Contributions to depth-based classification and computation of the Tukey depth

Inauguraldissertation
zur
Erlangung des Doktorgrades
der
Wirtschafts- und Sozialwissenschaftlichen Fakultät
der
Universität zu Köln

2014

vorgelegt
von

Pavlo Mozharovskyi

aus

Kiew, Ukraine

Referent:	Prof. Dr. Karl Mosler
Korreferent:	Prof. Dr. Jörg Breitung
Tag der Promotion:	07. Juli 2014

To my parents
and my grandmother

Acknowledgements

First of all I want to express my gratitude to my PhD-supervisor Karl Mosler, professor at the University of Cologne, for his patience and tireless interest during numberless discussions and his constant support of all activities connected with this thesis. He inspired and substantially complemented many ideas that have been worked out in this dissertation, including the topic of the research itself, and is a coauthor of papers which constitute three of four main chapters of this thesis. Special thanks belong to Tatjana Lange, professor at the Hochschule Merseburg, who supervised my scholarship there, which eventually led to this PhD-study. I thank her for her continuous support and her excellent ideas reflected in joint papers constituting two chapters of this thesis. This work would have never been born without these two people.

I would like to thank Rainer Dyckerhoff, professor at our department, for the permanent ability to consult me in theoretical and computational questions, and for sharing not only knowledge but implementations of his algorithms as well. Pavel Bazovkin, my colleague, is greatly acknowledged for dozens of fruitful in- and after-work discussions concerning all possible topics.

I am grateful to Oleksii Pokotylo, a colleague of mine, especially for his help in preparation and maintaining the R-package `ddalpha`. I thank Dr. Dominik Liebl, inter alia for his useful comments on an early version of the fourth chapter of this thesis. I am grateful to all participants of the research meetings of our Seminar of Statistics and Econometrics of the University of Cologne for their attention and live discussion, especially to Dr. Christoph Scheicher, Daniel Nowak, Prof. Oleg Badunenko, Dr. Tobias Wickern, Dr. Walter Orth, Dr. Konstantin Glombek, Dr. Stephan Nicklas.

I want to thank Yijun Zuo, professor at the Michigan State University, and Dr. Xiaohui Liu from the Jiangxi University of Finance and Economics, for providing their paper on exact computation of the Tukey depth with the Matlab source code of the algorithm. I am grateful to Robert Jumpertz and Holger Schützendorf, student assistants at our department, for the help in preparing the data-containing web-page `wisostat.uni-koeln.de/28969.html`. The maintainers and contributors of the repositories `archive.ics.uci.edu/ml`, `lib.stat.cmu.edu/datasets`, `stat.ethz.ch/Teaching/Datasets` are greatly acknowledged.

The very special grace is directed to my parents Liudmyla and Fedir, my grandmother Hanna, my brother Oleksandr, my cousin Kostiantyn, and my aunt Zinajida. My dear friends Marinela, Maria, Oleg, Gijs, I thank you all for your support.

Contents

Chapter 1

Introduction

People are thinking by means of patterns, unlike digital machines which operate with exact numbers. A hasty glance causes neurons of our brain to recognize dozens of patterns corresponding to familiar items of daily life. Today, possibilities and needs of mankind demand solving these tasks in an automatic way, according to precise algorithms and not involving direct human's work. Formally, regard an item as an object that is described by a number of reportable properties. It can be represented as a point in an appropriate space, with the properties constituting its dimensions. Statistically, a pattern or a class can be seen as a probability distribution of a random vector in this space. In practice these distributions are unknown, and only samples drawn from them are given. The task is then, based on these samples, to select the class for a new object that has been never seen before.

A huge variety of approaches to supervised classification has been developed during the last decades. This thesis explores the recently emerged idea of applying the notion of data depth to supervised classification. A statistical data depth is a function that describes the degree of centrality of a point w.r.t. a data cloud. That is, a data depth determines how representative an observation is w.r.t. the given sample, and thus to the class. A naïve way to proceed is to assign the new observation to the class in which it has the highest depth. Data depths have a number of attractive theoretical properties, they describe a random vector in a data driven way, often without moment assumptions, and can have nice robustness properties. Different shapes of the classes and their a priori probabilities intricate the task, making application of this naïve approach unreasonable. Thus, more subtle methods are required when designing depth-based classifiers. These are investigated in the current dissertation.

Below in this introductory chapter, a short overview of the area is given. Statistical data depth, its properties and computational issues, together with special depth notions used in this thesis are regarded in Section 1.1. Section 1.2 contains a brief introduction into supervised classification and some background of applying data depth to it. Depth-based classification in infinite-dimensional spaces is presented in Section 1.3. Section 1.4 structures the research part of the dissertation.

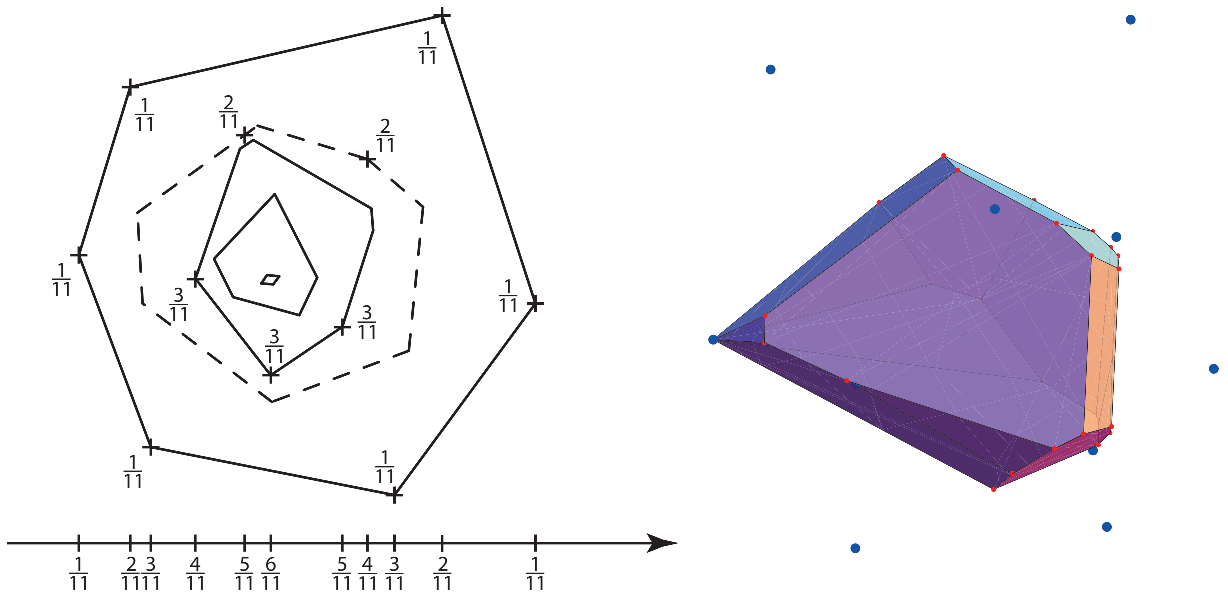

Figure 1.1: *Univariate, bivariate and trivariate Tukey depth regions.*

1.1 Data depth

Ordering w.r.t. a center has always been one of the most useful arts of statistical data description. Consider a sample consisting of eleven univariate observations. In the sorted data each point can be assigned an outward-decreasing number. To normalize, the number can be divided through the cardinality of the set, as it is shown in Figure 1.1, left, on the axis below. This value corresponds to the portion of the sample lying beyond the regarded point including itself. It tells us how centrally the point is located in the set. Now, if we have a new observation, it is reasonable to describe its relation to the sample by this value of the outer point closest to it. It represents the smallest portion of the sample to be removed to make the point lie outside of what remains.

Analyzing data in two dimensions requires a bivariate extension of the order. For this case, Tukey (1975) made a suggestion to separate outer points using lines, that was extended by Donoho & Gasko (1992) to outward-decreasing order. The centrality of a point is described by the smallest portion of the sample that can be cut off by a line through this point. For spaces of dimension three and more a line is naturally replaced by a hyperplane. Observations lying on this hyperplane are to be counted as well, so that the smallest fraction of the data is lying in a closed halfspace. Data depth allows to describe the shape of the data. See, e.g., Figure 1.1, left, where the five contours trim all points having order smaller than some given constant in $\{\frac{1}{11}, \frac{2}{11}, ..., \frac{5}{11}\}$. Thus, by any line through each point that belongs to the second outermost region (its boundary is depicted dashed, its depth is at least $\frac{2}{11}$) not less than two points are cut off from the sample. The approach can be generalized to higher dimensions by cutting hyperplanes. For example, observe trimmed region having depth $\frac{2}{11}$ when adding a third dimension to the data in Figure 1.1, right.

A function maintaining some center-outward ordering is a *statistical depth* function. Conditioned on a probability measure or a sample, it determines the centrality of the argument point. First we recall the concept of the statistical depth and the postulates it should satisfy in Section 1.1.1. After the pioneering work of Tukey (1975) many depth notions have emerged. To capture all of them is not the goal of this thesis. Here, only depth notions used in the following chapters are considered in detail, see Section 1.1.2. When employing the depths to multivariate data, questions about their computational tractability arise. These are briefly issued in Section 1.1.3.

1.1.1 The concept

Consider a point $\mathbf{z} \in \mathbb{R}^d$ and a random vector X distributed as P, especially one having empirical distribution on $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ in $\mathbb{R}^d$. A statistical data depth is a function $D(\mathbf{z}|X) : \mathbb{R}^d \mapsto [0, 1]$ that describes how deep, or central, the observation $\mathbf{z}$ is located w.r.t. X. In a natural way, it involves some notion of center. This is any point attaining the highest depth value in X, and not necessarily a single one. Attempts to find a reasonable center for multivariate X started long time ago. One of the first scientific publications has been made by Hayford (1902) defining the naïve componentwise *median*. Over the following hundred years various multivariate medians have been defined and extensively investigated, differing in applications and properties. For reviews the reader is referred to Donoho & Gasko (1987), Small (1990), Chaudhuri & Sengupta (1993), Niinimaa & Oja (1999), Dhar & Chaudhuri (2011), or Oja (2013) for the latest survey. In this view depth can also be seen as a center-outward ordering. Points closer to the center should have higher depth, and those more outlying a smaller one.

One of the first orderings of multivariate data has been proposed by Hodges (1955). He performed a sign test by counting the (positive) differences of observations in their univariate projections. Also for bivariate data, Tukey (1975) suggested to describe the shape by polygons defined as the intersection of halfplanes containing the same portion of a sample. Barnett (1976) did it by iteratively peeling the convex hulls. The first gave rise to the so-called *location* (=*halfspace*, also *Tukey*) *depth* (Donoho & Gasko, 1992), the latter to the *convex hull peeling* (see also Green & Silverman (1979) and Finch & Hueter (2004)), possessing no population version. Liu (1990) presents another notion of the center-outward ordering – the *simplicial depth*. By that she introduces the notion of *data depth*, and proves its useful properties. Many depth notions have arisen during the last several decades, see Serfling (2006), Romanazzi (2009), Cascos (2010) for surveys with properties. Reviews by Zuo & Serfling (2000) and Mosler (2013) additionally categorize depth notions. A very complete listing of existing depth notions can be found in the introductory chapter of Van Bever (2013).

The intuitive concept of a statistical data depth function can be formalized by stating postulates it should satisfy. Such requirements have been stated first by Liu (1990) for the simplicial depth. Later by Zuo & Serfling (2000) for general depth functions. Mosler (2013) suggests a somewhat differing set of properties. Following Dyckerhoff (2004) and Mosler (2013), a *depth function* is a function $D(\mathbf{z}|X) : \mathbb{R}^d \mapsto [0, 1]$ that is:

(*D1*) *translation invariant*: $D(\mathbf{z} + b|X + b) = D(\mathbf{z}|X) \, \forall \, b \in \mathbb{R}^d$,

(*D2*) *linear invariant*: $D(A\mathbf{z}|AX) = D(\mathbf{z}|X)$ for every $d \times d$ nonsingular matrix A,

(*D3*) *zero at infinity*: $\lim_{\|\mathbf{z}\| \to \infty} D(\mathbf{z}|X) = 0$,

(*D4*) *monotone on rays*: Let $\mathbf{z}^* = \operatorname{argmax}_{\mathbf{z} \in \mathbb{R}^d} D(\mathbf{z}|X)$, then $\forall \, \mathbf{r} \in S^{d-1}$ the function $\beta \mapsto D(\mathbf{z}^* + \beta\mathbf{r}|X)$ decreases, in the weak sense, $\forall \, \beta > 0$,

(*D5*) *upper semicontinuous*: the upper level sets $D_\alpha(X) = \{\mathbf{z} : D(\mathbf{z}|X) \geq \alpha, \mathbf{z} \in \mathbb{R}^d\}$ are closed $\forall \, \alpha$.

The first two properties state that $D(\mathbf{z}|X)$ is *affine invariant*. A in (D2) can be weakened to isometric linear transformations, which yields an *orthogonal invariant* depth. Taking instead of A some constant $\lambda > 0$ gives a *scale invariant* depth function. (D3) ensures that the upper level sets D_α, $\alpha > 0$ are bounded. According to (D4), the upper level sets are starshaped around $\mathbf{z}^*$, and $D_{\max_{\mathbf{z} \in \mathbb{R}^d} D(\mathbf{z}|X)}(X)$ is convex. Also, if X is *centrally symmetric* about $\mathbf{z}^*$, i.e. the distributions of $(X - \mathbf{z}^*)$ and $(\mathbf{z}^* - X)$ coincide, then $D(\mathbf{z}^*|X) = \max_{\mathbf{z} \in \mathbb{R}^d} D(\mathbf{z}|X)$. (D4) can be strengthened by requiring $D(\cdot|X)$ to be a *quasiconcave* function. Then the upper level sets are convex $\forall \, \alpha > 0$. (D5) is a useful technical restriction.

These postulates, different to those of Liu (1990), and Zuo & Serfling (2000), do not need to require maximality at the distribution's symmetry center, but imply this property. Additionally, (D5) requires upper semicontinuity, see also Dyckerhoff (2004).

Upper level sets $D_\alpha(X)$ of a depth function are called *depth-trimmed* or *central regions*. They describe the distribution's location, dispersion and shape. For given X, $D_\alpha(X)$ for $\alpha \in [0, 1]$ constitute a nested family of trimming regions. Note that due to (D1) and (D2) the central regions are affine equivariant.

In the variety of the depth notions developed many possess additional properties, aiming at certain practical goals and suited for particular application areas. Systematizations of the whole diversity have been offered by Zuo & Serfling (2000), and recently by Mosler (2013). Zuo & Serfling (2000) suggest four types of depth functions according to their construction. Thus *type A* depth functions are those defined by the average closeness of $\mathbf{z}$ from X, with their sample versions being U-statistics or V-statistics. These include in particular the simplicial depth and the *majority depth* (Singh, 1991). *Type B* is defined as scaled to $[0, 1]$ inverse of a nonnegative unbounded distance of $\mathbf{z}$ to X, including e.g. the L_p*-depth* (Zuo & Serfling, 2000) and the *simplicial volume depth* (Zuo & Serfling (2000), see also Oja (1983)) as representatives. *Type C* depth functions are defined similarly, but using an unbounded outlyingness measure. They include the *projection depth* (Stahel, 1981, Donoho, 1982) and the *Mahalanobis depth* (Mahalanobis, 1936) as examples. A data depth of *type D* is determined as the infimum of the probability mass of a set over some class of closed subspaces in $\mathbb{R}^d$, like location depth.

Mosler (2013) introduces three principal approaches to define a depth function. These result in different abilities to reflect asymmetries of the distribution, in varying robustness properties and in different computability.

- *Depths based on distances.* Here the depth function is defined using a distance from a properly defined central point or using a volume. Representatives mentioned are: L_2-depth, Mahalanobis depth, projection depth and simplicial volume depth.

- *Weighted mean depths.* Here depth is determined via weighted-mean trimmed regions (Dyckerhoff & Mosler, 2011, 2012), whose support function is a decreasingly weighted mean of order statistics (i.e., an L-statistics). Examples are: *zonoid depth* (Koshevoy & Mosler, 1997, Mosler, 2002), *expected convex hull depth* (Cascos, 2007), *geometrically weighted mean depth* (Dyckerhoff & Mosler, 2011).

- *Depths based on halfspaces or simplices.* These approaches concern halfspaces and simplices and regard only the combinatorial structure of the data. Typical examples are location depth and simplicial depth.

The depths based on the first two approaches are also mentioned as *metric depths*, and those based on the third one as *combinatorial depths.*

1.1.2 Some notions

Five depths, out of their thriving diversity, are used in this thesis for classification purposes. Here, they are briefly referred, in chronological order, and their most useful features are discussed. The corresponding depth functions, for a bivariate normal sample consisting of 250 observations drawn from $N\bigl(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right]\bigr)$, are plotted in Figure 1.2.

Mahalanobis depth is based on an outlyingness measure (Zuo & Serfling, 2000), *viz.* the Mahalanobis distance between $\mathbf{z}$ and a center of X, μ_X say (Mahalanobis, 1936):

$$d^2_{Mah}(\mathbf{x}; \mu_X, \Sigma_X) = (\mathbf{x} - \mu_X)'\Sigma_X(\mathbf{x} - \mu_X),$$

with "$'$" standing for the transposition operation. The depth of a point $\mathbf{z}$ w.r.t. X is then defined as (Liu, 1992)

$$D_{Mah}(\mathbf{z}|X) = \frac{1}{1 + d^2_{Mah}(\mathbf{z}; \mu_X, \Sigma_X)}, \tag{1.1}$$

see Figure 1.2, top middle, for illustration. If μ_X and Σ_X are chosen to be moment estimates, i.e. traditional *mean* and *covariance matrix*, the corresponding depth may be sensitive to outliers. A more robust depth is obtained with *minimum volume ellipsoid* (MVE) or *minimum covariance determinant* (MCD) estimators, see Rousseeuw & Leroy (1987) and Lopuhaä & Rousseeuw (1991).

This distance-based depth function satisfies all the above postulates and is quasi-concave, too (Mosler, 2013). It is defined by a finite number of parameters (namely $d(d + 1)$) and thus

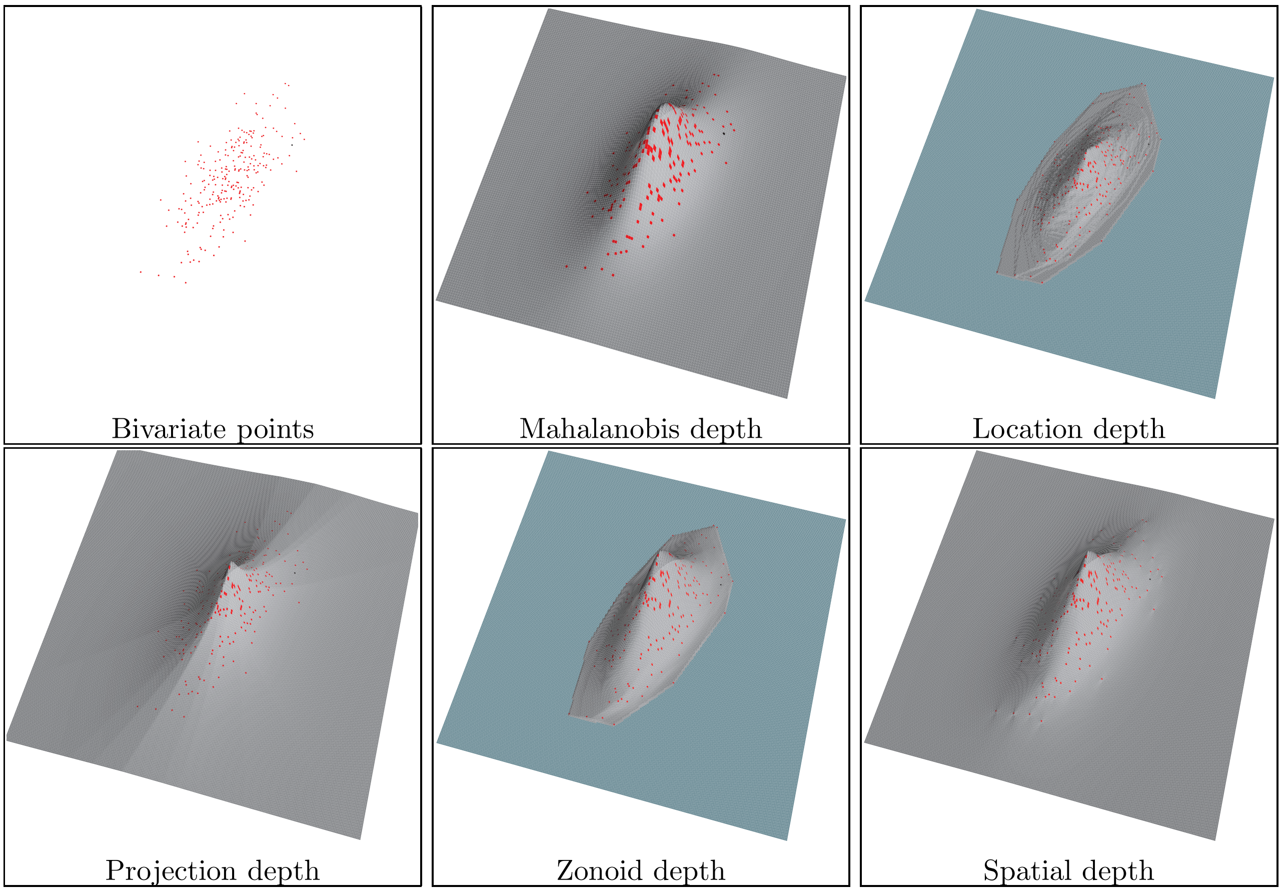

Figure 1.2: *Bivariate depths.*

can be regarded as a *parametric depth.* Based on the two first moments, its depth contours are ellipsoids, and it can identify the distributions defined by the two first moments only. Thus a single depth region suffices to identify a multivariate normal distribution, but also one within an affine family of nondegenerate multivariate distributions. Also, if the distribution of X is *angularly symmetric* around $\mathbf{c}_X \in \mathbb{R}^d$, i.e. if $\frac{X-\mathbf{c}_X}{\|X-\mathbf{c}_X\|}$ and $-\frac{X-\mathbf{c}_X}{\|X-\mathbf{c}_X\|}$ have the same distribution (with $\frac{0}{0}=0$), the Mahalanobis depth may not achieve its maximum value in $\mathbf{c}_X$.

Location depth follows the idea of Tukey (1975) (see also Donoho & Gasko (1992)). The location (=Tukey, halfspace) depth of $\mathbf{z}$ w.r.t. X is determined as:

$$D_{loc}(\mathbf{z}|X) = \inf\{P(H) \,:\, H \text{ is a closed halfspace, } \mathbf{z} \in H\}, \tag{1.2}$$

see Figure 1.2, top right, for illustration.

Location depth is a combinatorial depth, and it satisfies all the postulates prescribed for a depth function. It is additionally quasiconcave, and equals zero immediately outside the convex support of X. For any P, there exists at least one point having depth not smaller than $\frac{1}{1+d}$ (Mizera, 2002). For absolutely continuous P, it is a continuous function of $\mathbf{z}$, and its maximum value equals $1/2$. For angularly symmetric distributions, the maximum is attained at the symmetry center (Zuo & Serfling, 2000). If X has no Lebesgue density, location depth is a discrete function of $\mathbf{z}$, and the set of depth-maximizing locations – the *halfspace median* – can consist of more than one point. Location depth determines the empirical distribution uniquely (Struyf & Rousseeuw, 1999, Koshevoy, 2002), and converges for a sample from P almost surely to the depth of P (Donoho & Gasko, 1992).

Projection depth, similar to Mahalanobis depth, is based on a measure of outlyingness, used by Stahel (1981) and Donoho (1982), see Liu (1992). The worst case outlyingness is obtained by maximizing an outlyingness measure over all univariate projections:

$$o_{prj}(\mathbf{z}|X) = \sup_{\mathbf{u}\in S^{d-1}} \frac{|\mathbf{u}'\mathbf{z} - m(\mathbf{u}'X)|}{\sigma(\mathbf{u}'X)},$$

with $m(Y)$ and $\sigma(Y)$ being any univariate location and scatter measures. In practice most often *median*, $med(Y)$, and *median absolute deviation from the mediam*, $MAD(Y) = med(|Y - med(Y)|)$, are used as they are robust measures. Liu (1992) also suggests to use the gravity center of Oja (1983) as a median. Projection depth is then obtained as

$$D_{prj}(\mathbf{z}|X) = \frac{1}{1 + o_{prj}(\mathbf{z}|X)}, \qquad (1.3)$$

see Figure 1.2, bottom left, for illustration.

This distance-based depth satisfies all the postulates and quasiconcavity. By involving the symmetric scale factor MAD its contours retain a certain degree of symmetry and thus are not well suited for describing skewed data.

Zonoid depth has been first introduced by Koshevoy & Mosler (1997), see also Mosler (2002) for a discussion in detail. Unlike most of the other depths, the zonoid depth function is defined by means of depth contours – zonoid trimmed regions. The zonoid α-trimmed region of a probability measure P having finite expectation is defined as (Koshevoy & Mosler, 1997)

$$\begin{aligned} Z_\alpha(P) \;\; = \;\; & \Big\{ \int_{\mathbb{R}^d} \mathbf{x} g(\mathbf{x}) dP(\mathbf{x}) : \\ & \quad g : \mathbb{R}^d \mapsto \Big[0, \frac{1}{\alpha}\Big] \text{ measurable and } \int_{\mathbb{R}^d} g(\mathbf{x}) dP(\mathbf{x}) = 1 \Big\} \end{aligned}$$

for $\alpha \in (0, 1]$, and

$$Z_0(P) = \mathrm{cl}\Big(\bigcup_{\alpha\in(0,1]} D_\alpha(P) \Big)$$

for $\alpha = 0$, where "cl" denotes closure. Rather intuitive is the definition of the zonoid region of an empirical distribution. For $\alpha \in [\frac{k}{n}, \frac{k+1}{n}]$, $k = 1, ..., n-1$ the zonoid region is defined as

$$Z_\alpha(X) = \mathrm{conv}\Big\{ \frac{1}{\alpha n} \sum_{j=1}^{k} \mathbf{x}_{i_j} + \Big(1 - \frac{k}{\alpha n}\Big) \mathbf{x}_{i_{k+1}} : \{i_1, ..., i_{k+1}\} \subset N \Big\},$$

where $N = \{1, ..., n\}$, and for $\alpha \in [0, \frac{1}{n}]$

$$Z_\alpha(X) = \operatorname{conv}\bigl(\operatorname{supp}(X)\bigr) = \bigcap\{H \in \mathcal{H}^d : P(H) = 1\},$$

where “conv” is the convex hull, “supp” denotes the support, and $\mathcal{H}^d$ is the set of halfspaces in $\mathbb{R}^d$. Thus, e.g., $Z_{\frac{3}{n}}(X)$ is the convex hull over the set of all possible averages involving three points of the support of X, and $Z_0(X)$ is just the convex hull over the support.

The zonoid depth of a point $\mathbf{z}$ w.r.t. X is then defined as the largest $\alpha \in [0, 1]$ such that $Z_\alpha(X)$ contains $\mathbf{z}$ if $\mathbf{z} \in \operatorname{conv}\bigl(\operatorname{supp}(X)\bigr)$ and 0 otherwise:

$$D_{zon}(\mathbf{z}|X) = \begin{cases} \sup\{\alpha : \mathbf{x} \in Z_\alpha(X)\} & \text{if } \mathbf{x} \in \operatorname{conv}\bigl(\operatorname{supp}(X)\bigr), \\ 0 & \text{otherwise,} \end{cases} \tag{1.4}$$

see Figure 1.2, bottom middle, for illustration.

The zonoid depth belongs to the class of weighted mean depths; see Dyckerhoff & Mosler (2011, 2012), and additionally Mosler (2013) for detailed discussion. It satisfies all the above postulates and is quasiconcave. It fails to achieve maximality at the symmetry center of an angularly symmetric distribution. As well as location depth, zonoid depth vanishes beyond the convex support of X. Its maximum, located at $E[X]$, always equals 1. Thus the depth is not robust.

