Exact computation of the halfspace depth

Rainer Dyckerhoff Pavlo Mozharovskyi

Institute of Econometrics and Statistics, University of Cologne
Albertus-Magnus-Platz, 50923 Cologne, Germany

November 24, 2014

Abstract

We suggest a theoretical framework for computing the exact value of the halfspace depth of a point $\boldsymbol{z}$ w.r.t. a data cloud $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ of n points in arbitrary dimension. Based on this framework a whole class of algorithms can be derived. In all of these algorithms the depth is calculated as the minimum over a finite number of depth values w.r.t. proper projections of the data cloud. Three variants of this class are studied in more detail. All of these algorithms are capable of dealing with data that are not in general position and even with data that contain ties. As our simulations show, all proposed algorithms prove to be very efficient.

Keywords: Tukey depth; Exact algorithm; Projection; Combinatorial algorithm; Orthogonal complement.

1 Introduction

In 1975 John W. Tukey suggested a novel way of ordering multivariate data. He proposed to order the data according to their centrality in the data cloud. To achieve this, he defined what is nowadays known as *halfspace depth* (also called Tukey depth or location depth). Motivated by his proposal many notions of data depth have been proposed in the last decades, e.g., the simplicial

depth (Liu, 1988, 1990), the projection depth (Stahel, 1981, Donoho, 1982, Liu, 1992) and the zonoid depth (Koshevoy & Mosler, 1997).

Consider a data cloud $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ with data points $\boldsymbol{x}_i \in \mathbb{R}^d$. Conditioned on it, a statistical depth function assigns to an arbitrary point $\boldsymbol{z} \in \mathbb{R}^d$ its degree of centrality $\boldsymbol{z} \mapsto D(\boldsymbol{z}|\boldsymbol{X}) \in [0, 1]$.

The halfspace depth is determined as the smallest fraction of data contained in a closed halfspace containing $\boldsymbol{z}$. This classical depth function, based on ideas of Tukey (1975) and later developed by Donoho & Gasko (1992), is one of the most important depth notions and is historically the first one. It possesses a number of attractive properties, such as affine invariance, monotonicity on rays from any deepest point, quasiconcavity and upper semicontinuity, studied in Zuo & Serfling (2000), Dyckerhoff (2004), Mosler (2013). The halfspace depth determines uniquely the empirical distribution (Koshevoy, 2002), and takes a finite number of values in the interval from 0 (for the points that lie beyond the convex hull of the data) to its maximum value (1/2 if all data points are different from $\boldsymbol{z}$), increasing by a multiple of $1/n$. By its nature it has a high breakdown point.

The task of computation of the halfspace depth has a direct connection to the regression depth of Rousseeuw & Hubert (1999) and to the risk-minimizing separating hyperplane in classification. It coincides with the densest hemisphere problem (Johnson & Preparata, 1978), which is of non-polynomial complexity. For this reason a great part of the literature on the halfspace depth considers its computational aspects. Below, we give an overview of the history of its exact (in Section 1.1) and approximate (in Section 1.2) computation.

1.1 Previous approaches to calculation of the halfspace depth

The idea of the halfspace depth has been introduced in the conference paper by Tukey (1975). A similar mechanism of cutting by hyperplanes (lines in the two-dimensional case) has also been used for a bivariate sign test by Hodges (1955). During the last decades, a variety of attempts have been taken to compute the halfspace depth and its trimmed regions.

Rousseeuw & Ruts (1996) pioneered in exactly calculating the halfspace depth for bivariate data clouds and constructing its contours (Ruts & Rousseeuw, 1996a,b) exploiting the idea of a circular sequence (Edelsbrunner,

1987). Here, the depth of a point is computed with complexity $O(n \log n)$, and a single depth region is constructed with complexity $O(n^2 \log n)$, both essentially determined by the complexity of the QUICKSORT procedure. Johnson *et al.* (1998) suggest to account for a small subset of points only when constructing the first l depth contours, which yields a better complexity for small l (algorithm FDC). The halfspace depth describes a data cloud by a finite number of depth contours. Miller *et al.* (2003) compute all these with complexity $O(n^2)$, by which the depth of a single point can be calculated with complexity $O(\log^2 n)$ afterwards.

Rousseeuw & Struyf (1998) introduce an algorithm to compute the halfspace depth for $d = 3$ with complexity $O(n^2 \log n)$. They project points onto planes orthogonal to the lines connecting each of the points from $\boldsymbol{X}$ with $\boldsymbol{z}$ and then calculate the halfspace depth in these planes using the algorithm of Rousseeuw & Ruts (1996). Bremner *et al.* (2006) calculate the halfspace depth with a primal-dual algorithm by successively updating upper and lower bounds by means of a heuristic till they coincide. Bremner *et al.* (2008) design an output-sensitive depth-calculating algorithm that represents the task as two maximum subsystem problems for $d > 2$. The latter ones are then run in parallel.

When calculating depth trimmed regions, the first extension to $d > 2$ has been made by Mosler *et al.* (2009) for the zonoid depth (Koshevoy & Mosler, 1997, Mosler, 2002). Here, the unit sphere $\mathbb{S}^{d-1}$ is segmented according to direction domains — polyhedral cones defining the region's facets. These cones are all considered sequentially by wavelike spreading using the breadth-first search algorithm. The idea has been exploited in later algorithms for computing depth and depth regions, among others for the halfspace depth.

An interesting issue of updating the depth when points are added continuously to the data set is handled by Burr *et al.* (2011) for bivariate depth and depth contours.

Kong & Mizera (2012) employ direction quantiles defined as halfspaces corresponding to quantiles on univariate projections and prove their envelope to coincide with the corresponding halfspace depth trimmed region. Hallin *et al.* (2010) establish a direct connection between multivariate quantile regions and halfspace depth trimmed regions. The multivariate directional quantile for a given direction corresponds to a hyperplane that may carry a facet of a depth trimmed region. More than that, the authors define a polyhedral cone containing all directions yielding the same hyperplane; the union of the finite set of all these cones fills the $\mathbb{R}^d$. So, by the breadth-first search

algorithm a family of hyperplanes, each defining a halfspace, is generated, and the intersection of these halfspaces forms the halfspace depth trimmed region (see also Paindaveine & Šiman, 2012a,b).

Based on an idea similar to Hallin *et al.* (2010), Liu & Zuo (2014) compute the halfspace depth exactly by using a breadth-first search algorithm to cover $\mathbb{R}^d$ and QHULL to define the direction cones. In a most recent article Liu (2014) suggests another two algorithms for the exact computation of the halfspace depth, which seem to be very fast. One of these algorithms can be seen as a special case of the framework developed in the current paper.

1.2 Approximation of the halfspace depth

As even with the fastest available algorithms the exact calculation of the halfspace depth is very elaborate, and amounts to exponential complexity in n and d, one tries to save computation expenses by approximating the depth. Following Dyckerhoff (2004), the halfspace depth satisfies the weak projection property, i.e., it is the smallest achievable depth on all one-dimensional projections, and thus can be bounded from above by univariate depths.

Rousseeuw & Struyf (1998), when suggesting the algorithm computing the halfspace depth for $d = 3$, explored four algorithms differing in how the directions to project the data are generated. They propose to take a random subset of: (1) all lines connecting $\boldsymbol{z}$ and a point from $\boldsymbol{X}$, (2) all lines connecting two points from $\boldsymbol{X}$, (3) all lines normal to hyperplanes based on $\boldsymbol{z}$ and $d-1$ pairwise distinct points from $\boldsymbol{X}$, (4) all lines normal to hyperplanes based on d pairwise distinct points from X, and claim the last variant to work best. Cuesta-Albertos & Nieto-Reyes (2008) suggest to generate directions uniformly on $\mathbb{S}^{d-1}$. This method proves to be useful in classification. Afshani & Chan (2009) present a randomized data structure keeping the approximated depth value in some range of deviations from its real value.

