A New Approach for the Computation of Halfspace Depth in High Dimensions

Yijun Zuo

Department of Statistics and Probability, Michigan State University
East Lansing, MI 48824, USA
zuo@msu.edu

October 27, 2017

Abstract

Halfspace depth (HD), aka Tukey depth, is one of the most prevailing depth notions among all its competitors (Zuo and Serfling (2000), ZS00). To exactly compute the HD in $\mathbb{R}^d (d > 2)$ is a challenging task nevertheless, due to its definition involving infinitely many directional projections. Existing algorithms to compute HD in $\mathbb{R}^d$ $(d \geq 2)$, more or less involve data projection to the directions perpendicular to hyperplanes (therefore involving polytopes or polyhedral cones) are either dimension d-, or sample size n- limited, or slow.

Thanks to the work of Merkle (2010) (M10) and Bogićević and Merkle (2016) (BM16), algorithms for the fast computation of HD in ultra high dimension d for large n are proposed and addressed. They are sheer ball-based (or essentially point-wise distances) computations. The worst time complexity to approximately compute the depth m/n of a single point is $O((d + m + \log(n))n^2)$ which seems to be one of the best known results so far among the competitors for extra large d and n. Unlike most previous algorithms, they do not require the underlying data set to be *in general position*. As by-products of the algorithm, some depth regions (or depth level sets), and depth of sample points also could be obtained during the depth calculation of a given point.

AMS 2000 Classification: Primary 62G99; Secondary 62H99, 62P99, 90-08.

Key words and phrases: Halfspace depth; definition and computation; high dimension and large sample size; algorithm; time complexity.

1 Introduction

To visualize multivariate data, Tukey (1975) introduced the notion of data depth (known now as halfspace depth (HD)) and depth contours. HD is closely related to the test statistic in Hodges (1955). Besides HD, major competing depth notions include simplicial depth (Liu (1990)), projection depth (Liu (1992), ZS00, Zuo (2003)) and those in Mizera (2002). With the notions of data depth or depth function, one can extend nonparametric univariate rank/order based procedures and their advantages to the multidimensional setting, rank/order, and visualize multidimensional data.

Halfspace depth of a point $x \in \mathbb{R}^d$ with respect to (w.r.t.) underlying probability measure P(or distribution F) is defined to be

$$\begin{aligned} HD(x,P) &= \inf_{H_x}\{\ P(H_x) :\ \mid H_x \text{ is a closed halfspace in } \mathbb{R}^d \text{ and } x \in H_x\ \} \\ &= \inf_{u \in S^{d-1}} P\{y :\ \mid u'y \geq u'x,\ y \in \mathbb{R}^d\}, \end{aligned}$$

where $S^{d-1} := \{u \in \mathbb{R}^d :\ \mid \|u\| = 1\}$. In one dimension, one can readily see that $HD(x,F) = \min\{F(x), 1 - F(x^-)\}$.

All points x such that $HD(x,P) \geq \alpha \in [0,1]$ form a set which is called *an αth depth region*, denoted by $HD^\alpha(P)$, i.e.

$$HD^\alpha(P) = \{x :\ \mid HD(x,P) \geq \alpha,\},\ \alpha \in [0,1].$$

Note that $HD(x,P)$ is nonnegative, hence, $HD^0(P) = \mathbb{R}^d$. Define

$$\alpha^* := \sup_{x \in R^d} HD(x,P).$$

That is, α^* is the maximum possible halfspace depth. Hereafter, we confine our attention to the case: $\alpha \in (0, \alpha^*]$.

Sample versions of $HD(x,P)$ and $HD^\alpha(P)$ and α^* are obtained by replacing P by its sample version P_n which assigns mass $1/n$ to each sample point X_i, $i = 1, \cdots, n$. For illustration purpose, a special example of sample depth contours is given in Figure 1. This example has been discussed in other places. There are four halfspace depth contours with depth 1/8, 2/8, 3/8, 4/8, corresponding to the white, red, blue and yellow regions, respectively. We like to emphasize a special fact, that is, the interior depth regions contain no sample points.

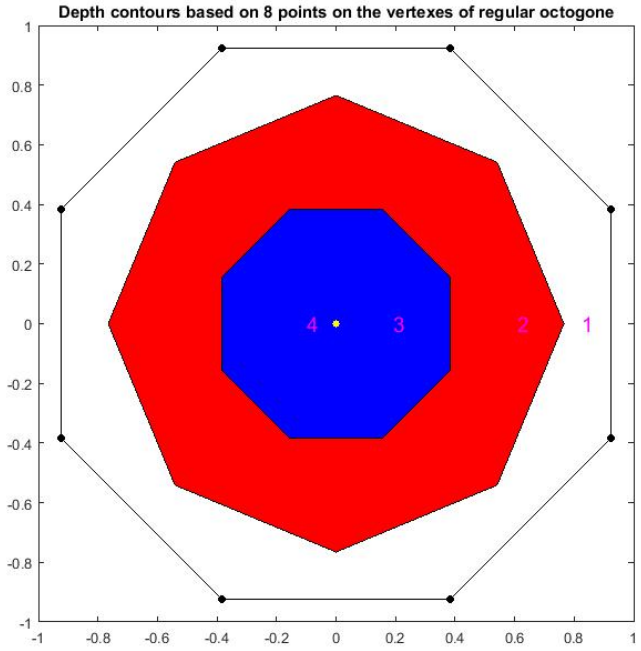

Figure 1: Four halfspace depth contours corresponding to depth $1/8, 2/8, 3/8, 4/8$

Many researchers focus on the computation of depth regions: $HD^{\alpha}(P_n)$. The latter are also called level sets in the literature, with level sets they can "order" data in a center-outward fashion (from the deepest region outward to least deepest one) and these sets also facilitate the visualization of data in $\mathbb{R}^d$. The deepest region is often called *median region* in $\mathbb{R}^d$. Note that

$$HD^{\alpha}(P_n) = \cap\{H : |\ H \text{ is a closed halfspace, } P_n(H) > 1 - \alpha\}, \qquad (1)$$

for a general treatment of HD^{α} see Section 3 or page 479 of ZS00, also see Kuelbs and Zinn (2017+a, b) and Brunel (2016). Therefore HD^{α^*} is the median region in $\mathbb{R}^d$.

Some researchers (e.g. BM16) believe that exactly computing depth regions is more important than that of depth of single point, and argue that the latter is seldom needed in real applications. On the contrary, this author believes that one does need to compute the depth of single point in some real applications. For example, in the depth based applications, one has to know the depth of the point in order to fulfil the task. Hereafter we focus on the *single point depth calculation*.

Depth region computation seems more complicated than that of single point depth computation. For both computations, existing algorithms

more or less involve consideration of data projections along the directions within ploytopes, or polyhedral cones. The latter are formed by the bounded intersection of a finite set of hyerplanes in $\mathbb{R}^d$ (see page 973 of Liu and Zuo (2014)). Within those polyhedral cones, a permutation $\{i_1, \cdots, i_n\}$ of $\{1, 2, \cdots, n\}$ keeps unchanged when one projects data onto a direction u, where $u'X_{i_1} \leq u'X_{i_2} \leq \cdots, \leq u'X_{i_n}$ for the unit direction $u = a/\|a\|$ and *any* vector $a \neq 0$ within one cone. Maximization or minimization problem could be solved using linear programming technique over single cones.

The same idea, called *circular sequence*, has been used in Rousseeuw and Ruts (1996) and Ruts and Rousseeuw (1996a,b), also in Zuo and Lai (2011). Overall time complexity of these algorithms is roughly of the order $O(n^d \log(n))$, *ad hoc* ones using computational geometry and other techniques may improve the time complexity a little bit, but not too much.

Utilizing a main result in M10, this article will introduce a novel approach to the computation of the depth of single points including a sample point. The main result also has been employed in BM16. The latter nevertheless focuses on *all depth contours computation* which turns out to be very difficult if not impossible. Here we focus on *depth computation of single point*. As a by-product, our algorithm could also be used to produce some depth regions. The new approach avoids $O(n^d)$ polyhedral cones, using instead essentially point-wise distances of the order of $O(n^2)$. Unlike most previous algorithms, the new one *does not* require the underlying data set is in general position. (A d-variate data set is called to be *in general position* (GP) if there are no more than d sample points in any $(d-1)$-dimensional hyperplane)

The rest of article is organized as follows. Section 2 briefly surveys the state of the art of computation of halfsapce depth and depth regions. Section 3 discusses and addresses a new approach and algorithms. Real data examples and simulation comparisons are given in Section 4. Section 5 ends the article with some concluding remarks.

2 State of the Art

The idea of a *circular sequence* in Edelsbrunner (1987) was first employed in Rousseeuw and Ruts (1996) and Ruts and Rousseeuw (1996a,b) (later it is also used in Zuo and Lai (2011) for projection depth). These were pioneer works in exact computing HD or depth contours for bivariate data. By restricting the computation to a small selected subset of data points and

a small number of depth contours to be computed, Johnson, et al. (1998) developed an algorithm FDC faster than the ISODEPTH given in Ruts and Rousseeuw (1996b), while achieving the detection of outliers in data cloud.

Using *duality* and *topological sweep* of an arrangement of lines, Miller, et al. (2003) achieved the best known algorithm so far for computing all halfspace depth contours and depth of a point in order of $O(n^2)$ time in $\mathbb{R}^2$.

Burr et al. (2011) using *dynamic maintenance* of halfspace depth for point and contours, proposed three algorithms with $O(\log n)$, $O(n \log n)$, and $O(n \log^2 n)$ in time per update respectively for single point, *rank-based* contours and *cover-based* contours (see Liu et al. (1999) also Rafalin and Souvaine (2004) for the definition of the two types of contours) for data sets in R^2 and *in general position*. Contours in this article are cover-based.

Bremner et al. (2006) proposed primal-dual algorithms which deal with arbitrary dimension d after Rousseeuw and Struyf (1998) algorithm dealing with $d = 3$. Their prima-dual algorithm uses *reserve search* technique and updates at every step an upper and a lower bound for the depth and terminates when and if they equal. The algorithm utilizes software like cdd. lrs, ZAMA (for parallelization), etc. It runs in the time of $O(n \cdot LP(n, d) \cdot |cells|)$, where $LP(n, d)$ denotes the time required to determine the feasibility of linear program having n constraints and d variables , and $|cells|$ stands for the total number of the cells used in the search.