‘Lifting’ all properly scaled zonoid regions into $\mathbb{R}^{d+1}$ yields the ‘lift zonoid’, which fully characterizes the distribution (Mosler, 2002). It generates an antisymmetric depth order (=set inclusion of lift zonoids) and a probability semi-metric (=Hausdorff distance between the lift zonoids). Additionally, zonoid regions are invariant to any linear transformation, their marginal projections are zonoid regions of the marginal distributions.

Spatial depth (also L_1-depth) is a distance-based depth exploiting the idea of spatial quantiles of Chaudhuri (1996) and Koltchinskii (1997), formulated by Vardi & Zhang (2000) and Serfling (2002). *Affine invariant spatial depth* is defined as:

$$D_{spt}(\mathbf{z}|X) = 1 - \|E_X[v\bigl(\Sigma^{-1/2}(\mathbf{z} - X)\bigr)]\|, \tag{1.5}$$

with $v(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ if $\mathbf{x} \neq \mathbf{0}$, and $v(\mathbf{0}) = \mathbf{0}$; see Figure 1.2, bottom right, for illustration. Σ_X is a scatter matrix of X, which provides the affine invariance.

Affine invariant spatial depth satisfies all the postulates, is continuous but not quasiconcave. Its maximum is referred to as the *spatial median.* In the one-dimensional case it coincides with the location depth.

1.1.3 Computational issues

When applying depth to multivariate data some depth notions cause substantial computational burden. Depending on the statistical methodology used, one may need to compute data depth for a number of points, also w.r.t. several data clouds. This leads to even higher computational expenses. On the other hand, in applications to supervised classification, exact depth values at high computational cost may be replaced by reasonable approximations. Here, we give an overview of computational possibilities for the depths introduced in Section 1.1.2, and specify those employed in this thesis.

The computational task is twofold: (1) calculation of the depth of a single point $\mathbf{z}$ w.r.t. a data cloud X, and (2) construction of a depth central region for a given depth value. Ad (1), such extensions as computing depth for a number of points at once, or of the data cloud itself are under consideration as well, and in some cases can yield an increase of efficiency. Ad (2), a geometric d-dimensional body has to be shaped, which can be a quite involved task. For depths like location depth, computing a certain range of regions (Johnson *et al.*, 1998) or their entire family (Miller *et al.*, 2003) at once can bring additional acceleration. Another practical problem is updating a depth and its central regions when X is dynamically changed by insertion or deletion of observations, see e.g. Burr *et al.* (2011).

Mahalanobis depth and *spatial depth* can be computed exactly from (1.1) and (1.5), correspondingly, in an efficient way. For Mahalanobis depth, a robust choice are MCD estimates which are still computationally very efficient. A robustness parameter has to be chosen though. Moment estimates can be taken for μ_X and Σ_X, to speed up the procedure extremely at the cost of robustness. In this dissertation, in Chapter 3 we employ both estimates, while in Chapter 4 only moment estimates are used. Central regions for Mahalanobis depth are ellipsoids located at μ_X and stretched in directions of the eigenvectors of Σ_X according to its eigenvalues. To the knowledge of the author, no algorithm for construction of the spatial depth central regions exists for $d > 1$.

Zonoid depth can be computed efficiently as the solution of a linear program. A very fast algorithm exploiting the idea of Dantzig-Wolfe decomposition has been proposed by Dyckerhoff *et al.* (1996). Dyckerhoff (2000) has suggested an algorithm to compute its bivariate contours. Mosler *et al.* (2009) construct an algorithm computing zonoid central region for $d > 2$. This was the first algorithm building central regions of dimension > 2. Later, Bazovkin & Mosler (2012) design a general algorithm for weighted-mean trimmed regions.

Location depth belongs to the class of so called combinatorial depths. Its exact computation is very expensive. Thus a substantial part of the literature on location depth considers computational issues. Here we name only a few sources that are most relevant for the content of this thesis. For extensive references the reader is referred to Sections 5.1.1 and 5.1.2 regarding exact and approximate computation, respectively. For bivariate data, Rousseeuw & Ruts (1996) propose an exact algorithm, and Ruts & Rousseeuw (1996a) construct central regions. Rousseeuw & Struyf (1998) publish an algorithm computing location depth for $d = 3$. Hallin *et al.* (2010) establish the connection between the multiple-output regression and location depth, and suggest an algorithm producing all the halfspaces determining a central region in their intersection (see also Paindaveine & Šiman (2012a,b)). Liu & Zuo (2014a) develop an algorithm computing location depth exactly in any dimension, which proves to be extremely time consuming. High computational cost is a general characteristic of all algorithms computing location depth or its central regions.

Dyckerhoff (2004) introduces the *weak projection property* of a depth, i.e. that the depth can be obtained as the minimum over the depths on all univariate projections. He proves that, among others, location depth possesses this property, which gives rise to an approximation. Cuesta-Albertos & Nieto-Reyes (2008) propose the random Tukey depth, and Chen *et al.* (2013) suggest a more systematic approach. In this thesis, the random Tukey depth with slight computational modifications is exploited.

Projection depth is another depth intrinsically defined as an optimization problem. By that it causes even larger computational effort than location depth does. The depth and its contours have been computed exactly for $d = 2$ by Zuo & Lai (2011) and Liu *et al.* (2013), respectively, and for $d > 2$ by Liu & Zuo (2014b). To avoid the enormous computational expenses, in this thesis projection depth is approximated in the same way as location depth.

1.2 Depth based classification

In Section 1.1 a way of center-outward ordering of data has been regarded – statistical data depth. It describes the data's location, scale and shape, defining relatedness of an observation to a data cloud. Here, a closer look is taken at its application to the task of the *supervised classification*, or *supervised statistical learning*, restricted w.l.o.g. to the case of two classes only (=binary classification). Consider the joint distribution (X, Y) of a d-dimensional input vector X and a binary nominal output Y taking values "0" and "1". These constitute two classes (labeled "0" and "1", respectively), also representable by conditional probability distributions P_0 and P_1 of X possessing densities f_0 and f_1. Let $\mathcal{R}$ be the class of all measurable mappings $\mathbb{R}^d \mapsto \{0, 1\}$. Given is a *training sample*: $X_0 = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ and $X_1 = \{\mathbf{x}_{m+1}, ...\mathbf{x}_{m+n}\}$ drawn from P_0 and P_1, respectively, which is a bunch of observations labeled by a teacher in accordance with belonging to a class, i.e. numbered "0" or "1". Based on this training sample $X_0 \cup X_1$, we construct a rule $\boldsymbol{r}(\cdot) : \mathbb{R}^d \mapsto \{0, 1\}$, which, for a new observation $\mathbf{x}_0 \in \mathbb{R}^d$, predicts to which of the classes it belongs: $\widehat{class}(\mathbf{x}_0) = \boldsymbol{r}(\mathbf{x}_0)$. $\boldsymbol{r}$ is then mentioned as *classification* or *decision rule*, and a mapping $\boldsymbol{c} : \mathbb{R}^{md} \times \mathbb{R}^{nd} \mapsto \mathcal{R}$ that constructs a decision rule based on a training sample is called *classification method* or *classifier*. Such notation is also convenient because the function $\boldsymbol{c}$ is applied in the training phase, where the classifying rule $\boldsymbol{r}$ is synthesized. Then the created $\boldsymbol{r}$ is applied in the classification phase to determine to which class a newly seen observation belongs.

One of the most common quality criteria for $\boldsymbol{r}$ is that $\boldsymbol{r}$ should minimize (the mathematical expectation of) the classification error,

$$\begin{aligned} \mathcal{E}(\boldsymbol{r}, P_0, P_1) &= \pi_0 \int_{\mathbb{R}^d} I\big(\boldsymbol{r}(\mathbf{x}) = 1\big) dP_0(\mathbf{x}) \\ &+ \pi_1 \int_{\mathbb{R}^d} I\big(\boldsymbol{r}(\mathbf{x}) = 0\big) dP_1(\mathbf{x}), \end{aligned} \tag{1.6}$$

with π_0 and π_1 being prior probabilities of the classes, while $I(\cdot)$ is an indicator function taking value 1 when the condition is fulfilled and 0 otherwise. For given P_0 and P_1 the minimal achievable error $\mathcal{E}^*(P_0, P_1) = \inf_{\boldsymbol{r}\in\mathcal{R}} \mathcal{E}(\boldsymbol{r}, P_0, P_1)$ is referred to as the *Bayes error*, and a classification method whose error converges to it when $m, n \to \infty$ is called *Bayes consistent* or *Bayes optimal*. Usually different classifiers are Bayes consistent under certain restrictions on P_0 and P_1. Those consistent for any distribution are referred to as *universally consistent*.

In practice, when P_0 and P_1 are unknown, $\mathcal{E}(\boldsymbol{r}, P_0, P_1)$ can be consistently estimated using the jack-knife (=leave-one-out cross-validation) method (Luntz & Brailovsky, 1969, Stone, 1974). For a given classification method $\boldsymbol{c}$ and a training sample $X_0 \cup X_1$ the jack-knife estimate is defined as

$$\begin{aligned}\hat{\mathcal{E}}(\boldsymbol{c}, X_0, X_1) &= \frac{1}{m+n}\Big(\sum_{i=1}^{m} I\big(\boldsymbol{c}(X_0 \setminus \{\mathbf{x}_i\}, X_1)(\mathbf{x}_i) = 1\big) \\ &\qquad + \sum_{i=m+1}^{m+n} I\big(\boldsymbol{c}(X_0, X_1 \setminus \{\mathbf{x}_i\})(\mathbf{x}_i) = 0\big)\Big).\end{aligned}$$

Often, for time saving reasons, generalized cross-validation with leaving out more than one observation at a time is employed.

Two principally different rule-constructing methodologies, relevant for this thesis, are briefly described below: classifiers of the plug-in type, and classifiers minimizing the empirical risk. Overviews of the methods of supervised classification include Ripley (1996), Devroye *et al.* (1996), Vapnik (1998), Hastie *et al.* (2009), Haykin (2009) and many others.

Plug-in classifiers constitute classification methods that construct rules of the form

$$\boldsymbol{r}(\mathbf{x}_0) = \operatorname*{argmax}_{i\in\{0,1\}} \pi_i f_i(\mathbf{x}_0) = I\big(\eta(\mathbf{x}_0) \geq \frac{1}{2}\big), \tag{1.7}$$

see, e.g., Devroye *et al.* (1996) for other analytical forms. In them, π_i and the functions f_i, or $\eta(\mathbf{x}_0) = E[Y|X = \mathbf{x}_0]$, are to be plugged-in, i.e. replaced by a reasonable estimator. In what follows we briefly review the so-called traditional classifiers of the plug-in type.

Introduced by Fisher (1936) *linear discriminant analysis* (LDA) is a particularly fast procedure constructing a linear decision rule. It separates the classes on the one-dimensional projection obtained by regarding f_0 and f_1 as normal densities. Their parameters, μ_{X_0}, μ_{X_1} and $\Sigma_{X_0} = \Sigma_{X_1}$, are estimated from the data as the classes' means and the pooled covariance matrix. Thus, LDA is Bayes consistent for multivariate normal distributions with the same covariance matrix and when the priors are known (see, e.g., Hastie *et al.* (2009)). In other cases it may perform very poorly, e.g. it can be shown that even when the classes are linearly separable the error of LDA can be arbitrarily high (Devroye *et al.*, 1996). On the other hand, when round-shaped classes lie far from each other, LDA can be a reasonable solution, see Chapter 3. Its robustness properties are very much determined by the mean and covariance estimators used (see Section 1.1.2). An extension of LDA by accounting for the covariance matrices of both classes is known as *quadratic discriminant analysis* (QDA). It yields a quadratic decision rule and proves Bayes optimality also for the Gaussian case when the classes' covariance matrices are different. Another classifier of the plug-in type can be constructed using *kernel density analysis* (KDA) to estimate f_0 and f_1 (Wand & Jones, 1995). Here kernel parameters have to be presumed, and are usually estimated by means of cross-validation. Priors can be taken equal to classes' portions, or alternatively cross-validated. A *k-nearest-neighbor* (kNN) classifier estimates $\eta = \frac{\pi_1 f_1}{\pi_0 f_0}$ as the relation of class memberships among k nearest to $\mathbf{x}_0$ neighbors (Fix & Hodges, 1951, Stone, 1977). KDA and kNN are universally consistent.

Sparseness of empirical data, presence of outliers, substantially differing classes' portions in the training sample tangle the estimation of η, and can lead to rather poor rules. In addition, making no assumptions about P_0 and P_1 asks for – usually time consuming – cross-validation when tuning the parameters of $\boldsymbol{r}$. Thus, another rule-constructing methodology is often preferred. This is regarded right below.

Empirical risk minimization is another methodology of construction of a classification rule. Here, from some class of decision rules $\mathcal{R}_c \in \mathcal{R}$, $\boldsymbol{r}$ is chosen, that minimizes some *loss function* L averaged over $X_0 \cup X_1$ for given $\boldsymbol{r}$. The most natural is the indicator loss function $L(\mathbf{x}) = I\big(\boldsymbol{r}(\mathbf{x}) \neq \mathit{class}(\mathbf{x})\big)$ that yields *empirical risk*, i.e. the portion of misclassified observations obtained by applying $\boldsymbol{r}$ to $X_0 \cup X_1$ used for training:

$$\mathcal{L}(\boldsymbol{r}, X_0, X_1) = \frac{1}{m+n}\Big(\sum_{i=1}^{m} I\big(\boldsymbol{r}(\mathbf{x}_i) = 1\big) + \sum_{i=m+1}^{m+n} I\big(\boldsymbol{r}(\mathbf{x}_i) = 0\big)\Big). \tag{1.8}$$

Even for the class of linear rules its search for generally located $X_0 \cup X_1$ has non-polynomial (NP) complexity (Höffgen *et al.*, 1995). If X_0 and X_1 are linearly separable this task can be formulated in terms of quadratic programming (Vapnik & Chervonenkis, 1974). To circumvent NP-completeness for linearly nonseparable X_0 and X_1, other loss functions (e.g., based on the distance to the separating hyperplane) are used, which are computationally tractable. These allow for complexity $O\big((m+n)^3\big)$ in time and $O\big((m+n)^2\big)$ in space, and even less when using specialized algorithms for support vector machines (see, e.g., Osuna *et al.* (1997), Platt (1999)).

Historically, the first empirical risk minimizing classifier is the *perceptron*, designed (also mechanically) by Rosenblatt (1958) and based on the neuron model of McCulloch & Pitts (1943). It constructs the separating hyperplane by iteratively correcting an arbitrary linear rule, which is done in a finite number of steps if the training sample is linearly separable (Novikoff, 1962). Later, the entire field of methods minimizing empirical risk has been developed. Below we regard only two techniques directly related to this thesis: support vector machine and the α-procedure.

Vapnik & Lerner (1963), later Vapnik & Chervonenkis (1974), introduce the *generalized portrait* method for linearly separable X_0 and X_1. It finds the separating hyperplane maximally distant from the closest points of X_0 and X_1 by solving the task of constraint quadratic optimization. This hyperplane can be represented by a weighted sum of so-called support vectors from $X_0 \cup X_1$, i.e. observations minimally distant from the separating hyperplane. As the method involves dot products but not the points from $\mathbb{R}^d$, Boser *et al.* (1992) apply the kernel trick, which allows to separate by non-linear rules. Later, Cortes & Vapnik (1995) develop the soft-margin-hyperplane method, which works also when X_0 and X_1 are not faultlessly separable. This is today known as the *support vector machine* (SVM). The loss function minimized in SVM is not the indicator function, for the sake of computational tractability. Today, SVM together with its modifications constitute a separate field of study. In the book by Vapnik (1998) SVM is one of the main topics, and the book by Steinwart & Christmann (2008) contains a complete theoretical workout and regards important practical issues. In this thesis, it is briefly introduced in Section 3.3.2.

The *α-procedure* (Vasil'ev, 1991, Vasil'ev & Lange, 1998), having already been used in the nineteen seventies, constructs the separating hyperplane by iteratively synthesizing the space of the most informative features. In the first step, it chooses an axis of the property space, on which separation of X_0 and X_1 is best. In each of the following steps, it is looking for the optimal separating rule in all two-dimensional spaces based on the axis from the previous step and a coordinate axis of the property space, that has not been involved yet. The space which minimizes empirical risk is then transformed into a one-dimensional feature by projecting onto the direction which is normal to the optimal separating line. The method proceeds subsequently adding new axes from the property space, and stops when no additional classification approvement is possible. For the general description of the α-procedure the reader is referred to Vasil'ev (2003) and Lange & Mozharovskyi (2014). Its adaptation to the depth-based classification in the unit cube can be found in Section 2.3, and in Section 3.2.2 applied to real data.

Some notions of statistical data depth are *directly connected* to the task of supervised classification. Thus, regression depth (Rousseeuw & Hubert, 1999) corresponds to the smallest empirical risk achievable when separating by a hyperplane (Christmann & Rousseeuw, 2001, Christmann *et al.*, 2002). Solving this problem exactly has NP-complexity though. This fact obstructs its application to high-dimensional data. Ghosh & Chaudhuri (2005a) also point out the relationship between calculating location depth and finding the optimal separating linear rule. They enlighten this by suggesting a transformation of $X_0 \cup X_1$. To circumvent the NP-completeness, they suggest to smooth the empirical risk functional (as a function of the parameters of a linear rule). They approximate the indicator function $I(u > 0)$ with the logistic function $\frac{1}{1+e^{-tu}}$. Then a steepest descent numerical optimization is applied. The parameter t has to be set, and multiple repetition of the optimization starting from different points copes with the problem of local minima.

Later, a number of plug-in classifiers, where data depth is used to replace $\pi_i f_i$, has been developed, see Section 1.2.1. Combining the plug-in technique with empirical risk minimization by applying the latter in a depth space opens new perspectives. Investigation of those constitutes the main part of this dissertation. A short introduction on this topic is given in Section 1.2.2.

1.2.1 Plug-in classifiers

Depth-based classifiers of the plug-in type use an appropriate depth function to reconstruct $\pi_i f_i$ for each of the classes. In their naïve form they can be written as

$$\boldsymbol{r}(\mathbf{x}_0) = \underset{i\in\{0,1\}}{\operatorname{argmax}}\, D(\mathbf{x}_0|X_i), \tag{1.9}$$

with $D(\mathbf{x}_0|X_i)$ being a multivariate depth. The so-called *maximum depth classifier* (Ghosh & Chaudhuri, 2005b) is Bayes consistent if the classes originate from the same strictly unimodal elliptical distribution and have equal priors. That is, the classes' distributions are equally probable, have the same, equally scaled structural matrix and the same strictly decreasing radial density. So, differing in location only, they constitute a *location-shift alternative.* To overcome this constraint, more complicated forms than (1.9) have to be considered.

Hoberg (2002) applies the zonoid depth to supervised classification, see also Mosler & Hoberg (2006). In the classification phase, a point lying beyond the convex hulls of both classes will have zonoid depth = 0 w.r.t. both classes, and thus cannot be readily classified. (Recall, the zonoid depth vanishes beyond the convex hull of the data.) Such a point is referred to as an *outsider.* To deal with this problem, Mahalanobis depth (positive for the entire $\mathbb{R}^d$) is applied, scaled in a way that it cannot be larger than zonoid depth, unless the latter is zero. The classification rule is:

$$\boldsymbol{r}(\mathbf{x}_0) = \underset{i\in\{0,1\}}{\operatorname{argmax}}\Big\{\max\big\{D_{zon}(\mathbf{x}_0|X_i), \frac{1}{\max_{j\in\{0,1\}}\{\#X_j\}} D_{Mah}(\mathbf{x}_0|X_i)\big\}\Big\}.$$

The issue of outsiders is thoroughly addressed in Chapters 2 and 3 below. Jörnsten (2004) employs a maximum depth classifier with L_1-depth, which is everywhere positive, so that the problem of outsiders does not arise.

Ghosh & Chaudhuri (2005b) construct a maximum depth classifier based on location, simplicial, majority, projection, spatial or spatial volume depths. To overcome the location-shift restriction, the authors apply a kernel density estimator, using the re-scaled Mahalanobis distance based on location depth. Choosing the scaling constant by means of cross-validation then makes the classifier Bayes-optimal for strictly unimodal elliptical classes (*location-scale alternative*). Note, that the classes may have unknown priors and differing radial densities. Dutta & Ghosh (2012) similarly employ a robust classifier based on projection depth. Also, Dutta & Ghosh (2011) suggest a classifier using L_p-depth, which is Bayes-consistent for L_p-symmetric location-scale alternatives. These classifiers involve scaling constants determined by means of cross-validation, where projection (Dutta & Ghosh, 2012) and L_p (Dutta & Ghosh, 2011) depth have to be calculated on each iteration. For this reason the classifiers seem to be computationally expensive. Cui *et al.* (2008) define an *extended projection depth* using a relative dispersion measure based on the product of squared MADs obtained by subsequently maximizing the latter ones on orthogonal directions. Their classifier of the form (1.9), based on the extended projection depth, is Bayes optimal for the location-scale alternative.

The depth-based approaches considered above make use of so-called global depths. These usually satisfy the postulates (D1) – (D5). They can be expressed as a monotone function of the population's density if the density is strictly unimodal elliptical. So they are not able to properly describe the population if it is not elliptically symmetric (or L_p-symmetric when using L_p-depth). In practical applications, classes mostly do not possess elliptical symmetry, but are often asymmetric and may have several modes. In this case the maximum depth classifier is insufficient to achieve optimal Bayes error. Paindaveine & Van Bever (2012) suggest a depth-based k-nearest neighbor classifier which is universally consistent. When classifying $\mathbf{x}_0$, they suggest to proceed as follows: First, for each class $i \in \{0, 1\}$, calculate the depth of each point of X_i being appended by its centrally symmetric reflection w.r.t. $\mathbf{x}_0$, then, apply the kNN-rule to the obtained depth values. Dutta *et al.* (2012) construct another universally consistent classifier by introducing a localized spatial depth based on kernels. Both methods allow for universal consistency but cause enormous computational burden.

1.2.2 DD-classifier

Li *et al.* (2012) suggest a new method of depth-based classification exploiting the probabilistic geometry of data: depth-vs-depth classification (DD-classification). First, for two classes X_0 and X_1, the DD-plot – a subset of the unit square – is obtained:

$$Z = \{(z_0, z_1) | z_0 = D(\mathbf{x}_i|X_0), z_1 = D(\mathbf{x}_i|X_1), i = 1, ..., m+n\}. \tag{1.10}$$

Then, a polynomial decision rule is constructed that separates the DD-plot and minimizes the error probability $\mathcal{E}$. This yields a classification rule of the form:

$$\boldsymbol{r}(\mathbf{x}_0) = I\big(D(\mathbf{x}_0|X_1) > \sum_{k=1}^{k_0} a_k D(\mathbf{x}_0|X_0)^k\big), \tag{1.11}$$

with $(a_1, ..., a_{k_0}) \in \mathbb{R}^{k_0}$ being the coefficient vector of the polynomial of degree k_0. The constant a_0 is set to zero, as the origin obviously is a separating point. (If the depth of $\mathbf{x}_0$ w.r.t. both classes = 0, no decision can be made: $\mathbf{x}_0$ is an outsider.) The DD-classifier based on the Mahalanobis depth is demonstrated in Figure 1.3 for a normal location-scale alternative. X_0 origins from $N\big(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right]\big)$, and X_1 origins from $N\big(\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}4&4\\4&16\end{smallmatrix}\right]\big)$, containing 250 points in each of the classes. The polynomial rule on the DD-plot (right) corresponds to the solid line on the scatter plot (left). The optimal Bayes rule is marked by the dashed line (left). As every smooth function can be approximated by a polynomial of suitable degree, an optimal separating line on the DD-plot can always be found for large enough k_0. To eventually achieve the optimal Bayes error, the DD-transform should not lead to loss of information. Loosely speaking, no two points in $\mathbb{R}^d$ that belong – according to an optimal Bayes rule – to different classes in the original space, must be projected into the same point of the DD-plot. This can be guaranteed if the classes are strictly unimodal elliptically distributed. DD-classifiers, as pointed out by Li *et al.* (2012), possess a number of advantages: the separating rule is determined automatically

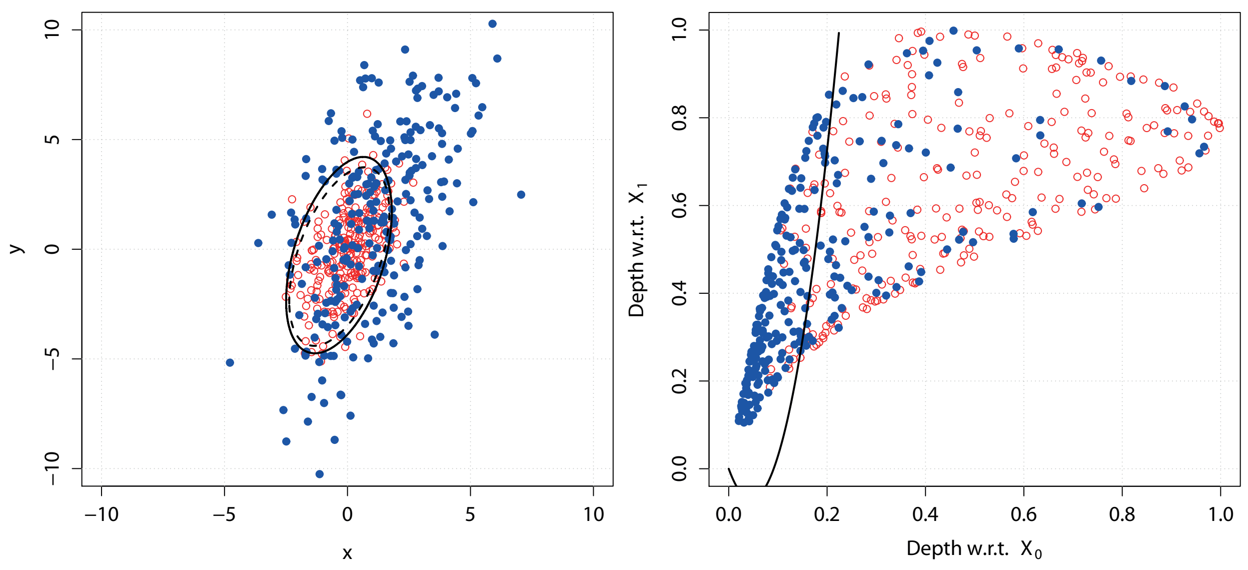

Figure 1.3: *DD-classifier based on the Mahalanobis depth for a normal location-scale alternative: scatter plot with DD- (solid line) and optimal Bayes (dashed line) rule (left) and corresponding DD-plot with separating second-degree polynomial (right).*

based on the data topology, no parametric assumptions are needed, and the separation can be well visualized.

Determining the optimal polynomial curve in the *DD*-plot computationally can be a challenging task though. Li *et al.* (2012) suggest to minimize the empirical risk for a given degree of the polynomial k_0 by employing numerical optimization. To provide smoothness, they approximate the indicator function $I(x > y)$ by the logistic function $\frac{1}{1+e^{-t(x-y)}}$, reporting that $t = 100$ is a good choice. k_0 ranges in $\{1, 2, 3\}$, and the one delivering the smallest classification error is chosen by means of cross-validation. Finding the optimal coefficient vector $(a_1, ..., a_{k_0})$ requires solving a rather complicated optimization problem. For the sample in Figure 1.3 the empirical risk functional (taking $k_0 = 2$) is shown in Figure 1.4, left. For a sample, where X_0 and X_1 are generated from Cauchy distributions with the same centers and shape matrices as before, the functional to be minimized (with $k_0 = 2$) is shown in Figure 1.4, right. To deal with this problem, Li *et al.* (2012) first pick 1000 polynomial lines passing through k_0 points of Z and the origin. Then, the one delivering the smallest empirical risk is selected as the initial value for the numerical optimization procedure. Such an approach can be computationally involved and deliver an unstable solution.