The latter work of Chen *et al.* (2013) determines the number of tries needed to achieve a required precision exploiting the third approximation method of Rousseeuw & Struyf (1998). The authors also present its generalization by projecting $\boldsymbol{z}$ and $\boldsymbol{X}$ onto affine spaces of dimension greater than one. They report the approximated depth values to be exact in most of the experiments and the approximation errors never to be larger than $2/n$.

1.3 Proposal

In this paper, we suggest a theoretical framework for computing the halfspace depth, which yields a whole class of algorithms. For $k \in \{1, \ldots, d-1\}$, consider a tuple of k (out of n) data points, which together with $\boldsymbol{z}$ span an affine subspace of dimension k. For each such tuple, the data are projected onto the corresponding orthogonal complement, and the halfspace depth is computed as the sum of the depth in these two orthogonal subspaces. Further, for some fixed k, the halfspace depth is obtained as a minimum of the depths over all such tuples. All proposed algorithms are capable of dealing with data that are not in general position and even with ties.

In what follows, we first develop the theoretical framework in Section 2, leading to the main result stated in Theorem 2, which yields the above mentioned class of algorithms. Further, three algorithms for $k = 1, d-2, d-1$ are presented in Section 3. In Section 4, we discuss implementation issues and provide a speed comparison of the three algorithms for data in general and non-general position. Some further ideas concerning the application of the proposed techniques are discussed in Section 5.

We will use the following notation. The number of elements of a set I is denoted by $\#I$ and the complement of a set I by I^c. The linear hull of some points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{R}^d$ is denoted by $\operatorname{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. For a subspace U of $\mathbb{R}^d$, we denote the orthogonal complement of U by $U^\perp$.

2 Theory

The *halfspace depth* of a point $\boldsymbol{z} \in \mathbb{R}^d$ w.r.t. n data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ is defined by

$$\mathrm{HD}(\boldsymbol{z} \,|\, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \frac{1}{n} \min_{\boldsymbol{p} \neq \boldsymbol{0}} \#\{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i \geq \boldsymbol{p}'\boldsymbol{z}\} \,.$$

The halfspace depth of a point $\boldsymbol{z}$ is therefore simply the minimum fraction of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ contained in a closed halfspace containing $\boldsymbol{z}$. For simplicity, we will also use the shorter notation $\mathrm{HD}(\boldsymbol{z} \,|\, \boldsymbol{X})$.

The halfspace depth is affine invariant, in particular it is location invariant. Therefore

$$\mathrm{HD}(\boldsymbol{z} \,|\, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \mathrm{HD}(\boldsymbol{0} \,|\, \boldsymbol{x}_1 - \boldsymbol{z}, \ldots, \boldsymbol{x}_n - \boldsymbol{z}) \,,$$

which shows that we can restrict ourselves w.l.o.g. to the case, that the halfspace depth of the origin has to be computed. Further, it will be useful to consider the integer version of the halfspace depth,

$$\mathrm{nHD}(\boldsymbol{z} \,|\, \boldsymbol{X}) = n \cdot \mathrm{HD}(\boldsymbol{z} \,|\, \boldsymbol{X}) \in \mathbb{N}\,.$$

In this section we will assume, that $\mathbf{0} \notin \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. If some of the data points are equal to $\mathbf{0}$, then these points are removed from the data set and their number is simply added to the halfspace depth of the origin w.r.t. the remaining points. We will further assume that $\mathrm{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \mathbb{R}^d$.

Under the assumptions from above, the (integer) halfspace depth can be written as

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) = \min_{\boldsymbol{p} \neq \mathbf{0}} \#\{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i \geq 0\}\,.$$

A vector $\boldsymbol{p}_0 \neq \mathbf{0}$ is called *optimal for the dataset* $\boldsymbol{X}$, if

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) = \#\{i \,|\, \boldsymbol{p}_0'\boldsymbol{x}_i \geq 0\}\,.$$

We will use the following notation: If $\boldsymbol{p} \neq \mathbf{0}$, then

$$I_{\boldsymbol{p}}^+ = \{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i > 0\}\,, \quad I_{\boldsymbol{p}}^0 = \{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i = 0\}\,, \quad I_{\boldsymbol{p}}^- = \{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i < 0\}\,,$$

and the corresponding cardinalities are denoted by

$$n_{\boldsymbol{p}}^+ = \#I_{\boldsymbol{p}}^+\,, \quad n_{\boldsymbol{p}}^0 = \#I_{\boldsymbol{p}}^0\,, \quad n_{\boldsymbol{p}}^- = \#I_{\boldsymbol{p}}^-\,.$$

With this notation

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) = \min_{\boldsymbol{p} \neq \mathbf{0}} (n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0) = n - \max_{\boldsymbol{p} \neq \mathbf{0}} n_{\boldsymbol{p}}^-\,.$$

For a subset I of indices $\boldsymbol{X}_I$ denotes the data set $(\boldsymbol{x}_i)_{i \in I}$ of all data points with indices in I. If the data are not in general position, then the linear hull of some data points $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$ may contain additional data points. For a set $I = \{i_1, \ldots, i_k\}$ of indices we denote by

$$I^* = \{j \,|\, \boldsymbol{x}_j \in \mathrm{span}(\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k})\}$$

the set of all indices j such that $\boldsymbol{x}_j$ is contained in the linear hull of $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$. Obviously, $I \subset I^*$. The cardinality of I^* is denoted by n_{I^*}.

Proposition 1. *If $\boldsymbol{p} \neq 0$ is optimal for $\boldsymbol{X}$, then $I_{\boldsymbol{p}}^0 = \emptyset$, i.e., no data points lie on the boundary of the closed halfspace defined by $\boldsymbol{p}$.*

Proof: Assume that $I^0_{\boldsymbol{p}} \neq \emptyset$. Then, there is an index $i_0 \in I^0_{\boldsymbol{p}}$, i.e., $\boldsymbol{p}'\boldsymbol{x}_{i_0} = 0$. Now consider $\tilde{\boldsymbol{p}} = \boldsymbol{p} - \lambda \boldsymbol{x}_{i_0}$, where $\lambda > 0$. By choosing λ sufficiently small we can guarantee that $\tilde{\boldsymbol{p}}'\boldsymbol{x}_i < 0$ for all indices $i \in I^-_{\boldsymbol{p}}$. On the other hand $\tilde{\boldsymbol{p}}'\boldsymbol{x}_{i_0} = -\lambda\|\boldsymbol{x}_{i_0}\|^2 < 0$. Thus, $i_0 \in I^-_{\tilde{\boldsymbol{p}}}$. Consequently, $n^-_{\tilde{\boldsymbol{p}}} > n^-_{\boldsymbol{p}}$, which contradicts the optimality of $\boldsymbol{p}$. So, $I^0_{\boldsymbol{p}} = \emptyset$. □

We will need the following lemma, whose proof can be found in the appendix.

Lemma 1. *If* $\boldsymbol{p} \neq \boldsymbol{0}$ *and* $\dim \operatorname{span}(\boldsymbol{X}_{I^0_{\boldsymbol{p}}}) = l < d-1$, *then there exists* $\boldsymbol{x}_{i_0} \notin \operatorname{span}(\boldsymbol{X}_{I^0_{\boldsymbol{p}}})$ *and a vector* $\tilde{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{q} \neq \boldsymbol{0}$ *with* $\boldsymbol{q} \in \operatorname{span}(\boldsymbol{X}_{I^0_{\boldsymbol{p}}}, \boldsymbol{x}_{i_0})$ *such that*

$$I^0_{\tilde{\boldsymbol{p}}} = \left[I^0_{\boldsymbol{p}} \cup \{i_0\}\right]^* , \quad I^+_{\boldsymbol{p}} \subset I^+_{\tilde{\boldsymbol{p}}} \cup I^0_{\tilde{\boldsymbol{p}}} , \quad I^-_{\boldsymbol{p}} \subset I^-_{\tilde{\boldsymbol{p}}} \cup I^0_{\tilde{\boldsymbol{p}}} .$$

Further, $\dim \operatorname{span}(\boldsymbol{X}_{I^0_{\tilde{\boldsymbol{p}}}}) = l + 1$.

