Approximate computation of projection depths

Rainer Dyckerhoff, Pavlo Mozharovskyi, Stanislav Nagy

Version of Record: https://www.sciencedirect.com/science/article/pii/S0167947320302577
Manuscript_679bfee9d407b27819b4556e70e4b665

Approximate computation of projection depths

Rainer Dyckerhoff[a], Pavlo Mozharovskyi[b,*], Stanislav Nagy[c]

[a] *Institute of Econometrics and Statistics, University of Cologne*
Albertus-Magnus-Platz, 50923 Köln, Germany
[b] *LTCI, Telecom Paris, Institut Polytechnique de Paris*
19 Place Marguerite Perey, 91120 Palaiseau, France
[c] *Faculty of Mathematics and Physics, Charles University*
Sokolovská 83, Praha 8, CZ-186 75, Czech Republic

Abstract

Data depth is a concept in multivariate statistics that measures the centrality of a point in a given data cloud in $\mathbb{R}^d$. If the depth of a point can be represented as the minimum of the depths with respect to all one-dimensional projections of the data, then the depth satisfies the so-called projection property. Such depths form an important class that includes many of the depths that have been proposed in literature. For depths that satisfy the projection property an approximate algorithm can easily be constructed since taking the minimum of the depths with respect to only a finite number of one-dimensional projections yields an upper bound for the depth with respect to the multivariate data. Such an algorithm is particularly useful if no exact algorithm exists or if the exact algorithm has a high computational complexity, as is the case with the halfspace depth or the projection depth. To compute these depths in high dimensions, the use of an approximate algorithm with better complexity is surely preferable. Instead of focusing on a single method we provide a comprehensive and fair comparison of several methods, both already described in the literature and original.

Keywords: data depth, projection property, approximate computation,

*Corresponding author
Email addresses: rainer.dyckerhoff@statistik.uni-koeln.de (Rainer Dyckerhoff), pavlo.mozharovskyi@telecom-paris.fr (Pavlo Mozharovskyi), nagy@karlin.mff.cuni.cz (Stanislav Nagy)

non-convex optimization, unit sphere, random search, grid search, simulated annealing, great circles, coordinate descent, Nelder-Mead
2010 MSC: 62G05, 62H12, 90C26

1. Introduction

A statistical depth function is a generalization of the concept of quantiles to multivariate data. Given a probability measure or a data sample in $\mathbb{R}^d$, the depth assigns to any point of the space $\mathbb{R}^d$ a real number, often scaled to $[0, 1]$, which characterizes its degree of centrality w.r.t. this distribution or data set. By providing a non-parametric, affine-invariant, and (often) robust multivariate ordering, data depth finds numerous applications, e.g., in descriptive statistics, statistical inference, or risk measurement to name only a few (Liu et al., 1999; Cascos, 2010). For more information on applications of data depth we refer to surveys Zuo and Serfling (2000) and Mosler (2013).

Application of the depth-based methods requires efficient algorithms for the computation of depths. Since many depth notions are fully data-driven and robust, their computation may constitute a challenge. For example, computation of the halfspace depth (Tukey, 1975) — one of the most important depth notions in the literature — is an NP-hard problem (Johnson and Preparata, 1978), and the only exact existing algorithm for computation of the projection depth (Liu and Zuo, 2014) is still very slow. For this reason, theoretical developments on the data depth are accompanied by a substantial body of literature on its computation, which still contains a number of open problems. Since exact computation of certain depth notions can come at a very high computational cost (see, e.g., Dyckerhoff and Mozharovskyi, 2016 for the halfspace depth and Liu and Zuo, 2014 for projection depth), approximations have been proposed.

Dyckerhoff (2004) described a class of depths which satisfy the weak projection property. Out of the existing variety, these can be calculated in a universal way by minimizing depth in univariate projections. For this, in each direction, only the univariate depth of n observations should be computed, which often has

computational time complexity only $O(n)$. This class of depths constitutes the focus of the current work. Among the depths that satisfy the projection property, Mahalanobis (Mahalanobis, 1936), zonoid (Koshevoy and Mosler, 1997), halfspace (Tukey, 1975), projection (Zuo and Serfling, 2000) and asymmetric projection (Dyckerhoff, 2004) depth are considered here.

Several authors have already applied approximation techniques to the (sample) depth computation, notably for the halfspace depth and projection depth. Purely random methods seem most intuitive and have been used, e.g., by Cuesta-Albertos and Nieto-Reyes (2008) for the halfspace depth and Liu and Zuo (2014) for the projection depth. More sophisticated procedures were proposed as well. Rousseeuw and Struyf (1998) minimize univariate halfspace depth after projecting data onto directions based on a random combination of sample points. Chen et al. (2013) first project data on subspaces orthogonal to linear spans of $0 < k \leq d$ points from the sample, and then employ a brute-force approximation of the halfspace depth in these projections. Mozharovskyi et al. (2015) accelerate the problem of approximation of the halfspace depth of many points (and of the sample itself) w.r.t. a sample by preliminary sorting the data in all projections. Dutta and Ghosh (2012) use the Nelder-Mead algorithm (Nelder and Mead, 1965, run in $\mathbb{R}^d$) for approximation of the projection depth.

In this work, a systematic experimental approach is used to study the behavior of the approximation of the sample depth by minimizing it on univariate directions. For minimization of the univariate depth, we contrast eight approximation algorithms in an extensive simulation study. As algorithms we considered: (i) random search, (ii) grid search, (iii) refined random search, (iv) refined grid search, (v) random simplices, (vi) simulated annealing, (vii) coordinate descent, (viii) Nelder-Mead. Since the performance of the algorithms depends on chosen parameters, we start by fine-tuning of the algorithms. After this, the algorithms are compared in different settings to provide a trustworthy conclusion.

Throughout this paper we use the following notation. Vectors are notated with bold letters, e.g., $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$. $\mathbb{R}_0^+$ denotes the set of non-negative real numbers.

The $d-1$ dimensional unit sphere in $\mathbb{R}^d$ is denoted by $\mathbb{S}^{d-1}$. The transpose of a vector $\boldsymbol{x}$ is denoted by $\boldsymbol{x}^T$. The $d \times d$ identity matrix is denoted by $\boldsymbol{I}_d$, or shortly $\boldsymbol{I}$, and $\mathbf{0}$ or $\mathbf{0}_d$ denotes the origin in $\mathbb{R}^d$. For a set A we denote by $\mathcal{U}(A)$ the uniform distribution on the set A, and $\mathcal{B}$ and $\mathcal{N}$ denote Bernoulli and Gaussian distributions, respectively. The elements of a vector $\boldsymbol{x} \in \mathbb{R}^d$ are denoted by $x_1, \ldots, x_d$. By $\overset{\mathrm{d}}{=}$ we mean equality in distribution.

2. Preliminary material

In Section 2.1 we give a short introduction into the notion of *depth* and discuss the *projection property* and its implications for the computation of depth. We will see that for depths satisfying the projection property, the computation of the depth is equivalent to minimizing an objective function over the unit sphere. Therefore, in Section 2.2 we will have a closer look on the unit sphere and describe its geometry.

2.1. Data depth

Depth is a concept that measures the centrality of a given point $\boldsymbol{z} \in \mathbb{R}^d$ w.r.t. a probability distribution P on $(\mathbb{R}^d, \mathcal{B}^d)$ where $\mathcal{B}^d$ denotes the Borel σ-algebra on $\mathbb{R}^d$. A generic depth is denoted by $\mathrm{D}(\boldsymbol{z}|P)$. In applications, the probability measure P is often the empirical measure on a set of data points $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. In that case we write $\mathrm{D}(\boldsymbol{z}|\boldsymbol{X})$. Every reasonable notion of depth should satisfy the following set of axioms.

D1: Affine invariance. For every non-singular $d \times d$-matrix $\boldsymbol{A}$ and $\boldsymbol{b} \in \mathbb{R}^d$ it holds true that $\mathrm{D}(\boldsymbol{z} \,|\, \boldsymbol{X}) = \mathrm{D}(\boldsymbol{A}\boldsymbol{z} + \boldsymbol{b} \,|\, \boldsymbol{A}\boldsymbol{X} + \boldsymbol{b})$, where $\boldsymbol{A}\boldsymbol{X} + \boldsymbol{b} = (\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{b}, \ldots, \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{b})$.

D2: Vanishing at infinity. $\lim_{\|\boldsymbol{z}\| \to \infty} \mathrm{D}(\boldsymbol{z} \,|\, \boldsymbol{X}) = 0$.

D3: Upper semicontinuity. For each $\alpha > 0$ the set $\{\boldsymbol{z} \in \mathbb{R}^d \,|\, \mathrm{D}(\boldsymbol{z}|\boldsymbol{X}) \geq \alpha\}$ is closed.

D4: Monotone decreasing on rays. For each point $\boldsymbol{x}_0$ of maximal depth and each $\boldsymbol{r} \in \mathbb{R}^d$, $\boldsymbol{r} \neq \boldsymbol{0}$, the function $\lambda \mapsto \mathrm{D}(\boldsymbol{x}_0 + \lambda \boldsymbol{r} \,|\, \boldsymbol{X})$, $\lambda \geq 0$, is monotone non-increasing.

Properties D1, D2, and D4 have been introduced by Liu (1990). A further set of axioms for a depth has been given by Zuo and Serfling (2000). The main difference between their axioms and ours is that they do not require a depth to be upper semicontinuous. In addition, they require that for distributions having a properly defined unique center of symmetry, the depth attains its maximum value at this center. However, for centrally symmetric distributions, this follows already from our axioms. For further discussion on these axioms, see e.g., Dyckerhoff (2004). In the rest of this paper we assume that a depth satisfies the four axioms D1, D2, D3 and D4.

We consider five depth notions; they all satisfy the four above mentioned properties.

For a point $\boldsymbol{z} \in \mathbb{R}^d$, its Mahalanobis depth (MD) (Mahalanobis, 1936) w.r.t. a data set $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ is defined as follows:

$$\mathrm{D}_M(\boldsymbol{z}|\boldsymbol{X}) = \left(1 + (\boldsymbol{z} - \overline{\boldsymbol{x}})^T \boldsymbol{S}_{\boldsymbol{X}}^{-1} (\boldsymbol{z} - \overline{\boldsymbol{x}})\right)^{-1}.$$

Here, $\overline{\boldsymbol{x}}$ and $\boldsymbol{S}_{\boldsymbol{X}}$ denote the mean and the empirical covariance matrix of $\boldsymbol{X}$, respectively.

The halfspace depth (HD) (Tukey, 1975; Donoho and Gasko, 1992) is defined by

$$\mathrm{D}_H(\boldsymbol{z}|\boldsymbol{X}) = \min_{\boldsymbol{p} \in \mathbb{S}^{d-1}} \frac{1}{n} \#\{i \,:\, \langle \boldsymbol{x}_i, \boldsymbol{p} \rangle \geq \langle \boldsymbol{z}, \boldsymbol{p} \rangle \,,\, i = 1, ..., n\}$$

where $\#A$ denotes the number of elements of a set A.

The zonoid depth (ZD) (Koshevoy and Mosler, 1997) is given by

$$\mathrm{D}_Z(\boldsymbol{z}|\boldsymbol{X}) = \sup\{\alpha \in (0, 1] : \boldsymbol{z} \in Z_\alpha(\boldsymbol{X})\},$$

where $Z_\alpha(\boldsymbol{X})$ is the zonoid α-trimmed region, defined by

$$Z_\alpha(\boldsymbol{X}) = \left\{ \sum_{i=1}^n \lambda_i \boldsymbol{x}_i : 0 \leq \lambda_i \leq \frac{1}{n\alpha}, \sum_{i=1}^n \lambda_i = 1 \right\}$$

and the convention $\sup \emptyset = 0$ is used.

The projection depth (PD) (Zuo and Serfling, 2000) is given by

$$\mathrm{D}_P(\boldsymbol{z}|\boldsymbol{X}) = \min_{\boldsymbol{p}\in\mathbb{S}^{d-1}} \Big(1 + \frac{|\langle \boldsymbol{z},\boldsymbol{p}\rangle - \mathrm{med}(\langle \boldsymbol{X},\boldsymbol{p}\rangle)|}{\mathrm{MAD}(\langle \boldsymbol{X},\boldsymbol{p}\rangle)}\Big)^{-1},$$

where $\langle \boldsymbol{X},\boldsymbol{p}\rangle$ is understood to be the univariate data set obtained by projecting each point of $\boldsymbol{X}$ on $\boldsymbol{p}$, med is the univariate median, and MAD is the median absolute deviation from the median.

Since PD is always symmetric around its deepest point, the asymmetric projection depth (APD) (Dyckerhoff, 2004) has been defined as

$$\mathrm{D}_{AP}(\boldsymbol{z}|\boldsymbol{X}) = \min_{\boldsymbol{p}\in\mathbb{S}^{d-1}} \Big(1 + \frac{(\langle \boldsymbol{z},\boldsymbol{p}\rangle - \mathrm{med}(\langle \boldsymbol{X},\boldsymbol{p}\rangle))_+}{\mathrm{MAD}^+(\langle \boldsymbol{X},\boldsymbol{p}\rangle)}\Big)^{-1},$$

with $(a)_+ = \max\{a, 0\}$ being the positive part of a and MAD^+ being the median of the positive deviations from the median.

All of the above depths satisfy the *(weak) projection property*, defined as follows.

Definition 1. *A depth* D *satisfies the (weak)* projection property*, if for each point* $\boldsymbol{y} \in \mathbb{R}^d$ *and each sample* $\boldsymbol{X}$ *it holds:*

$$\mathrm{D}(\boldsymbol{y}|\boldsymbol{X}) = \inf\{\mathrm{D}(\langle \boldsymbol{p},\boldsymbol{y}\rangle \,|\, \langle \boldsymbol{p},\boldsymbol{X}\rangle)|\boldsymbol{p} \in \mathbb{S}^{d-1}\}\,.$$

If a depth satisfies the projection property, its computation is equivalent to minimization of the (possibly non-differentiable) objective function

$$\phi_{\boldsymbol{z},\boldsymbol{X}} : \mathbb{S}^{d-1} \to \mathbb{R}_0^+,\ \boldsymbol{p} \mapsto \mathrm{D}(\langle \boldsymbol{p},\boldsymbol{z}\rangle \,|\, \langle \boldsymbol{p},\boldsymbol{X}\rangle)\,.$$

Therefore, classical optimization methods can be used to compute the depth. Particular attention should be paid here to the domain of the function $\phi_{\boldsymbol{z},\boldsymbol{X}}$ which is the the unit sphere $\mathbb{S}^{d-1}$. Of course, the function $\phi_{\boldsymbol{z},\boldsymbol{X}}$ could be easily extended to a function $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$ defined on $\mathbb{R}^d \setminus \{\boldsymbol{0}\}$ by setting $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}(\boldsymbol{p}) = \mathrm{D}(\langle \boldsymbol{p},\boldsymbol{z}\rangle \,|\, \langle \boldsymbol{p},\boldsymbol{X}\rangle)$. However, because of the affine invariance of the depth, $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$ is constant on lines through the origin. Therefore, we claim that it should be advantageous to use optimization methods which are adapted to the particular domain $\mathbb{S}^{d-1}$. This claim will be confirmed in the simulation studies in Section 4.

To get some insights in the behavior of the objective function $\phi_{\boldsymbol{z},\boldsymbol{X}}$, e.g., whether there are local minima or not, we present several plots of $\phi_{\boldsymbol{z},\boldsymbol{X}}$ in the case $d = 3$ for different depths and data sets in Figure 1. The figures suggest that (at least for common distributions and in the case $d = 3$) local minima seem not to be a major problem.