1.3 Functional classification

The need to classify functional data arises in such fields as biology, biomechanics, medicine, economics. The data on functional values is generally given at some discretization points which may be neither equidistant nor common. Let $\mathcal{F}$ be the space of real functions, defined on a compact interval, which are continuous and almost everywhere smooth. Consider the setting

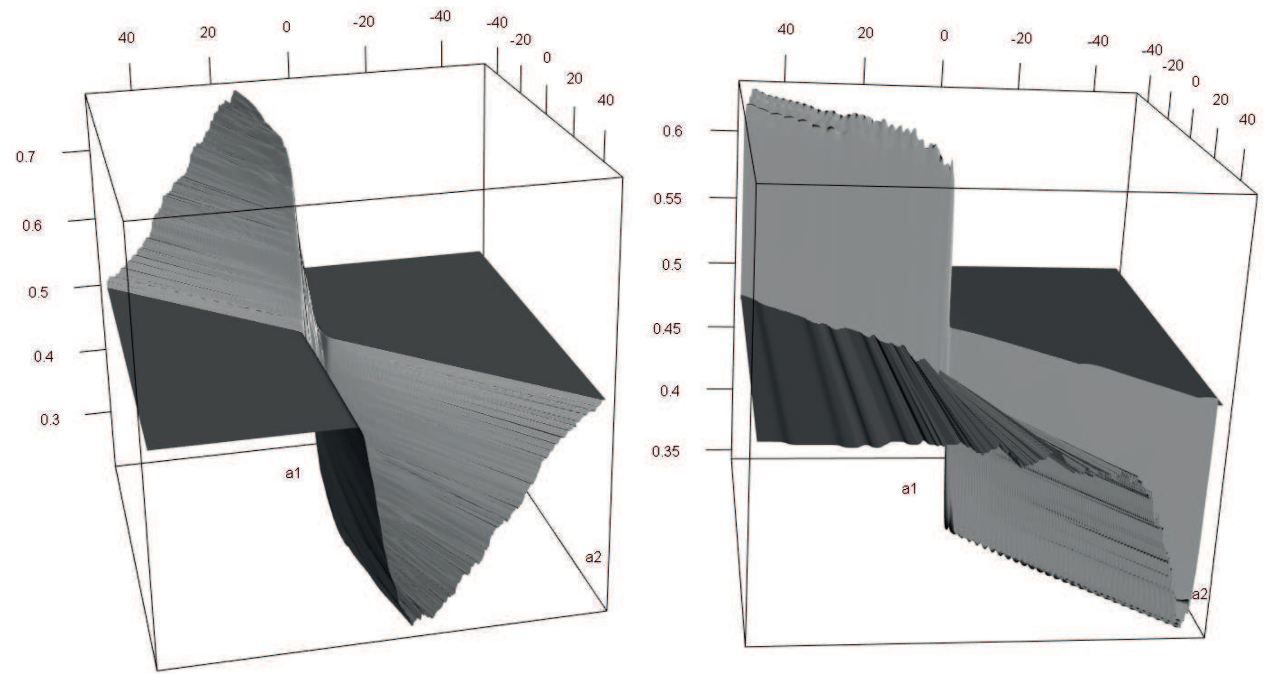

Figure 1.4: *Smoothed empirical risk (vertical axis) of DD-classifier based on the Mahalanobis depth obtained by the polynomial separating rule* $\mathbf{r}(\mathbf{x}_0) = I\big(D(\mathbf{x}_0|X_1) > a_1 D(\mathbf{x}_0|X_0) + a_2 D(\mathbf{x}_0|X_0)^2\big)$*, for normal (left) and Cauchy (right) location-scale alternatives.*

of binary supervised functional classification: Given two classes in $\mathcal{F}$, $X_0 = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ and $X_1 = \{\mathbf{x}_{m+1}, ...\mathbf{x}_{m+n}\}$, the task is to construct a rule categorizing a new observation $\mathbf{x}_0 \in \mathcal{F}$.

The wide range of constantly developing approaches can be roughly classified into three groups:

- approaches applying a finite-dimensional classification technique to projections onto a finite sub-basis, e.g. principal (Hall et al., 2001) or independent (Huang & Zheng, 2006) components, wavelets (Wang *et al.*, 2007) or functions of very simple and interpretable nature (Tian & James, 2013);

- approaches using an – in the classification sense – optimal subset of given or smoothed functional evaluations (Ferraty *et al.*, 2010, Delaigle *et al.*, 2012);

- approaches exploiting the idea of data depth (López-Pintado & Romo, 2006, Cuevas *et al.*, 2007, Cuesta-Albertos & Nieto-Reyes, 2010, Sguera *et al.*, 2014).

Advantages and shortfalls of each of them are discussed in Section 4.1. Another categorization of methods of functional classification can be found in Ferraty & Romain (2010), Chapter by Baíllo, Cuevas and Fraiman. They distinguish classification techniques based on linear discrimination rules, k-NN rules, kernel rules, partial least squares, reproducing kernels and depth measures, but also address some other decision-constructing methods. For general information on functional data, including discrimination, the reader is referred to Ramsay & Silverman (2005) and Ferraty & Vieu (2006).

In this new and rapidly evolving field, this dissertation addresses classification methods based on the idea of statistical data depth. Different notions of depth for functional data, which evolved in the last decade and which are relevant for this thesis, are briefly addressed in Section 1.3.1. Then, some classification techniques exploiting them are described in Section 1.3.2.

1.3.1 Functional depths

Regard a real-valued function $\mathbf{z} : [0,T] \mapsto \mathbb{R}$, and a random function X with the same support, distributed as P, in particular empirically on $\{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathcal{F}$. Fraiman & Muniz (2001) define the *integrated data depth* of $\mathbf{z}$ w.r.t. X as the average univariate depth at each argument value:

$$FD_{int}(\mathbf{z}|X) = \int_0^T D^1\big(\mathbf{z}(t)|X(t)\big)dt, \tag{1.12}$$

with $X(t)$ being a random vector, the evaluation of X at the point t, and $D^1(\cdot|\cdot)$ being a univariate depth. For an appropriate set of functions, they prove the uniform convergence of the empirical depth to its population version, and derive the strong consistency of the corresponding α-trimmed mean estimates.

López-Pintado & Romo (2011) define the *half-region depth* as the smallest of the probabilities of X lying in the hyper- and hypo-graph of $\mathbf{z}$:

$$\begin{aligned} FD_{hr}(\mathbf{z}|X) \;&=\; \min\big\{P\big(\mathbf{z}(t) \le X(t)\,\forall\, t \in [0,T]\big), \\ &\qquad P\big(\mathbf{z}(t) \ge X(t)\,\forall\, t \in [0,T]\big)\big\}. \end{aligned} \tag{1.13}$$

López-Pintado & Romo (2009) define the *band depth* as

$$FD_{band}(\mathbf{z}|X) = \sum_{j=2}^{J} P\big(\mathbf{z}(t) \in \text{conv}\{X_1(t), ..., X_j(t)\}\,\forall\, t \in [0,T]\big), \tag{1.14}$$

where $X_1, ..., X_j$ are i.i.d. drawings from P. The same authors also define the *generalized half-region depth* and the *generalized band depth* by averaging the mathematical expectation of the condition from $P(\cdot)$ over $[0,T]$. For the generalized band depth, López-Pintado & Romo (2009) report $J = 2$ to be generally sufficient. The two generalized depths correspond to the integrated depth of Fraiman & Muniz (2001) when taking $D^1(\cdot|\cdot)$ to be location and simplicial depths, respectively.

Mosler & Polyakova (2012) introduce a comprehensive class of functional depths, named Φ-*depth*. Consider a Banach space E having a norm $\|\cdot\|$, and the dual space E'^d of all continuous linear functionals $E \mapsto \mathbb{R}^d$. For $\mathbf{z}$ and a sample $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in E$ a Φ-depth is defined as:

$$FD_\Phi(\mathbf{z}|X) = \inf_{\phi \in \Phi} D^d\big(\phi(\mathbf{z})|\phi(\mathbf{x}_1), ...\phi(\mathbf{x}_n)\big), \tag{1.15}$$

where $D^d(\cdot|\cdot)$ is a d-variate depth, $\Phi \subset E'^d$. Φ-depth is the smallest multivariate depth among those achievable on all considered aspects, which are finite dimensional projections $E \mapsto \mathbb{R}^d$. (It correlates with the weak projection property of Dyckerhoff (2004).) Letting, e.g., $\Phi = E'^1$ and $D^1 = D_{loc}$ one obtains the *Tukey functional depth*, the natural extension of its finite-dimensional version.

Mosler & Polyakova (2012) formulate postulates a functional data depth should satisfy, and prove that any Φ-depth does so. These are analogous to the postulates for finite-dimensional depths (and differ only in substituting linear by scale invariance): (FD1) translation invariance, (FD2) scale invariance, (FD3) null at infinity, (FD4) monotonicity on rays (or stronger requirement – quasiconcaveness), (FD5) upper semicontinuity. Φ-depth gives substantial freedom when constructing a functional depth. The authors point out several meaningful subclasses. Thus, defining Φ as a (sub)set of all evaluations at $t \in [0, T]$, yields the subclass of *general graph depths*. These are represented by half-region depth (López-Pintado & Romo, 2011) and band depth (López-Pintado & Romo, 2009), when taking D^1 being location, respectively simplicial, depth for E being a set of real valued functions on $[0, T]$. Choosing Φ to be the set of projections of evaluations at k points onto $\mathbf{u} \in S^{k-1}$, defines the subclass of *grid depths*. If Φ is the (sub)set of two-dimensional points consisting of evaluations and their derivatives at $t \in [0, T]$, the subclass of *location-slope graph depths* is obtained. This is used as the basic idea of the *location-slope integral depth* defined and applied in Chapter 4.

Natural extensions of some finite dimensional depths to their functional versions can be meaningless. Chakraborty & Chaudhuri (2014) show that the above mentioned Tukey functional depth (see also Dutta *et al.* (2011)) as well as the projection depth are almost everywhere zero for a wide class of distributions. These include Gaussian distributions on the space $C[0, T]$ of continuous functions on $[0, T]$ with supremum norm. Also, half-region depth and band depth show degenerate behavior for some standard probability models, such as certain α-mixing sequences and Feller processes with continuous sample paths (Revuz & Yor, 1999). On the other hand, their generalized versions take depth values in the entire $[0, 1]$ for these settings. Also, the integrated depth of Fraiman & Muniz (2001), as well as the infinite extension of spatial depth suggested by Chakraborty & Chaudhuri (2014), do not suffer from this degenerate behavior.

For a d-variate stochastic process $\boldsymbol{z} = (\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(d)})$ with $\boldsymbol{z}^{(j)} : [0, T] \to \mathbb{R}$ continuous for $j = 1, ..., d$ and a random family of those $\boldsymbol{X}$ distributed as $\boldsymbol{P}$, Claeskens *et al.* (2014) define a multivariate functional depth as

$$MFD(\boldsymbol{z}|\boldsymbol{X}) = \int_0^T D^d\big(\boldsymbol{z}(t)|\boldsymbol{X}(t)\big) \cdot w(t)dt, \tag{1.16}$$

where $\boldsymbol{z}(t)$ is a d-variate vector which is the process's cut at the argument value t, $\boldsymbol{X}(t)$ is a d-variate random vector, and w is a weight function defined on $[0, T]$ and integrating to one. The authors show that, if the underlying finite dimensional depth is satisfying the postulates of Zuo & Serfling (2000), it satisfies the functional version of these postulates. Taking location depth as the building block they define the *multivariate functional halfspace depth*, which has a number of useful properties.

1.3.2 Discrimination

Data depth continuously proves to be useful in numerous applications, in particular when classifying multivariate data. Classification of functional data is a rather new, rapidly evolving field of statistics, finding its applications to data of functional and high-dimensional nature. Several articles of the last decade fuse both research areas together, employing depth based classification to functional data. These are briefly regarded below.

López-Pintado & Romo (2006) suggest to classify functional data by its *distance to the trimmed mean* and by *(trimmed) weighted average distance.* When employing classification based on the distance to the trimmed mean, on the training stage, the α-trimmed mean of each class $\mathbf{m}_i^\alpha, i \in \{0,1\}$ is calculated. Trimming is based on the band depth values $FD_{band}(\mathbf{x}|X_i)$, $\mathbf{x} \in X_i$. Then, taking, say, $d(\mathbf{x},\mathbf{x}') = \int_0^T |\mathbf{x}(t) - \mathbf{x}'(t)|dt$ as the distance measure, the classification rule is:

$$\boldsymbol{r}(\mathbf{x}_0) = \underset{i\in\{0,1\}}{\operatorname{argmin}}\, d\big(\mathbf{x}_0, \mathbf{m}_i^\alpha(t)\big). \tag{1.17}$$

When classifying by weighted average distance the classification rule looks as:

$$\begin{aligned} \boldsymbol{r}(\mathbf{x}_0) &= I\Big(\frac{\sum_{i=1}^m d(\mathbf{x}_0,\mathbf{x}_i)FD(\mathbf{x}_i|X_0)}{\sum_{i=1}^m FD(\mathbf{x}_i|X_0)} \\ &> \frac{\sum_{i=m+1}^{m+n} d(\mathbf{x}_0,\mathbf{x}_i)FD(\mathbf{x}_i|X_1)}{\sum_{i=m+1}^{m+n} FD(\mathbf{x}_i|X_1)}\Big). \end{aligned} \tag{1.18}$$

When the cardinalities of X_0 and X_1 are too different, the classification can be inaccurate. In this case, the authors propose to apply trimmed weighted average distance, i.e. to average over the k most central functions of each class with $k \le \min\{m,n\}$. Again, the central-outward order is provided by the data depth.

Cuevas *et al.* (2007) employ the naïve maximum depth classifier to functional data, with five notions of data depth: integrated depth of Fraiman & Muniz (2001), h-mode depth, and three variants of random depths. *Integrated depth* is taken in its usual form with $D^1(x|X) = 1 - \left|\frac{1}{2} - F(x)\right|$ being the univariate depth, and F being the cumulative distribution function of a univariate random variable X. *h-mode depth* is defined as $FD_h(\mathbf{z}|X) = E[K_h(\|\mathbf{z} - X\|)]$, where $\|\cdot\|$ is a suitable norm (e.g., L_2-norm), and $K_h(x)$ is a re-scaled kernel of type $K_h(x) = \frac{1}{h}K(\frac{x}{h})$, with K being a kernel function (e.g., Gaussian kernel $K(x) = \frac{1}{\sqrt{2\pi}}exp\big(\frac{-x^2}{2}\big)$) and h being a fixed tuning parameter. The authors use the normalized version of h-mode depth $\frac{FD_h(\mathbf{z}|X) - \min FD_h(\mathbf{z}|X)}{\max FD_h(\mathbf{z}|X) - \min FD_h(\mathbf{z}|X)}$. According to the *random projection method*, in the Hilbert space $L^2[0,T]$ data depth is averaged over univariate projections on random directions. The authors generate 50 directions standardized to norm 1, from a Gaussian distribution. Two *double random projection methods* are based on mapping the sample into $\mathbb{R}^2$ by applying the random projection method to functions (1st coordinate) and their derivatives (2nd coordinate). Then, to the bivariate data, either the random projection method or the h-mode depth is applied.

Together with the *random Tukey depth*, Cuesta-Albertos & Nieto-Reyes (2008) propose its *infinite-dimensional version* (in a separable Hilbert space of square-integrable functions). They suggest the practitioner to specify a distribution of random directions and their number, depending on the application. Also Cuesta-Albertos & Nieto-Reyes (2010) work out these issues, and investigate two strategies: First, random directions are generated from a fixed distribution, a standard Brownian motion. Second, they are drawn from the two-parameter distribution family obtained by scaling the standard Brownian motion by the absolute value of the difference between the classes' point-wise medians brought to some power. The authors choose the number of random directions by means of leave-one-out cross-validation, in both cases. They apply this functional data depth instead of band depth with techniques used by López-Pintado & Romo (2006).

Sguera *et al.* (2014) extend the spatial depth to infinite dimensional spaces by mapping the data to obtain a kernel-based similarity measure. They define the *kernelized functional spatial depth* of $\mathbf{z}$ w.r.t. X as

$$FD_{spt}(\mathbf{z}|X) = 1 - \|E[v\big(\phi(\mathbf{z}) - \phi(\mathbf{X})\big)]\|, \tag{1.19}$$

where $\phi : \mathbb{H} \mapsto \mathbb{F}$ is an embedding map from the Hilbert space with norm $\|\cdot\|$ inherited from the inner product, into an appropriate feature space $\mathbb{F}$, and $v(\cdot)$ as in (1.5). This leads to the sample version

$$\begin{aligned} FD^n_{spt}(\mathbf{z}|X) \quad = \quad 1 - \frac{1}{n}\Big(\sum_{\mathbf{x}_i,\mathbf{x}_j \in X} & \frac{K(\mathbf{z},\mathbf{z}) + K(\mathbf{x}_i,\mathbf{x}_j) - K(\mathbf{z},\mathbf{x}_i) - K(\mathbf{z},\mathbf{x}_j)}{\sqrt{K(\mathbf{z},\mathbf{z}) + K(\mathbf{x}_i,\mathbf{x}_i) - 2K(\mathbf{z},\mathbf{x}_i)}} \\ & \times \frac{1}{\sqrt{K(\mathbf{z},\mathbf{z}) + K(\mathbf{x}_j,\mathbf{x}_j) - 2K(\mathbf{z},\mathbf{x}_j)}} \Big)^{\frac{1}{2}}, \end{aligned}$$

where $K(\cdot,\cdot)$ is an appropriately chosen positive definite and stationary kernel. (A stationary kernel is a kernel that is translation invariant, also referred to as anisotropic stationary kernel, see, e.g., Genton (2001).) The authors employ the maximum depth classifier with the kernelized functional spatial depth, using a Gaussian kernel.

1.4 The structure of the thesis

This thesis consists of four main chapters. The second chapter named *Fast nonparametric classification based on data depth* introduces a new depth based classification method based on the idea of Li *et al.* (2012). A projective invariant method, the α-procedure (Vasil'ev, 1991, Vasil'ev & Lange, 1998, Vasil'ev, 2003), is applied in the DD-plot, which is a space of depths w.r.t. the classes of the training sample. The so-called $DD\alpha$-classifier possesses a number of useful properties: it is affine-invariant, robust, very fast, and can be visualized at every step of the training phase. Here the zonoid depth (Koshevoy & Mosler, 1997, Mosler, 2002) is employed, which has attractive properties and can be efficiently computed exactly in higher dimensions (Dyckerhoff *et al.*, 1996) by means of linear programming. As it vanishes beyond the convex hull of the data, the zonoid depth produces outsiders which are projected into the origin of the depth space and require an additional treatment. Several such treatments are considered in an extensive comparative study. The proposed method is applied to simulated and benchmark data together with traditional and existing depth-based classifiers. For several data sets it is contrasted with SVM (Vapnik, 1998) both in terms of performance and speed. Further, employed with the Tukey depth (Tukey, 1975), the $DD\alpha$-classifier is experimentally investigated by means of simulated data regarding performance, performance dynamics and speed. This chapter is a joint work with Prof. Tatjana Lange and Prof. Karl Mosler. Most material of the chapter has been published in the journal *Statistical Papers*; Section 2.7 concerning the random Tukey depth (Cuesta-Albertos & Nieto-Reyes, 2008), as well as the general description of the α-procedure with several experimental comparisons, constitute two chapters of the proceedings of the 36th Annual Conference of the German Classification Society *Data Analysis, Machine Learning and Knowledge Discovery.*

The third chapter named *Classifying real-world data with the $DD\alpha$-procedure* provides a broad comparative experience of application of the $DD\alpha$-classifier to fifty real-data problems with a number of depths and outsider treatments. Mahalanobis (Mahalanobis, 1936), spatial (Vardi & Zhang, 2000, Serfling, 2002) and projection (Stahel, 1981, Donoho, 1982, Liu, 1992) depths are everywhere positive so that no problem of outsiders arises, and thus every point of the space can be readily classified. On the other hand their central regions retain a certain degree of symmetry even if the real data is not symmetric. Tukey depth keeps the level sets closer to the data geometry, but vanishes beyond the convex hull of the data, and thus requires an additional outsider treatment. Here, treatments based on LDA, kNN, maximum Mahalanobis depth are used. As the calculation of projection and Tukey depths is computationally quite involved, their approximations from above are used, for Mahalanobis and spatial depth both moment and robust (MCD) estimates are tried. A new outsider treatment based on SVM is proposed, which also speeds up the classification phase enormously. All considered classifiers, together with the three traditional ones (LDA, QDA and kNN) taken as a benchmark, are compared. The comparison is based on five indicators, which rank the techniques w.r.t. their performance on the fifty real-data sets. A further investigation into the DD-plot features used by the α-procedure shows that in almost all cases two-dimensional linear separation is sufficient, and in 3/4 of cases no polynomial extension of the DD-plot is needed. The proposed technique has been implemented as an `R`-package named `ddalpha`. The classification tasks used, together with their descriptions and some statistics, can be accessed under `http://www.wisostat.uni-koeln.de/28969.html`. This chapter is a result of a close cooperation with Prof. Karl Mosler and Prof. Tatjana Lange. The material of the chapter is about to appear in the journal *Advances in Data Analysis and Classification.*

The fourth chapter named *Fast DD-classification of functional data* suggests an extension of the DD-classifier to the infinite-dimensional data based on the kNN and the α-procedure.

A new integral location-slope mapping is proposed, which is weakly continuous and preserves Bayes optimality. This first approximates the functional data with first-order splines, and then integrates its level (=location) and first derivative (=slope) over L, respectively S for slope, equally sized intervals. Afterwards, multivariate data depth is applied to this $\mathbb{R}^{L+S}$-representation. In such a way the entire functional support is considered, and changing of a function with the argument is accounted for in a nonparametric way. Being able to treat functional data of either intrinsically discrete or continuous nature, observed at a bunch of points, neither equidistant nor common ones, it provides the integral location-slope depth transform. To choose the dimensions of the location-slope space – L and S – a conservative Vapnik-Chervonenkis (Vapnik & Chervonenkis, 1974) bound is applied, followed by cross-validation over a restricted range of (L, S) pairs. This technique proves to be fast and efficient, outperforming traditional and maximum-depth classifiers as well as componentwise space-construction technique of Delaigle *et al.* (2012) in most of the considered cases. This chapter has been written commonly with Prof. Karl Mosler.

The fifth chapter named *Exact computation of the Tukey depth* proposes two algorithms computing location depth exactly. The first of them is based on the idea of Liu & Zuo (2014a), who regard successively, by means of a breadth-first spread algorithm, all direction cones, which are closures of polyhedral cones formed by all directions that maintain the same halfspace-defined subsets of the sample, and thus yield the same univariate location depth. To determine the cone's boundary, Liu & Zuo (2014a) use a vertex-facet enumeration and the `qhull` algorithm. Here, this is narrowed down to linear programming, which in this case can be much faster. Also, as the direction cones are coded with a binary sequence, only a relatively small number of them has to be stored in the memory. By that the algorithm is substantially faster than its predecessor and requires less memory. The idea of this algorithm is to regard – based on the spatial order – all dichotomies yielded by separating the sample with the hyperplane through the point $\mathbf{z}$ for which the depth is calculated. As all these dichotomies are to be regarded anyway, the second algorithm suggests to use a combinatorial order to do so. It is sufficient to go through univariate depths on the sample's projections onto directions that are orthogonal to hyperplanes containing $\mathbf{z}$ and $d-1$ points from the sample in $\mathbb{R}^d$. This approach is much less time consuming than the before-mentioned one, does not require memory-expensive structures, and can be very simply implemented in any programming environment.

Chapter 2

Fast nonparametric classification based on data depth

2.1 Introduction

A steady interest in statistical learning theory has intensified recently since nonparametric tools have become available. A new impetus has been given to supervised classification by employing depth functions such as Tukey's (1975) halfspace depth or Liu's (1990) simplicial depth. In supervised learning a function is constructed from labeled training data that classifies an arbitrary data point by assigning it one of the labels (Hastie *et al.*, 2009). Given two or more labeled clouds of training data in d-space, a data depth measures the centrality of a point with respect to these clouds. For any point in d-space it indicates the degree of closeness to each label. This can be employed in different ways for solving the classification task. Many authors have made use of data depth ideas in supervised classification. Liu *et al.* (1999) were the first who stressed the usefulness and versatility of depth transformations in multivariate analysis. They introduced the notion of a DD-plot, that is the two-dimensional representation of multivariate objects by their data depths regarding two given distributions. In a straightforward way, an object can be classified to the class where it is deepest, that is, according to its maximum depth. Jörnsten (2004) and Ghosh & Chaudhuri (2005b) have followed this and similar approaches; see also Mosler & Hoberg (2006). Dutta & Ghosh (2012, 2011) employ a separator that is linear in a density based on kernel estimates of the projection depth, respectively L_p-depth. Recently, Li *et al.* (2012) have used polynomial separators of the DD-plot to classify objects by their depth representation. These methods differ in the notion of depth used and allow for adaptive and other extensions.

The quoted literature has in common that a (possibly high-dimensional) space of objects is transformed into a lower-dimensional space of depth values of these objects and the classification task is performed in the depth space. In this context several questions arise:

1. Which particular notion of depth should be employed?

2. Which classification procedure should be applied to the depth-represented data?

3. How extends the procedure to $q > 2$ classes?

The above literature answers these questions in different ways. Ad (1), halfspace and simplicial depths, among others, have been employed in Ghosh & Chaudhuri (2005a), Li *et al.* (2012), Liu *et al.* (1999). They depend only on the combinatorial structure of the data, being constant in the compartments spanned by them. Consequently, these depths are rather robust to outlying data, but calculating them in higher dimensions can be cumbersome if not impossible. On the other hand Mahalanobis depth (Mahalanobis, 1936), which has also been used by these authors, is easily calculated but highly non-robust. Moreover, it depends on the first two moments only and does not reflect any asymmetries of the data. More robust forms of the Mahalanobis depth remain still insensitive to data asymmetries. L_1-depth as used in Jörnsten (2004) has similar drawbacks. Dutta & Ghosh (2011) employ L_p-depths, which are easily calculated if p is known, and choose p in an adaptive procedure; however the latter needs heavy computations. In Mosler & Hoberg (2006) the maximum zonoid depth and a combination of it with the Mahalanobis depth are used; both can be efficiently calculated also in high dimensions but lack robustness. Ad (2), Li *et al.* (2012) solve the classification problem of the DD-plot by designing a polynomial line that separates the unit square and provides a minimal average misclassification rate (AMR); the order (up to three) of the polynomial is selected by cross validation. Similarly, separators are determined in Dutta & Ghosh (2011) and Dutta & Ghosh (2012) by cross-validation.

Ad (3) with $q > 2$ classes a given point is usually classified in two steps according to majority rule: firstly $\binom{q}{2}$ classifications are performed that are restricted to pairs of classes in the object space, and secondly the point is assigned to that class where it was most often assigned in step 1.

In this chapter, ad (1), we employ the zonoid depth (Koshevoy & Mosler, 1997, Mosler, 2002), as it can be efficiently calculated also in higher dimensions (up to $d = 20$ and more) and has excellent theoretical properties regarding continuity and statistical inference. However the zonoid depth has a low breakdown point. If, in a concrete application, robustness is an issue the data have to be preprocessed by some outlier detection procedure. Ad (2), for final classification in the depth space a variant of the α-procedure is employed. It operates simply and very efficiently on low-dimensional spaces like the depth spaces considered here. The α-procedure has been originally developed by Vasil'ev (Vasil'ev, 1991, 2003) and Lange (Vasil'ev & Lange, 1998). Ad (3) we employ DD-plots if there are two classes and q-dimensional depth plots if there are $q > 2$ classes. Assignment of a given point to a class is based on $\binom{q}{2}$ binary classifications in the q-dimensional depth space plus a majority rule. Note that in each binary classification the whole depth information regarding all q classes is used.