Bremner *et al.* (2008) improved the previous result to a running time of $O(d^m \cdot LP(n, d-1))$ for a single point $x \in R^d$ with output depth $HD(x, P_n) = m/n$ calculation ($1 \leq m \leq n$). The algorithm thus is called "*Output-sensitive* algorithms for Tukey depth".

Using their directional quantiles defined based on halfspaces to form envelope, Kong and Mizera (2012) proved that the envelope coincides with the corresponding HD trimmed region. The idea of segmenting R^d into directional cones has been employed in Mosler, et al. (2009) for zonoid depth (Mosler (2002)) for zonid depth regions and later in Hallin et al. (2010), Paindaveine and Šiman (2012a,b) for HD regions, in Šiman (2011) for statistics based on project pursuit, and in Liu and Zuo (2014) for HD of a point. The worst case time complexity of these algorithms is roughly in the order of $O(n^d)$. Chan (2004) presented a randomized algorithm to compute the deepest point with an expected time of $O(n^{d-1})$ in $\mathbb{R}^d$ ($d \geq 3$).

Dyckerhoff and Mozharovskyi (2016) (DM16) proposed algorithms that

exactly computing the HD for *general data sets* (not necessary in GP). For each tuple of k ($1 \leq k \leq d-1$) data points selected from original n points, the data are projected onto the corresponding orthogonal complement, and the HD is computed as the sum of the depths in these two orthogonal subspaces. The calculation of one depth value in d-space is reduced to calculating many depth values in $(d-k)$-space and in k-space. The algorithm is of the order $O(n^{d-1}\log n)$ or $O(n^d)$ with $k = 1, d-2$ or $k = d-1$, respectively. Note that DM16 implies implicitly that data sets must content $n \geq d-1$.

BM16 employing a main result from M10 and reported that their algorithm for computing halfspace depth contours has a running time of order of $O(dn^2 + n^2 \log n)$ in $\mathbb{R}^d$ which *if true* (unfortunately it is problematic) is by far the best. They have a faultless theoretical result (from M10) but the realization has some problems (i.e., the computed regions or level sets are not the exact Tukey depth regions), see related discussions in Section 3.

Above we focus on the exact computation of depth or depth regions. Exact computation is very important, but often is costly (time intensive) though unaffordable in practice. Researchers therefore seek approximate algorithms to compute the depth of a point and depth regions.

For example, Rousseeuw and Struyf (1998) (RS98); Struyf and Rousseeuw (2000) (SR00) presented approximate algorithms to compute the halfspace depth of a single point in $\mathbb{R}^d$ ($d > 3$). An intuitively straightforward approximate method by randomly chosen directions over unit sphere was introduced by Cuesta-Albertos and Nieto-Reyes (2008) (CN08) (the authors attributed their main idea to Zuo (2006)) which was proved to be quite useful in experimental trials when one can afford to increase the number of chosen directions to be sufficiently large. Afshani and Chan (2009) employed a randomized data structure and halfspace range counting queries techniques to fast and quite accurately approximate depth value. Chen et al. (2013) (CH13) addressed the absolute approximation of Tukey depth, generalizing an algorithm presented by RS98.

Recent promising R packages for Tukey depth include 'depth' by Genest, Masse and Plante (2017), and 'ddalpha' by Pokotyl, Mozharovskyi and Dyckerhoff (2016) and packages cited therein.

3 A New Approach

3.1 Preliminary Theoretical Results

First, we present a general definition of so-called Type D depth functions which include HD function as a special case. Let $\mathscr{C}$ be a collection of closed Borel subsets in $\mathbb{R}^d$ and P be a probability measure on $\mathbb{R}^d$. *A Type D depth function* for a point $x \in \mathbb{R}^d$ is defined as (see page 472 of ZS00)

$$D(x; P, \mathscr{C}) := \inf_{C \in \mathscr{C}} \{P(C) | x \in C \in \mathscr{C}\}. \tag{2}$$

Remarks 3.1

(I) To insure the above is well defined, we assume that for any $x \in \mathbb{R}^d$, there always exists a $C \in \mathscr{C}$ such that $x \in C$. Examples of $\mathscr{C}$ include (i) all closed halfspaces $\mathscr{H}$, (ii) closed convex sets $\mathscr{V}$, or (iii) closed balls $\mathscr{B}$. With closed halfspaces $\mathscr{H}$, the above definition recovers the halfspace depth function defined in Section 1.

(II) Just from the definition point of view, the above closeness is not necessary but for the sake of depth region properties discussion in $\mathbb{R}^d$, we add the closeness requirement here. That is, a collection of open sets $\mathscr{C}$ in $\mathbb{R}^d$ can also serve the role in the definition above.

With the general definition, one now can define the *αth depth region* as

$$D^\alpha(P, \mathscr{C}) := \{ x \mid D(x; P, \mathscr{C}) \geq \alpha \in (0, 1] \}. \tag{3}$$

That is, the set of all points with depth at least α. For general properties of $D(x; P, \mathscr{C})$ and $D^\alpha(P, \mathscr{C})$, see ZS00 Theorem 2.11 and Zuo and Serfling (2000b). Hereafter we sometimes write D^α for $D^\alpha(P, \mathscr{C})$.

Generally speaking, depth regions could be employed to determine the depth of a single point. Specifically, $D(x; P, \mathscr{C}) = \alpha$ iff $x \notin D^{\alpha+\epsilon}$ and $x \in D^{\alpha-\epsilon}$, for any $\epsilon > 0$, see ZS00b or BM16. In halfspace depth and the sample case, we have the following practically very useful result.

Proposition 1. Let $\mathscr{C} = \mathscr{H}$, for any point $x \in \mathbb{R}^d$ and $\alpha \in (0, 1]$,

$$HD(x, P_n) = \alpha \text{ iff } \; x \in HD^\alpha(P_n) \text{ and } x \notin HD^{\alpha+1/n}(P_n). \tag{4}$$

The proof is straightforward and skipped here.

The following result gives a useful representation of D^α.

Proposition 2 The αth depth region $D^\alpha(P, \mathscr{C})$ with the given probability measure P and the collection $\mathscr{C}$ of Borel subsets in $\mathbb{R}^d$ can be expressed as

$$D^\alpha(P, \mathscr{C}) = \cap_{C \in \mathscr{C}} \{C | P(C) > 1 - \alpha\} = \cap_{C^c \in \mathscr{C}^c} \{C^c | P(C^c) > 1 - \alpha\}, \quad (5)$$

where $\mathscr{S}^c$ stands for the collection of all complements S^c of the set S in $\mathscr{S}$.

To insure the nonemptiness of the right hand side (RHS) of equality above, we assume that there always exist $C_1, C_2 \in \mathscr{C}$ such that $P(C_1) > 1-\alpha$ and $P(C_2) < \alpha$ for any $\alpha \in (0, 1]$.

Proof: The proof of the first equality is given in ZS00 (page 479), the proof of the second part follows directly from the definition of D^α and its compliment (also see M10).

Remarks 3.2 The result implies that

(I) the αth depth region is the intersection of all sets $C \in \mathscr{C}$ that contain probability mass more than $1 - \alpha$, or equivalently, all complements $C^c \in \mathscr{C}^c$ that possess probability mass more than $1 - \alpha$; that is, $\mathscr{C}$ and $\mathscr{C}^c$ is interchangeable.

(II) equivalently the complement of the αth depth region is the union of all sets $C \in \mathscr{C}$ that contain probability mass less than α.

(III) in the halfspace case, the αth halfspace depth region is the intersection of all closed halfspaces that possess probability mass more than $1 - \alpha$. Further more, one can replace the closed halfspaces with open ones.

The following result further indicates that one can replace the closed halfspaces with closed balls.

Proposition 3 Let $\mathscr{H}$ and $\mathscr{B}$ be the collection of all closed halfspaces and closed balls in $\mathbb{R}^d$ respectively and P is the underlying probability measure. Then

$$D(x; P, \mathscr{H}) = D(x; P, \mathscr{B}^c), \quad (6)$$

and

$$D^\alpha(P, \mathscr{H}) = \cap_{B \in \mathscr{B}} \{B | P(B) > 1 - \alpha\}. \quad (7)$$

Proof Equality (6) is the Corollary 4.1 of M10. Equality (7) follows directly from Proposition 2 and (6) (see also (4.3) of M10).

In the following we will focus on HD and sample distribution P_n and will suppress $\mathscr{H}$.

Corollary 1. Let $\mathscr{B}$ be the collection of all closed balls in $\mathbb{R}^d$. Then

$$HD^{\alpha}(P_n) = \cap_{B\in\mathscr{B}}\{B \mid \#\{i \mid X_i \in B\} > n(1-\alpha)\}, \tag{8}$$

where "#" is the counting measure and $X_1, \cdots, X_n$ are sample points.

Remarks 3.3

(I) That is, αth halfspace depth region is the intersection of all closed balls each of which contains at least $\lfloor n(1-\alpha)\rfloor + 1$ sample points.

(II) The centers of the balls however are undetermined in addition to the undetermined $\lfloor n(1-\alpha)\rfloor + 1$ sample points, and moreover there are infinitely many balls containing the same $\lfloor n(1-\alpha)\rfloor + 1$ sample points.

(III) That is, the result above, is seemingly useful for *exact computation of the depth region*, but actually falls short in practice (as already unsuccessfully experimented in BM16), due to the fact that there will be infinitely many *undetermined* balls on the RHS.

Furthermore, some depth regions are not *directly* determined by sample points (there are cases where no sample points are on its vertexes or boundaries or its interior of the HD regions, see Figure 1).

3.2 An exact algorithm for halfspace depth

The following result indicates that one does not have to consider *infinitely* many balls and can just focus on *finitely* many balls in order to exactly compute the halfspace depth of sample points within α depth regions. A ball with minimum radius that includes all n points $x_1, \cdots, x_n$ within (or on) its boundary is called a *minimum enclosing ball* for these n points.