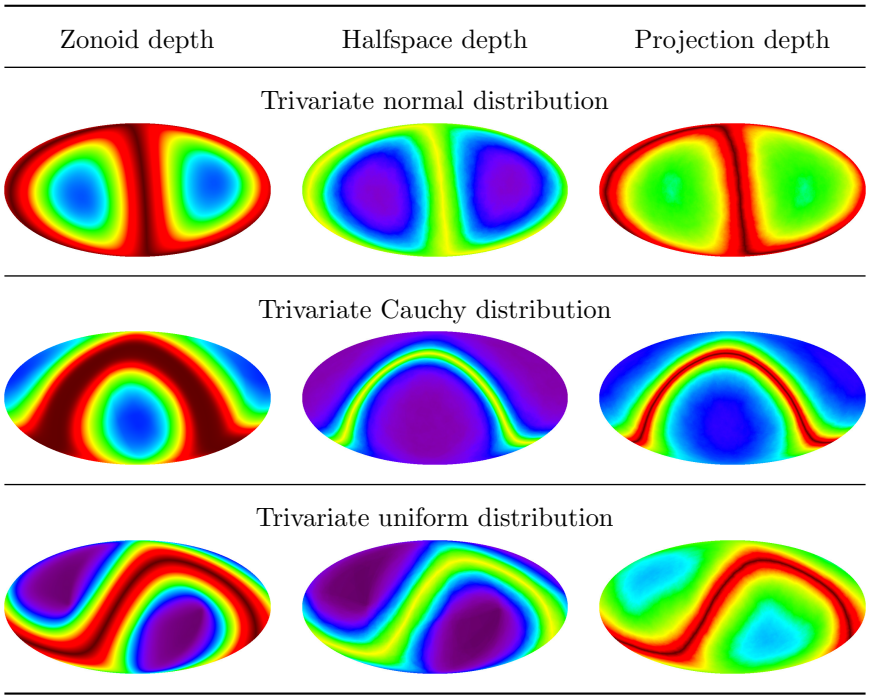

Figure 1: The map $\phi_{\boldsymbol{z},\boldsymbol{X}}$ for trivariate data. A total of $n = 1000$ data points were simulated from a trivariate distribution. The univariate depth (of a single randomly chosen point $\boldsymbol{z}$) in direction $\boldsymbol{p}$ is shown on a color scale from violet (low univariate depth) to dark red (high univariate depth). The sphere $\mathbb{S}^2$ is mapped on the plane using the so-called Mollweide projection, see Snyder (1987).

A further important observation is the following. All of the above depths are bounded above by unity. Therefore, the range of $\phi_{\boldsymbol{z},\boldsymbol{X}}$ depends on the depth of $\boldsymbol{z}$, i.e., if $\mathrm{D}(\boldsymbol{z} \,|\, \boldsymbol{X}) = c_0$, then the range of $\phi_{\boldsymbol{z},\boldsymbol{X}}$ is a subset of $[c_0, 1]$. The larger the depth of $\boldsymbol{z}$, the smaller is the variation of $\phi_{\boldsymbol{z},\boldsymbol{X}}$. In the extreme case that $c_0 = 1$ the function $\phi_{\boldsymbol{z},\boldsymbol{X}}$ is constant, $\phi_{\boldsymbol{z},\boldsymbol{X}}(\boldsymbol{p}) = 1$ for all $\boldsymbol{p} \in \mathbb{S}^{d-1}$. Therefore,

evaluating the univariate depth for a single direction only already gives the exact multivariate depth. In that sense, it should be easier to compute the depth of a point with high depth.

2.2. The geometry of the unit sphere

The set

$$S(\boldsymbol{a}, r) = \{\boldsymbol{x} \in \mathbb{R}^d | \|\boldsymbol{x} - \boldsymbol{a}\| = r\}$$

is called the *sphere* with center $\boldsymbol{a} \in \mathbb{R}^d$ and radius $r \geq 0$. The sphere with radius one and center $\mathbf{0}$ is called the *unit sphere* in $\mathbb{R}^d$, it is denoted by $\mathbb{S}^{d-1}$.

The intersection of the unit sphere with an affine subspace is called a *small sphere*. If the affine subspace contains the origin (i.e., if it is a linear subspace), then the intersection is called a *great sphere*. In the special case that the affine subspace is a plane, i.e., has dimension 2, then the intersection is called *small circle* (if the plane does not pass through the origin) or *great circle* (if the plane passes through the origin).

The intersection of the unit sphere with a closed (open) halfspace is called closed (open) *spherical cap*. If the bounding hyperplane of the halfspace passes through the origin, the spherical cap is called a *hemisphere*.

The distance between two points $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}^{d-1}$ on the sphere as *measured in the ambient space* $\mathbb{R}^d$ is given by the Euclidean distance,

$$d_e(\boldsymbol{u}, \boldsymbol{v}) = \|\boldsymbol{u} - \boldsymbol{v}\| = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2}\,.$$

However, when we measure the distance between two points $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}^{d-1}$ *in the sphere itself*, then the distance is given by the *great-circle distance*, that is the length of the shorter arc of a great circle passing through $\boldsymbol{u}$ and $\boldsymbol{v}$,

$$d_g(\boldsymbol{u}, \boldsymbol{v}) = \arccos(\langle \boldsymbol{u}, \boldsymbol{v} \rangle)\,.$$

It holds $0 \leq d_g(\boldsymbol{u}, \boldsymbol{v}) \leq \pi$, i.e., the great-circle distance between two points on the unit sphere is at most π.

The Euclidean distance and the great-circle distance are related as follows:

$$d_e(\boldsymbol{u}, \boldsymbol{v}) = 2 \sin\left(\frac{d_g(\boldsymbol{u}, \boldsymbol{v})}{2}\right).$$

The transformation $[0, \pi] \to [0, 2]$, $\phi \mapsto 2\sin(\phi/2)$, is continuous and strictly increasing. Therefore, both metrics generate the same topology of open sets on $\mathbb{S}^{d-1}$. The ϵ-neighborhood of a point $\boldsymbol{a} \in \mathbb{S}^{d-1}$ is given by

$$B(\boldsymbol{a}, \epsilon) = \{\boldsymbol{p} \in \mathbb{S}^{d-1} |\, d_g(\boldsymbol{p}, \boldsymbol{a}) < \epsilon\} = \{\boldsymbol{p} \in \mathbb{S}^{d-1} |\, \langle \boldsymbol{a}, \boldsymbol{p} \rangle > \cos(\epsilon)\}.$$

It consists of all points in the sphere that have a great-circle distance less than ϵ to $\boldsymbol{a}$. Since $B(\boldsymbol{a}, \epsilon)$ is the intersection of the unit sphere $\mathbb{S}^{d-1}$ with the halfspace

$$\{\boldsymbol{x} \in \mathbb{R}^d |\, \langle \boldsymbol{a}, \boldsymbol{x} \rangle > \cos(\epsilon)\}$$

the set $B(\boldsymbol{a}, \epsilon)$ is a spherical cap and its topological boundary is the small sphere

$$\{\boldsymbol{x} \in \mathbb{S}^{d-1} |\, \langle \boldsymbol{a}, \boldsymbol{x} \rangle = \cos(\epsilon)\}.$$

The shortest path between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ on the unit sphere $\mathbb{S}^{d-1}$ is given by the shorter arc of a great circle which passes through $\boldsymbol{x}$ and $\boldsymbol{y}$. Therefore, the great circles on a sphere are the *geodesics*, i.e., generalizations of straight lines from the usual Euclidean space. A great circle through $\boldsymbol{x}$ and $\boldsymbol{y}$ is unique as long as $\boldsymbol{y} \neq \pm\boldsymbol{x}$. Let $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \langle \boldsymbol{x}, \boldsymbol{y} \rangle \boldsymbol{x}$. Then, $\boldsymbol{x}$ and $\tilde{\boldsymbol{y}}$ are orthogonal. Therefore, the (unique) great circle through $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by

$$\left\{ \boldsymbol{z}(\phi) = \cos(\phi)\boldsymbol{x} + \sin(\phi)\frac{\tilde{\boldsymbol{y}}}{\|\tilde{\boldsymbol{y}}\|} \,\middle|\, \phi \in (-\pi, \pi] \right\}. \tag{1}$$

Simple algebra shows that the great-circle distance between $\boldsymbol{x}$ and $\boldsymbol{z}(\phi)$ is $|\phi|$. Further, $\boldsymbol{z}(0) = \boldsymbol{x}$ and $\boldsymbol{z}(d_g(\boldsymbol{x}, \boldsymbol{y})) = \boldsymbol{y}$.

3. Algorithms for approximating projection depths

In this section we present several algorithms to compute the depth of a given point $\boldsymbol{z}$ w.r.t. a data cloud $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. We assume that the depth satisfies the projection property so that the depth can be computed as the minimum of the projected univariate depths over the unit sphere $\mathbb{S}^{d-1}$. Most of the algorithms are presented in pseudocode.

3.1. Simple random search (RS)

If a depth satisfies the projection property, then for each $\boldsymbol{p} \in \mathbb{S}^{d-1}$ the value $\mathrm{D}(\langle \boldsymbol{p}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{p}, \boldsymbol{X} \rangle)$ is an upper bound for $\mathrm{D}(\boldsymbol{z}|\boldsymbol{X})$. Therefore, it seems reasonable to compute the univariate depths $\mathrm{D}(\langle \boldsymbol{p}, \boldsymbol{z} \rangle \mid \langle \boldsymbol{p}, \boldsymbol{X} \rangle)$ for several values of $\boldsymbol{p}$. The minima of these values constitute a decreasing sequence of upper bounds for the true value of the depth. One can show that under weak conditions this sequence of upper bounds converges to the true value. The following proposition can be found in Dyckerhoff (2004).

Proposition 1. *Let $\boldsymbol{z} \in \mathbb{R}^d$ and D a depth that satisfies the projection property. Further, let the mapping $\mathbb{S}^{d-1} \to [0, \infty)$, $\boldsymbol{p} \mapsto \mathrm{D}(\langle \boldsymbol{p}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{p}, \boldsymbol{X} \rangle)$, be upper semicontinuous. If $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots$ is a sequence of independent random vectors distributed uniformly on $\mathbb{S}^{d-1}$, then, almost surely,*

$$\lim_{N \to \infty} \min_{1 \le i \le N} \mathrm{D}(\langle \boldsymbol{p}_i, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{p}_i, \boldsymbol{X} \rangle) = \mathrm{D}(\boldsymbol{z}|\boldsymbol{X})\,.$$

Because of the affine invariance of the depth it holds true that $\mathrm{D}(\langle \boldsymbol{p}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{p}, \boldsymbol{X} \rangle) = \mathrm{D}(\langle -\boldsymbol{p}, \boldsymbol{z} \rangle \,|\, \langle -\boldsymbol{p}, \boldsymbol{X} \rangle)$. Therefore, one can restrict $\boldsymbol{p}$ to a hemisphere.

Proposition 1 motivates the following algorithm which we will call *simple random search*. Generate a "large number" N of random directions $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N \in \mathbb{S}^{d-1}$, independently drawn from the uniform distribution on $\mathbb{S}^{d-1}$. For each of these directions compute the univariate depth. The (multivariate) depth is then approximated by

$$\min_{1 \le i \le N} \mathrm{D}(\langle \boldsymbol{p}_i, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{p}_i, \boldsymbol{X} \rangle)\,.$$

It is well known that random vectors from the uniform distribution on the sphere $\mathbb{S}^{d-1}$ can easily be simulated by generating d independent random numbers from a standard normal distribution and normalizing the resulting vector to have norm one. For the sake of completeness we present this algorithm in pseudocode as Algorithm 1.

The simple random search is presented in pseudocode as Algorithm 2.

In implementing the above algorithm, one could think of two possibilities for choosing the number of iterations N. First, the number N could be chosen in

Algorithm 1 Random vectors from uniform distribution on the sphere

function RNDSPHERE(d)
  $s \leftarrow 0$
  for $i \leftarrow 1$ **to** d **do**
    $x_i \leftarrow$ RNDNORMAL() ▷ random number from a $\mathcal{N}(0,1)$-distribution
    $s \leftarrow s + x_i^2$
  for $i \leftarrow 1$ **to** d **do** $x_i \leftarrow x_i/\sqrt{s}$
  return $(x_1, \ldots, x_d)$

Algorithm 2 Simple random search

function RANDOMSEARCH($\boldsymbol{z}, \boldsymbol{X}$)
  $d_{min} \leftarrow \infty$
  for $j \leftarrow 1$ **to** N **do**
    $\boldsymbol{p} \leftarrow$ RNDSPHERE(d)
    $d_{min} \leftarrow \min(\mathrm{D}(\langle \boldsymbol{p}, \boldsymbol{z} \rangle \mid \langle \boldsymbol{p}, \boldsymbol{X} \rangle), d_{min})$
  return d_{min}

advance, depending on n, d and the desired accuracy. Second, random directions would be generated until a certain stopping criterion is satisfied. In the first case, it seems reasonable to assume that N (for a given accuracy) should depend on d but not on n. It further seems reasonable to require $N(d) \propto N(2)^{d-1}$. This would result in an overall complexity of the algorithm of order $O(N(2)^{d-1}n)$ provided the complexity of computing the univariate depth is of order $O(n)$, which holds for all the considered depths. However, this is not desirable since the complexity grows exponentially with the dimension d.

The convergence of $\min_{1 \le i \le N} \mathrm{D}(\langle \boldsymbol{p}_i, \boldsymbol{z} \rangle \mid \langle \boldsymbol{p}_i, \boldsymbol{X} \rangle)$ for the halfspace depth and the projection depth has been extensively studied in Nagy et al. (2020). Given a precision ϵ, these results can be used to find $N(\epsilon)$ such that the error is approximately ϵ.

Of course, an algorithm that uses a larger number of univariate depth evaluations should give a better result than with a smaller number. Therefore, for a fair comparison of different algorithms one should limit the number of depth evaluations to a given number. Following this logic, we choose not to use a stopping criterion in the random search but rather use a fixed number N of iterations. This number should be the same for all compared algorithms. In Sections 4.2 and 4.3 where we compared several algorithms and parameter

combinations, $N \in \{100, 1000, 10000\}$ was used.

3.2. Simple grid search (GS)

Since in the random search the directions are distributed randomly on the sphere, it might be tempting to use a deterministic grid of directions instead. Thus, the second algorithm that we discuss is a grid search on the sphere. A parametrization of the sphere $\mathbb{S}^{d-1}$ is given by *generalized spherical coordinates*:

$$\begin{aligned} x_1 &= \cos(\phi_1), \\ x_2 &= \sin(\phi_1)\cos(\phi_2), \\ &\vdots \qquad\qquad \vdots \\ x_{d-1} &= \sin(\phi_1)\ldots\sin(\phi_{d-2})\cos(\phi_{d-1}), \\ x_d &= \sin(\phi_1)\ldots\sin(\phi_{d-2})\sin(\phi_{d-1}), \end{aligned}$$

where $\phi_1, \ldots, \phi_{d-2} \in [0, \pi]$, $\phi_{d-1} \in [0, 2\pi)$. Here, ϕ_1 is called *polar angle*. For $\phi_1 = 0$ we get the *north pole* $(1, 0, \ldots, 0)^T = \boldsymbol{e}_1$, for $\phi_1 = \pi$ we get the *south pole* $(-1, 0, \ldots, 0)^T = -\boldsymbol{e}_1$. Since

$$d_g(\boldsymbol{x}, \boldsymbol{e}_1) = \arccos(\langle \boldsymbol{x}, \boldsymbol{e}_1 \rangle) = \arccos(x_1) = \phi_1 \,,$$

the polar angle ϕ_1 of a point $\boldsymbol{x} \in \mathbb{S}^{d-1}$ is the great-circle distance of $\boldsymbol{x}$ from the north pole.