We call our approach the $DD\alpha$-approach and apply it to simulated as well as real data. The results are contrasted with those obtained in Li *et al.* (2012), Dutta & Ghosh (2011), and Dutta & Ghosh (2012).

The contribution of this chapter is threefold. A classification procedure is proposed that

1. is efficiently computable for objects of higher dimensions,

2. employs a very fast classification procedure of the depth-transformed data,
3. uses the full multivariate information when classifying into $q > 2$ classes.

The rest of the chapter is organized as follows. Section 2.2 introduces the depth transform, which maps the data from the d-dimensional object space to the q-dimensional depth space, and provides a first discussion of the problem of 'outsiders', that are points having a vanishing depth vector. In Section 2.3 our modification of the α-procedure is presented in some detail. Section 2.4 provides a number of theoretical results regarding the behavior of the $DD\alpha$-procedure on elliptical and mirror symmetric distributions. Section 2.5 contains extensive simulation results and comparisons. Calculations of real data benchmark examples are reported in Section 2.6 as well as a comparison of the $DD\alpha$-procedure with the SVM approach. In Section 2.7 zonoid and Tukey depths are compared in a simulation study that involves fat-tailed and skewed distributions. Section 2.8 concludes.

The material of the chapter is based on Lange *et al.* (2014a), Section 2.7 is based on Lange *et al.* (2014b). A general description of the α-procedure can be found in Lange & Mozharovskyi (2014).

2.2 Depth transform

A data depth is a function that measures, in a certain sense, how close a given point $\mathbf{x}$ is located to the 'center' of a finite set X in $\mathbb{R}^d$, that is, how 'deep' it is in the set. More precisely, a data depth is a function

$$(\mathbf{x}, X) \mapsto D_X(\mathbf{x}) \in [0,1]\,, \quad \mathbf{x} \in \mathbb{R}^d\,, \quad X \subset \mathbb{R}^d\,,$$

that satisfies the following restrictions: affine invariant; upper semicontinuous in $\mathbf{x}$; quasiconcave in $\mathbf{x}$ (that is, having convex upper level sets), vanishing if $||\mathbf{x}|| \to \infty$. Sometimes two weaker restrictions are imposed: orthogonal invariant; decreasing on rays from a point of maximal depth (that is, starshapedness of the upper level sets). For surveys of these restrictions and many special notions of data depth, see e.g. Zuo & Serfling (2000), Mosler (2002), Dyckerhoff (2004), Serfling (2006), Cascos (2010).

Now, assume that data in $\mathbb{R}^d$ are to be classified into $q \geq 2$ classes and that $X_1, \ldots, X_q \subset \mathbb{R}^d$ are training sets for these classes each having finite size $n_j = |X_j|$. Let D be a data depth. The function $\mathbb{R}^d \to [0,1]^q$ mapping

$$\mathbf{x} \mapsto \mathbf{d} := \big(D_{X_1}(\mathbf{x}), \ldots, D_{X_q}(\mathbf{x})\big) \tag{2.1}$$

will be mentioned as a *depth representation.* Each object is represented by a vector whose q components indicate its depth or closeness regarding the q classes. In particular, the training sets $X_j \subset \mathbb{R}^d$ are transformed to sets in $[0,1]^q$ that represent the classes in the depth space. It should be noted that 'closeness' of points in the original space translates to 'closeness' of their representations. The classification problem then becomes one of partitioning the depth space $[0,1]^q$ into q parts.

A simple rule, e.g., is to classify a point to that class where it has the largest depth value; see Ghosh & Chaudhuri (2005b), Jörnsten (2004). This means that the depth space decomposes into q compartments which are separated by (parts of) q bisecting hyperplanes. Maximum depth classification is a linear rule. A nonlinear classification rule is used in Li *et al.* (2012), who treat the case $q = 2$ by constructing a polynomial line up to degree 3 that separates the depth space $[0,1]^2$; see also Dutta & Ghosh (2011, 2012).

With several important notions of data depth, $D_X(x)$ vanishes outside the convex hull of X. This is, e.g., the case with the halfspace, simplicial, and zonoid depths, but not with the Mahalanobis and L_p-depths. A point that is not within the convex hull of at least one training set then is mapped to the origin in the depth space. Such a point will be mentioned as an *outsider*. Of course, it can be neither regarded as correctly classified nor ignored. To classify this point we may consider three principal approaches, each allowing for several variants.

- Classify randomly, with probabilities equal to the expected proportions of origin of points to be classified.
- Use the k-nearest neighbors method with a properly chosen distance: Euclidean distance, L_p-distance, Mahalanobis distance with moment estimates, Mahalanobis distance with robust estimates (MCD, cf. e.g. Hubert & Van Driessen (2004)).
- Classify with maximum Mahalanobis depth (using moment estimates or MCD) or with the maximum of another depth that is properly extended beyond the convex hull as e.g. in Mosler & Hoberg (2006).

In the sequel we will use either random classification, k-nearest neighbors (with different distances), or maximum Mahalanobis depth (with moment and robust estimates).

2.3 The α-procedure

To separate the q classes in the multi-depth space we use the *α-procedure*, which has been developed by Vasil'ev (Vasil'ev, 1991, 2003) and Lange (Vasil'ev & Lange, 1998), see also Lange *et al.* (2011). Among others the regression depth method (see Rousseeuw & Hubert (1999), Christmann & Rousseeuw (2001), Christmann *et al.* (2002)) or the support vector machine (see Vapnik (1998) and Christmann *et al.* (2002)) seem to be good alternatives. In contrast with those the α-procedure, in application to the current task, is substantially faster and produces a unique decision rule. Besides that it focuses on features of the extended $[0,1]^q$, i.e. depths and their products, which, by their nature, are rather relevant. Moreover, by selecting a few important features only, the α-procedure yields a rather stable solution.

Let us first present the procedure in the case of $q = 2$ classes. As above consider two clouds of training data in $\mathbb{R}^d$, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_1}\}$ and $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_{n_2}\}$ and notate $\mathbf{x}_{n_1+m} = \mathbf{y}_m$,

$m = 1, ..., n_2$. By calculating the depth of all $\mathbf{x}_i$ with respect to each of the two clouds, their depth representation, $(D_X(\mathbf{x}_i), D_Y(\mathbf{x}_i))$, is obtained, $i = 1, 2, \ldots, n_1 + n_2$. The set

$$\mathcal{D} = \{\mathbf{d}_i \in [0,1]^2 | \mathbf{d}_i = \big(D_X(\mathbf{x}_i), D_Y(\mathbf{x}_i)\big), i = 1, \ldots, n_1 + n_2\}$$

is the DD-plot of the data (Liu *et al.*, 1999).

We use a modified version of the α-procedure to construct a nonlinear separator in $[0,1]^2$ that classifies the D-represented data points. The construction is based on depth values and the products of depth values up to some degree p that can be either chosen *a priori* or determined by cross-validation. For this, a linearized representation of the two classes in a depth feature space is

$$\begin{aligned}\mathbf{Z} = \{\mathbf{z}_i \quad | \quad & \mathbf{z}_i = \big(D_X(\mathbf{x}_i), D_Y(\mathbf{x}_i), D_X(\mathbf{x}_i) \cdot D_Y(\mathbf{x}_i), D_X^2(\mathbf{x}_i), D_Y^2(\mathbf{x}_i)\big), \\ & i = 1, ..., n_1 + n_2\} \,.\end{aligned}$$

Each element of the extended D-representation is mentioned as a *basic D-feature* and the space $[0,1]^r$ as the **feature space**. When the maximum exponent is $p \geq 1$, $\mathbf{z}_i$ is a vector in $\mathbb{R}^r$ having components

$$D_X(\mathbf{x}_i)^{k_\nu} \cdot D_Y(\mathbf{x}_i)^{\ell_\nu}, \quad \text{where } 1 \leq k_\nu + \ell_\nu \leq p \,, \quad \nu = 1, \ldots, r \,. \tag{2.2}$$

The number of basic D-features, that is the dimension of the feature space, equals $r = \binom{p+2}{2} - 1$, which is easily seen by induction. We index the basic D-features by ν and notate $\mathbf{z}_i = (z_{i\nu})_{\nu=1,\ldots,r}$.

The α-procedure now, in a stepwise way, performs linear discrimination in subspaces of the feature space. It is a bottom-up approach that successively builds new features from the basic D-features. In each step certain two-dimensional subspaces of $\mathbf{Z}$ are considered, and the projection of $\mathbf{Z}$ to each of these subspaces is separated by a straight discrimination line. Out of these subspaces the α-procedure selects a subspace whose discrimination line provides the least classification error. Clearly any discrimination line that separates the DD-plot must pass through the origin since $D_X(\mathbf{x}_i) = D_Y(\mathbf{x}_i) = 0$ implies that the point $\mathbf{x}_i$ cannot be classified to either of the two classes. The same must hold for all discrimination lines in subspaces of the extended depth space.

In a *first step* a pair (ν_1, ν_2) of D-features (2.2) is chosen with $(k_1 + k_2)(\ell_1 + \ell_2) > 0$. The latter restriction implies that the two D-features do not solely relate to one of the classes. A straight discrimination line is calculated in the two-dimensional coordinate subspace defined by the pair (ν_1, ν_2). As the line passes through the origin it is characterized by an angle $\alpha \in [0, 2\pi[$.

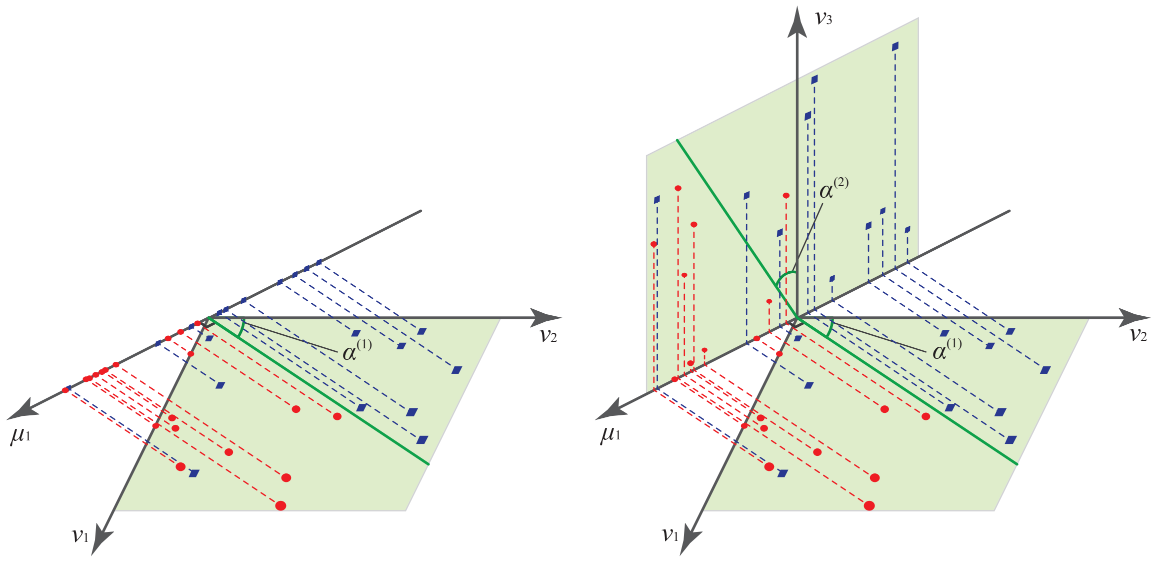

Figure 2.1: *α-procedure; step 1 (left) and step 2 (right).*

The best discriminating angle α_{ν_1,ν_2} is determined by minimizing the *average misclassification rate (AMR)*,

$$\begin{aligned}\Delta(\alpha;\nu_1,\nu_2) \quad &= \quad \frac{1}{n_1+n_2}\Big[\sum_{i=1}^{n_1} I(z_{i,\nu_1}\cos\alpha + z_{i,\nu_2}\sin\alpha < 0) \\ &\qquad + \sum_{i=n_1+1}^{n_1+n_2} I(z_{i,\nu_1}\cos\alpha + z_{i,\nu_2}\sin\alpha > 0)\Big]\,.\end{aligned} \tag{2.3}$$

Here $I(A)$ denotes the indicator function of A. If the minimum is attained in an interval, its middle value is selected for α_{ν_1,ν_2}; see Figure 2.1, left. The same is done for all pairs of D-features satisfying the above restriction, and the pair (ν_1^*,ν_2^*) is selected that minimizes (2.3). If the minimum is not unique the pair with the smallest k and ℓ is chosen. Let $\alpha^{(1)} = \alpha_{\nu_1^*,\nu_2^*}$ and denote the respective AMR by $\Delta^{(1)}$. Next the D-features ν_1^* and ν_2^* are replaced by a new D-feature which is indexed by μ_1 and gives value

$$z_{i,\mu_1} = z_{i,\nu_1}\cos\alpha^{(1)} + z_{i,\nu_2}\sin\alpha^{(1)}\,, \quad i=1,\dots n_1+n_2\,, \tag{2.4}$$

to each $\mathbf{x}_i$. Geometrically the values are obtained by projecting $(z_{i,\nu_1}, z_{i,\nu_2})$ to a straight line in the (ν_1,ν_2)-plane that is perpendicular to the discrimination line; see Figure 2.1, left. The first step results in the new D-feature μ_1 and the AMR $\Delta^{(1)}$ produced by classifying according to this feature.

The *second step* couples the new D-feature μ_1 with each of the basic D-features ν that have not been replaced so far. For each of these pairs of D-features a best discriminating angle $\alpha_{\mu_1,\nu}$ is determined, and among these the pair of D-features is selected that provides the minimum AMR. The minimum error is denoted by $\Delta^{(2)}$ and the angle at which it is attained by $\alpha^{(2)}$. This is visualized in Figure 2.1, right. The best pair of D-features is replaced by a new D-feature μ_2, where the values z_{i,μ_2} are calculated as in (2.4).

The last step is repeated with μ_2 in place of μ_1, etc. The procedure stops after step t if either the additional discriminating power $\Delta^{(t)} - \Delta^{(t+1)} = 0$ or $t = r$, that is, all basic D-features have been replaced. Then the angle $\alpha^{(t)}$ defines a linear rule for discriminating between two (up to) p-th order polynomials in $D_X(\mathbf{z})$ and $D_Y(\mathbf{z})$, which correspond to the two finally constructed D-features, according to their sign. This yields a polynomial separation of the classes in the depth space.

For example, let in step 1 the basic features D_X and D_Y^2 be selected and, consequently, $D_X \cdot D_Y$ and D_X^2 be included in steps 2 and 3. If the procedure terminates after step 3, the result is a polynomial in the two depths $D_X(\mathbf{x})$ and $D_Y(\mathbf{x})$ that has form

$$aD_X(\mathbf{x}) + bD_X^2(\mathbf{x}) + cD_Y^2(\mathbf{x}) + dD_X(\mathbf{x})D_Y(\mathbf{x}).$$

A given point $\mathbf{x}$ of the object space then is classified according to the sign of the polynomial.

If there are more than two classes, say $X_1, \dots, X_q$, each data point $\mathbf{x}_i$ is represented by the vector of depth values $\mathbf{d} = \big(D_{X_1}(\mathbf{x_i}), \dots, D_{X_q}(\mathbf{x}_i)\big)$ in $[0,1]^q$. Again a depth feature space is considered of some order p; it has dimension $r = \binom{p+q}{q} - 1$. With $q > 2$ classes every two training classes $X_j, X_k, j \neq k$, are separated by the α-procedure in the same way as above: In each step a pair of D-features is replaced by a new D-feature as long as the AMR decreases and basic D-features are left to be replaced. For each pair of classes the procedure results in a hypersurface that separates the q-dimensional depth space into two sets of attraction. A given point $\mathbf{x}$ is finally assigned to that class to which it has been most often attracted.

2.4 Some theoretical aspects

In order to investigate some properties of the $DD\alpha$-approach we transfer it to a more general probabilistic setting and define a depth function as the population version of a data depth. Let $\mathcal{P}$ be a properly chosen set of probability distributions on $\mathbb{R}^d$ that includes the empirical distributions. A *depth function* D is a function that assigns a value $D_P(\mathbf{x}) \in [\mathbf{0}, \mathbf{1}]$ to every $\mathbf{x} \in \mathbb{R}^d$ and $P \in \mathcal{P}$ in an affine invariant way (i.e. $D_{AP+b}(A\mathbf{x}+b) = D_P(\mathbf{x})$ for any nonsingular matrix $A \in \mathbb{R}^{d\times d}$ and any $b \in \mathbb{R}^d$, AP denoting the push-forward measure), and has convex compact upper level sets. Obviously, the restriction of a depth function D to the class of empirical distributions is an affine invariant quasiconvex data depth. For details on general depth functions, see e.g. the above cited surveys (Cascos, 2010, Mosler, 2002, Serfling, 2006, Zuo & Serfling, 2000).

While data depth is an intrinsically nonparametric notion, the behavior of depth functions and depth based procedures on parametric classes is of special interest as it indicates how the nonparametric approach relates to the more classical parametric one. As a generalization of multivariate Gaussian distributions, spherical and elliptical distributions play an important role in parametric multivariate analysis. A random vector $\mathbf{X}$ in $\mathbb{R}^d$ has a *spherical distribution* if $\mathbf{X} = R \cdot \mathbf{U}$, where $\mathbf{U}$ is a random vector uniformly distributed on the sphere S^{d-1} and R is a random variable having support $[0, \infty[$ and being independent of $\mathbf{U}$. A random vector $\mathbf{Y}$ has an *elliptical distribution* if it is an affine transform of a spherically distributed $\mathbf{X}$, $\mathbf{Y} = \mu + B\mathbf{X}$. If R has a density r we notate $\mathbf{Y} \sim \text{Ell}(\mu, BB', r)$. As, by definition, a depth function is affine invariant, it operates on elliptical distributions in a rather simple way. The following propositions give some insight into the the behavior of depth functions and the $DD\alpha$-procedure if the data generating processes are elliptical.

Proposition 2.1. *If D is an affine invariant depth function and P an elliptical distribution, then for every $\alpha \in]0, 1]$ the upper level set*

$$D_\alpha(P) = \{\mathbf{x} \in \mathbb{R}^d | D_P(\mathbf{x}) \geq \alpha\}$$

is an ellipsoid.

Proof. Let $P = \text{Ell}(\mu, BB', r)$ and $\alpha \in]0, 1]$. Consider $P_0 = \text{Ell}(\vec{0}, I_d, r)$. Then, for all $\beta \geq \alpha$, $\{\mathbf{x} \in \mathbb{R}^d | D_{P_0}(\mathbf{x}) = \beta\}$ is a sphere since D is, in particular, orthogonal invariant. Hence, $D_\alpha(P_0) = \{\mathbf{x} \in \mathbb{R}^d | D_{P_0}(\mathbf{x}) \geq \alpha\}$ is a ball and, by affine transformation with μ and B, $D_\alpha(P)$ is an ellipsoid. □

Proposition 2.2. **(i)** *Let D be the zonoid depth and P a unimodal elliptical distribution, that is $P = \text{Ell}(\mu, BB', r)$. Then, for every non-empty density level set $\{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) \geq \beta\}$, some $\alpha = \phi(\beta)$ exists such that*

$$\{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) \geq \beta\} = D_\alpha(P)\,.$$

(ii) *If, in addition, r has an interval support then ϕ is a continuous, strictly increasing function. It holds $D_P(\mathbf{x}) = \phi(f(\mathbf{x}))$ and therefore*

$$f(\mathbf{x}) \geq f(\mathbf{y}) \quad \Longleftrightarrow \quad D_P(\mathbf{x}) \geq D_P(\mathbf{y})\,. \tag{2.5}$$

Proof. (i): Note that $D_0 = \mathbb{R}^d$. Thus, if $\beta \leq 0$, the claim holds with $\alpha = 0$. Now let $\beta > 0$ and assume w.l.o.g. that P is spherical. Then $\{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) \geq \beta\}$ is a ball with center at the origin. Let $\mathbf{x}^*$ be a point on its surface. Also the central regions D_α are balls around the origin. By Theorems 3.9 and 3.14 in Mosler (2002), the D_α are continuous and strictly decreasing on the convex hull of the support of P and it holds $\alpha^* := D_P(\mathbf{x}^*) > 0$. We conclude $D_{\alpha^*} = \{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) \geq \beta\}$.
(ii): Under the additional premise, the density level sets are continuously and strictly decreasing in $\beta > 0$, which yields the result. □

Corollary 2.1. *Consider a mixture of unimodal elliptical distributions $P_j = \text{Ell}(\mu_j, B_j B_j', r_j)$, $j = 1, \ldots, q$, with mixing probabilities π_j and assume that all r_j have an interval support. Let D be the zonoid depth.*

Then, for each j and k exists a strictly increasing function ψ_{jk} so that

$$\pi_j \cdot f_j(\mathbf{x}) < \pi_k \cdot f_k(\mathbf{x}) \quad \Longleftrightarrow \quad D_{P_j}(\mathbf{x}) < \psi_{jk}(D_{P_k}(\mathbf{x}))\,.$$

Proof. From Proposition 2.2 continuous and strictly increasing functions ϕ_j and ϕ_k are obtained with $D_{P_j}(\mathbf{x}) = \phi_j(f_j(\mathbf{x}))$ and $D_{P_k}(\mathbf{x}) = \phi_k\big(f_k(\mathbf{x})\big)$. Consequently,

$$\pi_j \cdot f_j(\mathbf{x}) < \pi_k \cdot f_k(\mathbf{x}) \quad \Leftrightarrow \quad D_{X_j}(\mathbf{x}) < \phi_j\left(\frac{\pi_k}{\pi_j}\,\phi_k^{-1}\big(D_{X_k}(\mathbf{x})\big)\right)\,,$$

which proves the claim by use of the function $\psi_{jk}(\cdot) = \phi_j\left(\frac{\pi_k}{\pi_j}\,\phi_k^{-1}(\cdot)\right)$. ☐

A similar result holds for other data depths including the halfspace, simplicial, projection and Mahalanobis depths; see Prop. 1 in Li *et al.* (2012). In the rest of section we consider the limit behavior of the $DD\alpha$-procedure under independent sampling. For this, we assume that the empirical depth is a consistent estimator of its population version. This is particularly true for the zonoid, halfspace, simplicial, projection and Mahalanobis depths.

Theorem 2.1 (Bayes rule)**.** *Let F and G probability distributions on $\mathbb{R}^d$ having densities f and g, and let H be a hyperplane such that G is the mirror image of F with respect to H and $f \geq g$ in one of the half-spaces generated by H. Then based on a 50:50 independent sample from F and G the $DD\alpha$-procedure will asymptotically yield the linear separator that corresponds to the bisecting line of the DD-plot.*

Note that the rule given in the theorem corresponds to the Bayes rule, see Hastie *et al.* (2009). Especially the requirements of the theorem are satisfied if F and G are mirror symmetric and unimodal.

Proof. Due to the mirror symmetry of the distributions in $\mathbb{R}^d$ the DD-plot is symmetric as well. Symmetry axis is the bisector, which is obviously the result of the α-procedure when the sample is large enough. ☐

Theorem 2.2. *Let F, G be unimodal elliptical, $F = \mathrm{Ell}(\mu_F, BB', r)$, $G = \mathrm{Ell}(\mu_G, BB', r)$. Then based on a 50:50 independent sample from F and G the $DD\alpha$-procedure will asymptotically yield the linear separator that corresponds to the bisecting line of the DD-plot.*

Proof. If F and G are spherically symmetric, they satisfy the premise of the previous theorem. A common affine transformation of F and G does not change the DD-plot. ☐

2.5 Simulation study

The $DD\alpha$-procedure has been implemented on a standard PC in an R-environment. To explore its specific potencies we apply it to simulated as well as to real data. The same data have been analyzed with several classifiers in the literature. In this section results on simulated data are

Table 2.1: *Distributional settings used in the simulation study.*

No.	Alternative	1st class	2nd class
1	Normal location	$\mathrm{N}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$	$\mathrm{N}\left(\begin{bmatrix}1\\1\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$
2	Normal location-scale	$\mathrm{N}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$	$\mathrm{N}\left(\begin{bmatrix}1\\1\end{bmatrix},\begin{bmatrix}4&4\\4&16\end{bmatrix}\right)$
3	Cauchy location	$\mathrm{Cauchy}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$	$\mathrm{Cauchy}\left(\begin{bmatrix}1\\1\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$
4	Cauchy location-scale	$\mathrm{Cauchy}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$	$\mathrm{Cauchy}\left(\begin{bmatrix}1\\1\end{bmatrix},\begin{bmatrix}4&4\\4&16\end{bmatrix}\right)$
5	Normal contaminated location	Learning sample: 90% as No. 1, 10% from $\mathrm{N}\left(\begin{bmatrix}10\\10\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$. Testing sample: as No. 1	as No. 1
6	Normal contaminated location-scale	Learning sample: 90% as No. 2, 10% from $\mathrm{N}\left(\begin{bmatrix}10\\10\end{bmatrix},\begin{bmatrix}1&1\\1&4\end{bmatrix}\right)$. Testing sample: as No. 2	as No. 2
7	Exponential location	$\big(\mathrm{Exp}(1),\mathrm{Exp}(1)\big)$	$\big(\mathrm{Exp}(1)+1,\mathrm{Exp}(1)+1\big)$
8	Exponential location-scale	$\big(\mathrm{Exp}(1),\mathrm{Exp}(1/2)\big)$	$\big(\mathrm{Exp}(1/2)+1,\mathrm{Exp}(1)+1\big)$
9	Asymmetric location	$\big(\mathrm{MixN}(0;1,2),\mathrm{MixN}(0;1,4)\big)$	$\big(\mathrm{MixN}(1;1,2),\mathrm{MixN}(1;1,4)\big)$
10	Normal-exponential	$\mathrm{N}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}\right)$	$\big(\mathrm{Exp}(1),\mathrm{Exp}(1)\big)$

presented regarding the average misclassification rate of nine procedures besides the $DD\alpha$-classifier (Section 2.5.1). Then the speed of the $DD\alpha$-procedure is quantified (Section 2.5.2). The following Section 2.6 covers the relative performance of the the $DD\alpha$- and other classifiers on several benchmark data sets.

2.5.1 Comparison of performance

To simplify the comparison with known classifiers, we use the same simulation settings as in Li *et al.* (2012). These are supervised classification tasks with two equally sized training classes. Data are generated by ten pairs of distributions according to Table 2.1. Here N and Exp denote the Gaussian and exponentional distributions, respectively, and

$$\mathrm{MixN}(\mu,\sigma_1,\sigma_2)=\begin{cases}-\sigma_1 * |\mathrm{N}(0,1)|+\mu & \text{with probability } 1/2,\\ \sigma_2 * |\mathrm{N}(0,1)|+\mu & \text{with probability } 1/2.\end{cases}$$

The $DD\alpha$-classifier is contrasted with the following nine classifiers: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), k-nearest neighbors classification (k-NN), maximum depth classification based on Mahalanobis (MM), simplicial (MS), and halfspace (MH) depth, and DD-classification with the same depths (DM, DS and DH, correspondingly). For more details about the data and the procedures as well as for some motivation the reader is referred to Li *et al.* (2012).