This means that if we change $\boldsymbol{p}$ to $\tilde{\boldsymbol{p}}$, then some of the data points that were contained in one of the open halfspaces defined by $\boldsymbol{p}$ are now on the boundary hyperplane defined by $\tilde{\boldsymbol{p}}$, whereas no data points did change sides.

Proposition 2. *If* $\boldsymbol{p} \neq 0$ *is optimal for* $\boldsymbol{X}$, *then for every* k, $1 \leq k < d$, *there are* k *linearly independent data points* $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$ *and a vector* $\tilde{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{q} \neq \boldsymbol{0}$, *where* $\boldsymbol{q} \in \operatorname{span}(\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k})$, *such that*

$$I^0_{\tilde{\boldsymbol{p}}} = \{i_1, \ldots, i_k\}^* , \quad I^+_{\boldsymbol{p}} \subset I^+_{\tilde{\boldsymbol{p}}} \cup I^0_{\tilde{\boldsymbol{p}}} , \quad I^-_{\boldsymbol{p}} \subset I^-_{\tilde{\boldsymbol{p}}} \cup I^0_{\tilde{\boldsymbol{p}}} .$$

Proof: The proposition follows from repeated application of Lemma 1. □

Proposition 3. *Let* $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$ *be linearly independent and* $\boldsymbol{p} \neq \boldsymbol{0}$ *such that* $I = \{i_1, \ldots, i_k\} \subset I^0_{\boldsymbol{p}}$. *Then,*

$$\mathrm{nHD}(\boldsymbol{0} \,|\, \boldsymbol{X}) \leq (n^+_{\boldsymbol{p}} + n^0_{\boldsymbol{p}}) - \left(n_{I^*} - \mathrm{nHD}(\boldsymbol{0} \,|\, \boldsymbol{X}_{I^*})\right) .$$

Proof: Let $\boldsymbol{0} \neq \boldsymbol{q} \in \operatorname{span}(\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k})$ be optimal for the dataset $\boldsymbol{X}_{I^*}$. Consider the direction $\tilde{\boldsymbol{p}} = \boldsymbol{p} + \lambda \boldsymbol{q}$ with $\lambda > 0$. We have

$$\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = \boldsymbol{p}'\boldsymbol{x}_i + \lambda \boldsymbol{q}'\boldsymbol{x}_i .$$

By choosing λ sufficiently small we can guarantee that $\tilde{\boldsymbol{p}}'\boldsymbol{x}_i$ and $\boldsymbol{p}'\boldsymbol{x}_i$ have the same sign for all $i \in I^+_{\boldsymbol{p}} \cup I^-_{\boldsymbol{p}}$. For $\boldsymbol{x}_i \in \boldsymbol{X}_{I^*}$ it holds $\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = \lambda \boldsymbol{q}'\boldsymbol{x}_i$. Since

$\boldsymbol{q}$ is optimal for $\boldsymbol{X}_{I^*}$, there are $n_{I^*} - \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*})$ data points in $\boldsymbol{X}_{I^*}$ for which $\boldsymbol{q}'\boldsymbol{x}_i < 0$. Therefore,

$$\begin{aligned}\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) &\leq \#\{i \,|\, \tilde{\boldsymbol{p}}'\boldsymbol{x}_i \geq 0\} \\ &= \#\{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i \geq 0\} - \#\{i \in I^* \,|\, \boldsymbol{q}'\boldsymbol{x}_i < 0\} \\ &= (n^+_{\boldsymbol{p}} + n^0_{\boldsymbol{p}}) - \left(n_{I^*} - \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*})\right).\end{aligned}$$

□

Proposition 4. *For every k, $1 \leq k < d$, there is a set $I = \{i_1, \ldots, i_k\}$ of indices and a vector $\tilde{\boldsymbol{p}} \neq \mathbf{0}$, such that $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$ are linearly independent, $I^0_{\tilde{\boldsymbol{p}}} = I^*$, and*

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) \geq (n^+_{\tilde{\boldsymbol{p}}} + n^0_{\tilde{\boldsymbol{p}}}) - \left(n_{I^*} - \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*})\right).$$

Proof: Let $\boldsymbol{p}_0 \neq \mathbf{0}$ be optimal for $\boldsymbol{X}$. Choose $\boldsymbol{q}$ and $\tilde{\boldsymbol{p}} = \boldsymbol{p}_0 + \boldsymbol{q}$ as in Proposition 2. For $i \in I^-_{\boldsymbol{p}_0} \cap I^0_{\tilde{\boldsymbol{p}}}$ it holds $\boldsymbol{p}_0'\boldsymbol{x} < 0$ and $\tilde{\boldsymbol{p}}'\boldsymbol{x} = 0$ which implies $\boldsymbol{q}'\boldsymbol{x} > 0$ and thus $(-\boldsymbol{q})'\boldsymbol{x} < 0$. Therefore,

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^0_{\tilde{\boldsymbol{p}}}}) \leq n^0_{\tilde{\boldsymbol{p}}} - \#(I^-_{\boldsymbol{p}_0} \cap I^0_{\tilde{\boldsymbol{p}}}).$$

Then,

$$\begin{aligned}\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) &= \#\{i \,|\, \boldsymbol{p}_0'\boldsymbol{x}_i > 0\} \\ &= \#\{i \,|\, \tilde{\boldsymbol{p}}'\boldsymbol{x}_i \geq 0\} - \#(I^-_{\boldsymbol{p}_0} \cap I^0_{\tilde{\boldsymbol{p}}}) \\ &\geq (n^+_{\tilde{\boldsymbol{p}}} + n^0_{\tilde{\boldsymbol{p}}}) - \left(n^0_{\tilde{\boldsymbol{p}}} - \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^0_{\tilde{\boldsymbol{p}}}})\right).\end{aligned}$$

Because of Proposition 2 it holds $I^0_{\tilde{\boldsymbol{p}}} = I^*$ and thus, $n^0_{\tilde{\boldsymbol{p}}} = n_{I^*}$, which completes the proof. □

For the following denote by $\mathcal{L}_k$ the set of all subsets I of order k of $\{1, \ldots, n\}$ such that the points $(\boldsymbol{x}_i)_{i \in I}$ are linearly independent.

Theorem 1. *For each k such that $1 \leq k < d$ it holds*

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) = \min_{I \in \mathcal{L}_k} \left[\left(\min_{\boldsymbol{p} \in X_I^\perp \setminus \{\mathbf{0}\}} \left(n^+_{\boldsymbol{p}} + n^0_{\boldsymbol{p}}\right) \right) - \left(n_{I^*} - \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*})\right) \right].$$

Proof: The proof follows immediately from the preceding two propositions. □

We will now show how

$$\min_{\boldsymbol{p}\in X_I^\perp\setminus\{\mathbf{0}\}} (n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0)$$

can be computed as the halfspace depth of a projection of the data points.

