented by the **X** on the dotted line segment joining 2 points in each 2-set) of the pair of points. By joining all such means from all 2-sets of S, we obtain a 2-zonoid of S or $Z_2(S)$ as in Figure 2. In a similar fashion, zonoids up to depth n can be constructed.

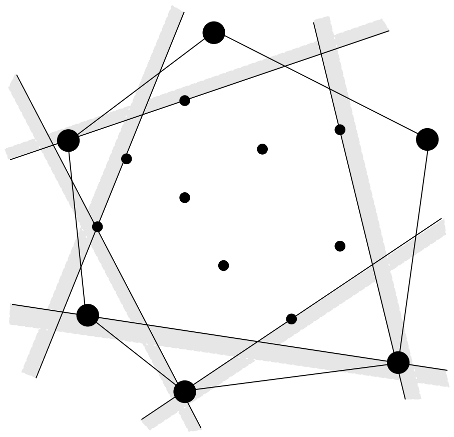

Figure 1: 1-sets on S and the 1-zonoid

Lemma 1 *For any integer $1 \le k \le n$, there is a bijection between the vertices of $Z_k(S)$ and the k-sets of S.*

Proof. We show that the relationship between k-zonoid vertices and k-sets is both one-to-one and onto.

One-to-one: Consider a k-set in a set S of n points. Allot $\lambda_i = \frac{1}{k}$ to each point in this k-set. This is in accordance to the definition of a k-zonoid in Section 1. Then $\sum_{i=1}^{n} \lambda_i p_i$ is the mean of the points in the k-set as well as an extreme point

[*]School of Computer Science, Carleton University, hgopala@scs.carleton.ca
[†]School of Computer Science, Carleton University, morin@scs.carleton.ca

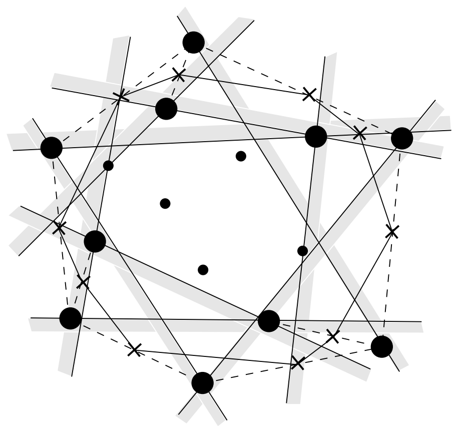

Figure 2: 2-sets on S and the 2-zonoid

of $Z_k(S)$ in the direction perpendicular to the separating line. If we take a different k-set and do as above, we get a different k-zonoid vertex. Also, two k-sets are not made up of the same points. So different k-sets correspond to different k-zonoid vertices.

Onto: A k-zonoid vertex is extreme in some direction and is the mean of the points in the k-set separated by a line perpendicular to that direction. □

3 Algorithms for zonoid depth

We now give algorithms for various zonoid depth problems, based on Lemma 1.

3.1 How to construct a k-zonoid

The top part of Figure 3 represents the primal and the bottom part, the dual. The upper (lower) convex hull of points in the primal corresponds to the upper (lower) envelope of the lines in the dual. In the primal, we construct a k-zonoid, for some k. In the dual, this is the shaded region. The upper and lower boundaries of the shaded region are also convex, because the corresponding boundaries of the k-zonoid in the primal are convex.

Each vertex of the k-level and the $(n-k)$-level in the dual maps to a line that separates a k-set in the primal. We constructed the k-zonoid in the primal by finding all possible k-sets, taking the mean of k points in each k-set and then joining these mean points of all k-sets. In the dual, we first construct the k-level (respectively the $(n-k)$-level) and then, for each vertex, we draw an upwards (respectively downwards) vertical ray through it, and compute the mean of the k lines that intersect this vertical ray. Such mean points are then joined to get the boundaries of the shaded region in Figure 3. It maybe observed here that for each vertex on the upper (respectively lower) boundary of the dual of the k-zonoid, there is a vertex directly below (respectively above) it on the k-level (respectively the $(n-k)$-level).