As was mentioned earlier, we can always restrict the direction $\boldsymbol{p}$ to a hemisphere. In the implementation we used a grid where the first angle ϕ_1 was restricted to the interval $[0, \pi/2]$ which corresponds to a grid on the northern hemisphere.

However, a severe drawback of using generalized spherical coordinates is the fact that the resulting grid is not uniform, the so-called *pole problem*. This is best illustrated in the case of the usual 2-sphere in $\mathbb{R}^3$. The meridians (the lines for which ϕ_2 is constant) converge at the two poles. Thus, near the poles the distance between grid points is smaller than at the equator which leads to an (unwanted) oversampling in the neighborhood of the poles. Further, the

representation of the poles is obviously not unique since all coordinate vectors $(0, \phi_2, \ldots, \phi_{d-1})$, $\phi_2, \ldots, \phi_{d-2} \in [0, \pi]$, $\phi_{d-1} \in [0, 2\pi)$, represent the north pole. More generally, if an angle ϕ_i is equal to 0 or π, then for all values of $\phi_{i+1}, \ldots, \phi_{d-1}$ we get the same point on the sphere.

Contrary to the trivial case of the 1-sphere, it is for dimension $d > 2$ not possible to perfectly uniformly distribute an arbitrary number of points on $\mathbb{S}^{d-1}$. This is closely connected to the existence of convex regular polytopes. As a consequence, a perfectly uniform grid on $\mathbb{S}^{d-1}$ is for $d > 2$ only possible for finitely many values of N. In particular, a regular grid on $\mathbb{S}^{d-1}$ cannot be made arbitrarily fine.

However, there are grids on the sphere which are nearly uniform. Such quasi-uniform grids do not suffer from the pole problem. They are used in geophysics, climate modeling, or astronomy. Examples of quasi-uniform grids include the Kurihara grid (Kurihara, 1965), the icosahedral grid (Williamson, 1968), the cubed sphere grid (Ronchi et al., 1996), or the Yin-Yang grid (Kageyama and Sato, 2004). The disadvantage of these grids is that they are more complicated to implement or that they only work for the case of the 2-sphere. Since it will become obvious that there are methods that are clearly superior to the simple random search or the simple grid search for the computation of depth, we decided not to spend much time on implementing these more complicated quasi-uniform grids. In our simulations we use a simple uniform grid in the hyper-cubic domain that is based on generalized spherical coordinates.

A major drawback of the grid search is that it suffers from the *curse of dimensionality*. For example assume that we only use four subdivisions per angle. Then, for a data set of dimension d, which means that the grid based on spherical coordinates is of dimension $d - 1$, the grid consists of 4^{d-1} grid points. For $d = 11$ we already have more than a million grid points, for $d = 20$ we already have $2.7 \cdot 10^{11}$ grid points, a number far exceeding the capacity of most computers. Therefore, we choose not to use the grid search for dimensions larger than $d = 10$. For the same reason we do not provide a description of the grid search in pseudocode.

In the simulations in Sections 4.2 and 4.3 we used the same mesh size for all spherical coordinates apart from the first (the polar angle) where we used half of this mesh size. This is motivated by the fact that points on the sphere that differ only in their polar angle have the same distance regardless of the other coordinates whereas the distances between points that differ in the other spherical coordinates converge to zero near the poles. The mesh size was chosen so that the number of depth evaluations was approximately equal to a given number N.

3.3. Refined random search (RRS)

In the simple random search the whole surface of the sphere is searched with the same intensity. Therefore, a lot of time is wasted searching areas which are far away from the minimum of the objective function. The idea of the *refined random search* is to concentrate the search in the neighborhood of directions with low depth.

At the start of the algorithm the neighborhood of a point on the sphere is defined to be the hemisphere with pole at that point. We choose N_1 directions at random in the neighborhood of the current best point. Every time a new direction with minimum depth is found, this point is chosen as the new center of the neighborhood. As the search continues we choose the neighborhood smaller and smaller.

We first discuss how to sample from a neighborhood of the north pole $\boldsymbol{e}_1$. The ϵ-neighborhood $B(\boldsymbol{e}_1, \epsilon)$ of the north pole consists of all points whose great-circle distance to the north pole is less than ϵ. Since the distance to the north pole is the polar angle, the *ϵ-polar cap* is the set of all points whose polar angle is less than ϵ. Therefore, in a first step we choose the polar angle ϕ_1 from a uniform distribution on $[0, \epsilon]$ and get the first coordinate of the sampled point, $x_1 = \cos(\phi_1)$. In the second step we choose a point from a uniform distribution on the small sphere $\{\boldsymbol{x} \in \mathbb{S}^{d-1} | x_1 = \cos(\phi_1)\}$ which has radius $\sqrt{1 - \cos^2(\phi_1)}$. This is described in pseudocode in Algorithm 4.

Sampling from a uniform distribution in the first step means that the distance of the points from $\boldsymbol{e}_1$ is uniformly distributed. Note that this does not yield a uniform distribution on the ϵ-polar cap but a distribution where points near the pole have a higher density than points farther away. However, this seems desirable in the refined random search since it means that points near the current minimum have a higher probability to get drawn.

On the contrary, assume that we would choose new points from a uniform distribution on the ϵ-polar cap. In high dimensions, most of the surface area of the spherical cap is concentrated near the base of the cap. Therefore, with a very high probability we would get points that have a large distance from the north pole $\boldsymbol{e}_1$. For example, if $d = 10$ and $\epsilon = 0.1$ there is a probability of 0.998 to draw a point which has a distance between 0.05 and 0.1 from $\boldsymbol{e}_1$, whereas the probability to draw a point that has a distance between 0 and 0.05 is only 0.002. This is even worse in higher dimensions. Therefore, we decided to draw the polar angle from a uniform distribution.

Note further that the density of $\boldsymbol{x}$ depends only on the polar angle ϕ_1 which measures the distance from the pole.

To sample from the neighborhood of an arbitrary point $\boldsymbol{p}$, we look for an transformation $\boldsymbol{Q}$ of the sphere that transforms the north pole $\boldsymbol{e}_1$ to $\boldsymbol{p}$ and does not change distances. Such a transformation is given by a *Householder matrix* (Golub and Van Loan, 1989):

$$\boldsymbol{Q} = \boldsymbol{I} - \frac{2}{\boldsymbol{v}^T\boldsymbol{v}}\boldsymbol{v}\boldsymbol{v}^T\,, \tag{2}$$

where $\boldsymbol{v}$ is some non-zero vector and $\boldsymbol{I}$ is the identity matrix. A Householder matrix is symmetric, orthogonal and thus involutory. Geometrically, $\boldsymbol{Q}$ is a reflection at the hyperplane through the origin whose normal is $\boldsymbol{v}$. It can easily be seen that choosing $\boldsymbol{v} = \boldsymbol{p} - \boldsymbol{e}_1$ does the trick, i.e., $\boldsymbol{Q}\boldsymbol{e}_1 = \boldsymbol{p}$. Therefore, if $\boldsymbol{x}$ is sampled from the ϵ-polar cap $B(\boldsymbol{e}_1, \epsilon)$, then $\boldsymbol{Q}\boldsymbol{x}$ is sampled from the spherical cap $B(\boldsymbol{p}, \epsilon)$. To compute $\boldsymbol{Q}\boldsymbol{x}$ we first note that $\boldsymbol{v}^T\boldsymbol{v} = (\boldsymbol{p} - \boldsymbol{e}_1)^T(\boldsymbol{p} - \boldsymbol{e}_1) =$

$\|\boldsymbol{p}\|^2 - 2p_1 + 1 = 2(1-p_1)$ since $\|\boldsymbol{p}\|^2 = 1$. Therefore,

$$\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{x} - \frac{2}{2(1-p_1)}(\boldsymbol{p}-\boldsymbol{e}_1)(\boldsymbol{p}-\boldsymbol{e}_1)^T\boldsymbol{x} = \boldsymbol{x} - \frac{\boldsymbol{p}^T\boldsymbol{x} - x_1}{1-p_1}(\boldsymbol{p}-\boldsymbol{e}_1) = \boldsymbol{x} - \lambda(\boldsymbol{p}-\boldsymbol{e}_1)$$

where $\lambda = (\langle \boldsymbol{p}, \boldsymbol{x}\rangle - x_1)/(1-p_1)$. For the first component we get $(\boldsymbol{Q}\boldsymbol{x})_1 = x_1 - \lambda p_1 + \lambda$ whereas for the remaining components we have $(\boldsymbol{Q}\boldsymbol{x})_i = x_i - \lambda p_i$. Algorithm 3 describes the Householder transformation in pseudocode.

Algorithm 3 Householder transformation

▷ The Householder transformation that maps $\boldsymbol{e}_1$ to $\boldsymbol{p}$ is applied to $\boldsymbol{x}$
function HOUSEHOLDER($\boldsymbol{x}, \boldsymbol{p}$)
  if $p_1 = 1$ **then return** $\boldsymbol{x}$
  $\lambda \leftarrow (\langle \boldsymbol{p}, \boldsymbol{x}\rangle - x_1)/(1-p_1)$
  $x_1 \leftarrow x_1 + \lambda$
  for $i \leftarrow 1$ **to** d **do** $x_i \leftarrow x_i - \lambda \cdot p_i$
  return $\boldsymbol{x}$

The generation of a random point from the spherical cap is described in Algorithm 4. The refined random search is presented in Algorithm 5.

Algorithm 4 Random vectors from a spherical cap

▷ Generate a random number from a spherical cap with size ϵ around $\boldsymbol{p}$
function RNDSPHERICALCAP($\boldsymbol{p}, \epsilon$)
  $x_1 \leftarrow$ RNDUNIF() ▷ random number from a $\mathcal{U}([0,1])$-distribution
  $x_1 \leftarrow \cos(\epsilon \cdot x_1)$
  $(x_2, \ldots, x_d) \leftarrow \sqrt{1-x_1^2} \cdot$ RNDSPHERE$(d-1)$
  $\boldsymbol{x} \leftarrow (x_1, \ldots, x_d)$
  return HOUSEHOLDER($\boldsymbol{x}, \boldsymbol{p}$)

Algorithm 5 Refined random search

function REFINEDRANDOMSEARCH($\boldsymbol{z}, \boldsymbol{X}$)
  $\boldsymbol{p}_{min} \leftarrow \boldsymbol{e}_1$ ▷ start with the North Pole
  $d_{min} \leftarrow \mathrm{D}(\langle \boldsymbol{p}_{min}, \boldsymbol{z}\rangle \,|\, \langle \boldsymbol{p}_{min}, \boldsymbol{X}\rangle)$
  $\epsilon \leftarrow \pi/2$ ▷ initial neighborhood is a hemisphere
  for $i \leftarrow 1$ **to** N_{ref} **do**
    for $j \leftarrow 1$ **to** N_{it} **do**
      $\boldsymbol{p}_{cur} \leftarrow$ RNDSPHERICALCAP($\boldsymbol{p}_{min}, \epsilon$)
      $d_{cur} \leftarrow \mathrm{D}(\langle \boldsymbol{p}_{cur}, \boldsymbol{z}\rangle \,|\, \langle \boldsymbol{p}_{cur}, \boldsymbol{X}\rangle)$
      if $d_{cur} < d_{min}$ **then**
        $d_{min} \leftarrow d_{cur}$
        $\boldsymbol{p}_{min} \leftarrow \boldsymbol{p}_{cur}$
    $\epsilon \leftarrow \epsilon \cdot \alpha$ ▷ shrinking the neighborhood
  return d_{min}

There, a geometric shrinking of the neighborhood is applied, i.e., in each refinement step the size of the neighborhood is multiplied by a factor $\alpha < 1$. Of course, other shrinking schemes are possible. For example, one could apply an arithmetic shrinking scheme. For our simulations we choose a geometric shrinking scheme.

The refined random search depends on the parameters N_{ref}, N_{it} and α. For N_{ref} and α we tried several parameter combinations and compared them in Section 4.2. The parameter N_{it} was always chosen such that the total number of depth evaluations was equal to a given number N, i.e., $N_{it} = N/N_{ref}$ was used. The fine-tuning of these parameters is described in Section 4.2, see also the figures in the Supplementary Materials.

A drawback of the refined random search is that the algorithm might be trapped in a local minimum. However, the plots in Figure 1 suggest that – at least in simple situations – this should not be a major problem.

3.4. Refined grid search (RGS)

Since the grid search suffers from the curse of dimensionality, we may use the same ideas as in the refined random search, i.e., we start with a relatively coarse grid and apply successive refinements of the grid in the neighborhood of the current minimum. As in the refined random search the *spherical cap neighborhood* may be used. The oversampling near the pole may be sensible in this case, since it means oversampling near the current minimum.

As the refined random search, the refined grid search depends on the parameters N_{ref}, the number of refinements, and α, the shrinking factor of the spherical cap. The mesh size of the grids was chosen such that the total number of depth evaluations was approximately equal to the given value of N. Again, the fine-tuning of these parameters is described in Section 4.2. For N_{ref} and α the same parameter sets as for the refined random search were used. For the same reasons as in the case of the grid search, the refined grid search was used only for $d \leq 10$ in the simulations in Section 4. Because of this limitation and for the sake of brevity we also do not provide a description of the algorithm in

pseudocode.

3.5. Random simplices (RaSi)

In this algorithm random directions are used that are derived from the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ themselves. Similar strategies have already been used by Rousseeuw and Struyf (1998) for the halfspace depth and by Christmann et al. (2002) for the regression depth and classification.

To be more specific, a random sample $\boldsymbol{x}_{i_0}, \boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_d}$ of size $d+1$ is drawn without replacement from the n data points. If these points are in general position, they form a simplex. In the next step, on the facet opposite to $\boldsymbol{x}_{i_0}$ a point $\boldsymbol{x}_o = \sum_{j=1}^{d} w_j \boldsymbol{x}_{i_j}$ is chosen, where the weights $(w_1, \ldots, w_d)^T$ are drawn from a symmetric Dirichlet distribution with parameter α. Finally, the direction $\boldsymbol{p} = \boldsymbol{x}_{i_0} - \boldsymbol{x}_o$ is used to project the data and compute the univariate depth. This process is repeated N times and the minimum attained depth is returned. The algorithm is described in pseudocode as Algorithm 6.

Algorithm 6 Random simplices

```
function randomSimplices(z, X)
    d_min ← ∞
    for i ← 1 to N do
        (i_0, i_1, ..., i_d) ← rndSubset(d + 1, {1, ..., n})
        (w_1, ..., w_d) ← rndDirichlet(d, α)
        p ← Σ_{k=1}^{d} x_{i_k} w_k − x_{i_0}
        p ← p/‖p‖
        d_min ← min(D(⟨p, z⟩ | ⟨p, X⟩), d_min)
    return d_min
```

Besides N, the number of iterations, the only parameter is the parameter α of the Dirichlet distribution. For $\alpha = 1$ we get a uniform distribution on the facet defined by $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_d}$. The higher α, the more the distribution is concentrated around the center of that facet. Fine-tuning of the parameter α is described in Section 4.2.