Let I be a subset of $\{1,\dots,n\}$ of order k such that the data points $\boldsymbol{x}_i$, $i \in I$, are linearly independent. Let further $\boldsymbol{a}_1,\dots,\boldsymbol{a}_{d-k}$ be a basis of the orthogonal complement of $\operatorname{span}(\boldsymbol{x}_{i_1},\dots,\boldsymbol{x}_{i_k})$ and $\boldsymbol{A}_I$ the matrix whose columns are the $\boldsymbol{a}_i$. Thus, every vector $\boldsymbol{p} \in \boldsymbol{X}_I^\perp$ is a linear combination of $\boldsymbol{a}_1,\dots,\boldsymbol{a}_{d-k}$, i.e., $\boldsymbol{p} = A_I\tilde{\boldsymbol{p}}$ for some $\tilde{\boldsymbol{p}} \in \mathbb{R}^{d-k}$ and the map $\mathbb{R}^{d-k} \to \boldsymbol{X}_I^\perp$, $\tilde{\boldsymbol{p}} \mapsto \boldsymbol{A}_I\tilde{\boldsymbol{p}} =: \boldsymbol{p}$, is a bijection. Since

$$\tilde{\boldsymbol{p}}'(\boldsymbol{A}_I'\boldsymbol{x}_i) \geq 0 \quad \iff \quad (\tilde{\boldsymbol{p}}'\boldsymbol{A}_I')\boldsymbol{x}_i \geq 0 \quad \iff \quad \boldsymbol{p}'\boldsymbol{x}_i \geq 0$$

we conclude

$$\begin{aligned}\min_{\boldsymbol{p}\in X_I^\perp\setminus\{\mathbf{0}\}} (n_{\boldsymbol{p}}^+ + n_{\boldsymbol{p}}^0) &= \min_{\boldsymbol{p}\in X_I^\perp\setminus\{\mathbf{0}\}} \#\{i \,|\, \boldsymbol{p}'\boldsymbol{x}_i \geq 0\} \\ &= \min_{\tilde{\boldsymbol{p}}\in\mathbb{R}^{d-k}\setminus\{\mathbf{0}\}} \#\{i \,|\, \tilde{\boldsymbol{p}}'(\boldsymbol{A}_I'\boldsymbol{x}_i) \geq 0\} \\ &= \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I'\boldsymbol{X})\,.\end{aligned}$$

A further simplification arises from the fact that all points $\boldsymbol{x}_i$, $i \in I^*$, are mapped to the origin by $\boldsymbol{A}_I'$. Therefore, these points can be removed from the data set and the halfspace depth is computed w.r.t. the data set $\boldsymbol{x}_i$, $i \in (I^*)^c$. Therefore,

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I'\boldsymbol{X}) = \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I'\boldsymbol{X}_{(I^*)^c}) + n_{I^*}\,,$$

and finally

$$\begin{aligned}\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) &= \min_{I\in\mathcal{L}_k} \left[\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I'\boldsymbol{X}) - \left(n_{I^*} - \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*})\right)\right] \\ &= \min_{I\in\mathcal{L}_k} \left[\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I'\boldsymbol{X}_{(I^*)^c}) + \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*})\right]\,.\end{aligned}$$

In the same way as above it can be shown that

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}_{I^*}) = \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{P}_I'\boldsymbol{X}_{I^*})\,,$$

where $\boldsymbol{P}_I = [\boldsymbol{x}_{i_1},\dots,\boldsymbol{x}_{i_k}]$. Therefore, the following theorem holds.

Theorem 2. *With the notation from above, for each* $1 \leq k < d$*, it holds*

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) = \min_{I \in \mathcal{L}_k} \Big[\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I' \boldsymbol{X}_{(I^*)^c}) + \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{P}_I' \boldsymbol{X}_{I^*}) \Big] .$$

Note, that for each subset I of k linearly independent data points, the data points fall in one of two categories: The points whose projections on the orthogonal complement of $\mathrm{span}(\boldsymbol{X}_I)$ are different from $\mathbf{0}$ and those who are equal to $\mathbf{0}$. The former points are taken into account by $\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I' \boldsymbol{X}_{(I^*)^c})$, whereas the latter ones are considered by $\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{P}_I' \boldsymbol{X}_{I^*})$. Usually there will be much more points of the first category than of the second one. The computation of $\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I' \boldsymbol{X}_{(I^*)^c})$ is done in dimension $d - k$, whereas the computation of $\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{P}_I' \boldsymbol{X}_{I^*})$ is done in dimension k. Thus, by the preceding theorem the calculation of one depth value in d-space is reduced to calculating many depth values in $(d-k)$-space (and in k-space). By choosing k we can control how much the dimension is reduced in each step. The price for a higher dimension reduction is that more subsets I have to be considered in that step.

An important special case of the algorithm arises, when the data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and $\mathbf{0}$ are in general position. In that case any linear subspace of dimension k, $1 \leq k < d$, contains at most k of the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Therefore, if $\#I = k$, then $I^* = I$ and $n_{I^*} = k$. Further, $\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{P}_I' \boldsymbol{X}_{I^*}) = 0$, which can be seen by choosing $\boldsymbol{p}$ such that $\boldsymbol{p}'(\boldsymbol{P}_I' \boldsymbol{x}_{i_1} - \boldsymbol{P}_I' \boldsymbol{x}_{i_j}) = 0$, $j = 2, \ldots, k$. Therefore, we get the following corollary of the above theorem.

Corollary 1. *If the data* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ *and* $\mathbf{0}$ *are in general position, then for each* $1 \leq k < d$*, it holds*

$$\mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{X}) = \min_{I \in \mathcal{L}_k} \mathrm{nHD}(\mathbf{0} \,|\, \boldsymbol{A}_I' \boldsymbol{X}_{I^c}) .$$

3 Algorithms

The result of the previous section gives rise to several algorithms. In these algorithms the dimensionality of the data is reduced at different rates. When the dimension is reduced to $d = 1$ or $d = 2$ specialized algorithms may be used. For the case $d = 1$ the standard algorithm of complexity $O(n)$ is used. For bivariate data the algorithm of Rousseeuw & Ruts (1996) with complexity $O(n \log n)$ (or any other algorithm with this complexity) may be used.

3.1 Combinatorial algorithm, $k = d - 1$

If we choose $k = d - 1$ this results in the so-called *combinatorial algorithm* (Algorithm 1). In this algorithm all hyperplanes defined by $d - 1$ linearly independent data points are considered. For each such hyperplane the data are projected in the direction normal to the hyperplane. Thus, the dimensionality is in one step reduced to dimension one. Only if there are more than $d - 1$ data points in the considered hyperplane (which can only occur when the data are not in general position), then for these data points $\boldsymbol{y}_i = \boldsymbol{P}_I' \boldsymbol{x}_i$ is calculated and the procedure nHD is recursively called for the data points $\boldsymbol{y}_{j_1}, \ldots, \boldsymbol{y}_{j_l}$. The algorithm is illustrated in Figure 1.

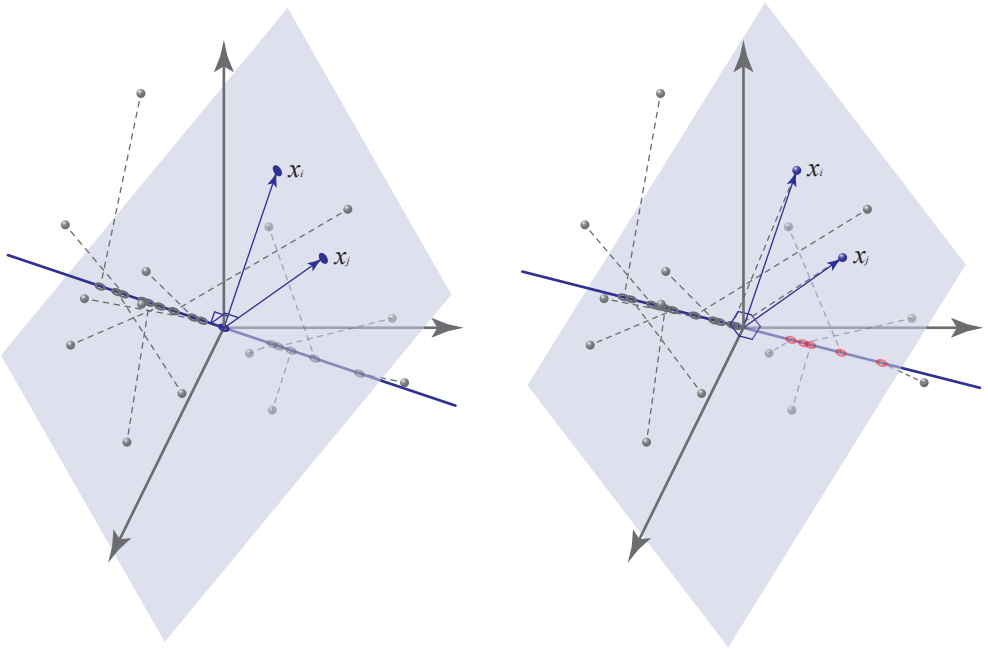

Figure 1: Illustration of the combinatorial algorithm, $k = d - 1$

In the case of data in general position the algorithm will never enter the recursion. Then, for each processed hyperplane the complexity of this algorithm is of order $O(n)$. Since there are $\binom{n}{d-1}$ subsets of $d-1$ data points, the overall complexity of the algorithm is $\binom{n}{d-1} O(n) = O(n^d)$.