Theorem 1 Let $\mathscr{B}$ be the collection of all closed balls in $\mathbb{R}^d$. Then

$$\{X_j \mid HD(X_j, P_n) \geq \alpha\} \quad \subset \quad \big\{X_i \mid X_i \in \cap_l^N B_l,\ \#\{i \mid X_i \in B_l\} \geq k_\alpha\big\},$$

where $X_i, X_j \in \{X_1, \cdots, X_n\}$, $k_\alpha := \lfloor n(1-\alpha)\rfloor + 1$, $N = \binom{n}{k_\alpha}$, and $B_l \in \mathscr{B}$, $l = 1, \cdots, N$ is the unique minimum enclosing ball (MEB) determined by k_α data points from $\{X_1, \cdots, X_n\}$, $\alpha \in (0, \alpha^*]$.

Proof of Theorem 1

Assume that $HD(X_j, P_n) \geq \alpha$, that is, X_j is a sample point in the set on the left hand side (LHS). We show that it belongs to the set on the RHS.

First, it belongs to the αth depth region based on the definition, i.e. it belongs to the LHS of (8) and hence RHS of (8). That is, X_j belongs to *any* ball that contains k_α sample points, hence it belongs to the MEB ball based on those k_α sample points and consequently to all N MEBs. That is, X_j belongs to the set on the RHS. □

Remarks 3.4

(I) The result provides an approach to identify exactly the sample points with its HD at least $\alpha \in (0, \alpha^*]$. Specifically, one just needs to take intersections of N balls each of which contains k_α sample points. The N balls are MEBs, each of which contains at least k_α sample points. One might need to shift the center of the MEBs if necessary so that it still contains $k_\alpha = n - k + 1$ sample points but *as few as possible total sample points*. See examples 3.1.

(II) The theorem implies that the set on the RHS contains all sample points with depth $\geq \alpha$. It does not exclude some sample points with depth $< \alpha$. In the algorithm (below), we can try to exclude the latter scenario. We start with the smallest possible depth value α of sample points (i.e. $1/n$); and increase it by $1/n$ at each step; meanwhile, update the set on the RHS (initial set is the entire data set) so that the sample points with depth value lower than α are excluded from the set.

(III) It is a well-known result that MEBs are unique and can be computed in linear time (see Welzl (1991), Fischer (2001), Fischer and Gärtner (2003)).

Algorithm 1 for the computation of sample points with fixed HD α
input: data matrix X with d rows and n columns;
output: sample points $X_i \in X$ with halfspace depth $\alpha = m/n$, $1 \leq m \leq n$.

For $1 \leq k \leq \min\{(m+1), n\}$
Let $S_0 = X$,$N_k = \binom{n}{n-k+1}$, construct MEBs B_j each containing the $n - k + 1$ (but as few as possible) sample points, $j = 1, \cdots, N_k$;
Let $S_k := S_{k-1} \cap_{j=1}^{N_k} B_j$;

if $S_k = \emptyset$, **print** "no sample points with depth $\geq k/n$"; **break**;
if $X_i \in S_{k-1}$ and $X_i \notin S_k$, **output** X_i and $\text{HD}(X_i) = (k-1)/n$;
if $X_i \in S_k$ and $k = n$, **output** X_i and $\text{HD}(X_i) = 1$.
end

Example 3.1 Let's illustrate the algorithm with some simple examples.

(I) Assume that we have a bivariate sample with $n = 3$: $X_1 = (0,1)'$, $X_2 = (0,0)'$ and $X_3 = (1,0)'$. Of course, in this case one can immediately determine that all sample points have HD 1/3. Assume that we have $\alpha = 1/3$, i.e. $m = 1$. When $k = 1$, we have $S_0 = X$, $N_1 = 1$, B_1 contains all sample points, and consequently $S_1 = S_0 = X$. In the next loop, $k = 2$, we have $N_2 = 3$, $B_1 = \{X_1, X_2\}$, $B_2 = \{X_1, X_3\}$, $B_3 = \{X_2, X_3\}$. Note that the MEB originally containing X_1 and X_3 actually also contains X_2, so the B_2 does not contain the *as few as possible* sample points. We can shift the center of B_2 a little bit, say to $(r, r)'$ instead of $(1/2, 1/2)'$, where $r > 1/2$, to content the requirement of containing *as few as possible but* $k_\alpha (= 2)$ sample points. Now we have $S_2 = \emptyset$, So all three sample points have HD 1/3.

(II) Now consider the case that there is fourth data point X_4 which is overlapped with X_3, i.e. we have a data set that is *not in general position*. Applying the above algorithm, one can successfully identify the exact depth of 1/4, 1/4, 2/4 and 2/4 for X_1, X_2, X_3 and X_4, respectively.

One can also consider (III) the case of four sample points in $\mathbb{R}^2$ which form arbitrary quadrilateral and (IV) then add one more sample point inside (or (V) on the boundary), applying the algorithm one can compute the exact depth of sample points in all three cases successfully.

In the classification or discriminant depth applications, one has a new observation x (might be a new cancer patient) which is no longer a sample point. How can one figure out its depth w.r.t. a given sample?

Let x be a non sample point w.r.t. the sample $X_1, \cdots, X_n$ in $\mathbb{R}^d$. Add x to the original data and denote the augmented data set as $X'_1, \cdots, X'_{n+1}$ and let P'_{n+1} be the empirical distribution corresponding to the augmented data set.

Then the following result tells that one just needs to focus on the depth calculation of *sample points*.

Proposition 4 $HD(x, P_n) = m/n$ iff $HD(x, P'_{n+1}) = (m+1)/(n+1)$.

Proof: It is straightforward and trivial.

Remarks 3.5

(I) The result provides a direct way to calculate the halfspace depth of an *arbitrary non sample point* $x \in \mathbb{R}^d$ w.r.t. the original data set and the

augmented data set. That is, if one could get the halfspace depth w.r.t. one data set and the depth w.r.t. other data set is automatically obtained and vice versa. This means the algorithm 1 could also be adapted to the calculation of halfspace depth of an *arbitrary* point $x \in \mathbb{R}^d$. This is especially important in the case of classification and discrimination for real applications.

(II) With α halfspace depth sample points, one may hope to reconstruct αth depth contours using the so-called *vertex or facet enumeration* (see Bremner, Fukuda, and Marzetta (1998)). This approach works for some α (such as $1/n$) but not for all α due to the fact that some depth regions contain no sample points on its vertexes, boundary, or entire convex region. If one records the results from the algorithm, then the depth of all sample points can be used to construct all the *rank-based* contours.

(III) In the algorithm 1, set intersections can be done in linear time and built-in function exists in packages like C++, Matlab, and R. However, there are totally *unaffordable* $N_k = \binom{n}{n-k+1}$ intersections for every $1 \leq k \leq \min\{(m+1), n\}$. That is, the *exact* algorithm 1 is still not feasible in practice except in the special cases below.

(IV) When n is small and d is large (also called *a big p small n problem*), as in the micro-array data cases or more general genetic data analysis (see Johnstone and Titterington (2009)), one has small subject number n (in tens) but huge dimensions d (in hundreds). The algorithm works effectively. On the other hand, classical approaches may fail in these cases, since when $n < d$, the hyperplane contains d sample points could not be constructed and hence the polytopes approaches fail. Meanwhile, MEBs still exist (see Fischer and Gärtner (2003)). Of course one could work on the subspace formed by (and containing) n points. Besides, if the data set is in GP, then each sample point should have HD $1/n$ in these degenerated (singular) cases.

When n is large (with small or large d), algorithm 1 is no longer feasible for the computation of halfspace depth. One, however, can (i) either count on research developments in computational geometry and computer science to have more feasible algorithms based on ball intersections, or (ii) propose fast and *approximate* algorithms like the one given in Section 3.3 below, where one just considers n balls initially.

3.3 Approximate Algorithms and Discussions

The following result provides a method which identifies sample points with halfspace depth at least α.

Theorem 2 Let $\mathscr{B}$ be the collection of all closed balls in $\mathbb{R}^d$. Then

$$\{X_j \mid HD(X_j, P_n) \geq \alpha\} \quad \subset \quad \{X_i \mid X_i \in \cap_{j=1}^{n} B_j\}, \tag{9}$$

where B_j is the minimum ball centered at X_j containing k_α sample points.

Proof: It is trivial.

Remarks 3.6

(I) Based on the result, one can identify sample points with depth at least α just by the intersection of n balls. Note that each of the n balls is determined with fixed center and easily identified radius.

(II) In one dimension, the smallest and largest sample points in the set of RHS of (9) exactly form the αth depth region (interval). This suggests an alternative way to define the halfspace depth and region in $\mathbb{R}^1$ and is a demonstration of Proposition 3 in $\mathbb{R}^1$.

(III) Although the set on the RHS of (9) is not exactly equal to the one on the LHS. But from the relationship of the two sets, we can develop an *approximate algorithm* to compute the HD of sample points. The algorithm is given below and is based on the onion-peeling idea.

(IV) The theorem implies that the set on the RHS contains all sample points with depth no less than α but it does not exclude some sample points with depth less than α. In the algorithm, we can try to exclude the latter scenario. We start with the smallest possible depth value α of sample points (i.e. $1/n$); and increase it by $1/n$ at each step; meanwhile, update the set on the RHS (initial set is the entire data set) so that the sample points with depth value lower than α are excluded from the set (by adding systematically balls centered at points uniformly selected from a region based on data and containing k_α sample points into intersection process). In this way, we can further identify all sample points with exact α depth by utilizing Proposition 1. Generally speaking, the algorithm based on Theorem 2 tends to be too liberal (*overestimate* HD of points, see examples in Section 4).

Combining Proposition 4 with Theorem 2, one can also compute the halfs-

pace depth of an *arbitrary non sample point* $x \in \mathbb{R}^d$ use the n balls intersection idea in Theorem 2 on the RHS of (9). The following is the algorithm.

Algorithm 2 for the computation of HD of an arbitrary point in $\mathbb{R}^d$:

Input: a data set $X = \{X_1, \cdots, X_n\}$ with d rows and n columns and an arbitrary point X_0 in $\mathbb{R}^d$ ($d \geq 2$).