3.6. Simulated annealing (SA)

In the refined random search we might get trapped in a local minimum. *Simulated annealing* avoids this by accepting also worse solutions with a certain

probability. Let d_{cur} denote the current depth and d_{new} the depth of a new trial solution. Then, d_{new} is chosen as the new solution with probability

$$p = \min\left\{\exp\left(-\frac{d_{new} - d_{cur}}{T}\right), 1\right\}.$$

If $d_{new} < d_{cur}$, the new solution is always accepted. However, if $d_{new} > d_{cur}$, the new solution is still accepted with positive probability p.

The parameter T is called *temperature*. If T is high, the probability of accepting a worse solution is high. Conversely, if T is low, worse solutions are accepted only with small probability. In the course of the algorithm the temperature (and thus the probability of accepting worse solutions) is slowly decreased. The way the parameter T is decreased is called *cooling schedule*. Often a linear cooling schedule, $T(t) = T(0) - \eta\, t$ where $\eta > 0$ is a chosen parameter, or a geometric cooling schedule, $T(t) = T(0)\alpha^t$ where $\alpha \in (0, 1)$, is used. Note, that the choice of the cooling schedule can have a significant impact on the performance of simulated annealing. If the temperature is decreased too fast, then the algorithm may be trapped in a local minimum. If the temperature is decreased too slowly, the convergence of the algorithm is also very slow. A description of simulated annealing where an exponential cooling schedule is used is shown in Algorithm 7.

Simulated annealing has already been applied to the computation of a special projection depth by Shao and Zuo (2012).

Since the sequence of values d_{cur} is in general not decreasing, the last value d_{cur} needs not necessarily be the minimum depth found during the course of the algorithm. Therefore, in Algorithm 7 we keep track of the minimum depth found so far and return this value.

Simulated annealing depends on a lot of parameters. The performance of the algorithm depends strongly on the fine-tuning of these parameters. In particular, to apply the algorithm (i) the starting solution, (ii) the used neighborhood, (iii) the probability distribution for generating a new candidate, (iv) the number of iterations, (v) the starting temperature, (vi) the temperature at which the algorithm is stopped, and (vii) the cooling schedule have to be specified.

Algorithm 7 Simulated annealing

function SIMULATEDANNEALING($\boldsymbol{z}, \boldsymbol{X}$)
  if `Start = Mn` **then** $\boldsymbol{u}_{cur} \leftarrow \boldsymbol{z} - \overline{\boldsymbol{x}}$
  if `Start = Rn` **then** $\boldsymbol{u}_{cur} \leftarrow$ RNDSPHERE(d)
  $\epsilon \leftarrow (\pi/2)/\beta$ ▷ size of the spherical cap
  $d_{cur} \leftarrow \mathrm{D}(\langle \boldsymbol{u}_{cur}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{u}_{cur}, \boldsymbol{X} \rangle)$
  $d_{min} \leftarrow d_{cur}$
  $T \leftarrow T_0$ ▷ starting temperature, $T_0 = 1$
  repeat
    for $j \leftarrow 1$ **to** N_{it} **do**
      $\boldsymbol{u}_{new} \leftarrow$ RNDSPHERICALCAP($\boldsymbol{u}_{cur}, \epsilon$)
      $d_{new} \leftarrow \mathrm{D}(\langle \boldsymbol{u}_{new}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{u}_{new}, \boldsymbol{X} \rangle)$
      $p \leftarrow \min\left(\exp\left(-\frac{d_{new} - d_{cur}}{T}\right), 1\right)$
      with probability p **do**
        $\boldsymbol{u}_{cur} \leftarrow \boldsymbol{u}_{new}$
        $d_{cur} \leftarrow d_{new}$
        $d_{min} \leftarrow \min(d_{cur}, d_{min})$
    $T \leftarrow T \cdot \alpha$ ▷ cooling schedule
  until $T < T_{min}$ ▷ $T_{min} = 0.001$
  return d_{min}

In our simulations the temperature $T(0)$ at the start of the algorithm was always chosen as 1, the temperature T_{min} at which the algorithm stops was always chosen as 0.001. For the starting solution a parameter `Start` having two possible values `Mn` and `Rn` was passed. For `Start=Rn` the algorithm was started with a direction randomly drawn from $\mathcal{U}(\mathbb{S}^{d-1})$. For `Start=Mn` the direction $\boldsymbol{u} = \boldsymbol{z} - \overline{\boldsymbol{x}}$ joining the mean of the data points and $\boldsymbol{z}$ was used. Especially for the case of spherical distributions this should give a good starting solution. For the neighborhood we used the spherical cap of size ϵ. The size ϵ of the spherical cap was controlled by a parameter β via $\epsilon = (\pi/2)/\beta$. A new candidate solution was drawn from the spherical cap using the function RNDSPHERICALCAP described in Section 3.3 and in Algorithm 4. In the simulations we always used a geometric cooling schedule. The speed of cooling was controlled by a parameter α. The number N_{it} of iterations was chosen such that the total number of univariate depth evaluations was equal to a given value N. The fine-tuning of the parameters α, β and `Start` is described in Section 4.2.

3.7. Coordinate descent (CD)

In the coordinate descent algorithm for finding the minimum of a function $f : \mathbb{R}^d \to \mathbb{R}$ in the Euclidean space $\mathbb{R}^d$ one proceeds as follows. Start with an initial value $\boldsymbol{x}^0 = (x_1^0, \ldots, x_d^0)^T$. In the k-th iteration we solve d minimization problems,

$$x_j^{k+1} = \operatorname*{arg\,min}_{y \in \mathbb{R}} f(x_1^{k+1}, \ldots, x_{j-1}^{k+1}, y, x_{j+1}^k, \ldots, x_d^k), \quad j = 1, \ldots, d.$$

This yields a sequence $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ for which $f(\boldsymbol{x}^0) \geq f(\boldsymbol{x}^1) \geq f(\boldsymbol{x}^2) \geq \ldots$.

The function $\phi_{\boldsymbol{z},\boldsymbol{X}} : \mathbb{S}^{d-1} \to \mathbb{R}_0^+$, $\boldsymbol{p} \mapsto \mathrm{D}(\langle \boldsymbol{p}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{p}, \boldsymbol{X} \rangle)$ can be extended to a function $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$ on the domain $\mathbb{R}^d \setminus \{\boldsymbol{0}\}$. To the function $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$ the coordinate descent could be applied without major modifications. However, because of the affine invariance of the depth, $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$ is constant on lines, i.e., for $\lambda \neq 0$ holds $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}(\lambda \boldsymbol{p}) = \tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}(\boldsymbol{p})$. We believe that the performance of the coordinate descent should profit from taking into account the special geometry of the domain of $\phi_{\boldsymbol{z},\boldsymbol{X}}$. Therefore, we adapted the coordinate descent for the special case that the domain of the objective is the unit sphere $\mathbb{S}^{d-1}$. Hence, we cannot use the coordinate directions since they are not contained in the unit sphere. Instead, we replace the straight lines by *great circles* which are the geodesics on the unit sphere. As noted earlier, we can always restrict ourselves to minimize the univariate depth over a hemisphere. So we do not have to minimize over a whole great circle, but only over a great semi-circle.

From (1) it follows that for two orthogonal directions $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}^{d-1}$ the great semi-circle between $\boldsymbol{v}$ and $-\boldsymbol{v}$ passing through $\boldsymbol{u}$ is given by

$$S(\boldsymbol{u}, \boldsymbol{v}) = \left\{ \cos(\alpha)\boldsymbol{u} + \sin(\alpha)\boldsymbol{v} \middle| \alpha \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right] \right\}.$$

In the k-th iteration we solve $d-1$ (univariate) minimization problems. Denote by $\boldsymbol{u}^{(k,0)}$ the current point at the beginning of the k-th iteration. We choose $d-1$ directions $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d-1}$ such that the vectors $\boldsymbol{u}^{(k,0)}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d-1}$ form an orthonormal system of vectors in $\mathbb{R}^d$. Thus, $\boldsymbol{u}^{(k,0)} + \operatorname{span}\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d-1}\}$ is a hyperplane which is tangent to the unit sphere at $\boldsymbol{u}^{(k,0)}$. Together with the

current point the directions $\boldsymbol{p}_j$ determine the great circles for the univariate optimization problems.

Denote by $\boldsymbol{u}^{(k,j)}$, $j = 1, \ldots, d-1$, the solution of the j-th minimization problem in the k-th iteration,

$$\boldsymbol{u}^{(k,j)} = \underset{\boldsymbol{v} \in S(\boldsymbol{u}^{(k,j-1)}, \boldsymbol{p}_j)}{\arg\min} \mathrm{D}(\langle \boldsymbol{v}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{v}, \boldsymbol{X} \rangle).$$

Note that, when we move from $\boldsymbol{u}^{(k,j-1)}$ to $\boldsymbol{u}^{(k,j)}$, also the tangent hyperplane has to be rotated accordingly. However, this rotation takes place in the plane spanned by $\boldsymbol{u}^{(k,j-1)}$ and $\boldsymbol{p}_j$ and affects only these two vectors, leaving the other vectors of the orthonormal system intact. Denote the image of $\boldsymbol{p}_j$ under this rotation by $\tilde{\boldsymbol{p}}_j$. Then, after j minimizations, the system of vectors is mapped to

$$\boldsymbol{u}^{(k,j)}, \tilde{\boldsymbol{p}}_1, \ldots, \tilde{\boldsymbol{p}}_j, \boldsymbol{p}_{j+1}, \ldots, \boldsymbol{p}_{d-1}\,.$$

It is easy to see that this is still an orthonormal system of vectors. Note also, that there is no need to compute the vectors $\tilde{\boldsymbol{p}}_j$ since for the $(j+1)$-th minimization we only need $\boldsymbol{u}^{(k,j)}$ and $\boldsymbol{p}_{j+1}$.

We still need to find the vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d-1}$. If $\boldsymbol{H}$ is a Householder matrix from equation (2) that maps $\boldsymbol{u}^{(k,0)}$ to the unit vector $\boldsymbol{e}_d$, then $\boldsymbol{H}\boldsymbol{u}^{(k,0)} = \boldsymbol{e}_d$ which is equivalent ($\boldsymbol{H}$ is orthogonal and symmetric) to $\boldsymbol{u}^{(k,0)} = \boldsymbol{H}\boldsymbol{e}_d$. Thus, the last column of $\boldsymbol{H}$ is equal to $\boldsymbol{u}^{(k,0)}$. Since the columns of $\boldsymbol{H}$ form an orthonormal system of vectors, $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{d-1}$ can be chosen as the first $d-1$ columns of $\boldsymbol{H}$. An easy calculation shows that

$$(\boldsymbol{p}_j)_i = \begin{cases} -\frac{u_j u_i}{1-u_d} & \text{if } i \neq j, d, \\ 1 - \frac{u_j^2}{1-u_d} & \text{if } i = j, \\ u_j & \text{if } i = d, \end{cases}$$

where u_i denotes the i-th component of $\boldsymbol{u}^{(k,0)}$ and $(\boldsymbol{p}_j)_i$ is the i-th component of $\boldsymbol{p}_j$. A description of the coordinate descent in pseudocode is given in Algorithm 10. The stopping criterion that we used in our simulations guaranteed

that a specified number N of evaluations of the univariate depth was not exceeded. In Algorithm 10 a procedure LineSearch is used. LineSearch$(\boldsymbol{u}, \boldsymbol{p}_j)$ tries to find the minimum of $\mathrm{D}(\langle \boldsymbol{a}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{a}, \boldsymbol{X} \rangle)$ for $\boldsymbol{a}$ in the great semi-circle $S(\boldsymbol{u}, \boldsymbol{p}_j)$. For this line search several strategies are possible, e.g., a uniform search (see Algorithm 8) over the semi great-circle or a golden section search (see Algorithm 9).

Algorithm 8 Line search (uniform spacing)

function lineSearch$(\boldsymbol{u}, \boldsymbol{p})$ ▷ search along the great circle defined by $\boldsymbol{u}$ and $\boldsymbol{p}$
  $f_{min} \leftarrow \infty$
  for $i \leftarrow 0$ **to** n_{LS} **do**
    $\lambda \leftarrow -\pi/2 + i \cdot \pi / n_{LS}$
    $\boldsymbol{w} \leftarrow \cos(\lambda)\boldsymbol{u} + \sin(\lambda)\boldsymbol{p}$
    $f \leftarrow \mathrm{D}(\langle \boldsymbol{w}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{w}, \boldsymbol{X} \rangle)$
    if $f < f_{min}$ **then**
      $f_{min} \leftarrow f$
      $\boldsymbol{u}_{min} \leftarrow \boldsymbol{w}$
  return $(\boldsymbol{u}_{min}, f_{min})$

Algorithm 9 Line search (golden section)

function lineSearch$(\boldsymbol{u}, \boldsymbol{p})$ ▷ search along the great circle defined by $\boldsymbol{u}$ and $\boldsymbol{p}$
  $\alpha \leftarrow (\sqrt{5} - 1)/2$
  $(a, b) \leftarrow (-\pi/2, \pi/2)$
  $\lambda \leftarrow \alpha a + (1 - \alpha) b$
  $\mu \leftarrow (1 - \alpha) a + \alpha b$
  $\boldsymbol{w} \leftarrow \cos(\lambda)\boldsymbol{u} + \sin(\lambda)\boldsymbol{p}$
  $f_1 \leftarrow \mathrm{D}(\langle \boldsymbol{w}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{w}, \boldsymbol{X} \rangle)$
  $\boldsymbol{w} \leftarrow \cos(\mu)\boldsymbol{u} + \sin(\mu)\boldsymbol{p}$
  $f_2 \leftarrow \mathrm{D}(\langle \boldsymbol{w}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{w}, \boldsymbol{X} \rangle)$
  $f_{min} \leftarrow \min(f_1, f_2)$
  while $b - a > \epsilon$ **do**
    if $f_1 > f_2$ **then**
      $(a, \lambda, f_1) \leftarrow (\lambda, \mu, f_2)$
      $\mu \leftarrow (1 - \alpha) a + \alpha b$
      $\boldsymbol{w} \leftarrow \cos(\mu)\boldsymbol{u} + \sin(\mu)\boldsymbol{p}$
      $f_2 \leftarrow \mathrm{D}(\langle \boldsymbol{w}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{w}, \boldsymbol{X} \rangle)$
      $f_{min} \leftarrow \min(f_2, f_{min})$
    else
      $(b, \mu, f_2) \leftarrow (\mu, \lambda, f_1)$
      $\lambda \leftarrow \alpha a + (1 - \alpha) b$
      $\boldsymbol{w} \leftarrow \cos(\lambda)\boldsymbol{u} + \sin(\lambda)\boldsymbol{p}$
      $f_1 \leftarrow \mathrm{D}(\langle \boldsymbol{w}, \boldsymbol{z} \rangle \,|\, \langle \boldsymbol{w}, \boldsymbol{X} \rangle)$
      $f_{min} \leftarrow \min(f_1, f_{min})$
  return $(\boldsymbol{w}, f_{min})$

In Section 4.2 we compare several strategies for the line search.