Algorithm 1 Combinatorial algorithm, $k = d - 1$

function nHD_Comb$(d, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ ▷ Halfspace depth of 0
  if $d = 1$ **then return** nHD1$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$
  $n_{min} \leftarrow n$
  for each subset $I \subset \{1, \ldots, n\}$ of order $d - 1$ **do**
    if $(\boldsymbol{x}_i)_{i \in I}$ linearly independent **then**
      Compute $\boldsymbol{p}_I$ such that $\boldsymbol{p}_I' \boldsymbol{x}_i = 0$ for all $i \in I$
      for all $\boldsymbol{x}_j$ **do**
        $z_j \leftarrow \boldsymbol{p}_I' \boldsymbol{x}_j$ ▷ project data points in direction $\boldsymbol{p}_I$
      $n_{new} \leftarrow \min\Big\{\#\{z_j > 0\}, \#\{z_j < 0\}\Big\}$
      if $\#\{z_j = 0\} > d - 1$ **then**
        $\boldsymbol{P}_I \leftarrow \text{Matrix}[(\boldsymbol{x}_i)_{i \in I}]$
        for all indices j with $z_j = 0$ **do**
          $\boldsymbol{y}_j \leftarrow \boldsymbol{P}_I' \boldsymbol{x}_j$
        $n_{new} \leftarrow n_{new} +$ nHD_Comb$(d - 1, \boldsymbol{y}_{j_1}, \ldots, \boldsymbol{y}_{j_l})$
      if $n_{new} < n_{min}$ **then** $n_{min} \leftarrow n_{new}$
  return n_{min}

3.2 Combinatorial algorithm, $k = d - 2$

Another possibility is to use $k = d - 2$ (Algorithm 2). In that case the data points are directly projected into the 2-dimensional space. This has the advantage that for the projected points the algorithm of Rousseeuw & Ruts (1996) for the bivariate halfspace depth can be used. Since this algorithm has a complexity of $O(n \log n)$ and there are $\binom{n}{d-2}$ subsets of order $d - 2$, the complexity of this algorithm is of order $\binom{n}{d-2} O(n \log n) = O(n^{d-1} \log n)$. Thus, this algorithm has a better complexity than the naive combinatorial algorithm with $k = d - 1$. This combinatorial algorithm has been independently proposed in a forthcoming paper by Liu (2014).

3.3 Recursive algorithm, $k = 1$

The other extreme is the case, where we choose $d = 1$. This yields the so-called *recursive algorithm* (Algorithm 3). In this algorithm, in the outer loop all data points $\boldsymbol{x}_i$ are considered and the data are projected on the hyperplane orthogonal to $\boldsymbol{x}_i$. For the projected data points (with the exception of the

Algorithm 2 Combinatorial algorithm, $k = d - 2$

function NHD_COMB2($d, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$) ▷ Halfspace depth of 0
  if $d = 1$ **then return** NHD1($\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$)
  if $d = 2$ **then return** NHD2($\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$)
  $n_{min} \leftarrow n$
  for each subset $I \subset \{1, \ldots, n\}$ of order $d - 2$ **do**
    if $(\boldsymbol{x}_i)_{i \in I}$ linearly independent **then**
      Compute a basis $\boldsymbol{a}_1, \boldsymbol{a}_2$ of the orthogonal complement of $(\boldsymbol{x}_i)_{i \in I}$
      $\boldsymbol{A}_I \leftarrow \text{Matrix}[\boldsymbol{a}_1, \boldsymbol{a}_2]$
      $\boldsymbol{P}_I \leftarrow \text{Matrix}[\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{d-2}}]$
      for all $\boldsymbol{x}_j$ **do**
        if $\boldsymbol{A}_I' \boldsymbol{x}_j \neq \boldsymbol{0}$ **then** $\boldsymbol{y}_j \leftarrow \boldsymbol{A}_I' \boldsymbol{x}_j$
          else $\boldsymbol{z}_j \leftarrow \boldsymbol{P}_I' \boldsymbol{x}_j$
      $l \leftarrow \#\{j : \boldsymbol{A}_I' \boldsymbol{x}_j \neq \boldsymbol{0}\}$
      $n_{new} \leftarrow$ NHD2($\boldsymbol{y}_{j_1}, \ldots, \boldsymbol{y}_{j_l}$)
      if $n - l > d - 2$ **then**
        $n_{new} \leftarrow n_{new} +$ NHD_COMB2($d - 2, \boldsymbol{z}_{j_1}, \ldots, \boldsymbol{z}_{j_{n-l}}$)
      if $n_{new} < n_{min}$ **then** $n_{min} \leftarrow n_{new}$
  return n_{min}

data points that are a multiple of $\boldsymbol{x}_i$ and are thus mapped to the origin) the algorithm is called recursively. Thus, in each step the dimensionality is reduced only by one. The recursion stops, when $d = 2$ in which case the algorithm of Rousseeuw & Ruts (1996) is applied. Note, that the recursive algorithm can be viewed as a generalization of the algorithm for the case $d = 3$ in Rousseeuw & Struyf (1998). Figure 2 shows an illustration of the recursive algorithm.

In the recursive algorithm the depth w.r.t. d-variate data is computed as the minimum over n depths w.r.t. $(d-1)$-variate data. Therefore, the complexity for d-variate data is n times the complexity for $(d-1)$-variate data. Since the recursion is stopped when $d = 2$ in which case the $O(n \log n)$-algorithm of Rousseeuw & Ruts (1996) is used, this results in an overall complexity of $n^{d-2}\, O(n \log n) = O(n^{d-1} \log n)$. Note, that this remains true, even if the data are not in general position.