Output: halfspace depth of X_0.

Initial Phase

(I) check if X_0 is a sample point, record the finding in a variable p, p=0, or 1 p=0 means yes. modify X so that it contains X_0, update n to $n^* = n + p$, w.l.o.g., still call it n;

$\cdots$ % in O(n) time

(II) using matrix operation, compute the $O(n(n-1)/2)$ point-wise distances;

$\cdots$ % in $O(d \cdot n^2)$ time

(III) sort for each point its point-wise distances, i.e., d_{ij}, $i, j = 1, \cdots, n$;

$\cdots$ % in $O(n^2 \log(n))$ time

(IV) initiating: $m = 0$, $\alpha_0 = 0$, $S_0 = \{X_i : X_i \in X\}$;

Iteration Phase

(V) updating: $m = m + 1$, $\alpha_m = m/n$; set $S_m = S_{m-1} \cap_{i=1}^{n} B_i$;

If ($X_0 \in S_{m-1}$ and $X_0 \notin S_m$), then hd $= (m - 1 - p)/(n - p)$;

$\cdots$ % converting the depth by Proposition 4 and 1;

if ($S_m = S_{m-1}$ and $X_0 \in S_m$ and $\alpha_m = 1$), then hd $= 1$

fprintf (' halfspace depth of X_0 is: $\%f\backslash n$ ', hd); **return**;

% where B_i is the set of points X_j's with $d_{ij} \leq d_{ik_\alpha}$, $i, j \in \{1, \cdots, n\}$ and
% $k_\alpha = \lfloor n(1-\alpha) \rfloor + 1 = (n - m) + 1$.

$\cdots$ % in $O(n^2)$ time,

(VI) if $S_m = S_{m-1}$ and $\alpha_m < 1$, then goto (V);

$\cdots$ % in $O(n)$ time,

% the loop above has been executed at most m times if the depth of X_0 is m/n.

The algorithm itself has already clearly indicated its **worse case time complexity**, i.e. $O(n + d \cdot n^2 + n^2 \log(n) + m \cdot n^2)$, where the halfspace depth of the given point is assumed to be m/n, or more precisely, $O((d + m + \log(n)) \cdot n^2)$. Note that the largest possible value of m is n.

Strictly speaking, d plays a role in almost all the steps above. For example, checking whether a point belongs to a set involves d coordinate comparisons besides going over all members in the set. Since they are all linear in d and the high order term $n^2 \log(n)$ does not involving the d (d_{ij} are just one-dimensional numbers), w.l.o.g., d factor is ignored in those terms.

That is, the time complexity depends on the dimension d only linearly and on sample size n logarithmically quadratically. This seems to be one of the best existing results for approximately computing the halfspace depth for arbitrary large d and n, as far as the author knows.

If one is interested in depth contours, then at each step after updating S_m, one can use **qhull** or **quickhull** to construct contours in $O(n \log(n))$ time. Here some caution must be taken, i.e., in some cases, the contours constructed are not exactly equal to the αth depth contours, see Remarks after Algorithm 1. On the other hand, one can safely store or output the depth of each sample points in S_m. In this way, one can obtain the depth for all sample points and therefore construct any *rank-based contours*.

For two extra options above, algorithm 2 needs very slight modification, adding no extra time complexity in terms of big O. With this information, one can even identify the halfspace median by borrowing the idea on page 833-834 of Rousseeuw and Ruts (1998) exactly in $\mathbb{R}^2$ or on page 419 of SR00 approximately in $\mathbb{R}^d$ ($d > 2$).

4 Examples and Simulations

4.1 Real data

4.1.1 Bivariate data set

Approximate Algorithm 2

We start with a data set(lib.stat.cmu.edu/DASL/Datafiles/nycrimedat.html) studied by SR00 and focus on the change (in percent) in policeman power and in weekly car thefts in New York city for 23 time periods in 1966-1967. The scatter plot of the data set is given in Figure 2 (to enhance the readabil-

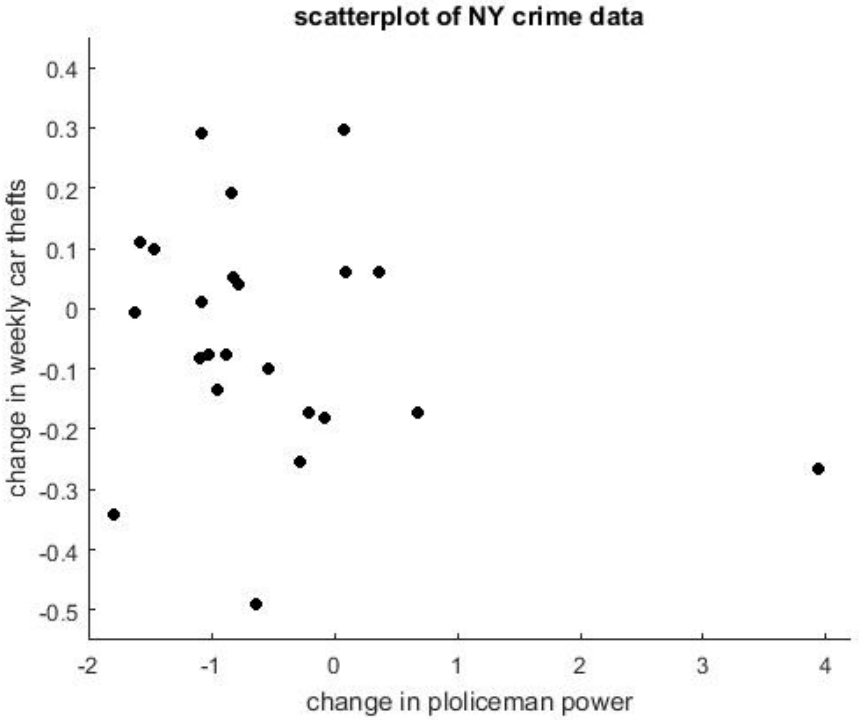

Figure 2: Scatter plot of crime data of New York City during 1966-1967.

ity, we uses a scale factor 0.1 for original data in this and other two plots) Inspecting the figure immediately reveals that there is an obvious outlier.

Employing our algorithm 2 to compute the HD of all sample points, the results are given in Figure 3. For illustration purpose, the depth of each point has been multiplied by sample size n so that we can simply display integers. Reviewing the figure one can see that many depth values are *overestimated* and there are just two (boundary) points that get the correct HD (1/23).

This is due to that fact that the approximate algorithm (called AA1) ignores many important balls, just looking at n balls. To get the exact HD of each sample point, we have to add more balls into the intersection process (called AA2).

General scheme for AA2: (i) pick a number N_b for total added balls (could be a fixed number (say 100) but usual $N_b \leq 1000$, depending on the structure of data, $N_b = 0$ in AA1 case); (ii) select the centers of N_b balls (could use a fixed scheme (say along axes), or as in this article randomly pick uniformly distributed points from a wide region determined by the data set); (iii) then calculate the distances from the center to all n sample points, sort them for each ball, keep just first k_α nearest sample points in

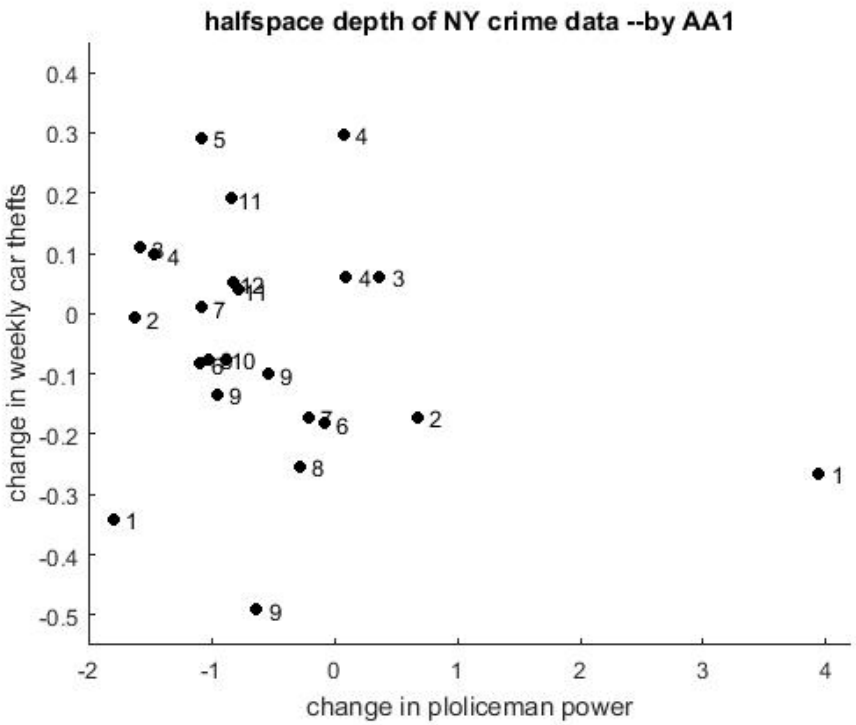

Figure 3: Halfspace depth of data points in NY crime data set, using approximate algorithm 2 only (AA1).

the intersection process; (iv) select a repeating number N_r and repeat the previous calculation N_r times to mitigate randomness, pick the minimum one as the final depth. N_r should coordinate with N_b (i.e., small N_r should be coupled with larger N_b, and vice versa, say, if $N_b = 1000$, then $N_r = 100$ is usual enough, on the other hand, if $N_r = 200$, then $N_b = 500$ might be enough. Generally, larger N_b is preferable to larger N_r).

By adding more balls into the intersection process we obtain the exact depth of all example points, in our experiment, we just add one more ball with strategically selected center for each point. The refined depth plot is given in Figure 4 . Comparing this with the Figure 1 of SR00, we see the refined algorithm (AA2) indeed captures exact depth of all points.

The time for the computation of depth in Figure 3 or Figure 4, is extremely short (about 1.29 or 1.60 seconds, respectively) based on matlab code (available upon request) on a server: Intel(R)Xeon(R) CPU E5-26670@2.90GHz 2.90GHz (2 processors), installed memory(RAM) 64.0GB.