To test our hypothesis that the coordinate descent should profit from taking

Algorithm 10 Coordinate descent

```
1: function coordinateDescent(z, X)
2:     u ← rndSphere(d)                          ▷ start with a random point
3:     repeat
4:         v ← u                                 ▷ v is u^(k,0)
5:         for j ← 1 to d − 1 do
6:             for i ← 1 to d − 1 do
7:                 p_i ← −v_i · v_j/(1 − v_d)
8:             p_j ← 1 + p_j
9:             p_d ← v_j                         ▷ p is p_j
10:            (u, d_cur) ← LineSearch(u, p)
11:    until stopping criterion is satisfied
12:    return d_cur
```

into account the spherical geometry of the domain, we implemented the coordinate descent such that by passing a parameter `Space`, having two possible values `Ec` (Euclidean space) or `Sp` (Sphere), we could choose between the naïve application of the coordinate descent and the version specifically adapted to the sphere. Further, a parameter `LS` can be passed that controls the line search algorithm used. For `LS` equal to `Eq` the line search is done on an equally spaced grid, for `LS` equal to `GS` the golden section method is used.

3.8. Nelder-Mead method (NM)

The *Nelder-Mead method* (also known as *downhill simplex method*), originally proposed by Nelder and Mead (1965), is a well-known optimization method that does not rely on derivatives. It is based on a simplex i.e., a polytope that is defined by $d+1$ vertices $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d+1} \in \mathbb{R}^d$. Assume that the vertices are ordered in such a way that the corresponding function values of an objective function f are increasing, $f(\boldsymbol{x}_1) \leq \cdots \leq f(\boldsymbol{x}_{d+1})$. In each step of the algorithm the vertex $\boldsymbol{x}_{d+1}$ with the worst function value is replaced by a new vertex in a specified manner. The new vertex is chosen in such a way that the simplex typically approaches a minimum of the function f. Denote by $\boldsymbol{x}_o = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{x}_i$ the centroid of all but the worst vertices. The updating of the simplex is based on the following operations:

- **Reflection:** $\boldsymbol{x}_r = \boldsymbol{x}_o + \alpha(\boldsymbol{x}_o - \boldsymbol{x}_{d+1})$, $\alpha > 0$;
- **Expansion:** $\boldsymbol{x}_e = \boldsymbol{x}_o + \gamma(\boldsymbol{x}_r - \boldsymbol{x}_o)$, $\gamma > 1$;

- **Inside Contraction:** $\boldsymbol{x}_{ic} = \boldsymbol{x}_o + \rho(\boldsymbol{x}_{d+1} - \boldsymbol{x}_o)$, $0 < \rho < 1$;
- **Outside Contraction:** $\boldsymbol{x}_{oc} = \boldsymbol{x}_o + \rho(\boldsymbol{x}_r - \boldsymbol{x}_o)$, $0 < \rho < 1$;
- **Shrinking:** $\boldsymbol{x}'_i = \boldsymbol{x}_1 + \sigma(\boldsymbol{x}_i - \boldsymbol{x}_1)$, $i = 2, \ldots, d+1$, $0 < \sigma < 1$.

A common choice for the parameters is $\alpha = 1$, $\gamma = 2$, $\rho = \sigma = 0.5$, see Figure 2. In the first four operations the new point lies on the straight line through $\boldsymbol{x}_o$ and $\boldsymbol{x}_{d+1}$, in the last operation new points are computed on the straight lines through $\boldsymbol{x}_1$ and $\boldsymbol{x}_i$, $i = 2, \ldots, d+1$. The new simplex is then formed by the points $\boldsymbol{x}_1$ and $\boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{d+1}$.

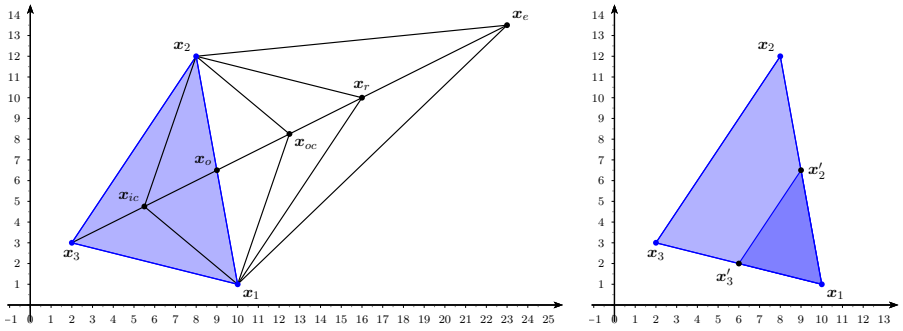

Figure 2: Illustration of the Nelder-Mead algorithm. The original simplex is given in light blue. In the left panel the points $\boldsymbol{x}_r, \boldsymbol{x}_e, \boldsymbol{x}_{ic}$ and $\boldsymbol{x}_{oc}$ are shown together with the newly formed simplices. In the right panel the shrunk simplex is shown in dark blue.

As in the case of the coordinate descent, it is possible to apply the Nelder-Mead algorithm without any modifications to the function $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$ that extends $\phi_{\boldsymbol{z},\boldsymbol{X}}$ on the domain $\mathbb{R}^d \setminus \{\boldsymbol{0}\}$. Again, we believe that it is better to take care of the special geometry of the domain $\mathbb{S}^{d-1}$ and adapt the Nelder-Mead method in a proper way. To do this the straight lines along which new points are computed have to be replaced by proper curves. The natural choice are again the geodesics on the sphere, i.e., the great circles.

Following (1) the great circle defined by $\boldsymbol{x}$ and $\boldsymbol{y}$ is given for $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \langle \boldsymbol{x}, \boldsymbol{y} \rangle \boldsymbol{x}$ by

$$\left\{ \boldsymbol{z}(\phi) = \cos(\phi)\boldsymbol{x} + \sin(\phi) \frac{\tilde{\boldsymbol{y}}}{\|\tilde{\boldsymbol{y}}\|} \,\middle|\, \phi \in (-\pi, \pi] \right\} .$$

Thus, the analogue of the point $\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})$ is given by $\boldsymbol{z}(t\alpha)$, where $\alpha = \arccos(\langle \boldsymbol{x}, \boldsymbol{y} \rangle)$ is the great-circle distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. We define the function $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}} : \mathbb{R} \to \mathbb{S}^{d-1}$ by

$$\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t) = \cos(t\alpha)\boldsymbol{x} + \sin(t\alpha)\frac{\tilde{\boldsymbol{y}}}{\|\tilde{\boldsymbol{y}}\|}\,.$$

Clearly, $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$ is a point that lies on the great circle defined by $\boldsymbol{x}$ and $\boldsymbol{y}$, and the great-circle distance between $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$ and $\boldsymbol{x}$ is t times the great-circle distance between $\boldsymbol{y}$ and $\boldsymbol{x}$ (provided that $|t\alpha| < \pi$). For $t \in [0, 1]$, the point $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$ can be seen as a *spherical convex combination* of $\boldsymbol{x}$ and $\boldsymbol{y}$.

For computing $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$, we first have to compute α. Although it is tempting to simply compute α using the inverse cosine, there is a problem when $\boldsymbol{x}$ and $\boldsymbol{y}$ are nearly parallel since the inverse cosine is numerically instable for values of the argument near unity. Therefore, we recommend to use the following formula to compute α:

$$\alpha = \begin{cases} 2\arcsin\left(\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|\right) & \text{if } \langle \boldsymbol{x}, \boldsymbol{y} \rangle \geq 0, \\ \pi - 2\arcsin\left(\frac{1}{2}\|\boldsymbol{x} + \boldsymbol{y}\|\right) & \text{if } \langle \boldsymbol{x}, \boldsymbol{y} \rangle < 0. \end{cases} \tag{3}$$

For computing $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$ we proceed as follows. First note that

$$\|\tilde{\boldsymbol{y}}\|^2 = \|\boldsymbol{y}\|^2 - 2\langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \cdot \|\boldsymbol{x}\|^2 = 1 - \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 = 1 - \cos^2(\alpha) = \sin^2(\alpha)$$

and therefore $\|\tilde{\boldsymbol{y}}\| = \sin(\alpha)$. Hence, $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$ can be computed as follows

$$\begin{aligned} \boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t) &= \cos(t\alpha)\boldsymbol{x} + \sin(t\alpha)\frac{\tilde{\boldsymbol{y}}}{\|\tilde{\boldsymbol{y}}\|} \\ &= \cos(t\alpha)\boldsymbol{x} + \frac{\sin(t\alpha)}{\sin(\alpha)}(\boldsymbol{y} - \langle \boldsymbol{x}, \boldsymbol{y} \rangle \boldsymbol{x}) \\ &= \left[\cos(t\alpha) - \frac{\sin(t\alpha)}{\sin(\alpha)}\langle \boldsymbol{x}, \boldsymbol{y} \rangle\right]\boldsymbol{x} + \frac{\sin(t\alpha)}{\sin(\alpha)}\boldsymbol{y} \\ &= \left[\frac{\cos(t\alpha)\sin(\alpha) - \sin(t\alpha)\cos(\alpha)}{\sin(\alpha)}\right]\boldsymbol{x} + \frac{\sin(t\alpha)}{\sin(\alpha)}\boldsymbol{y} \\ &= \frac{\sin(\alpha - t\alpha)}{\sin(\alpha)}\boldsymbol{x} + \frac{\sin(t\alpha)}{\sin(\alpha)}\boldsymbol{y} \\ &= \frac{\sin((1-t)\alpha)}{\sin(\alpha)}\boldsymbol{x} + \frac{\sin(t\alpha)}{\sin(\alpha)}\boldsymbol{y}\,. \end{aligned}$$

Note that $\sin(\alpha)$ can be computed as a byproduct when we compute α by (3),

$$\sin(\alpha) = \begin{cases} \|\boldsymbol{x}-\boldsymbol{y}\|\sqrt{\frac{1+\langle \boldsymbol{x},\boldsymbol{y}\rangle}{2}} & \text{if } \langle \boldsymbol{x},\boldsymbol{y}\rangle \geq 0, \\ \|\boldsymbol{x}+\boldsymbol{y}\|\sqrt{\frac{1-\langle \boldsymbol{x},\boldsymbol{y}\rangle}{2}} & \text{if } \langle \boldsymbol{x},\boldsymbol{y}\rangle < 0. \end{cases}$$

In the case $\langle \boldsymbol{x},\boldsymbol{y}\rangle \geq 0$, this follows from

$$\sin(\alpha) = 2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right) = 2\frac{\|\boldsymbol{x}-\boldsymbol{y}\|}{2}\sqrt{\frac{1+\cos(\alpha)}{2}} = \|\boldsymbol{x}-\boldsymbol{y}\|\sqrt{\frac{1+\langle \boldsymbol{x},\boldsymbol{y}\rangle}{2}}$$

and analogously when $\langle \boldsymbol{x},\boldsymbol{y}\rangle < 0$. Therefore, to compute $\boldsymbol{\gamma}_{\boldsymbol{x},\boldsymbol{y}}(t)$ we use the function GREATCIRCLE, given in pseudocode in Algorithm 11.

Algorithm 11 Point on a great circle

```
 1: function greatCircle(x, y, t)
 2:     sp ← ⟨x, y⟩
 3:     if sp ≥ 0 then
 4:         sum ← ‖x − y‖²
 5:         α ← 2 · arcsin(0.5 · sqrt(sum))
 6:         sina ← sqrt(sum · (1 + sp)/2)
 7:     else
 8:         sum ← ‖x + y‖²
 9:         α ← π − 2 · arcsin(0.5 · sqrt(sum))
10:         sina ← sqrt(sum · (1 − sp)/2)
11:     gx ← (1 − t) · α
12:     gy ← t · α
13:     if (Bound = y) and (abs(gy) > π/2) then
14:         if gy > 0 then gy ← π/2 else gy ← −π/2
15:         gx ← α − gy
16:     cx ← sin(gx)/sina
17:     cy ← sin(gy)/sina
18:     z ← cx · x + cy · y
19:     return z
```

In the Nelder-Mead method, the procedure GREATCIRCLE will be used for the operations Reflection, Expansion, Contraction and Shrinking. For t such that $t\alpha \notin [-\pi/2, \pi/2]$ the new point lies outside the closed hemisphere with pole $\boldsymbol{x}$. To avoid such a behavior the value of $t\alpha$ could be limited to the interval $[-\pi/2, \pi/2]$ in the routine GREATCIRCLE. This is described in lines 13 to 15 of Algorithm 11.

Besides generalizing the basic operations of the Nelder-Mead algorithm, we also have to generalize the notion of a centroid to the case of a sphere. Given points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in \mathbb{S}^{d-1}$, we have to define a *spherical centroid* $c(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) \in$

$\mathbb{S}^{d-1}$. It is not quite clear how to do this in a sensible way. A natural generalization seems to be the so-called *Frchet mean* (also called *Riemannian center of mass* or *Karcher mean*, see Afsari, 2011; Grove and Karcher, 1973; Grove et al., 1974a,b) of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ on a Riemannian manifold $\mathcal{M}$ with distance function $\operatorname{dist}(\cdot, \cdot)$, which is defined to be the minimizer of the sum of squared distances,

$$c_F(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathcal{M}} \sum_{i=1}^{n} \operatorname{dist}^2(\boldsymbol{x}, \boldsymbol{p}_i)\,.$$

One problem with the Frchet mean is that it needs not be unique. To see this, consider the case of two points on the 2-sphere $\mathbb{S}^2$. If the two points are the north and the south pole, respectively, then the whole equator minimizes the sum of squared distances which shows that the Frchet mean is not unique in this case. However, if the points are contained in an open hemisphere, then the Frchet mean on the sphere is unique, see Buss and Fillmore (2001). Nevertheless, there is no closed form expression for the Frchet mean, instead it has to be computed by solving an optimization problem.

A much easier possibility would be to compute the centroid in the ambient space $\mathbb{R}^d$ and then project the point back on the unit sphere $\mathbb{S}^{d-1}$, i.e.,

$$c_N(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) = \frac{\overline{\boldsymbol{p}}}{\|\overline{\boldsymbol{p}}\|}\,, \quad \text{where } \overline{\boldsymbol{p}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{p}_i\,.$$

Although this mean is also not unique when $\overline{\boldsymbol{p}} = \boldsymbol{0}$, it is widely used in directional statistics, see, e.g., Ley and Verdebout (2017). We will call this point the *naïve mean*.

Comparing the naïve mean and the Frchet mean, we decided to choose the naïve mean in our simulations. The reason is that in the long run the simplex formed by $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_d$ should be small. If we only consider a small region of the sphere then its geometry is "nearly flat". Therefore, if the simplex is small, the difference between the Frchet mean and the naïve mean should also be small. Thus, we expect no great difference in performance between using the naïve mean and the Frchet mean in the Nelder-Mead method.

The Nelder-Mead algorithm, adopted to the case of $\mathbb{S}^{d-1}$ as the domain, is

given as Algorithm 12. Since in the operations Reflection, Expansion and Contraction only the worst point $\boldsymbol{p}_d$ is changed, the sequence $(\boldsymbol{p}_1, f_1), \ldots, (\boldsymbol{p}_{d-1}, f_{d-1})$ is still ordered so that $f_1 \leq \cdots \leq f_d$. Therefore in line 28 of Algorithm 12 the routine inPlaceMerge can be used. This routine expects two sorted sequences as arguments and merges them into one sorted sequence. This is more efficient than sorting the whole sequence. Only in the case when Shrinking is applied the subsequence $(\boldsymbol{p}_1, f_1), \ldots, (\boldsymbol{p}_{d-1}, f_{d-1})$ has to be sorted which is done in line 27.