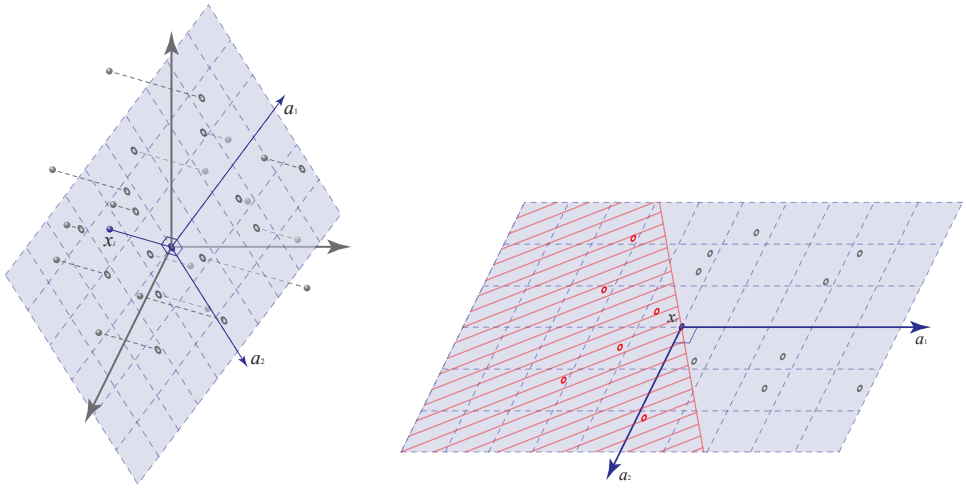

Figure 2: Illustration of he recursive algorithm, $k = 1$

Algorithm 3 Recursive algorithm, $k = 1$

function NHD_REC($d, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$) ▷ Halfspace depth of 0
  if $d = 1$ **then return** NHD1($\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$)
  if $d = 2$ **then return** NHD2($\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$)
  $n_{min} \leftarrow n$
  for all $\boldsymbol{x}_i$ **do**
    Compute a basis $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{d-1}$ of the hyperplane with normal $\boldsymbol{x}_i$
    $\boldsymbol{A}_I \leftarrow \text{Matrix}[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{d-1}]$
    for all $\boldsymbol{x}_j$ **do**
      if $\boldsymbol{A}_I'\boldsymbol{x}_j \neq \boldsymbol{0}$ **then** $\boldsymbol{y}_j \leftarrow \boldsymbol{A}_I'\boldsymbol{x}_j$
        else $z_j \leftarrow \boldsymbol{x}_i'\boldsymbol{x}_j$
    $l \leftarrow \#\{j : \boldsymbol{A}_I'\boldsymbol{x}_j \neq \boldsymbol{0}\}$
    $n_{new} \leftarrow$ NHD_REC($d-1, \boldsymbol{y}_{j_1}, \ldots, \boldsymbol{y}_{j_l}$)
      $+ \min\Big\{\#\{z_j > 0\}, \#\{z_j < 0\}\Big\}$
    if $n_{new} < n_{min}$ **then** $n_{min} \leftarrow n_{new}$
  return n_{min}

4 Implementation and experiments

All three variants of the algorithm, regarded in the previous section, have been implemented in C++, without using external libraries, thus maintaining maximum inter-system portability. The source code can be obtained upon request from the authors. For any k, the algorithm can be easily implemented in any programming environment. For $k = 1, ..., d-2$ the calculation of the orthogonal complement (which can easily be done, e.g., using the Gauss-Jordan method) and the routine `nHD2` by Rousseeuw & Ruts (1996) have to be implemented. For $k = d-1$ even the `nHD2`-routine is of no need.

The algorithms in the preceding section should not be called directly but rather be included in a wrapping function which does the necessary preprocessing of the data. In the preprocessing the point $\boldsymbol{z}$ is first subtracted from the data so that the halfspace depth of the origin has to be calculated. Second, all data points which are equal to the origin are removed from the data. Their number is stored and later added to the result of `nHD`. In an optional third step, the data could be scaled to have a norm of one. This does not change the halfspace depth, but has the advantage that all data have the same order of magnitude which should reduce numerical problems.

All algorithms based on Theorem 2 can be further improved by exiting the main loop as soon as n_{min} drops to zero, since in that case no further improvement is possible. However, this speed-up is data dependent and occurs only if the origin is outside the convex hull of the data. Therefore, we choose not to incorporate this modification into the algorithms that we used in our experiments (see below) to get stable computation times.

Due to the independent repetition of similar operations the algorithms based on Theorem 2 (for different values of k) possess high parallelization abilities, which grow with k. Clearly, for data in general position, the complexity of the algorithms for $1 \leq k \leq d-2$ is $O(n^{d-1} \log n)$, and is $O(n^d)$ for $k = d-1$. However, the exact execution time depends on the implementation of the single steps, routines, memory structures etc., and can differ in practice from the values reported in Tables 1 and 2. As we will see later in this section, each of the considered algorithms can show the best results (compared to the remaining two algorithms) for proper constellations of n and d. The algorithms' performance may differ a lot depending on whether the data are in general position or not. In all experiments, we used one kernel of the Intel Core i7-2600 (3.4 GHz) processor having enough physical memory.

First, we consider data $\boldsymbol{X}$ drawn randomly from a multivariate standard normal distribution $N\big(\mathbf{0}_d, \mathbf{I}_d\big)$, where the depth of the origin w.r.t. $\boldsymbol{X}$ is computed. Table 1 presents the execution times of the algorithms in seconds (in each cell the upper, middle, and lower lines correspond to $k = 1, 2, d-1$ respectively), averaged over 10 tries. Such a small number of tries is sufficient, as the execution times are extremely stable for all chosen values of n and d and differ by a few percents only. We vary d from 3 to 10 and increase $n = 10 \cdot 2^i$ with $i = 2, 3, ...$ till the execution time exceeds one hour. As one can see, because of the recurrent structure, for fixed n and with increasing d, the algorithm with $k = 1$ is outperformed by $k = d-2$, which is further outperformed by $k = d-1$. On the other hand, for fixed d and with increasing n, the algorithm with $k = d$ is outperformed by $k = 1$ and $k = d-2$ because of the better complexity of the latter algorithms. Comparing the algorithms with $k = 1$ and with $k = d-2$ the former one is superior for dimension $d = 3$ whereas the latter performs better when $d > 3$.

The designed framework allows for handling data for which the general position assumption is violated. The results for $\boldsymbol{X}$ distributed uniformly on $\{-2, -1, 0, 1, 2\}^d$ are presented in Table 2. As mentioned above (see also Corollary 1), for data in general position, if $k = d-1$ or $k = d-2$, no recursion is involved. If $k = d-1$, for data in non-general position, the recursive calls can increase the execution time, and, in general, the algorithmic complexity. Additionally, if n is not large enough (depending on d), the computation times depend heavily on the exact position of the points in $\mathbb{R}^d$ and become unstable. Therefore, for the algorithm with $k = d-1$ the median was taken instead of the mean when reporting the execution times for $d = 7, ..., 10$. The same effect occurs in the case $k = d-2$ as well, but the application of the two-dimensional routine (nHD2) designed by Rousseeuw & Ruts (1996) seems to compensate this increase in time by a quick handling of ties, especially when n gets larger. On the other hand, if $k = 1$, ties are rather an advantage, and the execution time of the algorithm decreases.

5 Conclusions

In this paper a class of combinatorial algorithms is presented, which calculate the halfspace depth of $\mathbf{0}$ w.r.t. $\boldsymbol{X}$ as the minimum over all combinations of k points, $k \in \{1, 2, ..., d-1\}$. For each combination, the depth is calculated as the sum of the depths of points lying in the affine space, spanned by these

k points and the origin, and its orthogonal complement. For each k, the algorithm can be easily implemented in any programming environment. None of the algorithms requires that the data be in general position or that the data have to be perturbed. For a given hardware and under the assumption of general position (or negligible violation of it), the computation times are stable for chosen d and n, and thus are predictable.

The empirical study shows that, for each of the presented algorithms, a pair (d, n) can be found where one algorithm outperforms the others in terms of speed. This suggests the development of a hybrid algorithm, which will choose k depending on d and n. This hybrid algorithm should be tuned for the corresponding implementation and hardware.

The developed framework may by extended to calculate further depths of combinatorial nature. Subsampling on the entire set of combinations for some (not necessarily equal) k's may be used for computation of approximate depth values.

Table 1: Execution times for data in general position, distributed as $N(\mathbf{0}_d, \mathbf{I}_d)$, averaged over 10 tries, in seconds. Variants of the algorithm with $k = 1, d-2, d-1$ are presented in the first, second, and third rows of each cell, respectively.