Exact Algorithm 1

We have investigated the performance of approximate algorithm 2 above.

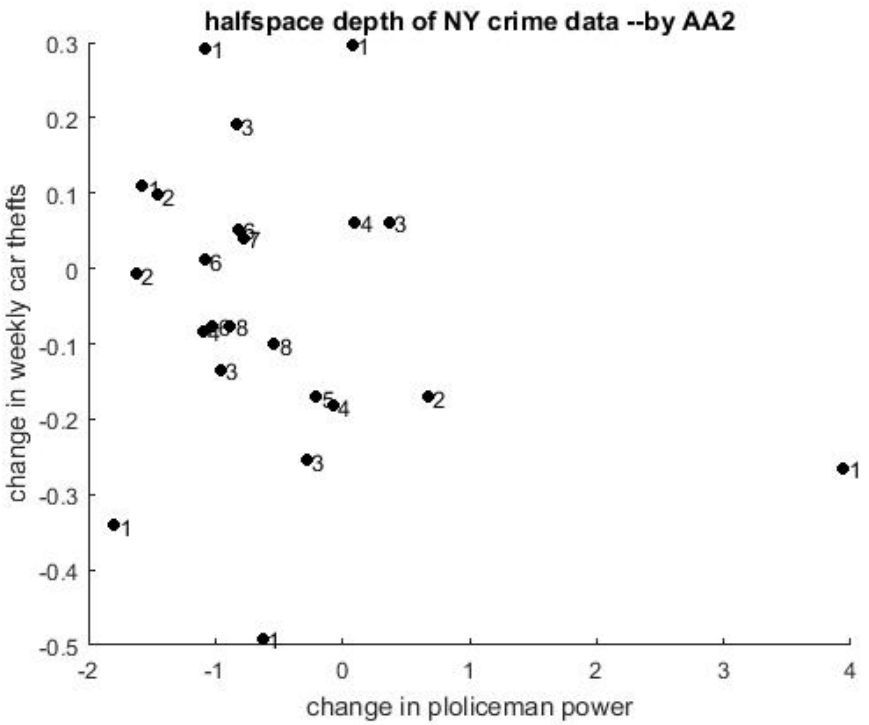

Figure 4: Halfspace depth of data points in NY crime data set, using refined algorithm 2 (AA2).

The running time 1.60 seconds for Figure 4 is somewhat misleading since it depends on "the strategically selected center". The latter however costs much more time to figure out and can be done just in two dimensional cases.

Without the special center of the added ball (circle), one can still obtain the exact depth by using AA2. One instead employs replication $N_r = 20$ times and adds $N_b = 100$ balls and costs 82.72 seconds for all points.

Now we like to investigate the performance of the exact algorithm (i.e. algorithm 1). We call it EA1. We first apply EA1 to **(I)** octagon data at the very beginning (i.e. 8 points located at the vertexes of the regular octagon), it costs 0.607 seconds for EA1 to get the exact depth of all points. We used $N_r = 10$ replications to alleviate the randomness of the sampling process. So the average time is 0.0607 seconds for all 8 points per replication.

We then apply the EA1 to **(II)** the three-point data set in Example 3.1 (I), note that there is a trouble point (0,0)', we have to shift the center of one of the MEBs (or try different centers, the scheme of selecting center is the same as the one in AA2) while keep the total sample points enclosed in the ball exactly $k_\alpha = n - k + 1$. We just selected $N_b = 10$ and $N_r = 10$, it costs 0.098 seconds for EA1 to get the exact halfspace depth 1/3 for all three

points, i.e. 0.0098 per replication. We also tried EA1 to **(III)** 10 standard normal points (center at (1.50, 3.75)'). EA1 did the job again with $N_r = 2$ and $N_b = 5$ and costed 0.60 seconds, see Figure 5. Note that the deepest point looks like to have depth 4/10 but actually 3/10.

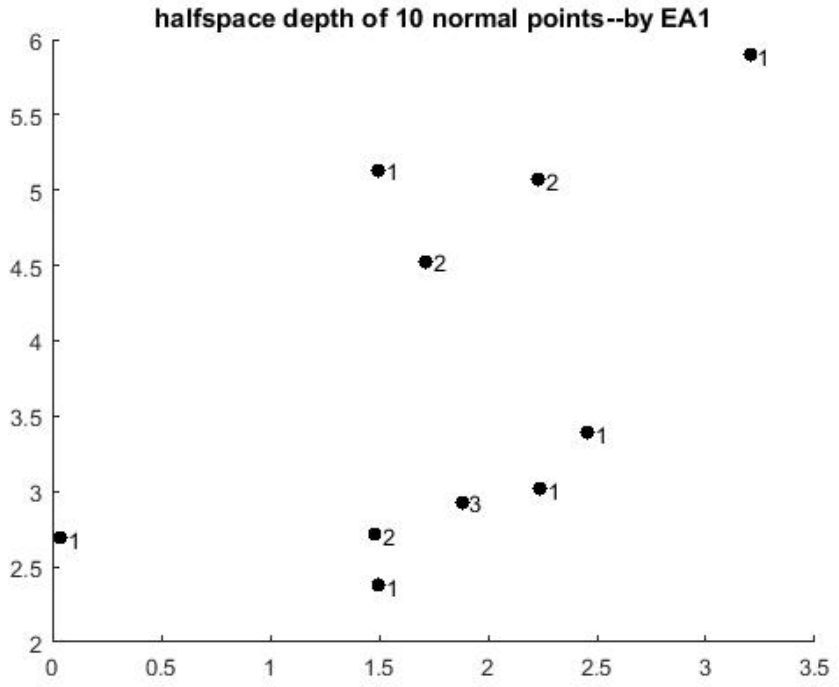

Figure 5: Halfspace depth of 10 normal data points, using exact algorithm 1 (EA1).

Finally, we apply EA1 to two real data sets. The first date set (from Seber (1984) (Table 9.12)) is **(IV)** the measurements of phosphate and chloride in the urine of 12 men with similar weights, and has been studied in Example 6.1 of Maronna, Martin, and Yohai (2006). The data set and its depth from EA1 are illustrated in Figure 6. The displayed numbers are again the $12 * HD$. It is easily to verify that they are indeed the exact HD. Here $N_r = 1$, $N_b = 5$, EA1 costed 0.35 seconds.

Of course, we also tried EA1 for **(V)** New York Crime data set. It turns out that the EA1 could get the exact depth for all 23 points. The time consumed per point varies widely, ranging from 0.0024 to 485 seconds corresponding to depth 1/23 and 8/23, respectively ($N_r = 1$, and $N_b = 1$ or

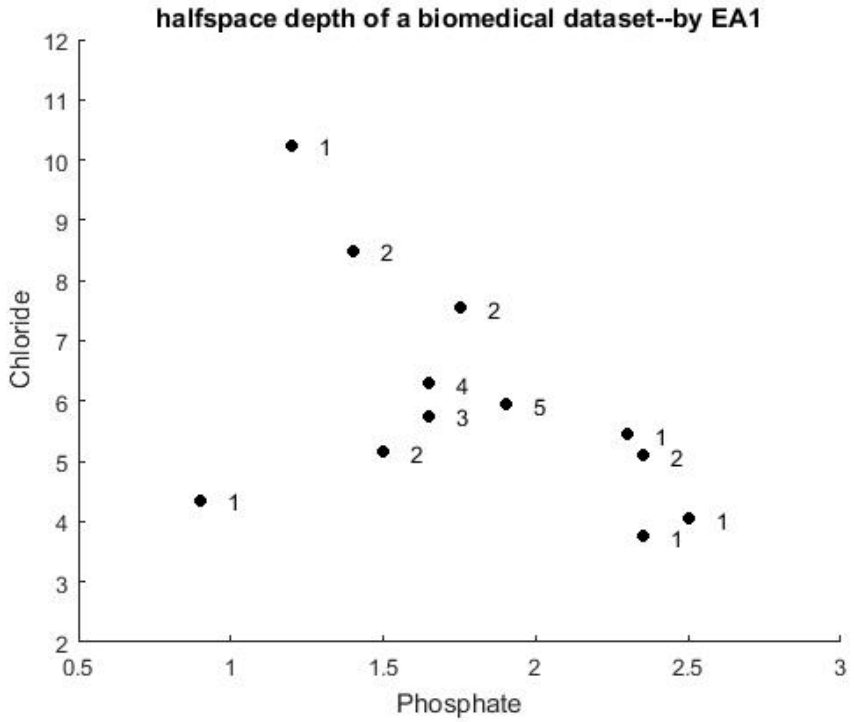

Figure 6: Halfspace depth of data points in a biomedical data set, using exact algorithm 1 (EA1).

$N_b = 10$, respectively). As excepted, lower depth point costs less time.

Remarks 4.1

(I) Our matlab code for EA1 (available upon request) is restricted to bivariate cases due to the limitation of existing executable code for MEB. Both Algorithm 1 and original MEB algorithm of Welzl (1991) are suitable for high dimensions though. That means further work to develop executable codes for both MEB and EA1 algorithms is needed.

(II) Compared with our AA algorithms, EA1, like other exact algorithms, is less efficient (much slower) and especially sample size n restricted. The latter is especially serious issue since no one can afford to have a $\binom{n}{n-k+1}$ loops in the computation, $2 \leq k \leq (m+1)$, where m/n is the HD to be computed. Note that EA1 depends on d only *linearly*.

4.1.2 Four dimensional data set

Now we focus on a data set in R^4, which has been studied in Liu and Zuo (2014) (LZ14). It consists of 64 four dimensional points, for details including the exact coordinates and depth of each point, please refer LZ14.

Here we compare our approximate depth results from AA1 and AA2 with the exact ones from LZ14, in terms of accuracy and average time consumed per data point and total time consumed for all 64 sample points (in seconds), where AA1 is the approximate algorithm based on Algorithm 2 and using only directly n balls intersections, whereas AA2 is the refined version by adding 1000 extra balls into the ball intersection process. Since AA2 involves randomness (using the random uniformly distributed points as the center of balls), the results are the minimum from 1000 replications.