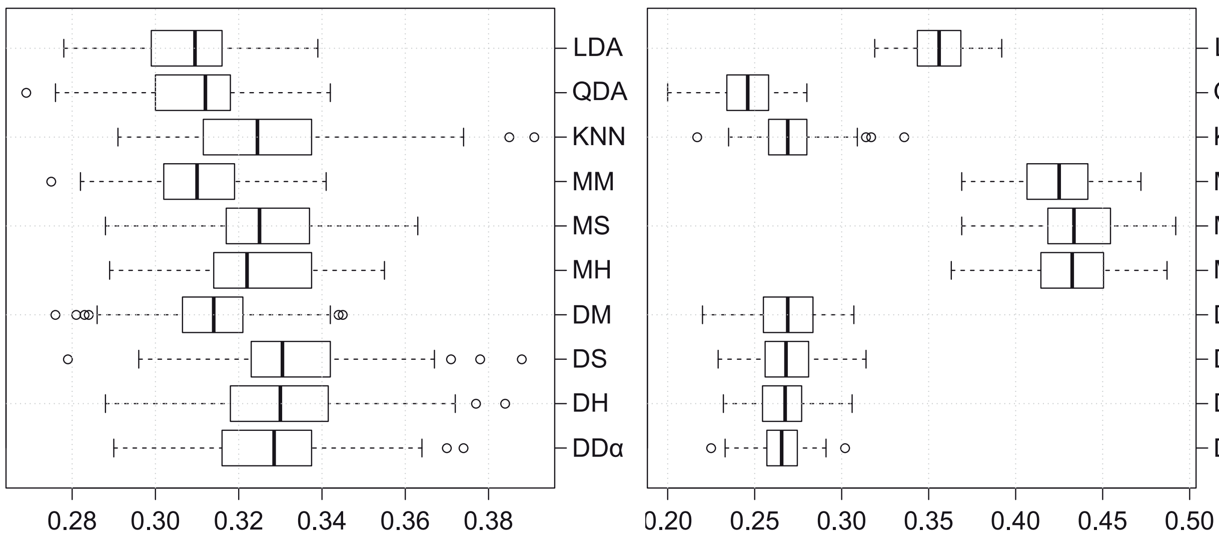

Figure 2.2: *Normal location (left) and location-scale (right) alternatives.*

All simulations of Li *et al.* (2012) are recalculated following their paper as close as possible. The LDA, QDA and k-NN classifiers are computed with the R-packages “MASS” and “class”, where the parameter k of the k-NN-classifier is selected by leave-one-out cross-validation over a relatively wide range. The simplicial, and halfspace depths have been determined by exact calculations with the R-package “depth”. The zonoid depth has been exactly computed by the algorithm in Dyckerhoff *et al.* (1996). Note that the problem of prior probabilities is avoided by choosing test samples of equal size from both classes.

For the DD-classifiers a polynomial line (up to degree three) is determined to discriminate in the two-dimensional DD-plot, a tenfold cross-validation is employed to choose the optimal degree of the polynomial, a smoothing constant t=100 is selected in the logistic function, and the DD-plot is never rotated. Each experiment includes a training phase and an evaluation phase: From the given pair of distributions 400 observations (200 of each class) are generated to train the classifier, and 1000 (500 of each) observations to evaluate its AMR. For each distribution pair and each classifier 100 experiments are performed, and the resulting sample of AMRs is visualized as a box-plot; see Figures 2.2 to 2.6.

As we have discussed at the end of Section 2.2, with depths like the simplicial, halfspace and zonoid depth the problem of outsiders arises. An outsider is, in the DD-plot, represented by the origin. A simple approach is to assign the outsiders randomly to the two classes. Throughout our simulation study we have chosen the random assignment rule, which results in kind of worst case AMR. Observe that this choice of assignment rule discriminates against the procedures that yield outsiders and advantages those that do not, in particular LDA, QDA, MM, DM and k-NN for all distribution settings.

The principal results of the simulation study are collected in Figures 2.2 to 2.6. Under the normal location-shift model (Figure 2.2, left) all classifiers behave satisfactorily, and the $DD\alpha$-classifier performs well among them. However LDA, QDA, MM and DM show slightly better results since they do not have to cope with outsiders like the other depth-based procedures.

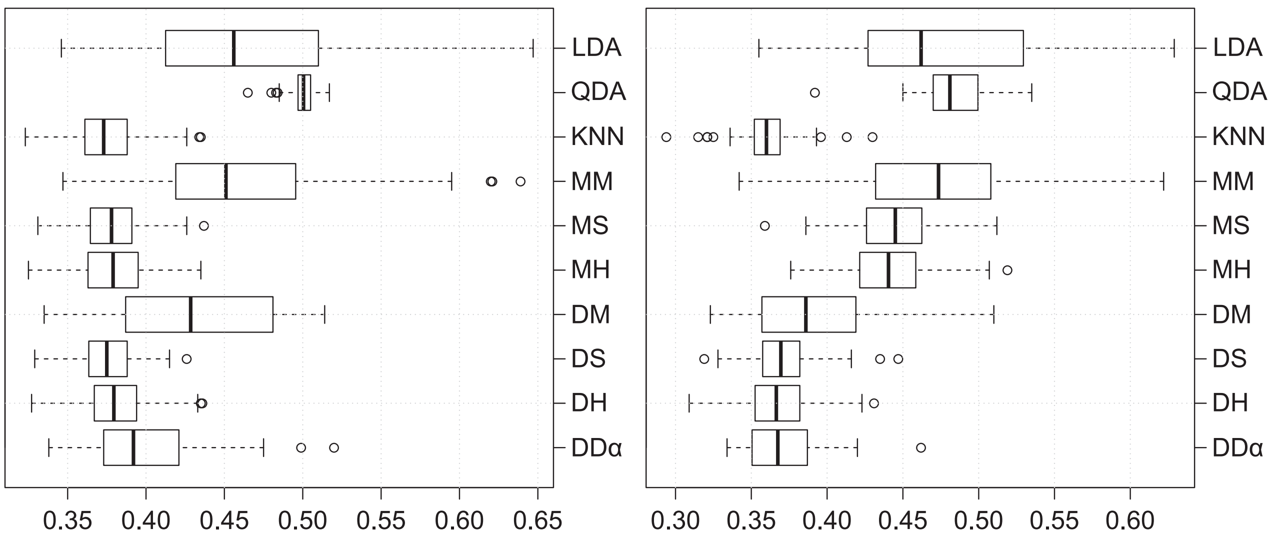

Figure 2.3: *Cauchy location (left) and location-scale (right) alternatives.*

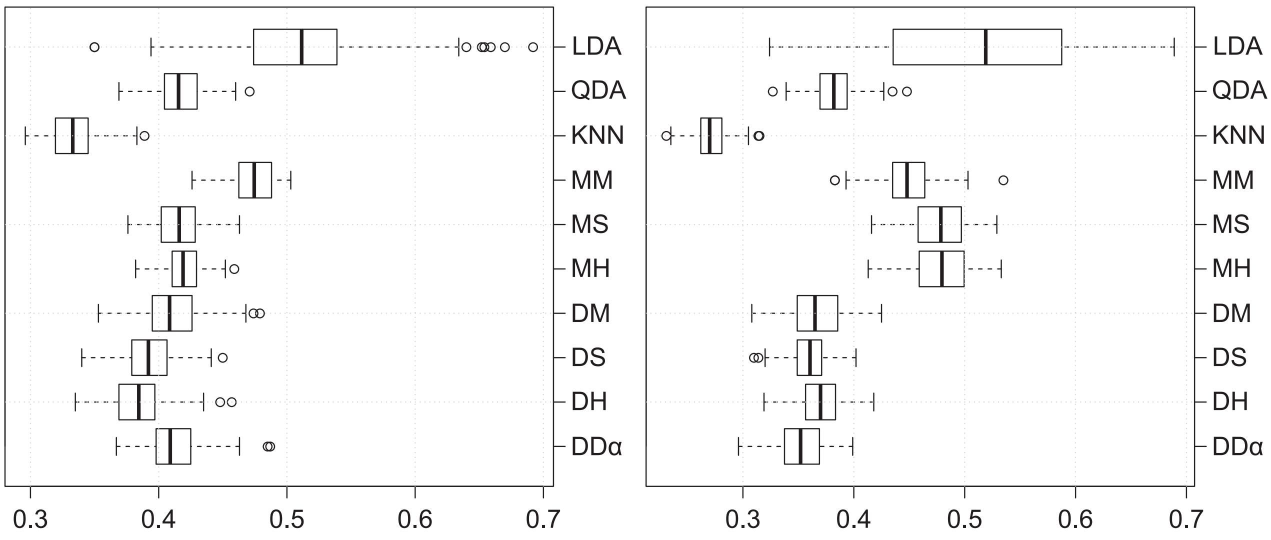

Figure 2.4: *Normal contaminated location (left) and location-scale (right) alternatives.*

Also under the normal location-scale alternative (Figure 2.2, right) the $DD\alpha$-classifier performs rather well, like all DD-classifiers. A slightly worse performance of the $DD\alpha$-classifier is observed when discriminating the Cauchy location alternative (Figure 2.3, left), but it is still close to the DD-classifiers. This can be attributed to the lower robustness of the zonoid depth. However, when scaling enters the game (Cauchy location-scale alternative, Figure 2.3, right), the $DD\alpha$-classifier again performs quite satisfactorily. The same picture arises when considering contaminated normal settings (Figure 2.4, left and right). Under a location alternative, the $DD\alpha$-classifier is a bit worse than the DD-classifiers, while it slightly outperforms them in a location-scale setting.

The relative robustness of the $DD\alpha$-classifier may be explained by two of its features: First it maps the original data points to a compact set, the q-dimensional unit hypercube. Second, for classification in the unit hypercube, it employs the α-procedure, which, by choosing a median angle in each step, is rather insensitive to outliers.

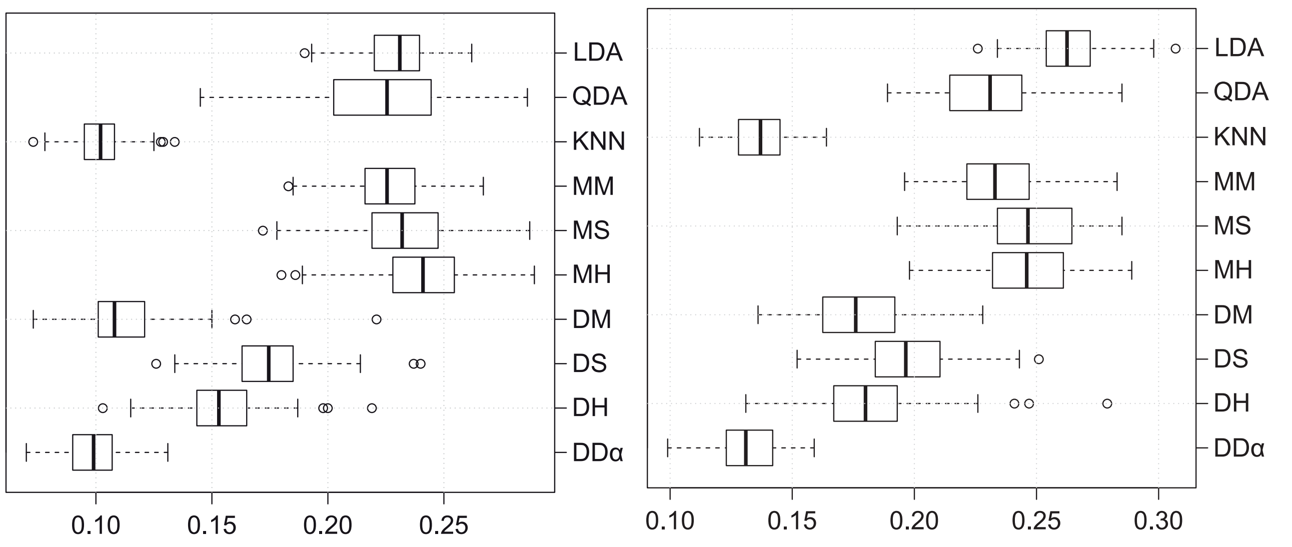

Figure 2.5: *Exponential location (left) and location-scale (right) alternatives.*

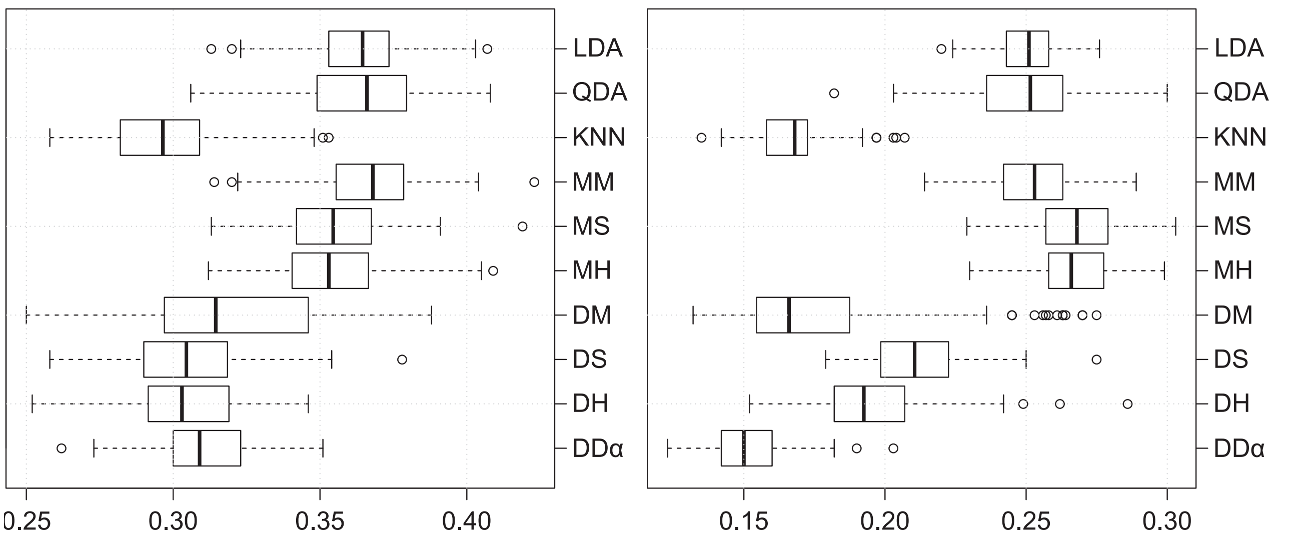

Figure 2.6: *Asymmetric location (left) and normal-exponential (right) alternatives.*

Under exponential alternatives (Figure 2.5, left and right) the $DD\alpha$-classifier shows excellent performance, which is even similar to that of the the k-NN for both location and location-scale alternatives. Its results for the asymmetric location alternative (Figure 2.6, left) are somewhat ambiguous, though still close to those of the DD-classifiers. Concerning the normal-exponential alternative (Figure 2.6, right) the $DD\alpha$-classifier performs distinctly better than the others considered here.

On the basis of the simulation study we conclude: The $DD\alpha$-classifier (1) performs quite well under various settings of elliptically distributed alternatives, it (2) is rather robust to outlier prone data, and (3) shows a distinctly good behavior under the asymmetrically distributed alternatives considered and when the two classes originate from different families of distributions.

Table 2.2: *Computing times of $DD\alpha$-classification, in seconds.*

	$\mathrm{N}(\mathbf{0}_d, \mathbf{I}_d)$ *vs.* $\mathrm{N}(0.25 \cdot \mathbf{1}_d, \mathbf{I}_d)$			
	$d = 5$	$d = 10$	$d = 15$	$d = 20$
$n = 200$	0.14 (0.00014)	1.55 (0.00014)	1.89 (-)	2.24 (-)
$n = 500$	1.04 (0.00046)	10.37 (0.00052)	12.58 (0.00062)	14.14 (-)
$n = 1000$	5.33 (0.0012)	42.54 (0.0014)	53.66 (0.0017)	59.18 (-)

	$\mathrm{N}(\mathbf{0}_d, \mathbf{I}_d)$ *vs.* $\mathrm{N}\big((0.25\ \mathbf{0}'_{d-1})', 5 \cdot \mathbf{I}_d\big)$			
	$d = 5$	$d = 10$	$d = 15$	$d = 20$
$n = 200$	0.15 (0.00014)	1.62 (0.00016)	1.94 (0.00021)	2.2 (0.00027)
$n = 500$	1.09 (0.00044)	11.33 (0.00059)	14.44 (0.00079)	15.18 (0.0010)
$n = 1000$	5.24 (0.0011)	47.63 (0.0016)	67.22 (0.0022)	74.15 (0.0026)

2.5.2 Speed of the $DD\alpha$-procedure

To estimate the speed of the $DD\alpha$-classification we have quantified the total time of training and classification times under two simulation settings, a shift and a location-shift alternative concerning d-variate normals (see Table 2.2, header), with various values of dimension d and of total size of training classes n. An experiment consists of a training phase based on two samples (each of size $n/2$) and an evaluation phase, where 2500 points (1250 from each distribution) are classified. Each experiment is performed 100 times, then the average computation time is determined. All these computations have been conducted on a single kernel of the processor Core i7-2600 (3.4 GHz) having enough physical memory.

Table 2.2 exhibits the average computation times (in seconds, with the standard deviations in parentheses) under the two distributional settings and for different d and n. As it is seen from the table, the $DD\alpha$-classifier is very fast, in the learning phase as well as in classifying high amounts of data. However, computation times increase considerably with the number of training points, which is due to the many calculations of zonoid depth needed. With dimension d computation time grows slower, which may be explained as follows. With increasing dimension of the data space, more points come to lie on the convex hull (thus having depth $= 2/n$) or outside it (thus having depth $= 0$). The algorithm from Dyckerhoff *et al.* (1996) computes the depth of such points much faster than that of points having larger depths.

2.6 Benchmark studies

Concerning real data, we take benchmark examples from Li *et al.* (2012), Dutta & Ghosh (2011, 2012) to compare the performance of the $DD\alpha$-classifier with respect to AMR (Section 2.6.1).

Table 2.3: *Overview of benchmark examples; dimension (d), number of classes (q), number of training points (# train), number of testing points (# test), total number of points (# total).*

No.	Dataset	Results	q	d	# train	# test	# total
1	Biomedical	Tables 2.5, 2.4	2	4	150	44	194
		Table 2.6	2	4	100	94	194
2	Blood Transfusion	Table 2.6	2	3	374	374	748
		Table 2.4	2	3	500	248	748
3	Diabetes (1)	Table 2.6	3	5	100	45	145
4	Diabetes (2)	Table 2.7	2	8	767	1	768
5	Ecoli	Table 2.7	3	7	271	1	272
6	Glass	Tables 2.5, 2.6	2	5	100	46	146
		Table 2.7	2	9	145	1	146
7	Hemophilia	Table 2.6	2	2	50	25	75
8	Image Segmentation	Table 2.4	2	10	500	160	660
9	Iris	Table 2.7	3	4	149	1	150
10	Synthetic	Tables 2.5, 2.6	2	2	250	1000	1250

Table 2.4: *Benchmark performance with DD- and other classifiers.*

Dataset	LDA	QDA	k-NN	MM	MH	DM	DH	$DD\alpha$
Biomedical	17.05 (0.49)	13.05 (0.38)	14.32 (0.45)	27.14 (0.6)	18.00 (0.49)	12.25 (0.4)	17.48 (0.51)	24.59 (0.63)
Blood Transfusion	29.49 (0.08)	29.11 (0.13)	29.74 (0.13)	32.56 (0.29)	30.47 (0.3)	26.82 (0.19)	28.26 (0.19)	32.27 (0.25)
Image Segmentation	8.17 (0.2)	9.44 (0.19)	5.59 (0.19)	9.12 (0.23)	11.87 (0.25)	9.54 (0.2)	13.98 (0.29)	43.58 (0.34)

In addition we use four real data sets from the UCI machine learning repository (Asuncion & Newman, 2007) to contrast the $DD\alpha$-classifier with the support vector machine (SVM) of Vapnik (1998) regarding both performance and time (Section 2.6.2).

2.6.1 Benchmark comparisons with nonparametric classifiers

As our benchmark examples are well known, we refer to the literature for their detailed description and restrict ourselves to mentioning the dimension d, the number of classes q, the number of points used for training (# train), the number of testing points (# test) and the total number of points (# total); see Table 2.3.

Tables 2.4, 2.5 and 2.6 exhibit the performance (in terms of AMR, with standard errors in parentheses) of the $DD\alpha$-classifier together with the performance of the different classifiers investigated in Li *et al.* (2012), Dutta & Ghosh (2011) and Dutta & Ghosh (2012) and based on the respective benchmark data. When applying the $DD\alpha$-classifier an auxiliary procedure has to be chosen by which outsiders are treated. In our benchmark study we employ several such procedures.

In Table 2.4 the $DD\alpha$-procedure is contrasted with the real data results in Li *et al.* (2012). Here we use the same settings as in Section 2.5.1 and classify the outsiders on a random basis. All results in Table 2.4 have been recalculated.

As we see from the Table, the performance of our new classifier is mostly worse than the classifiers considered in Li *et al.* (2012). Only in the Blood Transfusion case the AMR has comparable size. However, in this comparison the eventual presence and treatment of outsiders plays a decisive role. Observe that Li *et al.* (2012) in their procedures MH and DH use the random Tukey depth (Cuesta-Albertos & Nieto-Reyes, 2008) to approximate the halfspace depth of a data point in dimension three and more. But the random Tukey depth generally overestimates the halfspace depth so that some of the outsiders remain undetected. This implies that, in the procedures MH and DH, considerably fewer points (we observed around 16%, 4% and 11% correspondingly) are treated as outsiders and assigned on a random basis.

In fact, as exactly determined by calculating the zonoid depth, the rate of outsiders in the Biomedical Data (with $d = 4$) totals some 35%, in the Blood Transfusion Data ($d = 3$) about 11%, and in the Image Segmentation Data with $d = 10$ about 86%. This is in line with our expectation: the higher the dimension of the data the higher is the outsider rate. In contrast to the MH and DH procedures, the $DD\alpha$-procedure detects all outsiders and, in the comparison of Table 2.4, assigns them randomly. Obviously the performance of the latter can be improved with a proper non-random procedure of outsider assignment. In the subsequent benchmark comparisons several such procedures of non-random outsider assignment are included.

Dutta & Ghosh (2012) introduce classification based on projection depth and compare it with several variants of the maximum-Mahalanobis-depth (MD)classifier. The same authors (Dutta & Ghosh, 2011) propose an L_p-depth classifier (with optimized p) and contrast it with two types of MD. To compare the $DD\alpha$-classifier on a par with Dutta & Ghosh (2011, 2012) we implement the following rules for handling outsiders: First, k-nearest-neighbor rules are used with various k and either Euclidean or Mahalanobis distance, the latter with moment or, alternatively, MCD estimates. Second, maximum Mahalanobis depth is employed, again based on moment or MCD estimation. As the k-NN results of the benchmark examples do not vary much with k, we restrict to $k = 1$. (However, the performance of the classifiers can be improved by an additional cross-validation over k.) Consequently, five different rules for treating outsiders remain for comparison. Tables 2.5 and 2.6 exhibit the performance of the $DD\alpha$-classifier *vs.* the projection-depth classifiers of Dutta & Ghosh (2012) and the L_p-depth classifiers of Dutta & Ghosh (2011), respectively, regarding the benchmark examples investigated in these papers. The last five columns of Tables 2.5 and 2.6 report the AMR (standard deviations in parentheses) of the $DD\alpha$-classifier when one of the five outsiders treatments is chosen. The remaining columns are adopted as they stand in Dutta & Ghosh (2012) and Dutta & Ghosh (2011).

Regarding the Biomedical Data, Dutta & Ghosh (2012) do not specify the sample sizes they use in training and testing. For the $DD\alpha$-classifier, we select 100 observations of the larger class

Table 2.5: *Benchmark comparison with projection depth classifiers.*

Dataset	MD (SS)	MD (MS)	$\text{MD}_{\frac{3}{4}}$ (SS)	$\text{MD}_{\frac{3}{4}}$ (MS)	PD (SS)	PD (MS)
Synthetic	13.00	11.60	10.30	10.40	10.00	10.50
Glass	26.59 (0.25)	26.14 (0.25)	24.92 (0.25)	24.43 (0.25)	25.70 (0.34)	25.24 (0.33)
Biomedical	12.44 (0.13)	12.04 (0.12)	14.25 (0.13)	14.03 (0.14)	12.37 (0.14)	12.18 (0.13)

Dataset	$DD\alpha$-**classifier**				
	1-NN			Mahalanobis depth	
	Eucl. dist.	Mah. dist.			
		Mom.	MCD	Mom.	MCD
Synthetic	12.10	11.90	12.00	11.90	12.00
Glass	29.45 (0.20)	25.79 (0.17)	24.73 (0.18)	30.09 (0.18)	35.06 (0.22)
Biomedical	13.51 (0.14)	19.59 (0.18)	17.90 (0.17)	12.91 (0.14)	15.23 (0.16)

Table 2.6: *Benchmark comparison with L_p-depth classifiers.*

Data-set	MD		L_pD		$DD\alpha$-**classifier**				
					1-NN			Mahalanobis depth	
					Eucl. dist.	Mah. dist.			
	Mom.	MCD	Mom.	MCD		Mom.	MCD	Mom.	MCD
Syn.	10.20	10.60	9.60	10.70	12.10	11.90	12.00	11.90	12.00
Hem.	15.84 (0.30)	17.13 (0.32)	15.39 (0.32)	16.43 (0.32)	16.63 (0.20)	17.98 (0.20)	18.36 (0.19)	18.65 (0.22)	19.39 (0.22)
Gla.	26.80 (0.26)	24.80 (0.29)	27.64 (0.29)	24.75 (0.26)	30.13 (0.19)	28.37 (0.22)	26.63 (0.20)	32.88 (0.22)	36.82 (0.23)
Biom.	12.35 (0.14)	14.48 (0.15)	12.68 (0.15)	15.11 (0.15)	13.74 (0.09)	22.09 (0.16)	20.89 (0.14)	14.34 (0.12)	17.28 (0.14)
Diab.	8.22 (0.18)	11.49 (0.22)	9.39 (0.21)	11.92 (0.27)	10.77 (0.12)	18.36 (0.18)	18.33 (0.20)	12.70 (0.18)	15.90 (0.19)
B.Tr.	22.75 (0.07)	22.17 (0.08)	22.30 (0.07)	22.06 (0.07)	23.11 (0.06)	22.73 (0.06)	22.92 (0.06)	22.59 (0.06)	22.17 (0.06)

and 50 of the smaller class to form the training sample; the remaining observations constitute the testing sample. As it is seen from Table 2.5 the $DD\alpha$-classifier shows results similar to the projection-depth classifier (except with the Synthetic Data), while the performance of outsider-handling methods varies depending on the type of the data. Specifically, with the Glass Data 1-NN based on the Mahalanobis distance (both with the moment and the robust estimate) performs best in handling outsiders. On the other hand, with the Biomedical Data the same approach performs quite poorly, while treating outsiders with moment-estimated Mahalanobis depth or Euclidean 1-NN yields best results.

Table 2.6 presents a similar comparison of the $DD\alpha$-classifier with the L_p-classifier of Dutta & Ghosh (2011). The same approaches are included to treat outsiders. In all six benchmark examples the $DD\alpha$-classifier generally performs worse than the best L_p-depth classifier. However, its performance substantially depends on the chosen treatment of outsiders. In all examples the AMR of the $DD\alpha$-classifier comes close to that of the L_p-depth classifier, provided the outsider treatment is properly selected. On the Hemophilia Data, e.g., Euclidean 1-NN should be chosen. On the Glass Data a 1-NN outsider treatment with robust Mahalanobis distance performs relatively best, etc. On the Blood Transfusion Data all outsider-handling approaches show equally good performance, which appears to be typical when n is relatively large compared to d.

2.6.2 Benchmark comparisons with SVM

The support vector machine (SVM) is a powerful solver of the classification problem and has been widely used in applications. However, different from the $DD\alpha$-classifier, the SVM is a parametric approach, as in applying it certain parameters have to be adjusted: the box-constraint and the kernel parameters. The AMR performance of the SVM depends heavily on the choice of these parameters. In applications, optimal parameters are selected by some cross-validation, which affords extensive calculations. Once these parameter have been optimized, SVM-classification is usually very fast and precise.

In comparing the SVM with the $DD\alpha$-procedure, this step of parameter optimization has to be somehow accounted for. Here we introduce a two-fold view on the comparison problem: Two values of the AMR are calculated, first the *best AMR* when the parameters have been optimally selected, second the *expected AMR* when the parameters are systematically varied over specified ranges. Corresponding training times are also clocked. As ranges we choose the intervals between the smallest and the largest number that arise as an optimal value in one of our benchmark data examples. This seems to be a fair and, regarding the parameter ranges, rather conservative approach.