	$n = 40$	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
$d = 3$	0,000	0,000	0,000	0,011	0,047	0,184	0,780	3,191	13,173	54,512	224,919	932,628
	0,002	0,002	0,003	0,016	0,063	0,250	1,028	4,219	17,402	72,226	293,041	1210,018
	0,000	0,003	0,014	0,117	0,936	7,602	61,258	519,131	—	—	—	—
4	0,006	0,048	0,402	3,356	28,178	234,538	1958,967	—	—	—	—	—
	0,005	0,038	0,302	2,500	20,448	166,140	1357,806	—	—	—	—	—
	0,005	0,055	0,784	12,277	202,772	3288,079	—	—	—	—	—	—
5	0,205	3,617	62,544	1067,378	—	—	—	—	—	—	—	—
	0,055	0,952	16,110	269,388	—	—	—	—	—	—	—	—
	0,047	1,242	35,743	1113,671	—	—	—	—	—	—	—	—
6	7,322	274,638	—	—	—	—	—	—	—	—	—	—
	0,506	18,365	632,784	—	—	—	—	—	—	—	—	—
	0,392	21,629	1247,052	—	—	—	—	—	—	—	—	—
7	257,244	—	—	—	—	—	—	—	—	—	—	—
	3,603	278,096	—	—	—	—	—	—	—	—	—	—
	2,658	304,507	—	—	—	—	—	—	—	—	—	—
8	—	—	—	—	—	—	—	—	—	—	—	—
	21,404	3549,665	—	—	—	—	—	—	—	—	—	—
	14,315	3471,909	—	—	—	—	—	—	—	—	—	—
9	—	—	—	—	—	—	—	—	—	—	—	—
	107,068	—	—	—	—	—	—	—	—	—	—	—
	69,758	—	—	—	—	—	—	—	—	—	—	—
10	—	—	—	—	—	—	—	—	—	—	—	—
	439,424	—	—	—	—	—	—	—	—	—	—	—
	288,778	—	—	—	—	—	—	—	—	—	—	—

Table 2: Execution times for data in non-general position, distributed as $U(\{-2,-1,0,1,2\}^d)$, averaged over 10 tries, in seconds. Variants of the algorithm with $k = 1, d-2, d-1$ are presented in the first, second, and third rows of each cell, respectively. For $k = d-1$ and $d = 7, ..., 10$ the median is reported.

	$n = 40$	80	160	320	640	1280	2560	5120	10240	20480	40960	81920	163840
$d = 3$	0,000	0,002	0,002	0,009	0,036	0,139	0,530	2,114	8,325	33,075	132,113	529,965	2294,754
	0,000	0,002	0,005	0,013	0,052	0,198	0,773	3,139	12,594	49,857	195,232	786,086	3481,393
	0,002	0,019	0,188	2,113	26,056	383,915	—	—	—	—	—	—	—
4	0,005	0,045	0,375	3,055	24,564	196,412	1555,735	—	—	—	—	—	—
	0,005	0,036	0,291	2,352	18,532	141,400	1114,233	—	—	—	—	—	—
	0,203	5,474	234,871	—	—	—	—	—	—	—	—	—	—
5	0,202	3,560	60,547	1016,005	—	—	—	—	—	—	—	—	—
	0,063	1,059	17,376	282,512	—	—	—	—	—	—	—	—	—
	4,488	869,436	—	—	—	—	—	—	—	—	—	—	—
6	7,294	272,319	—	—	—	—	—	—	—	—	—	—	—
	0,570	—	—	—	—	—	—	—	—	—	—	—	—
	78,129	—	—	—	—	—	—	—	—	—	—	—	—
7	256,450	—	—	—	—	—	—	—	—	—	—	—	—
	4,335	315,036	—	—	—	—	—	—	—	—	—	—	—
	227,145	—	—	—	—	—	—	—	—	—	—	—	—
8	—	—	—	—	—	—	—	—	—	—	—	—	—
	24,190	—	—	—	—	—	—	—	—	—	—	—	—
	754,432	—	—	—	—	—	—	—	—	—	—	—	—
9	—	—	—	—	—	—	—	—	—	—	—	—	—
	144,395	—	—	—	—	—	—	—	—	—	—	—	—
	1745,995	—	—	—	—	—	—	—	—	—	—	—	—
10	—	—	—	—	—	—	—	—	—	—	—	—	—
	456,554	—	—	—	—	—	—	—	—	—	—	—	—
	1795,440	—	—	—	—	—	—	—	—	—	—	—	—

A Proof of Lemma 1

Denote by $\boldsymbol{y}_i$ the projection of $\boldsymbol{x}_i$ onto $\operatorname{span}(\boldsymbol{X}_{I_{\boldsymbol{p}}^0})^\perp$, the orthogonal complement of $\operatorname{span}(\boldsymbol{X}_{I_{\boldsymbol{p}}^0})$. Then, every data point can be uniquely represented as $\boldsymbol{x}_i = \boldsymbol{y}_i + \boldsymbol{d}_i$, where $\boldsymbol{y}_i \in \operatorname{span}(\boldsymbol{X}_{I_{\boldsymbol{p}}^0})^\perp$ and $\boldsymbol{d}_i \in \operatorname{span}(\boldsymbol{X}_{I_{\boldsymbol{p}}^0})$. Note, that $\boldsymbol{p}'\boldsymbol{x}_i = \boldsymbol{p}'\boldsymbol{y}_i$. Now, let

$$i_0 = \arg\min \left\{ \frac{|\boldsymbol{p}'\boldsymbol{y}_i|}{\|\boldsymbol{p}\|\|\boldsymbol{y}_i\|} \,|\, i \in I_{\boldsymbol{p}}^+ \cup I_{\boldsymbol{p}}^- \right\}$$

and

$$\epsilon = \frac{\boldsymbol{p}'\boldsymbol{y}_{i_0}}{\|\boldsymbol{p}\|\|\boldsymbol{y}_{i_0}\|} = \cos\alpha(\boldsymbol{p}, \boldsymbol{y}_{i_0})\,,$$

where $\alpha(\boldsymbol{p}, \boldsymbol{y}_{i_0})$ denotes the angle between $\boldsymbol{p}$ and $\boldsymbol{y}_{i_0}$. Now, define $\tilde{\boldsymbol{p}}$ by

$$\tilde{\boldsymbol{p}} = \boldsymbol{p} - \epsilon \frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|} \boldsymbol{y}_{i_0}\,.$$

Since $\boldsymbol{y}_{i_0} = \boldsymbol{x}_{i_0} - \boldsymbol{d}_{i_0} \in \operatorname{span}(\boldsymbol{x}_{i_0}, \boldsymbol{X}_{I_{\boldsymbol{p}}^0})$ it holds that $\tilde{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{q}$, where $\boldsymbol{q} \in \operatorname{span}(\boldsymbol{x}_{i_0}, \boldsymbol{X}_{I_{\boldsymbol{p}}^0})$. Further,

$$\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = \tilde{\boldsymbol{p}}'(\boldsymbol{y}_i + \boldsymbol{d}_i) = \boldsymbol{p}'\boldsymbol{y}_i + \boldsymbol{p}'\boldsymbol{d}_i - \epsilon \frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|} \boldsymbol{y}_{i_0}'\boldsymbol{y}_i - \epsilon \frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|} \boldsymbol{y}_{i_0}'\boldsymbol{d}_i\,.$$

Since $\boldsymbol{d}_i \perp \boldsymbol{p}$ and $\boldsymbol{d}_i \perp \boldsymbol{y}_i$ this reduces to

$$\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = \boldsymbol{p}'\boldsymbol{y}_i - \epsilon \frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|} \boldsymbol{y}_{i_0}'\boldsymbol{y}_i\,.$$

Now, consider the following cases:

If $i \in I_{\boldsymbol{p}}^+$, then

$$\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = \boldsymbol{p}'\boldsymbol{y}_i - \epsilon \frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|} \boldsymbol{y}_{i_0}'\boldsymbol{y}_i \geq 0\,,$$

since $\boldsymbol{p}'\boldsymbol{y}_i \geq |\epsilon|\|\boldsymbol{p}\|\|\boldsymbol{y}_i\|$ and $|\boldsymbol{y}_{i_0}'\boldsymbol{y}_i| \leq \|\boldsymbol{y}_{i_0}\|\|\boldsymbol{y}_i\|$. Thus, $I_{\boldsymbol{p}}^+ \subset I_{\tilde{\boldsymbol{p}}}^+ \cup I_{\tilde{\boldsymbol{p}}}^0$.

If $i \in I_{\boldsymbol{p}}^-$, then

$$\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = \boldsymbol{p}'\boldsymbol{y}_i - \epsilon \frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|} \boldsymbol{y}_{i_0}'\boldsymbol{y}_i \leq 0\,,$$

since $\boldsymbol{p}'\boldsymbol{y}_i \leq -|\epsilon|\|\boldsymbol{p}\|\|\boldsymbol{y}_i\|$ and $|\boldsymbol{y}_{i_0}'\boldsymbol{y}_i| \leq \|\boldsymbol{y}_{i_0}\|\|\boldsymbol{y}_i\|$. Thus, $I_{\boldsymbol{p}}^- \subset I_{\tilde{\boldsymbol{p}}}^- \cup I_{\tilde{\boldsymbol{p}}}^0$.