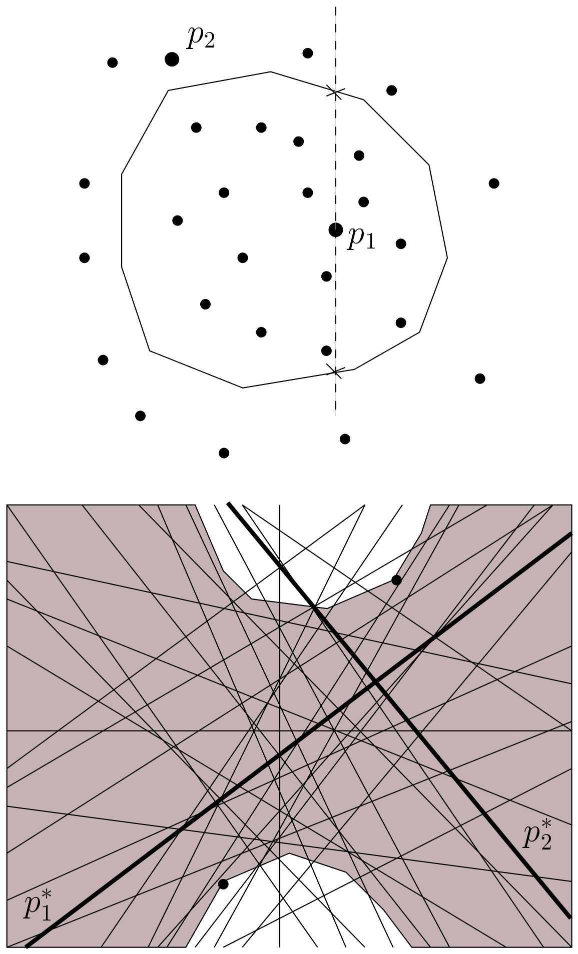

Figure 3: k-zonoid in primal and dual

Using Chan's algorithm [Cha99b], the k-level and the $(n-k)$-level can be constructed in $O(n \log n + nk^{\frac{1}{3}})$ expected time. This algorithm is easily augmented to output the k-zonoid.

Theorem 2 *The k-zonoid can be computed in $O(n \log n + nk^{\frac{1}{3}})$ expected time.*

3.2 Computing all zonoids

The relationship between k-zonoids, k-levels and $(n-k)$-levels allows us to compute all $\{1,\ldots,n\}$-zonoids in $O(n^2)$ time by computing all arrangements of the dual lines [Ede87].

Theorem 3 *$\{1,\ldots,n\}$-zonoids can be computed in $O(n^2)$ time.*

Once we have computed the $\{1,\ldots,n\}$-zonoids, we can preprocess them for point location using Kirkpatrick's algorithm [Kir83].

Theorem 4 *After $O(n^2)$ preprocessing, the zonoid depth of any query point p can be computed in $O(\log n)$ time.*

3.3 Depth computation - decision version

Given a set S of n points in general position and a query point p, we want to find out the largest integer k for which p lies inside $Z_k(S)$. This is the optimization version of the following decision problem: Given a set S of n points in general position, a query point p and an integer $1 \leq k \leq n$, report whether p lies inside or outside $Z_k(S)$. We first concentrate on the decision problem.

Consider again Figure 3. In the primal, if the point p_1 were to be moved upwards along a vertical line passing through p_1, then the line p_1^* also moves upwards in the dual, parallel to itself. When p_1 hits the k-zonoid boundary, p_1^* also hits the boundary of the dual of the k-zonoid. Since this boundary is convex, line p_1^* becomes a tangent to it. This leads us to the following idea: in the primal, draw a vertical line through the point p_1. It intersects the k-zonoid at 2 points (if the point p_1 is outside *and* to the right or to the left of the k-zonoid, then it is trivially outside and neglected). Finding these intersection points is equivalent to finding the points at which the vertical translation of line p_1^* becomes tangent to the boundaries of the dual of the k-zonoid. Once they are found, it can be easily said whether p_1 is inside or outside the k-zonoid by comparing the y-coordinates of the intersection points with that of p_1. Hereafter, we concentrate on finding that vertex on the upper boundary of the dual of the k-zonoid at which p_1^* is tangent. Such a vertex on the lower boundary can be found in a similar manner.