As benchmark four well-known data sets are employed in the sequel, Diabetes, Ecoli, Glass, and Iris Data being taken from Asuncion & Newman (2007). Following Dutta & Ghosh (2012) the two biggest classes of the Glass Data have been selected, and similarly to Dutta & Ghosh (2011) we have chosen three of the bigger classes from the Ecoli Data. The $DD\alpha$-classifier is calculated with the same outsider treatments as above. For the SVM-classifier we use radial basis function kernels as implemented in LIBSVM with the R-Package "e1071" as an R-interface. Leave-one-out cross validation is employed for performance estimation of the all classifiers. The computation has been done on the same PC as in Section 2.5.2.

The results on the best AMR together with time quantities and portions of outsiders are collected in the Table 2.7. The Iris Data appears twice in the Table. First the original are used, and second the same data after a preprocessing step. The preprocessing consists in the exclusion of an obvious outlier in the DD-plot that was identified by visual inspection of the plot.

Table 2.7: *Benchmark comparison with the support vector machine; γ – kernel parameter, C – box constraint.*

Data-set	Legend	$DD\alpha$-classifier					**SVM**
		1-NN			Mahalanobis depth		
		Eucl. dist.	Mah. dist. Mom.	Mah. dist. MCD	Mom.	MCD	Opt. (CV)
Diab.	Error	28.26	30.6	34.51	24.35	31.77	23.18
	Time:train	16.63	16.62	16.59	16.58	17.39	0.05 (875)
	Time:test	0.033	0.009	0.0092	0.0035	0.0037	0.0023
	γ/C						0.056/1
	% outsiders	62.24	62.24	62.24	62.24	63.54	
Ecoli	Error	10.29	11.4	12.13	12.13	16.18	3.68
	Time:train	0.26	0.26	0.26	0.26	0.26	0.0077 (105)
	Time:test	0.014	0.0026	0.0032	0.001	0.00044	0.0019
	γ/C						5.62/1.78
	% outsiders	75	75	75	75	75	
Glass	Error	18.49	26.03	31.51	34.93	34.93	21.23
	Time:train	0.31	0.32	0.31	0.32	0.32	0.0082 (36)
	Time:test	0.0083	0.0019	0.0016	0.00014	0.00055	0.0024
	γ/C						0.56/1
	% outsiders	95.89	95.89	95.89	95.89	95.89	
Iris	Error	37.33	37.33	37.33	36	46.67	4.67
	Time:train	0.07	0.07	0.07	0.07	0.07	0.0051 (30)
	Time:test	0.0046	0.0018	0.0013	0.00033	0.00047	0.0017
	γ/C						0.056/10
	% outsiders	50	50	50	50	50	
Iris (Pre.)	Error	3.36	3.36	4.03	2.68	13.42	2.68
	Time:train	0.07	0.07	0.07	0.07	0.07	0.0052 (30)
	Time:test	0.0046	0.0011	0.0013	0.0006	0.00027	0.0017
	γ/C						0.1/3.16
	% outsiders	51.68	51.68	51.68	51.68	51.68	

The overall analysis of the Table 2.7 shows that, even if using an arbitrary technique for handling outsiders, the $DD\alpha$-classifier mostly performs not much worse than an SVM where the parameters have been optimally chosen. In contrast, if the SVM is employed with some non-optimized parameters, its AMR can be considerably larger than that of the $DD\alpha$-classifier. For the regarded data sets average errors of the SVM over the relevant intervals varied from 44.99% to 66.67% (not reported in the Table).

The times needed to classify a new object (also given in Table 2.7) are quite comparable. But as the parameters of the SVM have to be adjusted first by running it many times for cross-validation, the computational burden of its training phase is much higher than that of the $DD\alpha$-classifier, which has to be run only once. Recall that the latter is nonparametric regarding tuning parameters. For example, in our implementation it took 875 seconds to determine approximate optimal values of SVM parameters for the Diabetes Data and similarly substantial times for the others (see Table 2.7, in parentheses).

2.7 $DD\alpha$-classification of asymmetric and fat-tailed data

When implementing the $DD\alpha$-classifier a data depth has to be chosen and the 'outsiders' have to be treated in some way. This section addresses the question which notion of data depth should be employed. To answer it we consider two depth notions and explore their sensitivity to fat-tailedness and asymmetry of the underlying class-specific distribution. The two depths are the zonoid depth (Koshevoy & Mosler, 1997, Mosler, 2002) and the location depth (Tukey, 1975). The zonoid depth is always exactly computed, while the location depth is either exactly or approximately calculated. In a large simulation study the average error rate of different versions of the $DD\alpha$-procedure is contrasted with that of standard classifiers, given data from asymmetric and fat-tailed distributions. Similarly the performance of different classifiers is explored depending on the distance between the classes, and their speed both at the training and classification stages is investigated. In this section, we restrict ourselves to the case $q = 2$.

2.7.1 Tukey depth *vs.* random Tukey depth

Location depth, also known as halfspace or Tukey depth, is defined as

$$HD(\mathbf{x}, X) = \frac{1}{n} \min_{\mathbf{u} \in S^{d-1}} \sharp\{i : \langle \mathbf{x}^i, \mathbf{u} \rangle \geq \langle \mathbf{x}, \mathbf{u} \rangle\}, \tag{2.6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. The location depth takes only discrete values and is robust (having a large breakdown point). The location depth can be exactly computed or approximated. *Exact computation* is described in Rousseeuw & Ruts (1996) for $d = 2$ and in Rousseeuw & Struyf (1998) for $d = 3$. For bivariate data we employ the algorithm of Rousseeuw & Ruts (1996) as implemented in the R-package "depth". In higher dimensions exact computation of the location depth is possible (Liu & Zuo, 2014a), but the algorithm involves heavy computations. Cuesta-Albertos & Nieto-Reyes (2008) instead propose to approximate the location depth, using (2.6), by minimizing the univariate location depth over randomly chosen directions $u \in S^{d-1}$. Here we explore two different settings where the set of randomly chosen u is either *generated once and for all* or *generated instantly* when computing the depth of a given point. By construction, the random Tukey depth is always greater or equal to the exact location depth. Consequently, it yields fewer outsiders.

2.7.2 Simulation study

A number of experiments with simulated data is conducted. Firstly, the error rates of 17 different classifiers (see below) are evaluated on data from asymmetric t- and exponential distributions in $\mathbb{R}^2$. Then the performance dynamics of selected ones is visualized as the classification error in dependence of the Mahalanobis distance between the two classes. The third study explores the speed of solutions based on the zonoid and the random Tukey depth.

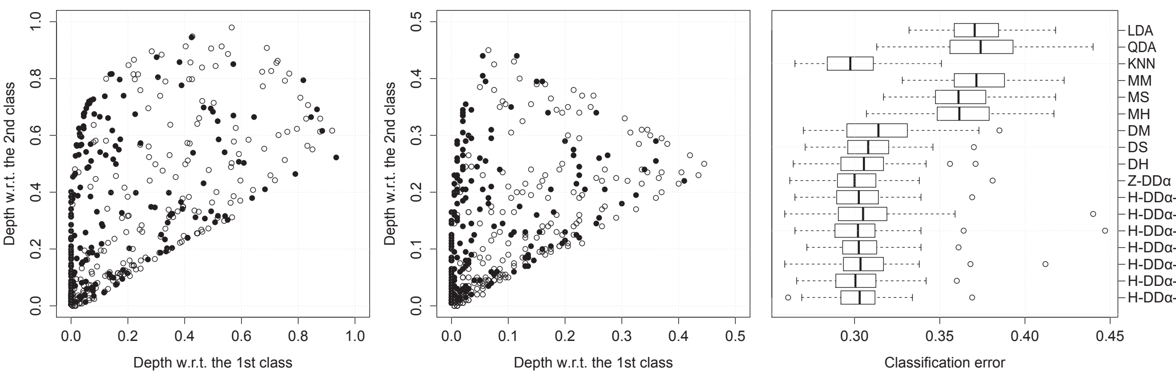

Figure 2.7: *DD-plots using zonoid (left) and location (middle) depth with black and open circles denoting observations from the two different classes and combined box-plots (right) for Gaussian location alternative*

Performance comparison

While the usual multivariate t-distribution is elliptically symmetric, it can be made asymmetric by conditioning its scale on the angle from a fixed direction, see Frahm (2004). For each degree of freedom, ∞ (= Gaussian), 5 and 1 (= Cauchy), two alternatives are investigated: one considering differences in location only (with $\mu_1 = \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$, $\mu_2 = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ and $\Sigma_1 = \Sigma_2 = \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right]$) and one differing in both location and scale (with the same μ_1 and μ_2, $\Sigma_1 = \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right]$, $\Sigma_2 = \left[\begin{smallmatrix}4&4\\4&16\end{smallmatrix}\right]$), skewing the distribution with reference vector $v_1^+ = (\cos(\pi/4), \sin(\pi/4))$, see Frahm (2004) for details. Further, the bivariate Marshall-Olkin exponential distribution (BOMED) is looked at: $(\min\{Z_1, Z_3\}, \min\{Z_2, Z_3\})$ for the first class and $(\min\{Z_1, Z_3\}+0.5, \min\{Z_2, Z_3\}+0.5)$ for the second one with $Z_1 \sim Exp(1)$, $Z_2 \sim Exp(0.5)$, and $Z_3 \sim Exp(0.75)$. Each time we generate a sample of 400 points (200 from each class) to train a classifier and a sample containing 1000 points (500 from each class) to evaluate its performance (= error rate).

DD-plots of a training sample for *the Gaussian location alternative* using zonoid (left) and location (middle) depth are shown in Figure 2.7. For each classifier, training and testing is performed on 100 simulated data sets, and a box-plot of error rates is drawn; see Figure 2.7 (right). The first group of non-depth classifiers includes linear (LDA) and quadratic (QDA) discriminant analysis and k-nearest-neighbors classifier (KNN). Then the maximal depth classifiers (MM, MS and MH; cf. Ghosh & Chaudhuri (2005b)) and the DD-classifiers (DM, DS and DH; cf. Li *et al.* (2012)) are regarded. Each triplet uses the Mahalanobis (Mahalanobis, 1936, Zuo & Serfling, 2000), simplicial (Liu, 1990) and location depths, respectively. The remaining eight classifiers are $DD\alpha$-classifiers based on zonoid depth (Z-$DD\alpha$), exactly computed location depth (H-$DD\alpha$-e), random Tukey depth for once-only (H-$DD\alpha$-$\sharp$s) and instantly (H-$DD\alpha$-$\sharp$d) generated directions, each time using $\sharp = 10, 20, 50$ random directions, respectively. The combined box-plots together with corresponding DD-plots using zonoid and location depth are presented for *the Cauchy location-scale alternative* (Figure 2.8) and *the BOMED location alternative* (Figure 2.9).

Based on these results (and many more not presented here) we conclude: In many cases $DD\alpha$-classifiers, both based on the zonoid depth and the random Tukey depth, are better than

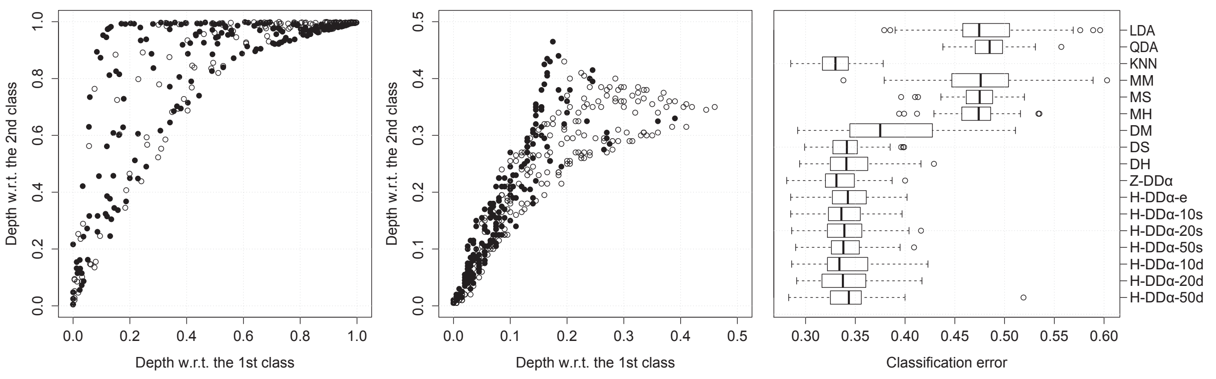

Figure 2.8: *DD-plots using zoonoid (left) and location (middle) depth and combined box-plots (right) for Cauchy location-scale alternative*

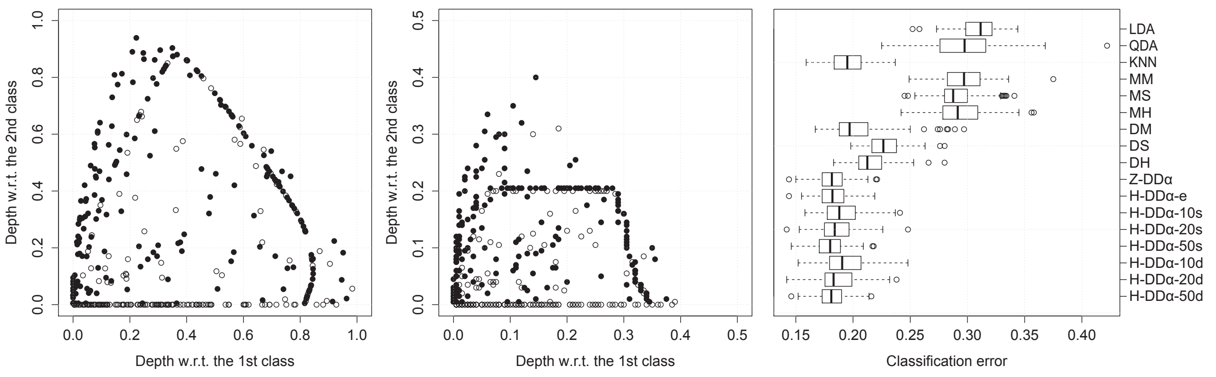

Figure 2.9: *DD-plots using zonoid (left) and location (middle) depth and combined box-plots (right) for BOMED location alternative*

their competitors. The versions of the $DD\alpha$-classifier that are based on the random Tukey depth are not outperformed by the exact computation algorithm. There is no noticeable difference between the versions of the $DD\alpha$-classifier based on the random Tukey depth using same directions and an instantly generated direction set. The statement "the more random directions we use, the better classification we achieve" is not necessarily true with the $DD\alpha$-classifier based on the random Tukey depth, as the portion of outsiders and their treatment are rather relevant.

Performance dynamics

To study the performance dynamics of the various $DD\alpha$-classifiers in contrast with existing classifiers we regard t-distributions with ∞, 5 and 1 degrees of freedom, each in a symmetric and an asymmetric version. The Mahalanobis distance between the two classes is systematically varied. At each distance the average error rate is calculated over 100 data sets and five shift directions in the range $[0, \pi/2]$. (As we consider two classes and have one reference vector two symmetry axes arise.) By this we obtain curves for the classification error of some of the classifiers considered, namely LDA, QDA, KNN, all $DD\alpha$-classifiers, and additionally those using five constant and instantly generated random directions. The results for two extreme

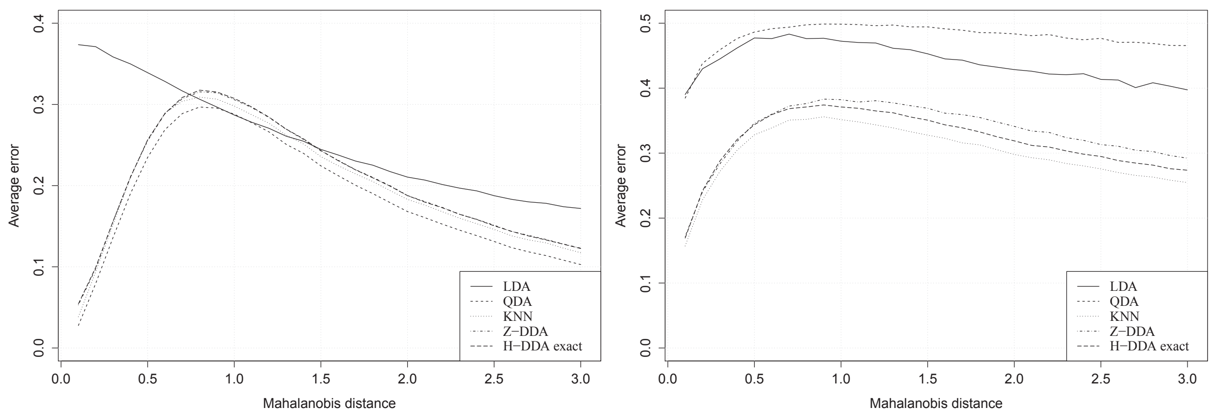

Figure 2.10: *Performance dynamic graphs for Gaussian (left) and asymmetric conditional scale Cauchy (right) distributions*

cases, Gaussian distribution (left) and asymmetric conditional scale Cauchy distribution (right) are shown in Figure 2.10.

Under bivariate elliptical settings (Figure 2.10, left) QDA, as expected from theory, outperforms other classifiers and coincides with LDA when the Mahalanobis distance equals 1. $DD\alpha$-classifiers suffering from outsiders perform worse but similarly, independent of the number of directions and the depth notion used; they are only slightly outperformed by KNN for the 'upper' range of Mahalanobis distances. (Note that KNN does not have the 'outsiders problem'.) But when considering an asymmetric fat-tailed distribution (Figure 2.10, right), neither LDA nor QDA perform satisfactorily. The $DD\alpha$-classifiers are still outperformed by KNN (presumably because of the outsiders). They perform almost the same for different numbers of directions. The $DD\alpha$-classifier based on zonoid depth is slightly outperformed by that using location depth, which is more robust.

The speed of training and classification

The third task tackled in this section is comparing the speed of the $DD\alpha$-classifiers using zonoid and random Tukey depth, respectively. (For the latter we take 1000 random directions and do not consider outsiders.) Two distributional settings are investigated: $\mathrm{N}(\mathbf{0}_d, \mathbf{I}_d)$ *vs.* $\mathrm{N}(0.25 \cdot \mathbf{1}_d, \mathbf{I}_d)$ and $\mathrm{N}(\mathbf{0}_d, \mathbf{I}_d)$ *vs.* $\mathrm{N}\big((0.25\ \mathbf{0}'_{d-1})', 5 \cdot \mathbf{I}_d\big)$, $d = 5, 10, 15, 20$. For each *pair of classes* and *number of training points* and *dimension* we train the classifier 100 times and test each of them using 2500 points. Average times in seconds are reported in Table 2.8.

Table 2.8 and the distributional settings correspond to those in Section 2.5.2, where a similar study has been conducted with the zonoid depth. We also use the same PC and testing environment. Note firstly that the $DD\alpha$-classifier with the random Tukey depth requires substantially less time to be trained than with the zonoid depth. The time required for training increases almost linearly with the cardinality of the training set, which can be traced back to the structure of the algorithms used for the random Tukey depth and for the α-procedure. The time decreases with dimension, which can be explained as follows: The α-procedure takes most of the time here; increasing d but leaving n constant increases the number of points outside

Table 2.8: *The average speed of training and classification (in parentheses) using the random Tukey depth, in seconds*

	$\mathrm{N}(\mathbf{0}_d, \mathbf{I}_d)$ *vs.* $\mathrm{N}(0.25 \cdot \mathbf{1}_d, \mathbf{I}_d)$			
	$d = 5$	$d = 10$	$d = 15$	$d = 20$
$n = 200$	0.1003 (0.00097)	0.097 (0.00073)	0.0908 (0.00033)	0.0809 (0.00038)
$n = 500$	0.2684 (0.00188)	0.2551 (0.00095)	0.2532 (0.00065)	0.252 (0.00059)
$n = 1000$	0.6255 (0.00583)	0.6014 (0.00289)	0.5929 (0.00197)	0.5846 (0.00148)
	$\mathrm{N}(\mathbf{0}_d, \mathbf{I}_d)$ *vs.* $\mathrm{N}\big((0.25\ \mathbf{0}'_{d-1})', 5 \cdot \mathbf{I}_d\big)$			
	$d = 5$	$d = 10$	$d = 15$	$d = 20$
$n = 200$	0.0953 (0.00098)	0.0691 (0.0005)	0.0702 (0.00034)	0.0699 (0.00025)
$n = 500$	0.2487 (0.0019)	0.2049 (0.00096)	0.1798 (0.00065)	0.1845 (0.00049)
$n = 1000$	0.5644 (0.0058)	0.5476 (0.00289)	0.4414 (0.00197)	0.4275 0.00148

the convex hull of one of the training classes, that is, having depth = 0 in this class; these points are assigned to the other class without calculations by the α-procedure.

2.7.3 Some remarks

The experimental comparison of the $DD\alpha$-classifiers, using the zonoid depth and the random Tukey depth, on asymmetric and fat-tailed distributions shows that in general both depths classify rather well, the random Tukey depth performs not worse than the zonoid depth and sometimes even outperforms it (cf. Cauchy distribution), at least in two dimensions. Though both depths can be efficiently computed, also for higher dimensional data, the random Tukey depth is computed much faster. Still when employing the random Tukey depth the number of random directions has to be selected; this as well as a proper treatment of outsiders needs further investigation.

2.8 Discussion and conclusions

A new classification procedure has been proposed that is completely nonparametric. The $DD\alpha$-classifier transforms the d-variate data to a q-variate depth plot and performs linear classification in an extended depth space. The depth transformation is done by the zonoid depth, and the final classification by the α-procedure. The procedure has attractive properties: First, it proves to be very fast and efficient in the training as well as in the testing phase; in this it highly outperforms existing alternative nonparametric classifiers, and also – regarding the training phase – the support vector machine. Second, in many settings of elliptically distributed alternatives, its AMR is of similar size than that of the competing classifiers. Moreover, it is rather robust to outlier prone data. As a nonparametric approach, the new procedure shows a particularly good behavior under asymmetrically distributed alternatives and, in certain cases, when the two classes originate from different families of distributions. Other than many competitors, it considers all classes in the multi-class classification problem even when performing binary classification. Different for KNN, SVM and other kernel based procedures our method does not need to be parametrically tuned. Also several theoretical properties of the $DD\alpha$-procedure have been derived: It operates in a rather simple way if the data generating processes are elliptical, and a Bayes rule holds if $q = 2$ and the two classes are mirror symmetric.

The zonoid depth has many theoretical and computational advantages: Most important here, it is efficiently computed also in higher dimensions. However, as it takes its maximum at the mean of the data, the zonoid depth lacks robustness. Nevertheless, the $DD\alpha$-classifier shows a rather robust behavior. Its relative robustness can be explained as follows: The original data points are mapped to a compact set, the q-dimensional unit hypercube, and then classified by the α-procedure. The latter, by choosing a median angle in each step, is rather insensitive to outliers.

Points that are not within the convex hull of at least one training set must be specially treated as their depth representation is zero. To classify those so called outsiders several approaches have been used and compared. Instead of assigning them randomly, which disadvantages the $DD\alpha$-procedure like other procedures based on halfspace or simplicial depth, one should classify outsiders by 1-NN and some distance or by a properly chosen maximum depth rule.

To contrast the $DD\alpha$-procedure with an SVM approach, a novel way of comparison has been taken: An optimal performance of an SVM has been evaluated, that arises under an optimal choice of the parameters, as well as an average performance, where the parameters vary over specified conservative intervals. It came out that, even with an arbitrary handling of outsiders, the $DD\alpha$-classifier mostly performs not much worse than an SVM whose parameters have been optimally chosen. However, if the SVM is employed with some non-optimized parameters, the AMR can be considerably larger than that of the $DD\alpha$-classifier.

More investigations are needed on the consistency of the $DD\alpha$-classifier, its behavior on skewed or fat-tailed data, the – possibly adaptive – choice of outsider treatments, and the use of alternative notions of data depth. These are intended for future research.

Chapter 3

Classifying real-world data with the $DD\alpha$-procedure

3.1 Introduction

Many statistical procedures have been developed to classify data into two or more given classes. Generally, if the data arise from a known class of distributions, properties of the classifiers are established through either theoretical considerations or simulation studies. By this, alternative classifiers are compared and procedures identified that are optimal under properly chosen assumptions. However, real-world data do often not fit into standard parametric distribution models. Classifying them requires nonparametric procedures, while, due to the lack of established general properties, selecting a good classifier has to be mainly based on empirical evidence. Usually such evidence is sought from simulation studies mimicking certain features of the data that arise in practical applications. At best, in a given field of application so called ‘stylized facts’ are identified and translated into a simulation setting. But often such ‘stylized facts’ do not exist. Then, we cannot learn much from simulation studies how a statistical procedure really works in practice. The adequacy and fitness of the procedure can only be demonstrated when it is applied to real-world data, and its *general* fitness can be only established by successful application to a *large variety* of such data.

In the sequel this is done for a newly developed nonparametric classifier, the $DD\alpha$-procedure (Lange *et al.*, 2014a). It is applied to fifty binary classification problems regarding real-world benchmark data.

The $DD\alpha$-procedure first transforms the data from their original property space into a *depth space*, which is a low-dimensional unit cube, and then separates them by a projective invariant procedure, called *α-procedure.* To each data point the transformation assigns its depth values with respect to the q given classes. The depth coordinates of the data reflect their degree of centrality w.r.t. each of the classes. This central ordering is carried out using a properly chosen depth function. The subsequent separation in the depth space accounts only for differences in the depth values: If $q = 2$ a binary separator is determined by the α-procedure. The α-procedure stepwise selects pairs of *extended depth properties* (that is, depth coordinates and powers and products of them) and separates them by a linear rule. The separator is a hyperplane in the extended depth space, which corresponds to a polynomial line in the basic depth plane, both containing the origin as an element. With $q > 2$ classes, $\binom{q}{2}$ such α-separations can be performed and a majority rule applied; alternatively q one-against-all separators can be used. We restrict the present study to the case $q = 2$, see Lange *et al.* (2014a) for $q > 2$.

In Lange *et al.* (2014a) the zonoid depth (Koshevoy & Mosler, 1997, Mosler, 2002) is applied, which is efficiently computed also in higher dimensions. Here we employ four alternative depths: the Mahalanobis depth, the spatial depth, the projection depth and the Tukey depth. The first three depths are positive everywhere, while the Tukey depth (like the zonoid depth) vanishes outside the convex hull of the data. However, the Tukey depth reflects the shape of the data much better than the previous three do and it is more robust against outliers than these (and than the zonoid depth as well). For computational reasons we use the *random Tukey depth* (Cuesta-Albertos & Nieto-Reyes, 2008), which approximates the Tukey (= location) depth by minimizing univariate Tukey depths over a finite number of directions.

When using the random Tukey depth (or another depth that vanishes outside the convex hull of the data) a first practical question is how outsiders, that is data points having zero depth in all classes, should be treated. These points are, by construction, represented by the origin of the depth space and, hence, arbitrarily assigned. With real data, often a large portion of the data turn out to be such outsiders. As our task is to classify all points, we need either a depth that does not produce outsiders, or a supplementary treatment of outsiders. By definition, the $DD\alpha$-procedure includes a treatment of outsiders if necessary. For the $DD\alpha$-classifier with the random Tukey depth, several possible treatments are introduced in the sequel. The paper considers the respective variants of the $DD\alpha$-classifier and compares them with the $DD\alpha$-classifiers based on Mahalanobis, spatial and projection depths. Recall that the latter depths, as they are positive on the entire $\mathbb{R}^d$, do not yield outsiders.

A second question is how many directions should be chosen to approximate the Tukey depth and how they should be generated; it is addressed in Lange *et al.* (2014b) by means of a simulation study. (For the random projection depth this question is less important as it has no outsiders.) Broad numerical experience is provided about the relative usefulness of the classifiers. Further we investigate how many features in the extended depth space are needed on an average to satisfactorily separate the data. Finally we demonstrate the robustness of our procedure when applied to real data containing substantial amounts of outliers.