Table 1. The performance of exact and approximate algorithms based on 64 four dimensional data points

Methods	accuracy	total time (seconds)	average time per point
LZ14	64/64	6194	96.80
AA1	4/64	15.87	0.248
AA2	58/64	97.23	1.52

Inspecting the table immediately reveals that the AA algorithms are unsurprisingly much faster than the exact one. AA1 just spent on average of 0.248 seconds whereas the exact one consumed about 97 seconds for a single point. For this advantage, the AA1 has to pay the heavy price of low accuracy (6% vs. 100%). AA2 on the other hand is also vary fast (1.52 vs. 97 seconds) but with an acceptable accuracy (about 90% vs. 100%). Note that, the accuracy of AA2 could be further improved by adding more balls (more than 1000) and of course that will cost more seconds.

On the other hand, if one can afford to wait for the exact results, one should still use the exact algorithm. Furthermore, without the slow exact algorithm as the benchmark, no one can develop fast approximate algorithms with known acceptable accuracy. *The contribution and importance of the exact algorithm can never be over emphasized.*

4.1.3 Thirteen dimensional data set

The LZ14 exact algorithm is restricted to dimension less than 8 (due to the limitation of some existing function with matlab), so for this data set in R^{13}, it is no longer applicable.

The original data set contains crime rate and 13 social variables (plus one binary variable) for 47 states of the USA in 1960, and there are 16 observations that come from southern states (corresponding to the binary variable labeled 1). For more details we refer to the R package "ddalpha".

We first get rid of the binary variable in the 14 variables, then compute the halfspace depth of all the 16 southern states w.r. t. all the 47 states in the USA in R^{13}. Due to the extremely long time needed (for a single point, it could cost more than 8 days to get the exact result), the exact algorithm given by DM16 which is theoretically capable of doing the job could not be practically employed here for gauge or comparison. Instead, we use another well vetted approximate algorithm results from Shao and Zuo (2016)(SZ16) as the benchmark (see the reason given below).

Results from other leading competitors (RS98, CN08, and CH13) are also summarized in Table 2 for comparison, where RS98 uses the fourth proposal on page 196, CH13 uses a modified third proposal and projects onto two dimensional affine space ($k = 2$). All three used 10^4 directions.

Reviewing the table, one can see that SZ16 is very fast yet most accurate, AA1 is even faster (0.135 v.s. 0.144 seconds) but least accurate (6% v.s. 100%). Whereas AA2 (adding 10^4 balls into the ball intersection process) is still quite fast (2.921 v.s. 0.135 seconds) but much more accurate (100% v.s. 6%) compared with AA1.

All results are based on the minimum halfspace depth from 50 times repetitions of the algorithms. Due to the consumed time factor, 50 is selected here. Changing this number, results might be slightly different. AA1 however is not sensitive at all since it has nothing to do with the randomness as long as the data set is fixed. It will serve as a good candidate of tools for ultra-high dimensional data screening to immediately identify outliers.

SZ16, RS98, CH13 and AA2 gets the smallest possible depth (1/47), i.e the exact depth, whereas CH08 is the second worst accurate yet the slowest one.

Table 2. The performance of approximate algorithms based on 16 points out of a thirteen dimensional data set of size 47

Methods	accuracy	total time (seconds)	average time per point per repeat
SZ16	16/16	115.0	0.144
AA1	01/16	108.2	0.135
AA2	16/16	2337.5	2.921
RS98	16/16	3767.0	4.709
CN08	5/16	4094.2	5.118
CH13	16/16	2548.7	3.186

Remarks 4.2

(I) Table 2 presents results from a *single run* of different methods. One may have concerns on the representativeness and fairness of the comparison since most methods (except AA1) involve randomness.

(II) To meet the concerns, Table 3 below lists the results based on $R = 10$ repetitions using different methods. The empirical mean squared error (EMSE) is defined as EMSE $= \frac{1}{R}\sum_{i=1}^{R} \|\hat{\theta}_i - \theta\|^2$ where θ is the target (exact) depth value vector with length 16 and each component 1/47, and $\hat{\theta}_i = \min_{k=1}^{N_r}\{\hat{d}_{i_k}\}$ is the depth vector estimate from the ith repetition and $\hat{d}_{i_k}$ is the depth vector estimate from the kth interior repetition within the ith exterior repetition, min function works column-wise, $N_r = 50$ is the interior repetition number of the methods.

Table 3. The performance of approximate algorithms based on 16 points out of a thirteen dimensional data set of size 47, repeated 10 times

Methods	EMSE	total time (seconds)	average time per replication
AA1	0.8302	1145.11	114.5
AA2	0.0064	3208.78	320.9
RS98	0.0000	3672.13	367.2
CN08	0.0399	4044.45	404.4
CH13	1.36e-4	3504.71	350.4

Inspecting the table reveals that speed of AA1 works again its accuracy. AA2 is the third most accurate and the second most fast. RS98 is the most

accurate. CH13, the improved version of RS98 that indeed consumed less time but less accurate. CH08 is the slowest yet the second worst accurate.

For RS98, CN08, and CH13, they used 10^3 random directions. Their performance could be improved by increasing this number (we could not afford in our simulation though), but see second point in Remark 4.2. If time permits one can increase N_r to get more accurate results. $N_b = 10^3$ is the total number of balls added into the intersection process in AA2, whereas AA1 corresponds to $N_b = 0$. One cannot afford to have $R = 100$ since it will consume roughly 10.2 and 11.2 hours for RS98 and CH08, respectively.

Remarks 4.3

(I) **Randomness** Note that AA1 actually has nothing to do with the randomness since the 16 points are fixed and the algorithm uses the same fixed steps in each replication. All others involve randomness. The purpose of repeating of the calculation 50 times is to alleviate their randomness. The potential advantage of the ball-based approach AA2 will be more clearly demonstrated for *larger* d in subsection 4.2.

(II) **Tuning parameters** All methods use parameters. In RS98, CN08, and CH13, one parameter is the number of random directions used. Increasing this number generally could improve their performance, but not always as one expected. For example, using 10^5 directions, CN08 costs 848 seconds but still misses 11 exact depth out of 16 in a single replication with its EMSE 0.0299 still significant. The same is true for the repetition number which can improve their performance but also could consume intolerable time.

Tuning N_b and the repeating number (N_r) in AA2 could change the performance of AA2. Generally speaking, smaller N_b or N_r will speed up the procedure but pay the price of losing accuracy.

(III) **Platform and programming** SZ16 is not included in the table since its original code and many of its employed stochastic optimization procedures are in R, the results above are based on matlab programming. From this point of view, the comparison of results in table 2 is somewhat unfair (since results are not obtained from the same platform and programming). One expects that SZ16 will still perform very well even using matlab in view of its single run result from R though (the two should not have very big difference).

4.2 Simulated data

After comparing the accuracy and efficiency of AA algorithms with other leading approximate algorithms, we next investigate the effect of the sample size n and dimension d on the performance of AA algorithms in terms of average consumed time (seconds) per data point (i.e. its scalability and feasibility).

As discussed in Section 3.3, AA1 costs at most $O((d+m+\log(n))\cdot n^2)$ time to compute the HD m/n of a point, whereas AA2 just adds N_b additional balls into the process. It then costs $O((d+m+\log(n))\cdot n^2 + N_b n\log(n))$.

Now we have experimental results for *finite numbers* d and n in order to better understand the *asymptotic* conclusions above. The average time consumed per data point by different AAs are plotted in Figure 7 and 8. Here we generalized random multivariate standard normal d dimensional n points, and computed the HD of each sample point and then determined the average time spent (seconds) per data point. We considered 7 different dimensions d, they are $d = [10, 20, 40, 80, 100, 150, 200]$ and 6 sample sizes $n = [50, 100, 200, 300, 400, 500]$. For smaller n, the results are obtained by averaging results from many replications (hundreds or thousands). Inspecting the Figures reveals that the average times consumed by the algorithms are *roughly* linear in dimension d and logarithmically quadratic in sample size n. Again here one has to *take randomness into consideration.*

There are two seemingly surprising and yet puzzling phenomena in the Figures. First, for fixed n, when d increases (from 10), the average consumed time per data point contrarily decreases in many cases. Second, when adding 1000 more balls into the intersection process (AA2), the average consumed time per data point decreases reversely ($d \neq 10$) compared with AA1.

The first phenomenon can be explained by the fact that for a fixed n, data points in the lower (smaller) d are more "dense" (or less "sparse") than in higher (larger) d, where points are more likely lying on the boundary, i.e. they are more likely to have lower depth m/n close to $1/n$ and therefore need fewer loops (time) to stop (get depth).

For the second phenomenon, it is due to the fact that AA1 tends to *overestimate* depth as m^*/n, whereas AA2 tends to quickly get exact depth m/n. Here $\delta = m^* - m > 0$ tends not to be a small integer, therefore $\delta n^2 \log(n)$ dominates the extra term $N_b^* n \log(n)$ in the time complexity of AA2, where N_b^* is much less than N_b ($= 1000$) since AA2 terminates when

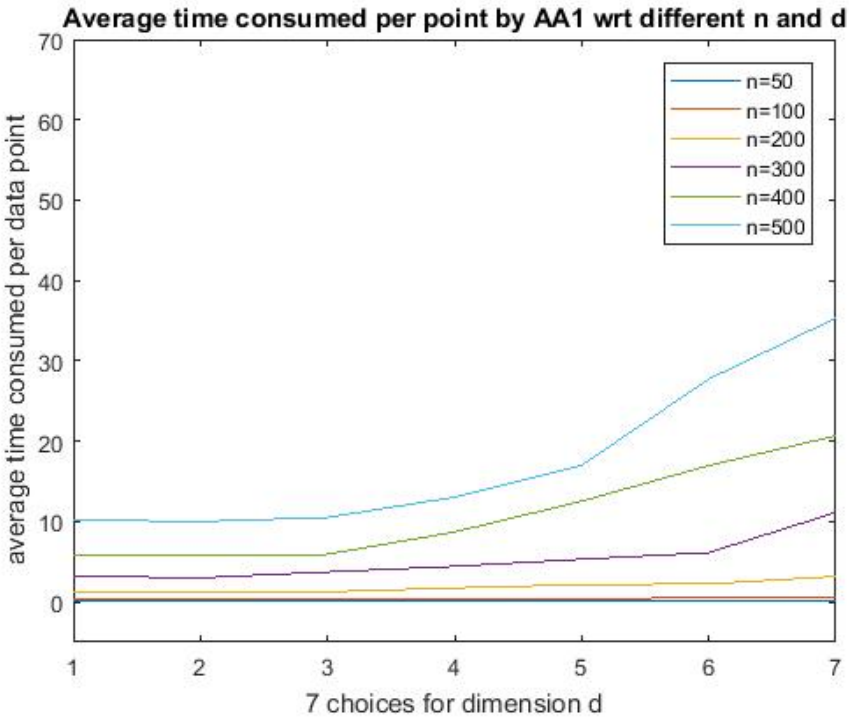

Figure 7: Average time consumed (in seconds) per multivariate standard normal point wrt different n and $d = [10, 20, 40, 80, 100, 150, 200]$ by AA1.

the stopping condition is satisfied and does not necessarily exhaust all N_b balls.