To conclude the proof, we now show that $I^0_{\tilde{\boldsymbol{p}}} = \left[I^0_{\boldsymbol{p}} \cup \{i_0\}\right]^*$. First, note that

$$\begin{aligned}
\tilde{\boldsymbol{p}}'\boldsymbol{x}_i &= \boldsymbol{p}'\boldsymbol{y}_i - \epsilon\frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|}\boldsymbol{y}'_{i_0}\boldsymbol{y}_i \\
&= \|\boldsymbol{p}\|\|\boldsymbol{y}_i\|\cos\alpha(\boldsymbol{p},\boldsymbol{y}_i) - \epsilon\frac{\|\boldsymbol{p}\|}{\|\boldsymbol{y}_{i_0}\|}\|\boldsymbol{y}_{i_0}\|\|\boldsymbol{y}_i\|\cos\alpha(\boldsymbol{y}_{i_0},\boldsymbol{y}_i) \\
&= \|\boldsymbol{p}\|\|\boldsymbol{y}_i\|\left[\cos\alpha(\boldsymbol{p},\boldsymbol{y}_i) - \cos\alpha(\boldsymbol{p},\boldsymbol{y}_{i_0})\cos\alpha(\boldsymbol{y}_{i_0},\boldsymbol{y}_i)\right] .
\end{aligned}$$

The term in brackets is zero if and only if

$$\frac{\cos\alpha(\boldsymbol{p},\boldsymbol{y}_i)}{\cos\alpha(\boldsymbol{p},\boldsymbol{y}_{i_0})} = \cos\alpha(\boldsymbol{y}_{i_0},\boldsymbol{y}_i) .$$

From the definition of ϵ the absolute value of the left hand side is at least one. So the only possible solution is, if $\boldsymbol{y}_i$ is a multiple of $\boldsymbol{y}_{i_0}$. Thus, $\tilde{\boldsymbol{p}}'\boldsymbol{x}_i = 0$ if and only if $\boldsymbol{y}_i$ is a multiple of $\boldsymbol{y}_{i_0}$, i.e.,

$$\boldsymbol{x}_i = \lambda\boldsymbol{y}_{i_0} + \boldsymbol{d}_i \in \operatorname{span}(\boldsymbol{y}_{i_0}, \boldsymbol{X}_{I^0_{\boldsymbol{p}}}) .$$

Therefore, $I^0_{\tilde{\boldsymbol{p}}} = \left[\{i_0\} \cup I^0_{\boldsymbol{p}}\right]^*$, as stated. Since $\boldsymbol{X}_{I^0_{\tilde{\boldsymbol{p}}}}$ is generated from $\boldsymbol{X}_{I^0_{\boldsymbol{p}}}$ by adjoining $\boldsymbol{x}_{i_0}$ (and maybe some other data points which are already in the linear hull of $\boldsymbol{X}_{I^0_{\tilde{\boldsymbol{p}}}} \cup \{\boldsymbol{x}_{i_0}\}$) its dimension equals $l+1$, which completes the proof of the lemma. □

References

Afshani, P. and Chan, T. (2009). On approximate range counting and depth. *Discrete and Computational Geometry* **42** 3–21.

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

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

Burr, M., Rafalin, E. and Souvaine, D.L. (2011). Dynamic maintenance of half-space depth for points and contours. `http://arxiv.org/abs/1109.1517`.

Chen, C., Morin, P. and Wagner, U. (2013). Absolute approximation of Tukey depth: theory and experiments. *Computational Geometry* **46** 566–573.

Cuesta-Albertos, J.A. and Nieto-Reyes, A. (2008). The random Tukey depth. *Computational Statistics and Data Analysis* **52** 4979–4988.

Donoho, D. (1982). *Breakdown properties of multivariate location estimators*. Ph.D. Qualifying Paper, Harvard University.

Donoho, D.L. and Gasko, M. (1992). Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics* **20** 1803–1827.

Dyckerhoff, R. (2004). Data depths satisfying the projection property. *AStA – Advances in Statistical Analysis* **88** 163–190.

Edelsbrunner, H. (1987). *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin Heidelberg.

Hallin, M., Paindaveine, D. and Šiman, M. (2010). Multivariate quantiles and multiple-output regression quantiles: from L_1-optimization to halfspace depth. *The Annals of Statistics* **38** 635–669.

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

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

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

Kong, L. and Mizera, I. (2012). Quantile tomography: using quantiles with multivariate data. *Statistica Sinica* **22** 1589–1610.

Koshevoy, G. (2002). The Tukey depth characterizes the atomic measure. *Journal of Multivariate Analysis* **83** 360–364.

Koshevoy, G. and Mosler, K. (1997). Zonoid trimming for multivariate distributions. *Annals of Statistics* **25** 1998–2017.

Liu, R. Y. (1988). On a notion of simplicial depth. *Proceedings of the National Academy of Sciences of the USA* **85** 1732–1734.

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

Liu, R. Y. (1992). Data depth and multivariate rank tests. In Y. Dodge, ed., L_1*-Statistical Analysis and Related Methods*. Elsevier.

Liu, X. (2014). Fast implementation of the Tukey depth. *Mimeo*.

Liu, X. and Zuo, Y. (2014). Computing halfspace depth and regression depth. *Communications in Statistics – Simulation and Computation* **43** 969–985.

Miller, K., Ramaswami, S., Rousseeuw, P., Sellarès, J.A., Souvaine, D., Streinu, I. and Struyf, A. (2003). Efficient computation of location depth contours by methods of computational geometry. *Statistics and Computing* **13** 153–162.

Mosler, K. (2002). *Multivariate Dispersion, Central Regions and Depth: The Lift Zonoid Approach*. Springer-Verlag, New York.

Mosler, K. (2013). Depth statistics. In: Becker, C., Fried, R., Kuhnt, S. (eds.), *Robustness and Complex Data Structures, Festschrift in Honour of Ursula Gather*, Springer-Verlag, Berlin, 17–34.

Mosler, K., Lange, T. and Bazovkin, P. (2009). Computing zonoid trimmed regions of dimension $d > 2$. *Computational Statistics and Data Analysis* **53** 2500–2510.

Paindaveine, D. and Šiman, M. (2012a). Computing multiple-output regression quantile regions. *Computational Statistics and Data Analysis* **56** 840–853.

Paindaveine, D. and Šiman, M. (2012b). Computing multiple-output regression quantile regions from projection quantiles. *Computational Statistics* **27** 29–49.

Rousseeuw, P.J. and Hubert, M. (1999). Regression depth. *Journal of the American Statistical Association* **94** 388–433.

Rousseeuw, P.J. and Ruts, I. (1996). Algorithm AS 307: bivariate location depth. *Journal of the Royal Statistical Society. Series C: Applied Statistics* **45** 516–526.

Rousseeuw, P.J. and Struyf, A. (1998). Computing location depth and regression depth in higher dimensions *Statistics and Computing* **13** 153–162.

Ruts, I. and Rousseeuw, P.J. (1996a). Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis* **23** 153–168.

Ruts, I. and Rousseeuw, P. (1996b). Isodepth: a program for depth contours. In: Prat, A. (Ed.), *Proceedings in Computational Statistics*, COMPSTAT 1996, Physica-Verlag, Heidelberg, 441–446.

Stahel, W. (1981). *Robuste Schätzungen: Infinitesimale Optimalität und Schätzungen von Kovarianzmatrizen*. Ph.D. Thesis, ETH Zürich.

Tukey, J.W. (1975). Mathematics and the picturing of data. In: James, R.D. (Ed.), *Proceeding of the International Congress of Mathematicians (Volume 2)*, Canadian Mathematical Congress, Vancouver, 523–531.

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