For comparison several indicators are introduced, two of which refer to a combination of classical procedures as benchmark. To evaluate the performance of the $DD\alpha$-procedure under different depths and outsider treatments (when using the random Tukey depth) we have set up experiments with a large number of binary classification tasks with real data. These data sets have been selected from open internet sources and different fields of application. Some of them have already served as benchmark sets in other classification studies. By this they are well suited for evaluating and comparing our new approach. The data can be downloaded in standardized form from `www.wisostat.uni-koeln.de/28969.html`. The complete $DD\alpha$-procedure is available as an R-package named `ddalpha`.

The $DD\alpha$-classifiers are also compared with three traditional procedures: *linear (LDA)* and *quadratic (QDA) discriminant analysis* and the *k-nearest-neighbors (KNN)* classifier, as in many cases (including the data sets considered here) at least one of them performs satisfactorily. We exclude neural network methods because they offer too many possible architectures, among which it is difficult to select in an automatic and computationally feasible way. While we expect that, given the specific data set, a properly adapted neural network performs rather well, such an approach affords a by hand tuning for each data set. Therefore we do not regard neural networks as fair competitors to the $DD\alpha$-classifiers. We exclude as well the usual support vector machine (SVM) as a classifier because for each data set it has to be specially tuned.

Overview: Sect. 3.2 describes the training phase of the $DD\alpha$-classifier, which consists of the depth transformation and the α-separation in the extended depth space. The problem of generating directions for the random Tukey depth is discussed. Sect. 3.3 regards the classification phase, where the problem of outsiders arises. Several classical approaches to classify the outsiders (LDA, maximum Mahalanobis depth, KNN) are introduced, as well as a simplified SVM approach, which liaises with the $DD\alpha$-separation in two ways. Sect. 3.4 first describes the 50 classification tasks, which vary by absolute and relative sizes of training classes and include different portions of outliers and ties. Then the settings and results of the empirical study are presented. In Sect. 3.5 further evidence on outsider treatments and the number of (extended) depth properties needed is discussed. Sect. 3.6 concludes.

The material of the chapter is based on Mozharovskyi *et al.* (2014).

3.2 Constructing the $DD\alpha$-classifier

Consider a q-class classification problem, $q \geq 2$. The $DD\alpha$-classifier has been recently proposed by Lange *et al.* (2014a). Its classification phase consists of two parts: a transformation of the data from the original space into the *depth space* (depth transformation) and their subsequent separation using a modified version of the α-procedure (α-separation), see Lange & Mozharovskyi (2014), Vasil'ev (1991), Vasil'ev (2003), Vasil'ev & Lange (1998). This procedure is a projective invariant method to separate the depth transformed data.

3.2.1 Depth transformation

The *depth transformation* maps $\mathbf{z} \in \mathbb{R}^d$ into $[0,1]^q$, the depth space, where the coordinates of the transformed data reflect their degree of centrality w.r.t. each of the q classes, so that the subsequent class separation accounts only for the depth ordering. This central ordering is carried out using a properly chosen depth function $D(\cdot|\cdot)$. For more information on depth functions the reader is referred to the literature: e.g. Zuo & Serfling (2000) for properties and Mosler (2013) for a recent survey. Here we only briefly recall definitions of those used in the current work. The depth representation of the training sets in $[0,1]^q$ is called the *depth plot*.

First we briefly regard three depths whose empirical versions take positive values beyond the convex hull of the data. For a point $\mathbf{z} \in \mathbb{R}^d$ and a random vector X in $\mathbb{R}^d$ (especially one having an empirical distribution on a set of d-variate observations $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$) the *Mahalanobis depth* (Mahalanobis, 1936) of $\mathbf{z}$ w.r.t. X is defined as

$$D^{Mah}(\mathbf{z}|X) = \big(1 + (\mathbf{z} - \mu_X)'\Sigma_X^{-1}(\mathbf{z} - \mu_X)\big)^{-1}, \tag{3.1}$$

where μ_X measures the location (e.g. the mean) of X, and Σ_X the scatter (e.g. the covariance matrix) of X.

The *affine invariant spatial depth* (Vardi & Zhang, 2000, Serfling, 2002) of $\mathbf{z}$ regarding X is defined as

$$D^{Spt}(\mathbf{z}|X) = 1 - \|E_X\left[v(\Sigma_X^{-1/2}(\mathbf{z} - X))\right]\|\,. \tag{3.2}$$

Here $v(\mathbf{y}) = \|\mathbf{y}\|^{-1}\mathbf{y}$ for $\mathbf{y} \neq \mathbf{0}$ and $v(\mathbf{0}) = \mathbf{0}$, and Σ_X is the covariance matrix of X. As the Mahalanobis and spatial depths lack robustness when using standard moment estimates for Σ_X and μ_X, we consider also robustified versions of them, where Σ_X and μ_X are estimated by MCD (minimum covariance determinant).

The *projection depth* (Zuo & Serfling, 2000) of $\mathbf{z}$ regarding X is given by

$$D^{Prj}(\mathbf{z}|X) = \big(1 + O^{Prj}(\mathbf{z}|X)\big)^{-1}, \tag{3.3}$$

with

$$O^{Prj}(\mathbf{z}|X) = \sup_{\mathbf{u}\in S^{d-1}} \frac{|\mathbf{u}'\mathbf{z} - m(\mathbf{u}'X)|}{MAD(\mathbf{u}'X)}\,, \tag{3.4}$$

where $m(\mathbf{u}'X)$ denotes the (univariate) median of $m(\mathbf{u}'X)$ and $MAD(\mathbf{u}'X)$ the (univariate) median of the absolute deviation of $\mathbf{u}'X$ from its median.

The *Tukey depth* or *location depth* (Tukey, 1975, Zuo & Serfling, 2000) of $\mathbf{z}$ w.r.t. X is defined as the minimal probability of X lying in a halfspace bounded by a hyperplane through $\mathbf{z}$,

$$D^{loc}(\mathbf{z}|X) = \inf\{P(H) : \, H \text{is a closed halfspace containing}\, \mathbf{z}\}, \tag{3.5}$$

where P is the probability distribution of X. (A closed halfspace is the set of all points that lie on one side of a hyperplane including that hyperplane.) If P is an empirical distribution, this means that $D^{loc}(\mathbf{z}|X)$ is the minimal portion of the data that can be cut off by a hyperplane through $\mathbf{z}$. Obviously, in general, the Tukey depth vanishes outside the convex hull of the distributions' support.

The Mahalanobis, spatial, projection depths are everywhere positive; thus outsiders cannot occur. However they are not very sensitive to the shape of the underlying distribution, which is illustrated in Figure 3.1. It exhibits the data of the "non-donating" class in the "blood-transfusion" task. The Figure shows the level sets of Mahalanobis, spatial, projection and Tukey depths. Observe that the Mahalanobis depth yield ellipses, while with the spatial and projection depth rather symmetric, ellipsis-like level sets are obtained. The asymmetric shape

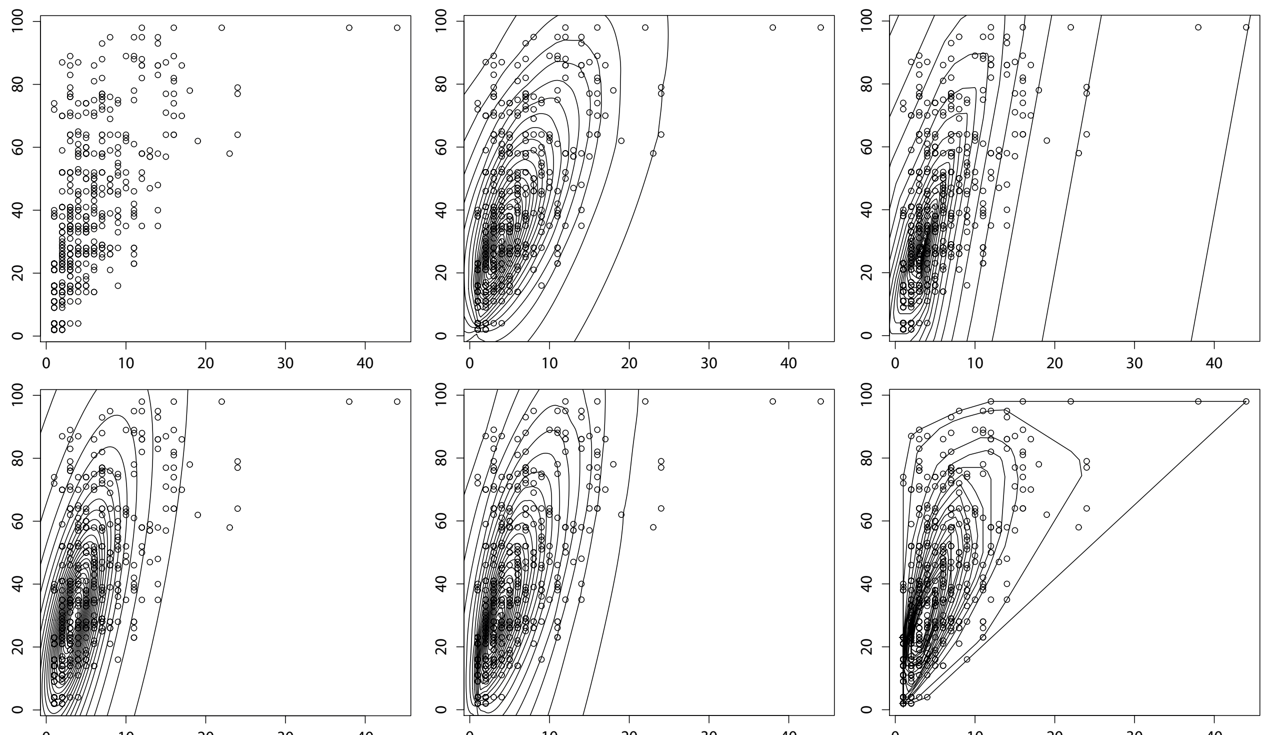

Figure 3.1: *Depth level sets for two dimensions (total number of donations and months since first donation) of the "blood-transfusion" data set, class of not donating in March 2007. Top, from left to right: data; spatial depth using moment estimate; projection depth. Bottom: Mahalanobis depth using MCD estimates with robustness parameter* 0.75*; spatial depth using the same MCD estimates; Tukey depth. The level sets are pictured for the depth values* $1/570, 1/114, 1/57, 2/57, ..., 19/57$ *for the Tukey depth, and for* $0.04, 0.08, ..., 1$ *for the rest of the depths.*

of the data is much more reflected by the Tukey depth (bottom right); therefore it appears to be better suited to extracting the relevant information from the training classes. Note that, by construction, the Mahalanobis depth has exact elliptical regions and the projection depth contains a symmetric factor, namely MAD, which accounts for the quasi-symmetric shape of the level sets. Moreover, the projection depth comes with an enormous computational cost. For these reasons we include the Tukey depth in our study, in spite of its need for extra outsider treatments.

D^{Mah} and D^{Spt} are easily computed. To estimate μ_X (for D^{Mah}) and Σ_X we use empirical moments and minimum covariance determinant (MCD) estimates that have outlyingness parameter 0.75, see Hubert & Van Driessen (2004), Rousseeuw & Van Driessen (1999). The Tukey depth as well as the projection depth satisfy the weak projection property Dyckerhoff (2004), i.e. the depth of a point can be represented as the minimum of the depths on all unidimensional projections. Based on this we approximate the Tukey depth by the random Tukey depth (Cuesta-Albertos & Nieto-Reyes, 2008), which is the minimum univariate Tukey depth over a set of unidimensional projections in randomly selected directions. As the exact calculation of D^{Prj} is rather elaborate (Liu & Zuo, 2014b), we approximate it in the same way.

3.2.2 α-separation

For each binary separation the *α-procedure* constructs a decision hyperplane in the extended property space $E_1 = [0,1]^r$, $r = \binom{p+q}{q} - 1$. The extended property space includes the powers and products of objects' attributes up to some degree p as additional coordinates. We mention the original depth coordinates as the *basic*, and the other coordinates as the *extended properties*. From all these properties, the relevant ones are stepwise chosen by the α-procedure. The final separation then is performed by an 'optimal' hyperplane in $[0,1]^r$.

Let X_1 and X_2 be two training classes in $\mathbb{R}^d$ having n_1 and n_2 elements respectively. Each data point $\mathbf{x} \in X_1 \cup X_2$ is transformed to $(d_1, d_2) \in [0,1]^2$, with $d_1 = D(\mathbf{x}|X_1), d_2 = D(\mathbf{x}|X_2)$. In the *first step* all pairs of (basic or extended) properties are selected that involve depths in both classes and the respective two-dimensional coordinate subspaces are considered. In each of these planes we separate the two classes by a straight line passing through the origin. The separating line is determined by the angle α (formed with one of the axes) that yields the minimal *empirical misclassification rate (EMR)*. Among all considered two-dimensional property spaces we choose the one delivering the smallest *EMR*; then we project its points onto a straight line f_1 that is orthogonal to the separating one. Now, separation in f_1 is performed by the origin, and f_1 becomes the first *projection axis* of a synthesized space.

We illustrate the procedure with data from the Pima Indians diabetes study; see `www.stats.ox.ac.uk/pub/PRNN/pima.tr2`. A subsample consisting of $q = 2$ classes (68 diseased and 132 not diseased females) has been selected, having seven attributes (number of pregnancies, 2 hours glucose concentration, blood pressure, triceps skin thickness, body mass index, diabetes pedigree function, age). The data points are represented in the unit square by their random Tukey depths regarding the two classes (using 10 000 random directions), and powers and products of depth values are considered up to degree $p = 2$; thus the extended depth space has dimension $r = 5$.

Figure 3.2 (left) shows the first step applied to the Pima data. Here, after mapping $X_1 \cup X_2$ into $[0,1]^2$, the depth space was extended up to degree $p = 2$, which yields the extended depth space $[0,1]^5$. With the Pima data the smallest *EMR* is achieved at the two basic depth properties d_1 and d_2.

After the first step the extended space is reduced by removing the two (possibly extended) properties that have obtained minimal *EMR*, and a similar second step is performed. In the *second step* we consider all planar subspaces based on f_1 and one of the properties of $[0,1]^{r-2}$ and find a separating straight line minimizing *EMR* in each of them. Again, among these planes we choose the one that yields the smallest *EMR* and obtain the second projection axis f_2. It separates the data at its zero point, as the first axis did in the initial step; see Figure 3.2 (right) for the Pima example. Similarly, after Step 2 the extended space is reduced to an $(r-3)$-dimensional unit hypercube. The steps are iterated, and properties are selected from the extended space, as long as the minimum *EMR* decreases and the remaining extended space is non-void. Note that with the Pima data, the procedure stops after Step 2.

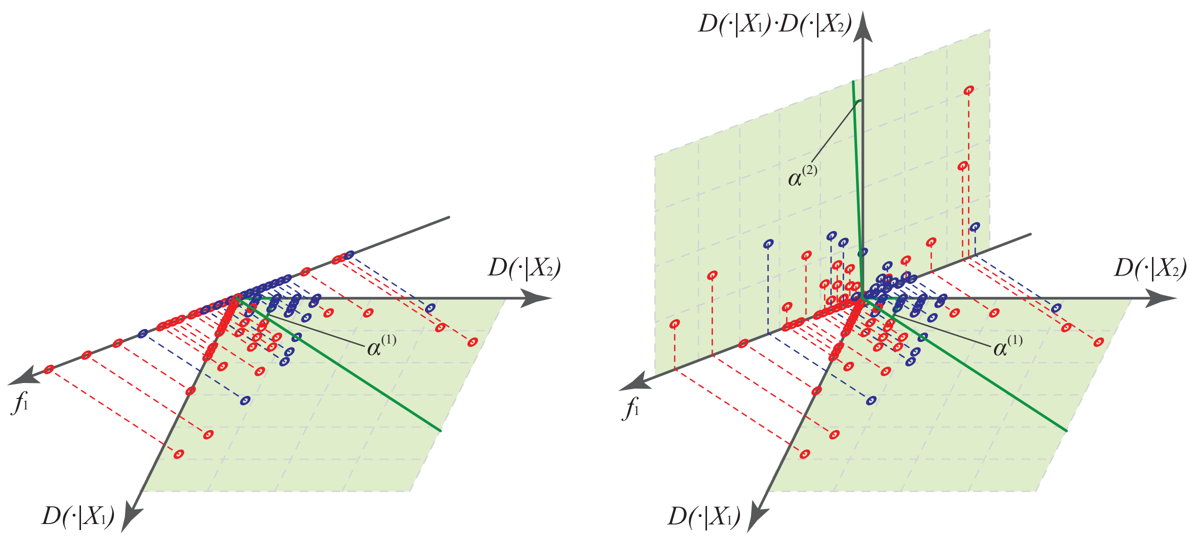

Figure 3.2: *α-separation; step 1 (left) and step 2 (right).*

3.2.3 Directions for the random Tukey depth

As mentioned above, we approximate the d-variate Tukey depth by the minimum univariate depth of the (on lines in several directions) projected data. The random Tukey depth inherits the robustness from the Tukey depth. When implementing the random Tukey depth we have to answer the following questions in a way that makes our classifier reasonable and computationally feasible.

(1) How should the random directions be generated?

(2) How many directions should we consider?

(3) Shall we generate a new set of directions for each point $\mathbf{z}$?

Ad (1): As in Cuesta-Albertos & Nieto-Reyes (2008) we generate directions that are uniformly and independently distributed on the unit sphere and independent of the data. Alternatively we could proceed as in Christmann *et al.* (2002), Christmann & Rousseeuw (2001), i.e. search through the normals of randomly formed $(d-1)$-simplices. However we find it more efficient to spend computational time on generating random directions and evaluating univariate depths rather than computing exact directions from the data; moreover, a moderate number of random directions proves to be enough.

Ad (2): We mention two qualitative arguments concerning this number. Firstly, a larger dimension of the space needs a larger number of directions, which is obvious from the geometry of $\mathbb{R}^d$. However, we are not able to indicate the precise dependency of this number on the dimension d. Secondly, we point out that there exists a trade-off between the number of directions used for the random Tukey depth and the number of outsiders. A simulation study that illuminates this trade-off has been conducted in Lange *et al.* (2014b). If the data stems from a centrally symmetric distribution (e.g. Gaussian or another elliptical distribution) the depth can be rather precisely approximated with a small number of directions. But if the data exhibit asymmetries and possible outliers, as real-world data mostly do, we need a larger number of directions to adequately represent them in the depth space. The degree of asymmetry and fat-tailedness found in the data may guide us in choosing this number; see Lange *et al.* (2014b). To be able to compare the procedure on different data sets we have chosen a fixed number of directions. This number has been set to 10 000 as a practical compromise between accuracy of the depth calculation and computational load.

It is obvious (see e.g. Liu & Zuo (2014a)) that the random Tukey depth of a point can widely deviate from the Tukey depth. On the other hand, the α-separation is rather robust, since the only invariant it uses is whether a point belongs to a class or not. Therefore an upward bias at a few points does not much influence the final separation. Further, a trade-off exists between the number of directions used for the random Tukey depth and the number of effective outsiders, which favors a moderate number of directions.

Ad (3): We use the same set of directions for each data point. Though the generation of a new set of directions for each point produces different depth values, there is no reason to expect them to be more precise. By keeping the set of directions constant we increase the speed of calculations in the training phase. The computations are boosted by avoiding the generation of the directions and projection of the points onto them. Let k be the number of random directions. When instantly generating the direction set for each point the complexity of the depth calculation amounts to $\mathrm{O}\big(kd(n_1+n_2)^2\big)$. If the direction set is not constantly changed, first, the time for their generation is saved. Second, recall that in the training phase the depth of all points of the training sample w.r.t. each class has to be computed. Hence, univariate projections are ordered, and all the univariate depths on each projection can be determined in a single pass. This yields complexity $\mathrm{O}\big(k(d+\log(n_1+n_2))(n_1+n_2)\big)$. Note that $d(n_1+n_2) > d+\log(n_1+n_2)$ holds. Therefore, for all d, n_1 and n_2 that are large enough to suppress eventual constants, the constant direction approach is substantially faster, see also Lange *et al.* (2014b). It also enhances the classifier's stability in the classification phase, as the same directions are used to approximate the depth of a new point to be classified.

3.2.4 Directions for the projection depth

Exact calculation of the projection depth appears to be a heavy computational task (Liu & Zuo, 2014b). Therefore we approximate the projection depth in the same way as the Tukey depth by minimizing the univariate depth of projections in randomly drawn directions. Compared with the Tukey depth, many more directions are needed to produce a reasonable approximation of the projection depth: When traversing the unit sphere the projection depth (which is a piecewise linear function) changes direction much more frequently because of the median and MAD estimates. To be able to compare the procedure on different data sets we have chosen two fixed numbers of directions. These have been set to 10 000 and 100 000 as a practical compromise between accuracy of the depth calculation and computational load.

3.3 Classifying outsiders

The *classification phase* proceeds as follows. Consider a point $\mathbf{z}$ to be classified. First the depth transform of $\mathbf{z}$ is calculated as $(d_1, d_2) = \big(D(\mathbf{z}|X_1), D(\mathbf{z}|X_2)\big)$. If d_1 is zero but not d_2, the point $\mathbf{z}$ is assigned to class X_2, and viceversa. If both d_1 and d_2 are non-zero the point is classified according to the separation rule determined by the α-procedure. This is always the case when using the Mahalanobis, spatial or projection depths. When employing the random Tukey depth some $\mathbf{z}$ may have $(d_1, d_2) = (0, 0)$, then $\mathbf{z}$ is regarded as an *outsider.* An outsider, being represented by the origin of the depth plot, cannot be readily classified but needs some special treatment. Specifically, a point $\mathbf{z}$ that, in the original data space, lies outside the convex hulls of the two training sets has zero Tukey depth in both classes and, thus, is an outsider. If a point $\mathbf{z}$ is no outsider it is mentioned as an *insider.* Insiders are instantaneously classified by the α-procedure.

As in Lange *et al.* (2014a), outsiders may be classified by determining their nearest neighbors. In doing so, Euclidean and Mahalanobis distances can be employed, the latter to account for scatter within the classes. Alternatively outsiders can be classified according to their maximum Mahalanobis depth, which is always positive. Mosler & Hoberg (2006) introduce a depth function, which is the maximum of the zonoid depth and a properly scaled Mahalanobis depth, and thus circumvent the outsider problem. Paindaveine & Van Bever (2012) propose an approach that avoids the outsider problem as well. In the classification phase, for each point $\mathbf{z}$ to be classified, (1) the sample is extended by reflecting the training classes symmetrically at center $\mathbf{z}$, (2) the depth of points in the extended sample is considered, and (3) a k-nearest-neighbor rule that uses depth in place of distance is applied for classifying $\mathbf{z}$. Here not only the classification phase is computationally hard (by instantaneous calculation of the depths of all data points), but also the training phase, where the classifier has to be validated in order to determine k. This requires onerous computations.

In the sequel we compare several alternative outsider treatments, which are classifiers applied to data in the original space. The treatments include three well known classifiers: linear discriminant analysis, maximum Mahalanobis depth, and k-nearest neighbors as well as a new one, which we call *SVM-simplified.* The performance of the $DD\alpha$-classifier with Mahalanobis, spatial and projection depth is contrasted as well. Note that all depths (Tukey, Mahalanobis, spatial and projection) are affine invariant as well as all treatments used (LDA, KNN with an affine invariant distance, Mahalanobis depth and the support vector machine). Therefore all considered $DD\alpha$-classifiers are affine invariant (under appropriate moment assumptions), if the exact versions of the depths are calculated. Since the random Tukey depth and the random projection depth converge to the exact versions, using them makes the $DD\alpha$-classifiers approximately affine invariant.

The random Tukey depth (RTD) used for the depth transform is very efficiently calculated, but yields outsiders. As it approximates the Tukey depth (TD) from above, some TD-outsiders will have non-zero RTD and, by this, be assigned to one of the classes. A smaller number of directions yields a worse approximation of the TD, but reduces the number of outsiders. The remaining RTD-outsiders still need a special treatment, though. Thus, when using the RTD, we face a trade-off between the quality of depth approximation and the extent of outsider treatment needed. As we will see below with real data, choosing a moderate number of random directions gives best results.

3.3.1 Classical approaches as treatments

Linear discriminant analysis (LDA), introduced in Fisher (1936), separates the classes by a hyperplane in the original data space; see also Hastie *et al.* (2009). The LDA classifier is particularly simple. It is optimal if the data follow a Gaussian or, more general, a unimodal elliptical distribution and the classes differ by location shifts only. However, in classifying real data it is often outperformed by other approaches. In many applications, the real data cannot be assumed to be Gaussian and ask for procedures different from LDA. However, after having classified the RTD-insiders, the remaining task of classifying the outsiders appears to be a much less exigent task and may be successfully done by a simple procedure like LDA.

The *maximum-Mahalanobis-depth* classifier is given by

$$\text{class} = \underset{i}{\operatorname{argmax}}\, \pi_i D^{Mah}(\mathbf{z}|X_i)\,, \tag{3.6}$$

where π_i is the prior probability for class i. The priors are estimated by the training class portions. Again, this classifier has optimality properties under ellipticity. Applied to outsiders it is expected to perform satisfactorily.

The *k-nearest-neighbors* classifier is still another option for treating outsiders. Its parameter k, the number of the nearest neighbors, has to be chosen by cross-validation. Often a relatively small k is enough; see, e.g., Lange *et al.* (2014a), where already $k = 1$ produces satisfying results. To make the procedure affine invariant we use Mahalanobis distances (based on the pooled data set) for finding nearest neighbors.

3.3.2 SVM-simplified as an outsider treatment

As another way to handle the outsider problem we propose to supplement the $DD\alpha$-classifier by an additional SVM-rule, which is restricted to classifying the outsiders. It has a particularly simple structure. Recall that the $DD\alpha$-procedure delivers a separator which is a hyperplane in the extended depth space. This hyperplane induces a decision rule in the original data space. Next, we remove all training points which are not correctly classified by this rule (so that $EMR = 0$) and subject the remaining points to an additional SVM classification step that involves determining a single kernel parameter but no box-constraint. This new approach is named *SVM-simplified* (*SVM-s*). As the *SVM-s* rule is defined on the whole $\mathbb{R}^d$, it is able to assign points which are outsiders in the $DD\alpha$-classification.

Figure 3.3, left panel, shows two classes, each containing 250 points, which are simulated from $\mathrm{N}\big(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right]\big)$ and $\mathrm{N}\big(\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}4&4\\4&16\end{smallmatrix}\right]\big)$, together with the separating lines of the optimal Bayes

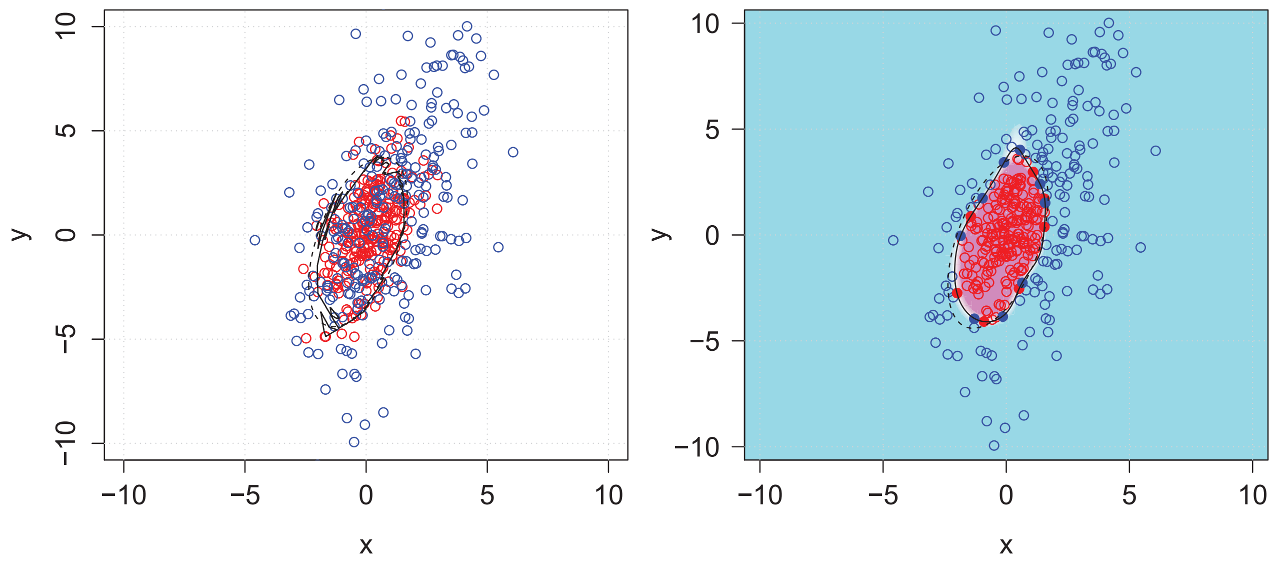

Figure 3.3: *Decision rules of different classifiers: optimal Bayes (dashed lines), DDα (left, solid line), and SVM-simplified (right, solid line) classifiers.*

(dashed) and $DD\alpha$ (solid) classifiers. The left panel regards the original data, while the right panel exhibits the data after the removal step.