Remarks 4.4

(I) Assuming that X is in GP and $n \leq d+1$, then one can conclude that the HD of all sample points is $1/n$. In this sense, one just needs to focus on the cases $n > d+1$ in the simulations. The other cases are kept in the figures for the purpose given below.

(II) It is impossible to compute the HD in most combinations above using exact algorithms (or even most existing approximate algorithms). However, we checked the accuracy of AA2 by considering different cases of $n \leq d+1$ and found that it was capable of getting the exact depth $1/n$ very fast in all the cases considered. For its performance in other cases, see Table 6.

(III) For comparison purpose, we list results in Table 6 below. Note that RS98 ($n \leq d$) and CH13 ($n \leq d-2$) are no longer applicable, whereas CN08 and AA1 and AA2 make no restrictions on d and n.

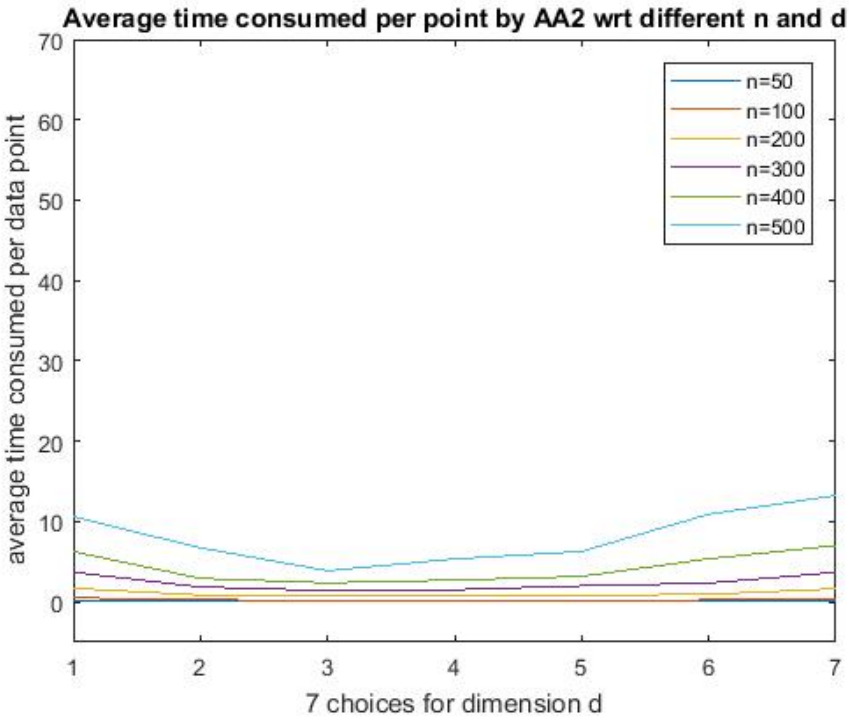

Figure 8: Average time consumed (in seconds) per multivariate standard normal point wrt different n and $d = [10, 20, 40, 80, 100, 150, 200]$ by AA2.

Thanks to AA2 which is efficient (very fast) yet accurate (produced the smallest possible HD, $1/n$, thus exact depth), we luckily have a benchmark for the evaluation of the performance of other competitors in the Table, otherwise no exact algorithms could be invoked to handle all the cases.

Inspecting the table immediately reveals that AA2 can very fast get exact HD in all the combinations whereas other leading competitors fail to get exact HD in most cases (except RS98 which can get most exact HD but costs *much more* time than AA2 though). 'NA' means that the corresponding algorithm collapses or fails to produce results within 96 hours.

To compute the HD of a point w.r.t. a data set in $\mathbb{R}^d$, RS98 and CH13 run in the time of $O(N_d N_r(d^3 + dn))$ and $O(N_d N_r(n \log n + 2dn))$ respectively, while AA2 and CN08 run in the time of $O(N_r((N_b + n)n \log n + (d + m)n^2))$ and $O(N_d N_r dn)$ respectively, where $N_d = 10^3$: the random directions used in RS98, CN08, CH13; $N_b = 10^3$: the balls used in AA2; $N_r = 50$: the interior repetitions in all methods; and m/n is the depth of point to be computed. These results are manifested in Table 6. Only AA2 can freely

handle the ultra-high dimensional case (e.g. $(200 \le d \le 800, n = 800)$, etc.).

Table 6. Performance of approximate algorithms based on standard normal points with various d and n in 10 replications. Table entries $(a; b)$ are a=EMSE, and b=time consumed (seconds) *per replication*.

AA2	AA1	RS98	CN08	CH13
		$d = 20, n = 18$		
(0; .3020)	(.6358; 15.10)	(NA; NA)	(0; 495.3)	(16.06; 543.5)
		$d = 20, n = 22$		
(0; .7101)	(.8616; 28.65)	(0; 857.4)	(0; 622.8)	(0; 1273.9)
		$d = 20, n = 34$		
(0; 3.247)	(.9585; 100.2)	(0; 936.9)	(3.4e-4; 977.0)	(0; 2130.9)
		$d = 20, n = 80$		
(0; 36.51)*	(1.268; 109.9)	(0; 3607.7)	(.0088; 2281.5)	(.1420; 2230.8)
		$d = 20, n = 100$		
(0; 108.1)*	(1.853; 1997)	(7e-5; 5565.8)	(.0307; 2733.7)	(.3308; 2822.1)
		$d = 100, n = 200$		
(0; 154.9)	(.8782; 19175)	(NA; NA)	(NA; NA)	(NA; NA)

* entries with $N_r = 100$.

For fixed d, when n gets larger, one should increase N_r, say, to 100, depending on the accuracy wanted and time allowed. In fact, for $d = 20$ and $n = 80$ or 100, AA2 used $N_r = 100$ to get the exact depth. Increasing N_d may also improve the accuracy of RS98, CN08, and CH13 with the price of paying intolerably long times (one cannot afford to increase N_r for them as well). This is the reason why the overall replication is just 10 times. In our simulation, the *same data sets* are generated and used for all methods.

When $n \le (d+1)$, the exact HD of each point *in* ***any*** *GP data set* is $1/n$. Can this still hold when $n > d+1$? The answer is negative. This nevertheless

does not exclude that it still can happen for a *specifically* generated random data which is in GP. In fact, it happens for a wide range of n. For example, in our simulation, when $d = 20$, for all n selected, the simulated normal points all have HD $1/n$ (for $n = 22$ this is verified in 10^4 replications).

This is a rather unexpected result. It gives an astonishing perspective of the position of *sparse data* in high dimensions. That is, all simulated points lie at the corners of the convex hull formed by general position data in high dimensions. In fact, when $n = 300$ the phenomenon above still happened.

Remarks 4.5

(I) Discussions and simulations above assume that the underlying data sets are in GP (indeed the simulated points are in GP with probability one). In practice, one may have non-GP data sets. Authors in the literature often state that assume (w.l.o.g.) that the data sets are in GP. They claim that one can always employ some small perturbation to convert the data set to one that is in GP.

(II) Let's first assume that the technique works. The real issue here is the complexity of checking whether the data set is in GP. It is an $O(n^d)$ problem that is as costly as computing the halfspace depth.

(III) The observation above calls for procedures *robust* towards the position of the data set, i.e. the procedures that can treat both in GP or not in GP data. What about the robustness of procedures discussed so far?

(IV) It turns out that the ball-based approach (i.e.Algorithm 1 and 2)(see Example 3.1) and CN08 is robust against the position of the data set, while the traditional hyperplane (determined by d points) and projection to its orthogonal unit vector approaches (including RS98 and CH13) will fail for some non-GP data sets.

5 Concluding Remarks

A closed balls based approach for the definition and therefore computation of Tukey's halfspace depth is proposed and discussed. An exact computing algorithm and approximate ones are presented. The exact algorithm is based on $\binom{n}{(n-m+1)}$ minimum enclosing balls (MEBs), where m/n is the halfspace depth the algorithm is computing. Therefore, like all other existing exact algorithms based on the classical polyhedral cones approach, it is

not practically feasible for large n.

Approximate algorithms thereafter proposed are much more promising, especially AA2 which is efficient (fast) and yet much more reasonably accurate compared with other leading competitors. To compute HD for large d (especially ultra-high d) and large n, AA2 as one of the best choices among the leading competitors has been demonstrated whereas AA1 can serve as a good candidate of tools for quickly detecting outliers in those scenarios.

The exact algorithm proposed here is anticipated to be improved very soon by researchers with computational geometry and combinatorial techniques and other tricks (such as dynamic maintenance and different search methods and data structures). For approximate algorithm AA2, issues like (i) how many balls are needed to be added and how many interior replications are needed (ii) how to determine the optimal range of the center of these balls are needed to be further addressed.

Recently, another type of feasible approximate algorithm was proposed by SZ16, where multiple try Metropolis algorithm combining with a simulated annealing algorithm is employed in the sampling scheme to speed up the searching for a global optimization solution. SZ16 however needs tuning parameters (at least four) and is much more sophisticated than AA2 which just has parameter N_b (the number of balls added, usually $N_b = 1000$) and N_r (the number of interior replication, usually $N_r = 100$) that need to be tuned and is more feasible for lay people in practice.