Algorithm 12 Spherical Nelder-Mead method

```
 1: function sphericalNelderMead(z, X)
 2:     if Start = Mn then u ← z − x̄
 3:     if Start = Rn then u ← rndSphere(d)
 4:     ϵ ← (π/2)/β                                   ▷ size of the spherical cap
 5:     for i ← 1 to d do                             ▷ finding the starting simplex
 6:         p_i ← rndSphericalCap(u, ϵ)
 7:         f_i ← D(⟨p_i, z⟩ | ⟨p_i, X⟩)
 8:     sort([(p_1, f_1), ..., (p_d, f_d)])           ▷ sort pairs (p_i, f_i) such that f_1 ≤ ··· ≤ f_d
 9:     repeat
10:         x_o ← c_N(p_1, ..., p_{d−1})
11:         x_r ← greatCircle(x_o, p_d, −α)           ▷ reflected point
12:         f_r ← D(⟨x_r, z⟩ | ⟨x_r, X⟩)
13:         if f_1 ≤ f_r < f_{d−1} then (p_d, f_d) ← (x_r, f_r)
14:         else if f_r < f_1 then
15:             x_e ← greatCircle(x_o, x_r, γ)        ▷ expanded point
16:             f_e ← D(⟨x_e, z⟩ | ⟨x_e, X⟩)
17:             if f_e < f_r then (p_d, f_d) ← (x_e, f_e) else (p_d, f_d) ← (x_r, f_r)
18:         else                                      ▷ f_{d−1} ≤ f_r
19:             if f_r < f_d then x_h ← x_r else x_h ← p_d
20:             x_c ← greatCircle(x_o, x_h, ρ)        ▷ contracted point
21:             f_c ← D(⟨x_c, z⟩ | ⟨x_c, X⟩)
22:             if f_c < f_d then (p_d, f_d) ← (x_c, f_c)
23:             else                                  ▷ reduction
24:                 for i ← 2 to d do
25:                     p_i ← greatCircle(p_1, p_i, σ)
26:                     f_i ← D(⟨p_i, z⟩ | ⟨p_i, X⟩)
27:                 sort([(p_1, f_1), ..., (p_{d−1}, f_{d−1})])
28:         inPlaceMerge([(p_1, f_1), ..., (p_{d−1}, f_{d−1})], [(p_d, f_d)])   ▷ put (p_d, f_d) in the
                                                      ▷ correct position
29:     until stopping criterion is satisfied
```

In our implementation we used a common choice for the parameters, namely $\alpha = 1$, $\gamma = 2$, $\rho = \sigma = 0.5$. The stopping criterion was again chosen in order to guarantee that a specified number N of evaluations of the univariate depth was not exceeded. The starting simplex was chosen as follows. From an ϵ-spherical cap d points were chosen randomly using the procedure rndSpericalCap. The

pole of this cap was determined by a parameter `Start` with possible values `Mn` and `Rn`. For `Start=Rn` the pole of the cap was randomly drawn from $\mathcal{U}(\mathbb{S}^{d-1})$ whereas for `Start=Mn` the pole is given by $\boldsymbol{u} = \boldsymbol{z} - \overline{\boldsymbol{x}}$. The size ϵ of the cap was controlled by a parameter β via $\epsilon = (\pi/2)/\beta$. A further parameter `Bound` controlled whether the movement along the great circle in the routine GREAT-CIRCLE$(\boldsymbol{x}, \boldsymbol{y}, t)$ was limited to a maximum distance of $\pi/2$ between $\boldsymbol{x}$ and the new point (`Bound=y`) or not (`Bound=n`). Further, we compared the adapted version of the Nelder-Mead algorithm with applying the ordinary Nelder-Mead algorithm to the function $\tilde{\phi}_{\boldsymbol{z},\boldsymbol{X}}$. This was controlled by a parameter `Space` having possible values `Ec` (Euclidean space) and `Sp` (Sphere). The tuning of the parameters `Start`, β, `Bound` and `Space` is again described in Section 4.2.

4. Simulation comparison

This section comprises the description and the results of our simulation comparison. In Section 4.1 we define the benchmark distributions and describe the experimental study. Section 4.2 indicates the sets over which parameters are tuned, and concludes on their final choice. Section 4.3 analyses the results of the simulation study.

4.1. Distributional and simulation settings

The simulation study is based on the depth computation of a point $\boldsymbol{z}$ w.r.t. a sample $\boldsymbol{X}$ of n i.i.d. d-variate points. The point $\boldsymbol{z}$ is taken to be the average of 10 arbitrary points of the sample. This guarantees that it belongs to the convex hull of the data so that the depth is always strictly positive, but also does not place it too deep in the data set to preserve the random nature of the choice of $\boldsymbol{z}$ (since only 10 points out of 1000 are averaged). The following six distributions are used for the fine-tuning (due to the affine-invariance of the considered depths we do not introduce any correlation structure):

- the standard normal distribution $\mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$;
- the spherically-symmetric Student t_5 distribution;

- the spherically-symmetric Student t_1 (Cauchy) distribution;
- the uniform distribution on $[0,1]^d$;
- the skewed normal distribution generated in the following way (Azzalini, 2013): let $U \sim \mathcal{U}([0,1])$, $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$, U and $\boldsymbol{Z}$ stochastically independent, and $\boldsymbol{\delta} \in \mathbb{R}^d$ be a skewness parameter. Then the skewed normal random vector equals

$$\boldsymbol{X} \stackrel{\mathrm{d}}{=} \begin{cases} \boldsymbol{Z} & \text{if } U \leq \Phi(\boldsymbol{\delta}^T \boldsymbol{Z}), \\ -\boldsymbol{Z} & \text{if } U > \Phi(\boldsymbol{\delta}^T \boldsymbol{Z}), \end{cases}$$

  where $\Phi(\cdot)$ is the c.d.f. of the standard normal distribution. We set $\boldsymbol{\delta} = (5, 0, ..., 0)^T$;
- a product of d independent exponential distributions (with parameter $\lambda = 1$).

In the simulation study, we consider the five depth notions described in Section 2.1. An explicit formula makes it unnecessary to approximate the Mahalanobis depth, which is in addition a quadratic (and thus everywhere smooth) function. Since rather good results for optimization techniques are expected, we include it for a qualitative comparison with random algorithms. Since the zonoid depth can be computed efficiently even in higher dimensions using the algorithm of Dyckerhoff et al. (1996), it is also included as a benchmark.

When fine-tuning the algorithms, for each depth, for $n = 1000$ points, in dimensions $d = 5, 10, 15, 20$, we use $N \approx 1000$ random directions for each of the algorithms and each combination of parameters. For each of the distributions, we repeat the computation of the depth 1000 times and summarize the results using two statistics based on the following idea. Since all the considered algorithms report an upper bound on the actual depth, for the same $\boldsymbol{z}$ and a data set one can compare parameters according to the obtained depth values, because lower obtained depth is always closer to the exact value. We do it by reporting:

- the average rank of the obtained depth approximation (among all considered parameter combinations) over 1000 runs (the lower the better, with 1 being the best), which we shortly denote as AveRank;

- the percentage when the considered set of parameters achieved the smallest depth value (among all parameter combinations) over 1000 runs (the higher the better, with 100% being the best), which we shortly denote as PercBest.

We consider AveRank as the more important criterion. The reason is that AveRank not only focuses on how many times a method is the best method (as does PercBest), but also takes into account the rank of a method when it is not the best method. Often the best methods according to AveRank and PercBest coincide. Therefore, we restrict to AveRank in the presentation, and present both statistics in the Supplementary Materials.

4.2. Fine-tuning of the algorithms

Before running the simulation study in Section 4.3, the variety of possible settings of the algorithms should be reduced to a proper choice of (nearly optimal) parameters. We do this by means of a comparative benchmark. The chosen parameters are then fixed for each algorithm throughout the subsequent sections. Since the task of parameter tuning is to choose parameters for a single algorithm, we run (and analyze) each algorithm separately.

For each algorithm, out of a preliminary chosen range, we select a suitable set of parameter values based on visual inspection and additional analysis in detail if necessary. Note, that even with such an extensive simulation the validity of the chosen parameters is still limited. We restrict the tuning process to rather small ranges of parameters since only several distributions are considered. Also, such a tuning provides a certain degree of robustness, which is especially desirable because the single chosen set of parameters will be used for all further experiments. Finally, this simplifies visual presentation and manual analysis. Table 1 summarizes the parameters' ranges and their choices.

Method	Parameters	Values	Description
RRS	N_{ref}	5, **10**, 15	Number of refinement steps
	α	0.2, **0.5**, 0.8	Shrinking factor of the spherical cap
RGS	N_{ref}	5, **10**, 15	Number of refinement steps
	α	0.2, **0.5**, 0.8	Shrinking factor of the spherical cap
RaSi	α	**1.25**, 1.5, 1.8	Parameter of the Dirichlet distribution
SA	α	0.5, 0.8, **0.95**	Cooling factor
	β	5, **10**, 15	Size of the spherical cap
	`Start`	**Mn**, `Rn`	Starting value (mean, random)
CD	`Space`	`Ec`, **Sp**	Euclidean space or sphere
	`LS`	`Eq`, **GS**	Line search: equally spaced or golden section
NM	`Space`	`Ec`, **Sp**	Euclidean space or sphere
	`Start`	**Mn**, `Rn`	Starting value (mean, random)
	β	**1**, 2, 4	Size of the spherical cap
	`Bound`	**y**, `n`	Bound movement on great circles (yes/no)

Table 1: Considered and selected (in bold) parameters, obtained by the fine-tuning.

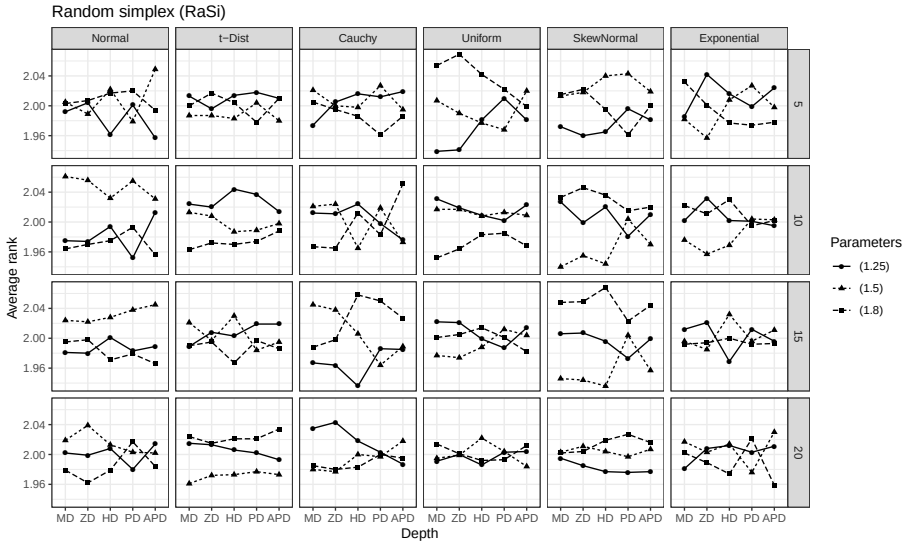

Figure 3: AveRank for the random simplices (RaSi) algorithm. The only parameter is α, taking values in $\{1.25, 1.5, 1.8\}$.

Algorithms *random search* (RS) and *grid search* (GS) do not require hyper-parameters and thus need not to be tuned. For the six remaining algorithms, we detail the parameter-tuning process below. All figures indicating computed statistics can be found in the Supplementary Materials in Section 1. Here we

575 only place two illustrative ones: an example of a mixed result (random simplices, Figure 3) and of a clear winner (coordinate descent, Figure 4).

- *Refined random search* (RRS): Visual inspection and summary statistics (average over all experiments) give no clear winner. However, the combinations with a high value of N_{ref} and a small value of α or vice versa, are
580 clearly inferior. Therefore we choose a pair where both parameters are in the middle of the ranges.

- *Refined grid search* (RGS): AveRank statistics mostly hint on the best pair here.

- *Random simplices* (RaSi): AveRank statistics, plotted in Figure 3, do
585 not help in choosing the best parameter. One observes that the concentration parameter of the Dirichlet seems not to have much influence on the performance of the algorithm. We decided to allow for most freedom

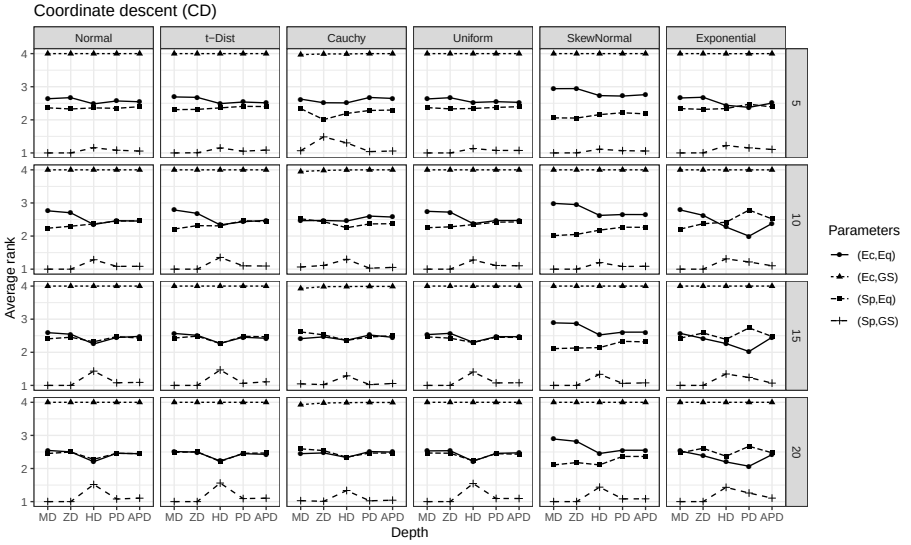

Figure 4: AveRank for the coordinate descent (CD) algorithm. The first parameter (`Space`) determines the space to be used: Euclidean space (`Ec`) or sphere (`Sp`). The second parameter (`LS`) defines the algorithm to be used for the line search: evaluation over an equidistant grid (`Eq`) or the golden section algorithm (`GS`).

when drawing the direction.

- *Simulated annealing* (SA): Substantial number of the considered alternatives complicates the choice of the best performing parameter combination. The parameter choice is thus made based on the summary statistics, i.e., by averaging over all considered simulation settings.

- *Coordinate descent* (CD): AveRank of the four considered settings is depicted in Figure 4. One clearly distinguishes superiority of using great circles' coordinate system and the golden section technique for the line search.

- *Nelder-Mead* (NM): First, running usual NM in the Euclidean space performs worse than when sticking to the geometry of the hyper-sphere. One can further notice that starting with the spherical cap around the direction from the data mean to $\boldsymbol{z}$, drawing the initial simplex from the entire hemisphere, as well as forcing the simplex to be always contained in a hemisphere gives on average better results.

Several limitations of the above fine-tuning need to be mentioned, which hold true for the simulation comparison in Section 4.3 as well:

- Strictly speaking, the tuning is subject to the chosen distributions, dimensions and ranges of parameters. Further parameters of the algorithms, not mentioned here (e.g. constants of the Nelder-Mead algorithm, see Section 3.8) were kept unchanged on their default values.

- The number of random directions is fixed to $N \approx 1000$, which is a budget constraint. Thus, it is possible that one algorithm would approximate better with a few more directions, while another one would not make use of these additional directions. On contrary, with less directions the results could look differently. On the other hand, the simulation study of Section 4.3 illustrates only weak dependence of our conclusions on the change of the number of random directions.