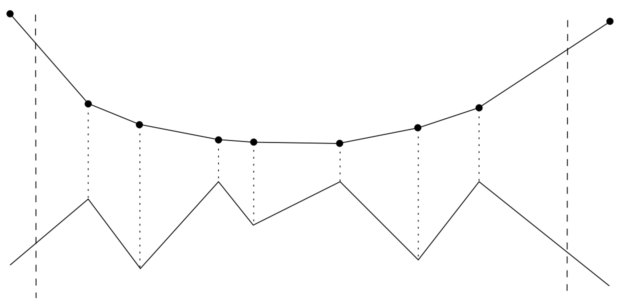

Figure 4: Vertical strip V showing the k-level and corresponding upper boundary of the dual of the k-zonoid

Consider an open vertical strip V in the dual, as in Figure 4, showing the k-level and the corresponding convex upper boundary of the dual of the k-zonoid. To find out whether or not the line p_1^* is tangent to some boundary vertex inside V, we do the following: we count the first k lines from the top intersected by the left vertical line of V, and compute their mean. This actually involves finding the mean of the slopes and intercepts of these lines (which are the x- and y-coordinates of the points corresponding to these lines in the primal). This gives us the line containing the segment of the upper boundary of the dual of the k-zonoid intersected by the left vertical line of V. We do similarly for the right vertical line of V. Now we compare the slopes s_1 and s_2 of the left and right intersected segments respectively with slope s_p of line p_1^*. If $s_1, s_2 \leq s_p$, then p_1^* is a tangent to the upper boundary *to the right* of strip V. Similarly, if $s_1, s_2 \geq s_p$, then p_1^* is a tangent to the upper boundary *to the left* of strip V. But if $s_1 < s_p < s_2$, then p_1^* is a tangent to the boundary inside strip V. The running time of this check is $O(n)$, because we count k lines on the left and right vertical line and compute their means. The slope comparison is, of course, done in constant time. Hence we have the following lemma.

Lemma 5 *If V is an open vertical strip in the dual, we can find out whether or not this strip contains the boundary vertex at which p_1^* is tangent, in $O(n)$ time.*

Lo et al [LMS94] give an algorithm for ham sandwich cuts by looking for a specific point on the median of an arrangement of lines. The main tool required by their algorithm is a method to determine whether the point in question lies to the left, right or in a vertical strip. Combining Lemma 5 with their algorithm gives us the following theorem.

Theorem 6 *Given a set S of n points in the plane in general position, an integer $1 \leq k \leq n$ and a query point p, we can find out in $O(n)$ time whether or not p lies inside or outside the k-zonoid $Z_k(S)$. More generally, we can compute the intersection of $Z_k(S)$ with any line in $O(n)$ time.*

3.4 Depth computation - full version

To compute the depth of a point we make use of a general technique due to Chan [Cha99a] which requires (*a*) a decision algorithm and (*b*) a decomposition into subproblems. The decision algorithm comes from Section 3.3. We now describe the decomposition of our problem into subproblems.

Given a set of points $S = \{p_1, p_2, \ldots, p_n\}$ in general position and a set of weights $w = \{w_1, w_2, \ldots, w_n\}$ where $W = \sum_{i=1}^{n} w_i$, a *weighted* zonoid of depth k is defined as

$$Z_k(S, w) = \left\{ \frac{1}{W} \sum_{i=1}^{n} \lambda_i w_i p_i \mid 0 \leq \lambda_i \leq \frac{1}{k}, \sum_{i=1}^{n} \lambda_i = 1 \right\}$$

We partition our problem into 4 subproblems S_1, S_2, S_3 and S_4 as follows: we first partition the set S of n points into 4 quadrants Q_1, Q_2, Q_3 and Q_4, each containing roughly $\frac{n}{4}$ points, using Megiddo's algorithm [Meg85]. Subproblem S_1 contains 3 consecutive quadrants, say Q_1, Q_2, Q_3, and a single point whose weight is the weighted average of all the points in Q_4. So S_1 has $\frac{3n}{4} + 1$ points. We define the sets S_2, S_3 and S_4 in a similar manner. We define $depth(p, S_i)$ as the zonoid depth of point p in problem S_i. We solve the subproblems recursively. Note that this merging produces a strictly smaller zonoid, i.e. $Z_k(S_i, w) \subseteq Z_k(S)$.