The essence of SZ16 is that it shares the dividend of latest developments in sampling techniques (e.g. MCMC) and stochastic optimization schemes, etc. while AA2 benefits directly from the ball-based new definition. Both procedures depend on d linearly and are robust against position of data sets. This begs for a further comparison investigation into the two and which deserves to be pursued elsewhere.

Acknowledgments

The author thanks Dr. Shao, W. for stimulating discussions and his help on matlab codes for RS98 and CH13. Thanks also go to his dear colleague Dr. Stapleton, J. for his careful English proofreading of the manuscript. The author greatly appreciates the thoughtful and constructive remarks and suggestions of a referee, which led to improvements in the paper.

References

[1] P. Afshani, and T. M. Chan. "On approximate range counting and depth", *Discrete and Computational Geometry* 42, 3-21, 2009.

[2] M. Bogićević and M. Merkle, "ABCDepth: efficient algorithm for Tukey depth", arXiv:1603.05609v1, 2016.

[3] 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.

[4] D. Bremner, K. Fukuda, and A. Marzetta "Primal-Dual Methods for Vertex and Facet Enumeration", *Discrete and Computational Geometry*, 20, pp. 333–357, 1998.

[5] D. Bremner, K. Fukuda, and V. Rosta. "Primal-dual algorithms for data depth", In: Liu, R., Serfling, R., and Souvaine, D. (eds.), *Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, American Mathematical Society, Providence RI, 171-194, 2006.

[6] M. A. Burr, E. Rafalin, and D. L. Souvaine, "Dynamic Maintenance of Half-Space Depth for Points and Contours", arXiv:1109.1517 [cs.CG], 2011. *In 14th Annual Fall Workshop on Computational Geometry*, pp.3-4, 2005.

[7] V. E. Brunel, "Concentration of the empirical level sets of Tukeys halfspace depth", arXiv:1605.09456v1,2016.

[8] T. M. Chan, "An optimal randomized algorithm for maximum Tukey depth", In *Proc. ACM-SIAM Sympos. Discrete Algorithms*, 430–436, 2004.

[9] D. Chen, P. Morin, and U. Wagner. "Absolute approximation of Tukey depth: Theory and experiments", *Computational Geometry* 46, pp. 566-573, 2013.

[10] J. A. Cuesta-Albertos, and A. Nieto-Reyes, "The random Tukey depth", *Computational Statistics and Data Analysis* 52, pp. 4979–4988, 2008.

[11] R. Dyckerhoff, and P. Mozharovskyi, "Exact computation of the halfspace depth", *Computational Statistics and Data Analysis*, 98,19-30, 2016.

[12] K. Fischer. "Smallest enclosing ball of balls", Diploma thesis, Institute of Theoretical Computer Science, ETH Zurich, 2001.

[13] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, Berlin, Heidelberg, 1987.

[14] K. Fischer and B. Grtner. "The Smallest Enclosing Ball of Balls: Combinatorial Structure and Algorithms", *In Proc. 19th annual ACM Symposium on Computational Geometry (SCG)*, pp. 292-301, 2003.

[15] A. K. Ghosh and P. Chaudhuri. "On data depth and distribution free discriminant analysis using separating surfaces". *Bernoulli* 11, 127, 2005

[16] M. Genest, J. C. Masse, and J. F. Plante, "depth: Nonparametric Depth Functions for Multivariate Analysis", R package version 2.1-1, URL http://CRAN.R-project.org/package=depth, 2017

[17] I. M. Johnstone, and D. M. Titterington, "Statistical challenges of high-dimensional data", *Philosophical Transactions of the Royal Society*, ADOI: 10.1098/rsta.2009.0159, 2009.

[18] M. Hallin, D. Paindaveine, and M. Šiman. "Multivariate quantiles and multiple-output regression quantiles: From L1-optimization to halfspace depth", *The Annals of Statistics* 38, 635-669, 2010.

[19] J. L. Hodges. "A bivariate sign test", *The Annals of Mathematical Statistics* 26, 523-527, 1955.

[20] T. Johnson, I. Kwok, and R. Ng. "Fast computation of 2- dimensional depth contours", *In: Agrawal, R., and Stolorz, P. (eds.), Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, AAAI Press, New York, 224-228, 1998.

[21] L. Kong, and I. Mizera. "Quantile tomography: Using quantiles with multivariate data", *Statistica Sinica* 22, 1589-1610, 2012.

[22] J. Kuelbs, and J. Zinn, "Limit Theorems for Quantile and Depth Regions for Stochastic Processes", Accepted in High dimensional probability VII-Progress in Probabiliy, 2017+a.

[23] J. Kuelbs, and J. Zinn, "Convergence of Quantile and Depth Regions", Accepted in Special Issue of SPA, 2017+b.

[24] P. Kumar, J. S. B. Mitchell, E. A. Yldrm, "Approximate minimum enclosing balls in high dimensions using core-sets", *Journal of Experimental Algorithmics (JEA)*, 8, 1.1, 2003.

[25] J. Li, J. A. Cuesta-Albertos, R. Y. Liu. "DD-Classifier: Nonparametric Classification Procedure Based on DD-Plot". *J. Amer. Statist. Assoc*, 107, 737753, 2012

[26] R. Y. Liu. "On a notion of data depth based on random simplices", *The Annals of Statistics* 18, 405-414, 1990.

[27] R. Y. Liu. "Data depth and multivariate rank tests", *In Y. Dodge (ed.), L1-Statistical Analysis and Related Methods*, North-Holland, Amsterdam, 279-294, 1992.

[28] R. Y. Liu, J. M. Parelius, and K. Singh. "Multivariate analysis by data depth: Descriptive statistics, graphics and inference", *The Annals of Statistics* 27, 783-858. With discussion, 1999.

[29] R. Y. Liu and K. Singh "A quality index based on data depth and multivariate rank tests". *J. Amer. Statist. Assoc.* 88:252260, 1993

[30] X. Liu, and Y. Zuo. "Computing halfspace depth and regression depth", *Communications in Statistics Simulation and Computation*, 43, 969-985, 2014.

[31] R. A. Maronna, R. D. Martin, V. J. Yohai. "Robust Statistics, Theorey and Methods", New York: John Wiley & Sons Ltd, 2006.

[32] M. Merkle, "Jensens inequality for multivariate medians", *J. Math. Anal. Appl.* 370, pp. 258-269, 2010.

[33] K. Miller, S. Ramaswami, P. Rousseeuw, J. A. Sellarès, D. Souvaine, I. Streinu, and A. Struyf, "Efficient computation of location depth contours by methods of computational geometry", *Statistics and Computing*, 13, 153-62, 2003.

[34] I. Mizera, "On Depth and Deep Points: a Calculus", *The Annals of Statistics*, 30(6), 1681-1736, 2002.

[35] K. Mosler. *Multivariate Dispersion, Central Regions and Depth: The Lift Zonoid Approach*. Springer, New York, 2002.

[36] K. Mosler, T. Lange, and P. Bazovkin. "Computing zonoid trimmed regions of dimension $d > 2$", *Computational Statistics and Data Analysis* 53, 2500-2510, 2009.

[37] D. Paindaveine, and M. Šiman. "Computing multiple-output regression quantile regions", *Computational Statistics and Data Analysis* 56, 840-853, 2012a

[38] D. Paindaveine, and M. Šiman. "Computing multiple-output regression quantile regions from projection quantiles". *Computational Statistics* 27, 29-49, 2012b

[39] O. Pokotylo, P. Mozharovskyi and R. Dyckerhoff. "Depth and depth-based classification with R-package ddalpha", *arXiv:1608.04109*, 2016

[40] P. J. Rousseeuw, and I. Ruts. "Algorithm AS 307: Bivariate location depth", *Journal of the Royal Statistical Society. Series C: Applied Statistics* 45, 516-526, 1996.

[41] P. J. Rousseeuw, and I. Ruts. "Constructing the bivariate Tukey median", *Statist. Sinica* 8, 827-839, 1998.

[42] P. J. Rousseeuw, and A. Struyf. "Computing location depth and regression depth in higher dimensions" *Statistics and Computing* 8, 193-203, 1998.

[43] I. Ruts, and P. J. Rousseeuw. "Computing depth contours of bivariate point clouds", *Computational Statistics and Data Analysis* 23, 153-168,1996a

[44] I. Ruts, and P. J. Rousseeuw. "Isodepth: A program for depth contours", *In: A. Prat(ed.), Proceedings in Computational Statistics, COMPSTAT*, Physica, Heidelberg, 441-446, 1996b

[45] G. A. F. Seber. "Multivariate Observations", New York: John Wiley & Sons, Inc., 1984.

[46] M. Šiman (2011). "On Exact Computation of Some Statistics Based on Projection Pursuit in a General Regression Context", *Communications in Statistics - Simulation and Computation*, 40(6), 948-956, 2010.

[47] W. Shao, and Y. Zuo. "Computing the halfspace depth with multiple try algorithm and simulated annealing algorithm", *Preprint*, 2016.

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

[49] E. Welzl. "Smallest enclosing disks (balls and ellipsoids)", *In H. Maurer, editor, New Results and New Trends in Computer Science*, volume 555 of Lecture Notes Comput. Sci., pp. 359-370. Springer-Verlag, 1991.

[50] J. W. Tukey. "Mathematics and the picturing of data", *In: James, R.D. (ed.), Proceeding of the International Congress of Mathematicians*, Vancouver 1974 (Volume 2), Canadian Mathematical Congress, Montreal, 523-531, 1975.

[51] Y. Zuo. "Multi-dimensional trimming based on projection depth", *The Annals of Statistics* 34(5), 2211-2251, 2006.

[52] Y. Zuo. "Projection-based depth functions and associated medians", *The Annals of Statistics* 31, 1460-1490, 2003.

[53] Y. Zuo, and S. Lai. "Exact computation of the bivariate projection depth and Stahel-Donoho estimator." *Computational Statistics and Data Analysis* , 53(3), 1173-1179, 2011.

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

[55] Y. Zuo, and R. Serfling. "Structural Properties and Convergence Results for Contours of Sample Statistical Depth Functions", *The Annals of Statistics*. 28(2): 483-499, 2000b.