4.3. Results of the simulation study

To compare the performance of the algorithms, we run a simulation study for the distributional settings from Section 4.1 and three additional distributions that were not used for fine-tuning:

- the hemispherical shell distribution (abbreviated as "Shell") generated as follows: let $(S_1, ..., S_{d-1}, S_d)^T \overset{\mathrm{d}}{=} \boldsymbol{S} \sim \mathcal{U}(\mathbb{S}^{d-1})$, $U \sim \mathcal{U}([0.9, 1])$, $\boldsymbol{S}$ and U stochastically independent. The random vector stemming from the shell distribution equals

$$\boldsymbol{X} \overset{\mathrm{d}}{=} U \cdot (S_1, ..., S_{d-1}, |S_d|)^T ;$$

- the bimodal normal mixture generated as follows: let $B \sim \mathcal{B}(0.5)$, $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$, B and $\boldsymbol{Z}$ stochastically independent. To model the bimodal normal mixture we use the following random vector:

$$\boldsymbol{X} \overset{\mathrm{d}}{=} \begin{cases} \boldsymbol{Z} + 2\boldsymbol{e}_1 & \text{if } B = 1\,, \\ \boldsymbol{Z} - 2\boldsymbol{e}_1 & \text{if } B = 0\,, \end{cases}$$

  where $\boldsymbol{e}_j$ is the j-th canonical unit vector;

- the multimodal normal mixture generated as follows: let $I \sim \mathcal{U}(\{1, ..., d\})$, $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$, I and $\boldsymbol{Z}$ stochastically independent, then the considered random vector equals

$$\boldsymbol{X} \overset{\mathrm{d}}{=} \boldsymbol{Z} + 3\boldsymbol{e}_I\,.$$

Further, to gain more insight into the performance of the methods, the two ordinal measures AveRank and PercBest are complemented with two cardinal measures: The mean absolute error (MAE) is calculated as the difference between the obtained depth approximation and the exact depth, averaged over all $N_{sim} = 1000$ runs, and the mean relative error (MRE) is calculated as MAE divided by the exact depth and averaged again over all $N_{sim} = 1000$ runs. However, note that the computation of the exact depth values is (in reasonable time) possible only for the Mahalanobis depth and the zonoid depth.

Therefore, when computing MAE and MRE for the other depths, the exact depth value is replaced by the minimum depth achieved by any of the methods to be compared.

We fix the parameters of the methods to the values chosen in Section 4.2 and take $N \approx 100$, 1000, and 10000 projections. Refined grid search (RGS) could not be used for $N \approx 100$ projections since for $N_{ref} = 10$ only ten projections could be used for the grid in each refinement step which is too small even for $d = 5$. In this section, only results for $N \approx 1000$ directions are analyzed, since those for the other numbers of directions are similar. The complete results in graphical and tabular form can be found in the Supplementary Materials, Sections 2.1 and 2.2, respectively.

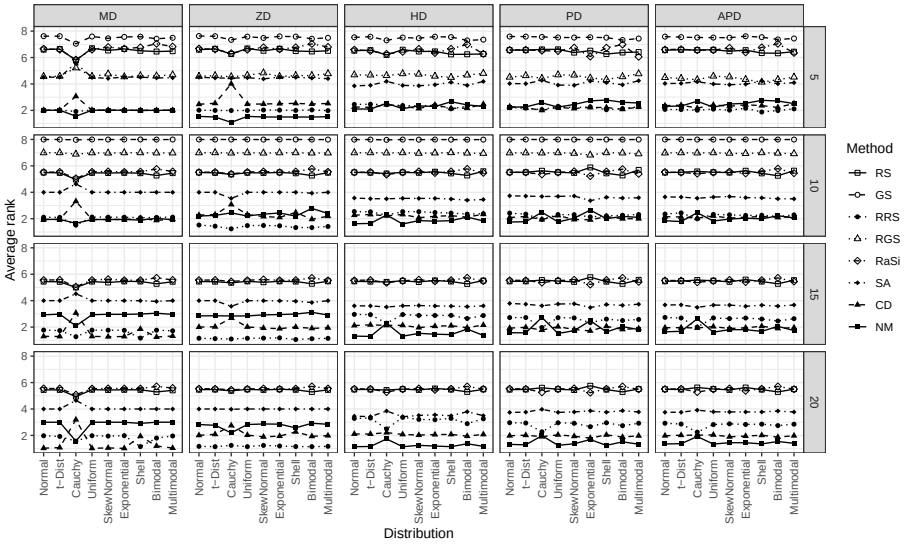

Figure 5: AveRank statistics for the eight considered approximation methods when using the parameter settings from Section 4.2 (see also Table 1) and $N \approx 1000$ projections.

Figure 5 exhibits AveRank for each of the eight considered algorithms for different depth notions, distributions, and dimensions (see the Supplementary Materials for further statistics and values of N). Several observations can be made:

- There is a group of methods which have poor performance that further de-

grades with increasing dimension: random search (RS), grid search (GS), refined grid search (RGS), and random simplices (RaSi). Moreover, GS and RGS are not considered in dimension $d > 10$ (as explained in Sections 3.2 and 3.4), because 1000 directions are not sufficient to generate even a very sparse grid in such a high dimension.

- Refined random search (RRS), coordinate descent (CD), and Nelder-Mead (NM) show rather good performance.

- NM shows superior behavior in this latter group, since it possesses almost always lower AveRank compared to the two other methods for the halfspace, projection, and asymmetric projection depths. Thus it can be seen as a general winner. However, it is closely followed by CD.

Comparison of AveRank with PercBest, MAE and MRE statistics reveals similarity of the results with very close overall ranking of the methods. Table 2 illustrates concordance of the four statistics for the halfspace depth (in most of the cases). Since only RRS, CD and NM appear as best-performing methods w.r.t. the four considered performance statistics, only these three methods are included in the table. Tables which show the best performing methods for the other four dephts are contained in the Supplementary Materials, Section 2.3.

To get more insights into the dynamic of the optimization process, we regard the flow of the minimal reached depth with the number of random directions. A typical behavior of the optimization, on the example of the normal distribution in dimension 20, is indicated in Figure 6. Similar figures for all nine considered distributions are gathered in Section 2.4 of the Supplementary Materials.

Our most important observation is a high performance of the optimization techniques (SA, RRS, CD, and NM) compared with the random methods (RS, RaSi). The two latter ones seem to (approximately) follow the bounds derived in Nagy et al. (2020) and are outperformed already before reaching 100 random directions. Further inspection shows that the improvement of simulated annealing (SA) is very weak, and minor improvement can be expected for even higher

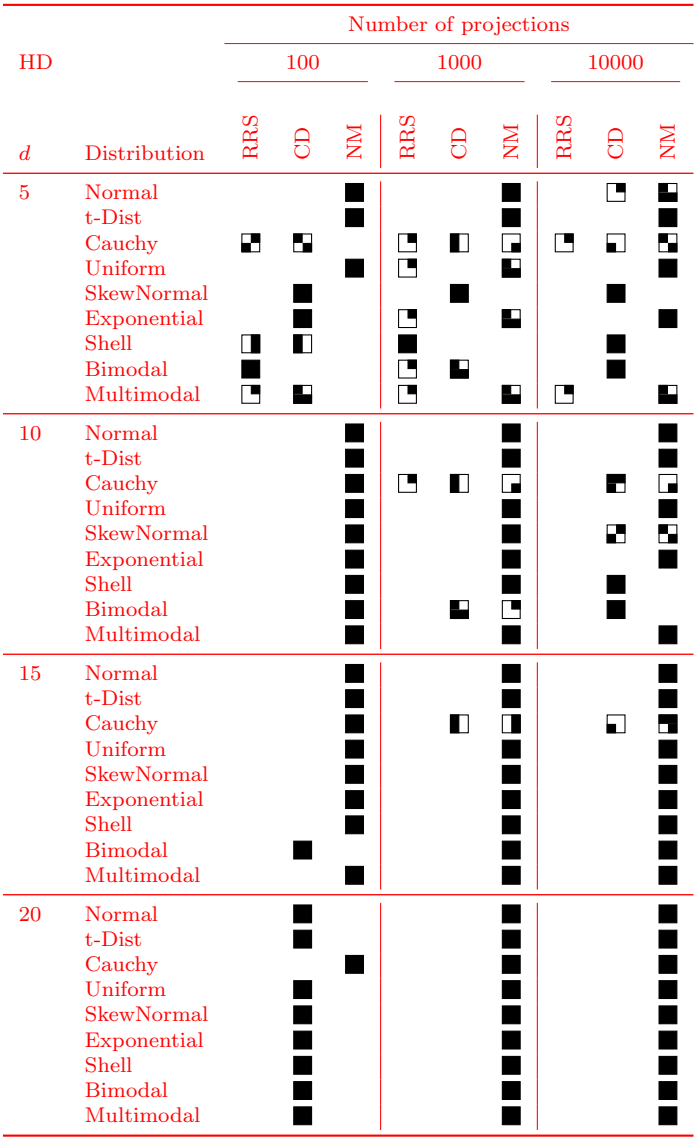

Table 2: Best performing methods in sense of AveRank (top-left square), PercBest (top-right square), MAE (bottom-left square), and MRE (bottom-right square) for the halfspace depth (HD).

number of directions. A possible explanation would be that the parameters of SA should be tuned separately for each setting.

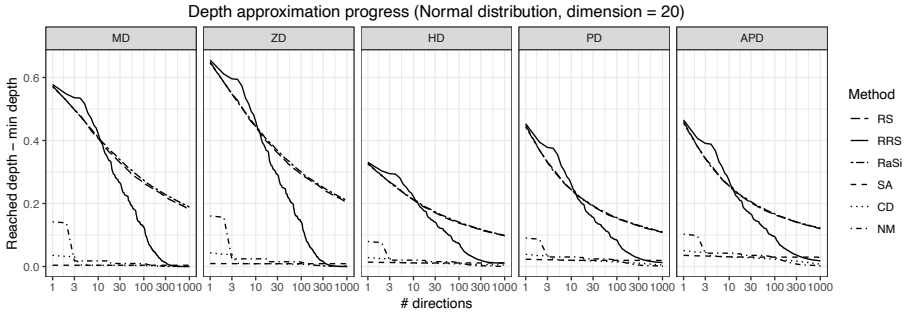

Figure 6: Average (over 1000 runs) difference between the reached depth and the minimally achieved depth (by all methods for the current triplet distribution—depth—dimension) during the optimization process (normal distribution, dimension $d = 20$). (The lines of RS and RaSi almost coincide in the graphs.)

5. Results for the approximation error

Apart from knowing which of the discussed approximation methods gives the best approximation, it is also of interest to have information on the approximation error. To calculate the approximation error, the exact values of the depths have to be known. From the considered depths only the Mahalanobis depth and the zonoid depth can be exactly computed in high dimensions in reasonable time. For the halfspace depth there exists an exact algorithm (Dyckerhoff and Mozharovskyi, 2016) to compute the depth in arbitrary dimension which has a complexity of $O(n^{d-1} \log n)$. For the considered sample size of $n = 1000$ the exact computation of the halfspace depth is (in reasonable time) only possible when $d \leq 5$. Although there is an exact algorithm for the projection depth (Liu and Zuo, 2014), the considered sample size of $n = 1000$ in dimension $d = 5$ is already too large to have the value of the depth computed in reasonable time. For the asymmetric projection depth no exact algorithm exists. Therefore we decided to have a closer look at two situations. First, we choose to examine the approximation of the halfspace depth since it probably is the most prominent depth. Because of the high computational cost we computed approximation errors for the halfspace depth only in dimension $d = 5$. Second, to get some intuition on approximation errors in high dimensions we analyzed the mean

absolute error (MAE) and the mean relative error (MRE) which we exactly computed for the zonoid (and also the Mahalanobis) depth in all dimension $d = 5, 10, 15, 20$. The zonoid depth was chosen since apart from the trivial case of the Mahalanobis depth, it is the only depth considered in this study for which exact computation is possible when $d = 20$. The time for computing the zonoid depth of a single point w.r.t. a sample of size $n = 1000$ in dimension $d = 20$ is still under one second. For both setups (halfspace depth, $d = 5$, and zonoid depth, $d \leq 20$) we used the same simulated datasets that were already used in Section 4.3. For the halfspace depth we only used the first fifteen simulated datasets to compute the approximation errors. The approximated depth values as well as the exact depth values for the fifteen datasets simulated from the normal distribution are shown for the halfspace depth in Table 3. The last two lines of the table show the mean absolute error (MAE) and the mean relative error (MRE) of the considered approximation methods. The respective tables for all nine distributions and $N \approx 100, 1000, 10000$ projections are given in Section 3 of the Supplementary Materials.

	RS	GS	RRS	RGS	RaSi	SA	CD	NM	Exact
1	0.279	0.286	0.267	0.273	0.279	0.269	0.268	0.269	0.265
2	0.143	0.160	0.132	0.139	0.148	0.136	0.132	0.130	0.128
3	0.244	0.259	0.238	0.245	0.247	0.241	0.238	0.238	0.236
4	0.329	0.340	0.310	0.336	0.324	0.316	0.311	0.311	0.309
5	0.220	0.235	0.200	0.208	0.225	0.204	0.199	0.200	0.197
6	0.236	0.259	0.214	0.229	0.227	0.219	0.216	0.217	0.213
7	0.238	0.242	0.230	0.235	0.244	0.233	0.227	0.229	0.226
8	0.228	0.229	0.218	0.225	0.229	0.223	0.219	0.218	0.215
9	0.171	0.169	0.152	0.153	0.164	0.154	0.151	0.152	0.149
10	0.241	0.248	0.224	0.230	0.228	0.225	0.223	0.223	0.221
11	0.187	0.205	0.170	0.182	0.185	0.169	0.169	0.169	0.166
12	0.284	0.280	0.269	0.273	0.288	0.274	0.272	0.271	0.268
13	0.221	0.228	0.201	0.218	0.212	0.206	0.206	0.202	0.200
14	0.171	0.182	0.161	0.161	0.176	0.160	0.158	0.157	0.154
15	0.168	0.188	0.157	0.158	0.168	0.161	0.161	0.157	0.157
MAE	0.017	0.027	0.003	0.011	0.016	0.006	0.003	0.003	0.000
MRE	0.105	0.169	0.017	0.063	0.099	0.035	0.019	0.016	0.000

Table 3: Exact and approximate values of the halfspace depth for 15 points together with the corresponding MAE and MRE: normal distribution, $n = 1000$, $d = 5$, $N \approx 1000$ projections.

Because of its good computability, for the zonoid depth we used all 1000

simulated datasets for each combination of dimension and distribution. In Table 4 the mean relative errors (MRE) are shown for the case where 1000 projections were used. The respective tables for both the mean relative errors (MRE) and the mean absolute errors (MAE) for all nine distributions and $N \approx 100, 1000, 10000$ projections are shown in the Supplementary Materials in Section 2.2.2.