Lemma 7

a. $depth(p, S_i) \leq depth(p, S)$ for each $1 \leq i \leq 4$, *b.* $depth(p, S) = \max\{depth(p, S_i) \mid 1 \leq i \leq 4\}$

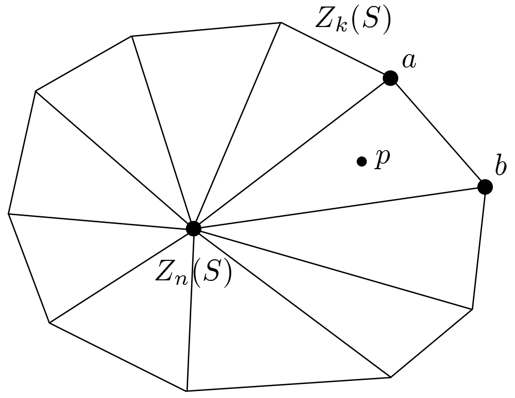

Figure 5: Partitioning the $Z_k(S)$ zonoid into triangles

Proof. Part a follows from the observation that $Z_k(S_i) \subseteq Z_k(S)$. To see why Part b is true, suppose $depth(p, S) = k$ and partition $Z_k(S)$ into triangles by drawing segments joining $Z_n(S)$ to each of the vertices of $Z_k(S)$, as shown in Figure 5. The point p lies in one of these triangles, say with vertices $Z_n(S)$, a and b. The points a and b correspond to two k-sets that have $k - 1$ points in common. Indeed, there are two infinitesimally close lines l_a and l_b such that l_a defines the k-set for a and l_b defines the k-set for b. Since l_a and l_b are infinitesimally close, they intersect at most three of the quadrants $Q_1, \ldots, Q_4$. Wlog suppose they miss Q_4. Then it is not hard to see that $Z_k(S_1)$ has a and b as vertices. Furthermore, $Z_k(S_1)$ contains $Z_n(S)$ and is convex, so it contains p. Therefore $depth(p, S_1) \leq k = depth(p, S)$ as required. □

This satisfies the requirements of Chan's optimization algorithm [Cha99a], and therefore yields the following theorem.

Theorem 8 *Given a set S of n points in the plane in general position and a query point p, we can find the largest k for which p lies inside $Z_k(S)$, in $O(n)$ time.*

Acknowledgment

The authors would like to thank Stefan Langerman for several helpful discussions.

References

[Cha99a] Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete and Computational Geometry*, 22(4):547–567, Dec 1999.

[Cha99b] Timothy M. Chan. Remarks on k-level algorithms in the plane. Manuscript, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, July 7 1999.

[DKM96] Rainer Dyckerhoff, Gleb Koshevoy, and Karl Mosler. Zonoid data depth : Theory and computation. In *COMPSTAT 1996*, pages 235–240, 1996.

[Ede87] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag New York, Inc., 1987.

[Kir83] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12:28–35, 1983.

[LMS94] C.-Y. Lo, J. Matousek, and W. Steiger. Algorithms for ham-sandwich cuts. *Discrete and Computational Geometry*, 11:433–452, 1994.

[Meg85] N. Megiddo. Partitioning with two lines in the plane. *J. Algorithms*, 3:430–433, 1985.

[Mos02] Karl Mosler. *Multivariate Dispersion, Central Regions and Depth. The Lift Zonoid Approach*, volume 165 of *Lecture notes in statistics (Springer-Verlag)*. Springer-Verlag New York, Inc, 2002.

[Raf] Eynat Rafalin. Computational Geometry at Tufts - Data Depth. Internet website. `http://www.cs.tufts.edu/research/geometry/data_depth/`.