		Approximation algorithm							
d	Distribution	RS	GS	RRS	RGS	RaSi	SA	CD	NM
5	Normal	0.018782	0.040681	0.000002	0.002805	0.019107	0.000535	0.000004	0.000001
	t-Dist	0.021093	0.047018	0.000002	0.003281	0.021418	0.000597	0.000008	0.000001
	Cauchy	0.043137	0.080894	0.000465	0.016957	0.045022	0.010639	0.017534	0.000056
	Uniform	0.018799	0.041453	0.000002	0.003367	0.018021	0.000537	0.000004	0.000001
	SkewNormal	0.021533	0.046762	0.000002	0.005977	0.029959	0.000643	0.000005	0.000001
	Exponential	0.021569	0.047682	0.000002	0.003068	0.020915	0.000653	0.000007	0.000001
	Shell	0.022559	0.052451	0.000002	0.003401	0.032908	0.000665	0.000005	0.000001
	Bimodal	0.023904	0.052215	0.000002	0.003452	0.037974	0.000727	0.000005	0.000001
	Multimodal	0.022954	0.048468	0.000002	0.006329	0.033007	0.000691	0.000006	0.000001
10	Normal	0.217197	1.118694	0.000025	0.942766	0.219397	0.007122	0.000057	0.000094
	t-Dist	0.222892	1.289776	0.000036	1.094230	0.228138	0.008009	0.000094	0.000168
	Cauchy	0.303627	2.212219	0.006813	1.661869	0.307660	0.064152	0.052861	0.034096
	Uniform	0.215653	1.090465	0.000023	0.937243	0.219798	0.006999	0.000054	0.000098
	SkewNormal	0.218591	1.081675	0.000028	0.923719	0.237190	0.008551	0.000062	0.000228
	Exponential	0.229525	1.201967	0.000037	1.001438	0.232001	0.009666	0.000090	0.000489
	Shell	0.222987	1.076789	0.000028	0.916450	0.244420	0.007856	0.000122	0.000214
	Bimodal	0.234134	1.111908	0.000030	0.940334	0.301895	0.008984	0.000063	0.002062
	Multimodal	0.219745	1.078995	0.000027	0.906989	0.225948	0.009091	0.000072	0.000280
15	Normal	0.585529	—	0.000182	—	0.609475	0.022943	0.000392	0.001717
	t-Dist	0.575809	—	0.000277	—	0.596498	0.027329	0.000737	0.002582
	Cauchy	0.676690	—	0.020609	—	0.692426	0.166416	0.084469	0.097217
	Uniform	0.579684	—	0.000160	—	0.588454	0.022043	0.000365	0.001557
	SkewNormal	0.581892	—	0.000200	—	0.619134	0.027594	0.000437	0.003568
	Exponential	0.607952	—	0.000343	—	0.622455	0.033453	0.000909	0.006960
	Shell	0.581782	—	0.000186	—	0.600450	0.025505	0.000951	0.008322
	Bimodal	0.600136	—	0.000206	—	0.719552	0.030843	0.000546	0.014352
	Multimodal	0.576272	—	0.000180	—	0.594009	0.026946	0.000436	0.002727
20	Normal	1.143799	—	0.001079	—	1.177983	0.052060	0.002086	0.006032
	t-Dist	1.024786	—	0.001421	—	1.052585	0.060773	0.003030	0.007248
	Cauchy	1.110807	—	0.049412	—	1.125693	0.369281	0.158725	0.099501
	Uniform	1.131364	—	0.001026	—	1.148654	0.050696	0.002042	0.005657
	SkewNormal	1.141836	—	0.001293	—	1.178756	0.070894	0.002138	0.008604
	Exponential	1.151532	—	0.002333	—	1.180201	0.126961	0.004631	0.014133
	Shell	1.195071	—	0.001521	—	1.233843	0.076760	0.004516	0.006101
	Bimodal	1.189500	—	0.001256	—	1.376561	0.119274	0.003015	0.014454
	Multimodal	1.117096	—	0.001127	—	1.157293	0.066462	0.002162	0.006479

Table 4: Mean relative error (MRE) for the approximation of the zonoid depth, $n = 1000$ data points, $N \approx 1000$ projections.

For the halfspace depth and normally distributed data in dimension $d = 5$, when $N \approx 1000$ projections are used, the best methods are NM and RRS, followed be CD, whereas the worst methods are RS, GS and RaSi. Even though there is one case (dataset 15) where RRS and NM found the exact halfspace depth, MAE suggests that on average the best halfspace found by RRS or NM

contains three points more than the optimal halfspace. Furthermore, relative depth approximation error remains (again on an average) below 2% of the exact depth value.

For the zonoid depth, when the approximation was done using $N \approx 1000$ projections, RRS gets the first place (when $d > 5$) followed by CD, NM and SA. An important point to note is that the very basic methods like RS and RaSi are unusable when dimension is high with relative error rates beyond 50% ($d = 15$) or even beyond 100% ($d = 20$). It is noteworthy that the same holds for RRS when the number of projections is low (see the Supplementary Materials), which suggests that RRS needs a substantial number of projections to work well. However, the more elaborate methods like CD, NM and SA perform well regardless of the number of projections. With these methods, relative errors can be kept reasonably low, even below 1.5% (except the Cauchy distribution) of the exact value.

6. Guidelines for practitioners

The current article shows that even depths that require substantial burden for exact computation can be (potentially) well approximated even in higher dimensions in reasonable time. For this, exploiting the geometry of the unit hyper-sphere is definitely advantageous (see, e.g., parameter tuning of CD or NM in Section 4.2). Further, methods based on random projections are clearly outperformed by those launching optimization over the surface of the hyper-sphere. Among the latter ones, NM performs the best, closely followed by CD and RRS. For the optimization techniques, the direction from the sample average to the point of interest seems to be a good initial argument. For the considered depth notions, the time complexity of all algorithms is $O(Nn)$ only, i.e., linear in both number of sample points and random directions, while the running time of the algorithms is very small (on average always below 0.05 second for 1000 directions), see Table 5 for the run time of the algorithms. At the same time, approximation precision seems to be high as well.

d	Depth	Approximation algorithm							
		RS	GS	RRS	RGS	RaSi	SA	CD	NM
5	MD	0.007	0.004	0.007	0.003	0.008	0.007	0.006	0.007
	ZD	0.014	0.007	0.015	0.006	0.016	0.015	0.015	0.014
	HD	0.009	0.004	0.007	0.003	0.010	0.007	0.006	0.007
	PD	0.027	0.015	0.026	0.012	0.028	0.027	0.027	0.026
	APD	0.020	0.011	0.020	0.008	0.022	0.020	0.019	0.019
10	MD	0.011	0.001	0.011	0.001	0.013	0.011	0.010	0.011
	ZD	0.018	0.001	0.019	0.002	0.021	0.020	0.018	0.018
	HD	0.013	0.001	0.010	0.001	0.016	0.011	0.010	0.010
	PD	0.031	0.001	0.030	0.003	0.035	0.031	0.030	0.030
	APD	0.024	0.001	0.024	0.002	0.029	0.024	0.023	0.024
15	MD	0.012	—	0.012	—	0.016	0.012	0.012	0.012
	ZD	0.020	—	0.020	—	0.024	0.021	0.020	0.021
	HD	0.015	—	0.012	—	0.019	0.013	0.011	0.012
	PD	0.033	—	0.032	—	0.037	0.033	0.031	0.032
	APD	0.026	—	0.025	—	0.030	0.026	0.025	0.024
20	MD	0.014	—	0.014	—	0.019	0.014	0.013	0.014
	ZD	0.022	—	0.022	—	0.027	0.023	0.021	0.022
	HD	0.017	—	0.014	—	0.022	0.015	0.013	0.014
	PD	0.035	—	0.034	—	0.040	0.036	0.033	0.033
	APD	0.028	—	0.028	—	0.033	0.028	0.026	0.026

Table 5: Average time (in seconds) of the algorithms for the budget of 1000 directions, over 1000 repetitions. Averaging is also performed over all considered distributions, since running times are independent of distributions. GS and RGS cannot be run in dimensions $d = 15$ and $d = 20$, while in dimension $d = 10$ only a very sparse grid (11 and 110 directions for GS and RGS, respectively) satisfies the condition $N \leq 1000$.

In applications, if possible, it is recommended to first fine-tune the method using either available real data or similar simulated ones. If computational budget allows, it is further advised to benchmark several methods, since they are comparable while having the upward bias, as in Section 4.2. This data-driven fine-tuning can of course be directly incorporated in the depth calculation procedure, which however can scale up computational time depending on the considered choices of parameters and methods.

It is necessary to emphasize the limitations of the entire simulation study. First of all, the presented results and the accompanying analysis is — strictly speaking — valid only for statistical processes which are similar enough to the nine considered distributions. Further, the behavior of the explored performance indicators is unpredictable outside the considered parameter ranges since it can have a non-linear character; this holds for the tuning procedure in Section 4.2

as well. Also, the study is restricted to a sample size of $n = 1000$ observations, while considered dimensions $d = 5, 10, 15, 20$ and numbers of random directions $N = 100, 1000, 10000$ are somewhat liming as well. Likewise, one should be careful when interpreting results of the aggregated (averaging) statistics since they might hide information that could be of use in particular cases. Finally, the approximation algorithms are compared with each other with respect to minimum achieved depth, while exact depth values — being unknown (in most of the cases) — are not addressed, and are only studied for the halfspace depth (in $d = 5$) and the zonoid depth in Section 5.

The presented simulation study can be beneficial beyond the data depth context, i.e., when a function other than univariate depth is optimized over the surface of the unit sphere. As an example, projection pursuit makes use of an optimization algorithm that maximizes a certain projection index. This problem, within the framework of skewness measures, is tackled in Loperfido (2018); see also Franceschini and Loperfido (2017) for an implementation.

The source codes of implementations of the methods described in Section 3 and the reproducing scripts of the experiments, as well as results of the fine-tuning simulation study from Section 4.2 are gathered in the Supplementary Materials. The disk space occupied by the complete results of the main simulation study of Section 4.3 is 103.4 `GB` (with 5.6 `GB` for the generated data, and 0.9 `GB`, 8.8 `GB` and 88.1 `GB` for the simulation results with 100, 1000 and 10000 directions, respectively) and thus cannot be uploaded online. Nevertheless, these results in a compact form, sufficient to reproduce all the illustrations and tables of the article (including the Supplementary Materials, apart from the figures in Section 2.4) are contained in the Supplementary Materials.

Supplementary materials

Supplementary materials to this article include:

- **Additional figures and tables:** Additional figures and tables illustrating more comprehensive results of the experimental study, i.e., illus-

trations to the fine-tuning, simulation results and approximation error. ("ACPD_Supplement.pdf")

- **Reproducing codes:** `C++` codes of all the methods from Section 3 as well as the scripts for reproduction of experiments. ("codes.zip")

- **Experimental results:** Results of the fine-tuning simulation study of Section 4.2 as well as results of the main simulation study in Section 4.3 in condensed form. ("results.zip")

Acknowledgements

The research of Stanislav Nagy was supported by the Czech Science Foundation [grant number 19-16097Y] and by the PRIMUS/17/SCI/3 project of Charles University. The authors greatly acknowledge the comments of the two anonymous referees.

References

Afsari, B., 2011. Riemannian l^p center of mass: existence, uniqueness, and convexity. Proceedings of the American Mathematical Society 139, 655–673.

Azzalini, A., 2013. The Skew-Normal and Related Families. Institute of Mathematical Statistics Monographs, Cambridge University Press.

Buss, S.R., Fillmore, J.P., 2001. Spherical averages and applications to spherical splines and interpolation. ACM Transactions on Graphics 20, 95–126.

Cascos, I., 2010. Data depth: Multivariate statistics and geometry, in: New perspectives in Stochastic Geometry. Oxford University Press, pp. 398–423.

Chen, D., Morin, P., Wagner, U., 2013. Absolute approximation of Tukey depth: Theory and experiments. Computational Geometry: Theory and Applications 46, 566–573.

Christmann, A., Fischer, P., Joachims, T., 2002. Comparison between various regression depth methods and the support vector machine to approximate the minimum number of misclassifications. Computational Statistics 17, 273–287.

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

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

Dutta, S., Ghosh, A.K., 2012. On robust classification using projection depth. Annals of the Institute of Statistical Mathematics 64, 657–676.

Dyckerhoff, R., 2004. Data depths satisfying the projection property. Allgemeines Statistisches Archiv 88, 163–190.

Dyckerhoff, R., Koshevoy, G., Mosler, K., 1996. Zonoid data depth: Theory and computation, in: Prat, A. (Ed.), COMPSTAT '96 – Proceedings in Computational Statistics, Physica-Verlag, Heidelberg. pp. 235–240.

Dyckerhoff, R., Mozharovskyi, P., 2016. Exact computation of the halfspace depth. Computational Statistics and Data Analysis 98, 19–30.

Franceschini, C., Loperfido, N., 2017. MaxSkew: Orthogonal data projections with maximal skewness. R Foundation for Statistical Computing. URL: `https://CRAN.R-project.org/package=MaxSkew`. R package version 1.1.

Golub, G.H., Van Loan, C.F., 1989. Matrix Computations. 2. ed., Johns Hopkins University Press.

Grove, K., Karcher, H., 1973. How to conjugate C^1-close group actions. Mathematische Zeitschrift 132, 11–20.

Grove, K., Karcher, H., Ruh, E.A., 1974a. Group actions and curvature. Inventiones mathematicae 23, 31–48.

Grove, K., Karcher, H., Ruh, E.A., 1974b. Jacobi fields and Finsler metrics on compact Lie groups with an application to differentiable pinching problems. Mathematische Annalen 211, 7–21.

Johnson, D., Preparata, F., 1978. The densest hemisphere problem. Theoretical Computer Science 6, 93–107.

Kageyama, A., Sato, T., 2004. "Yin-Yang grid": An overset grid in spherical geometry. Geochemistry, Geophysics, Geosystems 5, 1–15.

Koshevoy, G., Mosler, K., 1997. Zonoid trimming for multivariate distributions. The Annals of Statistics 25, 1998–2017.

Kurihara, Y., 1965. Numerical integration of the primitive equations on a spherical grid. Monthly Weather Review 93, 399–415.

Ley, C., Verdebout, T., 2017. Modern Directional Statistics. Chapman and Hall/CRC, Boca Raton, FL.

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

Liu, R.Y., Parelius, J.M., Singh, K., 1999. Multivariate analysis by data depth: Descriptive statistics, graphics and inference. The Annals of Statistics 27, 783–858.

Liu, X., Zuo, Y., 2014. Computing projection depth and its associated estimators. Statistics and Computing 24, 51–63.

Loperfido, N., 2018. Skewness-based projection pursuit: A computational approach. Computational Statistics & Data Analysis 120, 42–57.

Mahalanobis, P.C., 1936. On the generalized distance in statistics. Proceedings of the National Institute of Sciences of India 12, 49–55.

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, Berlin, pp. 17–34.

Mozharovskyi, P., Mosler, K., Lange, T., 2015. Classifying real-world data with the $DD\alpha$-procedure. Advances in Data Analysis and Classification 9, 287–314.

Nagy, S., Dyckerhoff, R., Mozharovskyi, P., 2020. Uniform convergence rates for the approximated halfspace and projection depth. Electronic Journal of Statistics 14, 3939–3975.

Nelder, J.A., Mead, R., 1965. A simplex method for function minimization. The Computer Journal 7, 308–313.

Ronchi, C., Iacono, R., Paolucci, P., 1996. The "cubed sphere": A new method for the solution of partial differential equations in spherical geometry. Journal of Computational Physics 124, 93–114.

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

Shao, W., Zuo, Y., 2012. Simulated annealing for higher dimensional projection depth. Computational Statistics & Data Analysis 56, 4026–4036.

Snyder, J.P., 1987. Map Projections: A Working Manual. Professional Paper 1395. U. S. Geological Survey. Washington D.C.

Tukey, J.W., 1975. Mathematics and the picturing of data, in: James, R. (Ed.), Proceedings of the International Congress of Mathematicians, Canadian Mathematical Congress. pp. 523–531.

Williamson, D.L., 1968. Integration of the barotropic vorticity equation on a spherical geodesic grid. Tellus 20, 642–653.

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