The *SVM-s* step consists in solving the following quadratic programming problem (Cortes & Vapnik, 1995):

$$\underset{\lambda}{\text{maximize}} \quad W(\lambda) = \lambda'\mathbf{1} - \frac{1}{2}\lambda'\mathbf{D}\lambda \tag{3.7}$$

subject to the constraints

$$\lambda \geq \mathbf{0}, \tag{3.8}$$

$$\lambda'\mathbf{y} = 0. \tag{3.9}$$

Here we notate $l = n_1 + n_2$, $\lambda = (\lambda_1, ..., \lambda_l)'$, $\mathbf{1} = (1, \dots, 1)'$ and $\mathbf{0} = (0, \dots, 0)' \in \mathbb{R}^l$. $\mathbf{y}$ stands for the l-dimensional vector of responses $y_1, ..., y_l \in \{-1, 1\}$, and $\mathbf{D}$ is a symmetric $l \times l$-matrix with elements

$$D_{ij} = y_i y_j K_\gamma(\mathbf{x}_i, \mathbf{x}_j), \;\; i, j = 1, ..., l, \tag{3.10}$$

where $K_\gamma(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ is a Gaussian kernel. Note that no box-constraint condition is needed here as the points are separable without error. But still the kernel parameter γ has to be chosen. For given γ a solution $\lambda^0 = (\lambda_1^0, ..., \lambda_l^0)$ of (3.7) is obtained, provided the two classes are linearly separable in the reproducing kernel Hilbert space that corresponds to $K_\gamma(\cdot,\cdot)$. Every such solution λ^0 determines a margin between the classes $\rho_0 = \sqrt{\frac{2}{W(\lambda^0)}}$ and a number of support vectors, $\sharp\{\lambda_i^0 | \lambda_i^0 > 0, i = 1, ..., l\}$.

In Figure 3.4, depending on γ, the values of ρ_0 (dashed line) and the corresponding numbers of support vectors (solid line) are plotted for the above example. A zero value of the margin ρ_0 or of the number of support vectors indicates that with the given γ no errorless discrimination is possible.

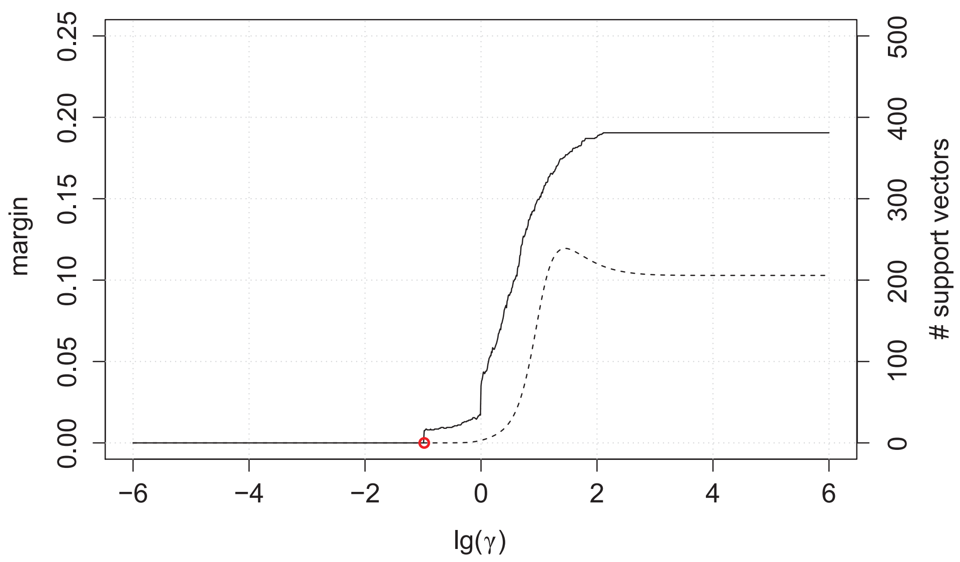

Figure 3.4: *The number of support vectors (solid line, right scale) and the margin between the classes (dashed line, left scale) for different values of the parameter lg(γ) (logarithm to the base 10).*

Loosely speaking, γ controls the complexity of the *SVM-s* rule; for small values of γ the decision rule is not able to separate the classes at all. Therefore it suggests itself to use the simplest separating rule, i.e. selecting the smallest γ for which the classes are separated without error (as indicated by the small circle in Figure 3.4). This also comes out as a most stable decision rule. Figure 3.3 exhibits the corresponding decision rule in its right panel as a solid line, while the dashed line indicates the same optimal Bayes decision rule as on the left panel of the Figure; the rules appear to be very similar. Most important, calculating the *SVM-s* rule needs no parameter tuning besides selecting the parameter γ, which is a straightforward task.

Needless to say, with the usual support vector machine (SVM) a solution can be obtained that is at least as good as the one achieved here. However, obtaining this solution needs a tuning of parameters that is computationally much more intensive.

To summarize the above procedure: The training phase consists of two steps, first determining the $DD\alpha$-classifier and then determining an *SVM-s* rule based on the correctly $DD\alpha$-classified points. Note that the classification performance of this procedure is determined by the $DD\alpha$-classifier, and the *SVM-s* step just extrapolates this classifier to treat the outsiders. In our experiments the whole training phase took between a few seconds and several minutes of computation time (64-bit, 1 kernel of the iCore 7-2600 having enough operative memory). The time reached a maximum of 10 minutes with four very large data sets only.

In the classification phase we have two choices: Either using the obtained *SVM-s* rule for all points $\mathbf{z}$ to be classified, or first check for each $\mathbf{z}$ whether it is an insider or an outsider and then classify it with the $DD\alpha$-rule if it is an insider and with the *SVM-s* rule otherwise. The first choice yields a particularly fast procedure, as the *SVM-s* rule does not involve any depth calculations in the classification phase. We choose the second one as it is in line with the application of the outsider treatments mentioned above. Its results are presented in column 'SVM-s' of Tables 3.3 and 3.4.

3.4 Real data experiments

To evaluate the $DD\alpha$-procedures with different outsider treatments and to judge their usefulness in practical applications we have set up experiments based on a large variety of real data sets. The methodology is applied to 50 binary classification tasks, which have been obtained from partitioning 33 freely accessible data sets, see Tables 3.1 and 3.2. The subset of the "Pima Indian Diabetes" described above is included in the Table as "pima" (No. 35).

The authors and introducers of the accessed data sets are Cox *et al.* (1982) ("biomed"), Miller *et al.* (1979) ("cloud"), Greaney & Kellaghan (1984) ("irish-ed"), McGilchrist & Aisbett (1991) ("kidney"), Nierenberg *et al.* (1989) ("plasma-retinol"), Biblarz & Raftery (1993) ("socmob") and Kalbfleisch & Prentice (1980) ("veteran-lung-cancer"); these data sets have been downloaded from `lib.stat.cmu.edu/datasets`.Data sets "chemdiab" (Reaven & Miller, 1979) and "hemophilia" (Habemma *et al.*, 1974) have been taken from the `R`-packages 'locfit' and 'rrcov' respectively. The "pima" data set constitutes a training subsample of the "diabetes" (see below) and can be downloaded from `www.stats.ox.ac.uk/pub/PRNN` (Ripley, 1996).Datasets "baby", "banknoten" (Flury & Riedwyl, 1988), "crab" (Ripley, 1996), "gemsen" , "groessen" (Galton, 1886), "tennis", "tips" and "uscrime" (Hand *et al.*, 1994) have been downloaded from the teaching data base `stat.ethz.ch/Teaching/Datasets`.The rest of the data sets is taken from `archive.ics.uci.edu/ml` (Asuncion & Newman, 2007);it in particular originates from Yeh *et al.* (2009) ("blood-transfusion"), Wolberg & Mangasarian (1990) ("breast-cancer-wisconsin") and Turney (1993) ("vowel").

Multiclass problems were reasonably split into binary classification problems, and some of the data sets were slightly processed by removing correlated attributes, by dropping objects with missing values, and by selecting prevailing classes. For detailed descriptions of the data considered we refer to the corresponding literature and public repositories; the fifty tasks together with short descriptions of the data can be found on the web page `www.wisostat.uni-koeln.de/28969.html`.

3.4.1 Data

As we see from Tables 3.1 and 3.2, the classification tasks are much different. The Tables show their basic parameters: dimension d of the original space, log ratio of the cardinalities n_1 and n_2 of the training classes (so that the sign reflects which is larger), total sample length $n = n_1 + n_2$, percentages of outliers and outsiders. As we see, up to 13 attribute dimensions are considered. The total sample sizes range from 47 to 1349, while the relative size of the two classes varies between 1 and 6.5.

Almost all data contain *outliers*; see column '% outl.'. In particular "diabetes", "glass", and "segmentation" contain substantial portions of them. The outliers of each data set have been

Table 3.1: *Data set parameters; Part 1.*

No.	Data set	$n_1 + n_2$	$\ln(n_1/n_2)$	d	$(n_1 + n_2)/d$	# tied	% outl.	% outs.
1	baby	247	0.63	5	49.4	0	5.3	31.6
2	banknoten	200	0.00	6	33.3	0	4.5	68.0
3	biomed	194	-0.63	4	48.5	0	4.6	27.8
4	blood-transfusion	748	-1.17	3	249.3	246	5.1	2.9
5	breast-cancer-wisconsin	699	0.64	9	77.7	236	8.3	46.9
6	bupa	345	-0.33	6	57.5	4	8.4	44.3
7	chemdiab_1vs2	112	-0.76	5	22.4	0	2.7	67.9
8	chemdiab_1vs3	69	0.09	5	13.8	0	2.9	81.2
9	chemdiab_2vs3	109	0.83	5	21.8	0	2.8	67.9
10	cloud	108	0.00	7	15.4	0	6.5	91.7
11	crab_BvsO	200	0.00	5	40.0	0	2.0	57.5
12	crab_MvsF	200	0.00	5	40.0	0	1.5	59.0
13	crabB_MvsF	100	0.00	5	20.0	0	4.0	76.0
14	crabF_BvsO	100	0.00	5	20.0	0	4.0	71.0
15	crabM_BvsO	100	0.00	5	20.0	0	1.0	74.0
16	crabO_MvsF	100	0.00	5	20.0	0	1.0	69.0
17	cricket_CvsP	156	0.00	4	39.0	7	1.3	26.9
18	diabetes	768	-0.62	8	96.0	0	8.9	56.8
19	ecoli_cpvsim	220	0.62	5	44.0	0	5.9	42.3
20	ecoli_cpvspp	195	1.01	5	39.0	0	5.1	43.1
21	ecoli_imvspp	129	0.39	5	25.8	0	8.5	61.2
22	gemsen_MvsF	1349	0.36	6	224.8	27	1.9	32.3
23	glass	146	-0.08	9	16.2	1	11.6	91.1
24	groessen_MvsF	230	0.02	3	76.7	0	2.6	20.4
25	haberman	306	1.02	3	102.0	23	2.9	7.5

Table 3.2: *Data set parameters; Part 2.*

No.	Data set	$n_1 + n_2$	$\ln(n_1/n_2)$	d	$(n_1 + n_2)/d$	# tied	% outl.	% outs.
26	heart	270	-0.22	13	20.8	0	4.4	100.0
27	hemophilia	75	-0.40	2	37.5	0	1.3	13.3
28	indian-liver-patient_1vs2	579	0.92	10	57.9	13	7.8	68.4
29	indian-liver-patient_MvsF	579	-1.14	9	64.3	13	7.9	54.7
30	iris_setosavsversicolor	100	0.00	4	25.0	2	4.0	48.0
31	iris_setosavsvirginica	100	0.00	4	25.0	3	4.0	48.0
32	iris_versicolorvsvirginica	100	0.00	4	25.0	1	2.0	51.0
33	irish-ed_MvsF	500	0.00	5	100.0	44	6.2	14.2
34	kidney	76	-1.02	5	15.2	0	2.6	73.7
35	pima	200	0.66	7	28.6	0	5.0	86.0
36	plasma-retinol_MvsF	315	1.87	13	24.2	0	8.3	98.1
37	segmentation	660	0.00	10	66.0	62	9.4	57.7
38	socmob_IvsNI	1156	0.00	5	231.2	45	4.2	14.1
39	socmob_WvsB	1156	0.00	5	231.2	8	3.0	15.1
40	tae	151	-1.43	5	30.2	43	1.3	26.5
41	tennis_MvsF	87	-0.07	15	5.8	0	6.9	100.0
42	tips_DvsN	244	0.95	6	40.7	1	5.3	48.8
43	tips_MvsF	244	-0.60	6	40.7	1	5.7	37.3
44	uscrime_SvsN	47	-0.65	13	3.6	0	0.0	100.0
45	vertebral-column	310	0.74	6	51.7	0	4.8	54.5
46	veteran-lung-cancer	137	0.01	7	19.6	0	8.8	80.3
47	vowel_MvsF	990	0.13	13	76.2	0	2.1	99.7
48	wine_1vs2	130	-0.19	13	10.0	0	3.8	100.0
49	wine_1vs3	107	0.21	13	8.2	0	0.9	100.0
50	wine_2vs3	119	0.39	13	9.2	0	3.4	100.0

identified by cutting moment Mahalanobis regions at a $\chi^2_d(0.975)$-quantile as, e.g., in Rousseeuw & Van Driessen (1999). We take a pure data-analytic view and thus treat a potential outlier in the same way as any other point. Observe that regarding eventual outliers the $DD\alpha$-procedure is highly robust for two reasons: Firstly, the classification is done by the α-procedure – a very robust approach – in a low-dimensional compact set, the unit cube of $\mathbb{R}^q$. Secondly, a robust depth like the Tukey depth can be employed.

Nevertheless the Tukey-DD-classifier suffers from the existence of *outsiders* as the Tukey depth vanishes outside the convex hulls of the training classes. The performance of this procedure obviously depends on the portion of outsiders in the data. We measure the *outsider proneness* of a training set by the portion of points lying outside the convex hulls of all classes. I.e. for each point we check (leaving it out) whether it lies inside the convex hull of at least one class of the remaining training sample. As shown in the two Tables (last column) the portion of outsiders varies from 0.029 to 1; see Sect. 3.5.1 for discussion.

An important parameter of a data set is the ratio of the sample size over the dimension, $(n_1 + n_2)/d$. It relates to the ability of the trained procedure to classify new data. The ratio varies from 3.6 to 249.3.

Finally, real data can contain ties, which require additional consideration by learning algorithms like KNN and depth based classifiers, and thus increase computation time. The number of tied points (in the pooled classes) is shown in column '# tied' of both Tables. It is determined as the smallest number of points that has to be removed from the training sample to make the remaining ones pairwise distinct.

3.4.2 Study settings

In constructing the depth transform of the $DD\alpha$-classifier, the Mahalanobis depth (based on moment estimates as well as robust MCD estimates with outlyingness parameter 0.75) is computed exactly, while the projection depth is approximated using 10 000 and 100 000 random directions. Instead of the exact Tukey depth we calculate the random Tukey depth (RTD) with 10 000 randomly chosen directions which are the same for all points of a given data set. Since the RTD approximates the Tukey depth from above and points having depth zero in both classes are treated as outsiders, the number of outsiders is systematically underestimated when using the RTD. These outsiders are treated with the techniques described in Sect. 3.3.1 and Sect. 3.3.2.

The α-separation is performed in a polynomially extended depth space, where the degree of the polynomial is chosen by 50-fold cross-validation. The complexity of the separation rule is, in a natural way, characterized by the dimension of the space needed. See Sect. 3.5.2 for results on the expected number of such features.

In classifying outsiders by KNN the number k is selected by the same cross-validation strategy as with the traditional KNN classifier. We determine the number k of neighbors in the KNN classifier by leave-one-out cross-validation, performed over a wide range of neighborhoods, but – to save computation time – not over the whole sample size; the performance is still highly satisfactory.

The max-Mahalanobis-depth classifier is calculated either with moment or MCD estimates, setting $\alpha = 0.75$. As a basis for the SVM-simplified classifier Joachims's C++ implementation of SVM^{light} (Joachims, 1999) is used with slight modifications and interfaced to the R-environment.

We use an R-implementation for the traditional KNN with ties broken at random; similarly when treating the outsiders by the affine invariant KNN. In SVM-simplified ties are neglected. The α-procedure is tie-immune as well, but in contrast to SVM, it accounts for the number of tied points.

3.4.3 Empirical comparison

We solve the above fifty classification problems by the following fourteen approaches: three classical approaches (LDA, QDA and KNN), the $DD\alpha$-classifier with the random Tukey depth and five outsider treatments from Sect. 3.3 (LDA, KNN, maximum Mahalanobis depth classifier with both moment and robust estimates and SVM-s), the $DD\alpha$-classifier based on Mahalanobis, spatial (both using moment estimates and robust MCD-estimates with outlyingness parameter equal to 0.75), and projection depth (using 10 000 and 100 000 random directions). The performance of each classifier is evaluated by leave-one-out cross-validation; we refer to this as the *average error rate* (AER). Tables 3.3 and 3.4 exhibit their average error rates for each of the fifty settings, and for eleven classifiers. (For reasons of space, the $DD\alpha$-classifiers based on Mahalanobis and spatial depth with MCD-estimates and the one based on projection depth with 10 000 random directions are left out; short cumulative results are given in Table 3.5 and Figure 3.5.)

The results are mixed. Some are surprising, e.g. the classification tasks "bupa", "glass", "indian-liver-patient_1vs2/FvsM" show error rates for QDA that are substantially higher than those for LDA. We attribute this to the poor estimation of the covariance matrix of the smaller class.

In Tables 3.3 and 3.4 classification error of the best classifier for each task is printed in bold. In almost all tasks none of the considered procedures dominates the others. Exceptions are "blood-transfusion", "indian-liver-patient_FvsM" and "irish-ed_MvsF", where the $DD\alpha$-classifier with the robust Mahalanobis depth (not shown in the Tables) dominates; also "haberman", where the $DD\alpha$-classifier with the projection depth using on 10 000 directions dominates, and "cricket_CvsP",where they both are prevailing.

While at first sight the error rates of the diverse $DD\alpha$-classifiers show comparable sizes among each other and with the classical approaches (LDA, QDA, and KNN), it seems worthwhile to have a closer look at the performance of the different depths and treatments of outsiders.

Five aggregating measures are used to compare the overall performance of the considered classifiers. The absolute performance is measured by the *average classification error* (ACE), which is the average error of a classifier over all the classification tasks. The relative measure is the *average relative classification edge* (ARCE) calculated as the average of $\big((1-\text{AER})-(1-\text{AER}_{\text{trd}})\big)/(1-\text{AER}_{\text{trd}})$ over all classification tasks, with AER_{trd} being the AER of the best of the three traditional classifiers (LDA, QDA and KNN) for each data set. We mention it as ARCE_{trd}. Its negative values relate to the best of the traditional classifiers, i.e. to an absolute reference. We also use ARCE with a relative reference (AER_{best}), which is the smallest AER among all considered classifiers for a given task (the bold values in Tables 3.3 and 3.4); it depends on the variety of the classifiers chosen. The corresponding measure is mentioned as $\text{ARCE}_{\text{best}}$, which is always non-positive. Two indicator measures denoted '$\#\geq$ trd' and '$\#\geq$ best' count how often AER of a classifier is not worse than AER_{trd}, respectively AER_{best}.

The five measures are given in Table 3.5, the best classifier w.r.t. each of the measures is printed in bold. Note, that all proposed classifiers have negative ARCE_{trd}, i.e. none of them can outperform (on an average) the best of the traditional triplet. On the other hand, the traditional classifiers perform mostly not satisfactory as well, although LDA shows favorable indicator values and competitive ACE.

To visualize the empirical evidence, the measures have been standardized to values in $\in[0,1]$, with larger numbers indicating better performance of the classifier, see Figure 3.5. Three groups of the classifiers are easily distinguishable. The first group consists mainly of the $DD\alpha$-classifier based on spatial depth followed by the one with Mahalanobis depth, both calculated using moment estimates. These two perform best. They also perform close to the best of the traditional classifiers (in terms of ARCE_{trd}, see Table 3.5), and not worse than this in (approximately) half of the cases (in terms of '$\#\geq$ trd', see Table 3.5). Results of the second group are mixed, only the $DD\alpha$-classifier based on the random Tukey depth supplemented with the LDA-treatment lies in parts on the positive border.

The $DD\alpha$-classifier based on random Tukey depth as transformation and moment-based Mahalanobis depth as outsider treatment performs worst. Similarly, that based on projection depth with 10 000 random directions is mostly outperformed; this can be explained by insufficient approximation. (Note that with 100 000 random directions its performance increases).

3.5 Some evidence from the empirical study

The empirical study sheds light on the nature and possible treatment of potential outsiders. It also provides practical evidence on the number of *features* (= extended depth properties) that is normally needed in the linear separating procedure.

3.5.1 Outsiders

It is seen from Tables 3.1 and 3.2 see that 10 (resp. 20%) data sets contain more than 90% outsiders, which actually means that less than 10% of the points are classified by the $DD\alpha$-procedure, when the Tukey depth is employed. It is clear that this variant of the $DD\alpha$-procedure may be not the best solution, as it constructs a separation rule only from a small

Table 3.3: *Comparison of classification performance (average error rate); Part 1.*

					$DD\alpha$ + random Tukey depth + treatment					$DD\alpha$ +		
							Mah.depth			Mah.	Spt.	Prj.
No.	Dataset	LDA	QDA	KNN	LDA	KNN	Moment	$\text{MCD}_{0.75}$	SVM-s	depth	depth	depth
1	baby	21.86	22.27	21.86	**21.46**	25.10	25.51	23.48	**21.46**	23.89	25.51	24.29
2	banknoten	**0.50**	**0.50**	**0.50**	**0.50**	1.00	4.00	**0.50**	2.00	1.00	1.00	1.50
3	biomed	15.98	12.37	11.34	11.34	11.34	13.92	**10.82**	11.86	12.37	13.40	13.40
4	blood-transfusion	22.86	22.19	22.59	22.33	22.19	21.93	22.86	23.93	20.86	22.06	20.72
5	breast-cancer-wisconsin	4.86	5.01	4.29	4.72	6.87	11.44	6.72	6.72	3.43	**3.00**	4.43
6	bupa	30.72	40.58	31.30	26.96	28.70	30.43	31.88	**26.09**	30.72	29.57	32.17
7	chemdiab_1vs2	3.57	7.14	7.14	3.57	5.36	14.29	5.36	**0.89**	3.57	3.57	8.93
8	chemdiab_1vs3	10.14	8.70	7.25	8.70	8.70	17.39	13.04	**2.90**	10.14	7.25	10.14
9	chemdiab_2vs3	3.67	0.92	0.92	3.67	6.42	1.83	0.92	**0.00**	1.83	1.83	1.83
10	cloud	53.70	50.93	66.67	54.63	64.81	**40.74**	48.15	59.26	51.85	49.07	49.07
11	crab_BvsO	**0.00**	**0.00**	3.00	**0.00**	1.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
12	crab_MvsF	4.00	5.00	9.00	4.50	7.00	5.50	5.50	**3.50**	4.50	**3.50**	6.00
13	crabB_MvsF	9.00	10.00	15.00	8.00	9.00	10.00	8.00	7.00	**6.00**	**6.00**	9.00
14	crabF_BvsO	**0.00**	1.00	5.00	**0.00**	5.00	2.00	1.00	**0.00**	1.00	1.00	2.00
15	crabM_BvsO	**0.00**	**0.00**	5.00	**0.00**	1.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
16	crabO_MvsF	3.00	**2.00**	7.00	3.00	6.00	3.00	3.00	3.00	**2.00**	**2.00**	**2.00**
17	cricket_CvsP	68.59	64.10	59.62	64.74	67.31	64.10	62.82	57.05	61.54	57.69	58.33
18	diabetes	**22.53**	26.04	24.61	26.04	25.52	29.04	27.21	27.34	23.57	24.22	34.24
19	ecoli_cpvsim	**1.36**	1.82	2.27	1.82	2.73	5.91	1.82	4.55	**1.36**	**1.36**	2.73
20	ecoli_cpvspp	**3.08**	4.10	4.10	4.62	4.62	5.64	6.15	9.23	4.62	5.13	5.13
21	ecoli_imvspp	5.43	3.88	5.43	5.43	4.65	9.30	6.20	5.43	**2.33**	3.88	5.43
22	gemsen_MvsF	19.13	14.16	14.01	16.46	15.86	16.90	16.53	18.53	14.97	**13.94**	23.28
23	glass	27.40	39.73	**19.18**	29.45	27.40	34.93	30.82	28.77	30.14	28.08	39.73
24	groessen_MvsF	10.87	10.43	14.35	12.61	13.48	13.48	13.48	13.04	12.61	**7.83**	**7.83**
25	haberman	25.16	24.51	25.82	28.43	27.12	28.76	26.80	28.43	26.14	25.16	23.53

Table 3.4: *Comparison of classification performance (average error rate); Part 2.*

					$DD\alpha$ + random Tukey depth + treatment					$DD\alpha$ +		
							Mah.depth			Mah.	Spt.	Prj.
No.	Dataset	LDA	QDA	KNN	LDA	KNN	Moment	$MCD_{0.75}$	SVM-s	depth	depth	depth
26	heart	**16.30**	16.67	33.70	22.59	21.85	22.96	22.96	24.44	20.37	18.15	27.78
27	hemophilia	**14.67**	16.00	16.00	16.00	17.33	18.67	18.67	18.67	17.33	16.00	21.33
28	indian-liver-patient_1vs2	29.88	44.56	31.26	30.92	**28.15**	30.57	31.61	36.96	29.88	28.50	29.19
29	indian-liver-patient_FvsM	24.53	63.04	25.39	27.63	26.08	26.60	26.25	33.51	25.39	25.39	24.70
30	iris_setosavsversicolor	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	12.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
31	iris_setosavsvirginica	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	12.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
32	iris_versicolorvsvirginica	**3.00**	4.00	**3.00**	**3.00**	**3.00**	6.00	4.00	10.00	**3.00**	6.00	5.00
33	irish-ed_MvsF	45.00	43.40	45.40	42.80	44.60	42.60	42.40	44.60	40.20	43.60	39.60
34	kidney	**28.95**	**28.95**	34.21	**28.95**	**28.95**	31.58	35.53	30.26	**28.95**	31.58	**28.95**
35	pima	**24.50**	27.50	29.00	26.00	28.50	31.00	31.00	28.00	30.50	30.00	27.50
36	plasma-retinol_MvsF	14.29	13.97	**13.33**	15.24	16.83	16.51	15.24	25.08	15.24	13.97	15.56
37	segmentation	8.33	9.24	4.55	4.55	**4.09**	7.58	5.30	14.55	7.73	8.79	12.73
38	socmob_IvsNI	34.34	34.34	33.48	32.70	33.30	32.61	33.13	33.04	32.09	**30.36**	34.17
39	socmob_WvsB	28.89	29.15	19.12	17.99	17.91	**17.65**	18.08	18.94	20.16	19.64	21.89
40	tae	17.22	19.87	23.18	**11.92**	13.91	14.57	17.22	15.89	17.22	16.56	17.22
41	tennis_MvsF	41.38	44.83	43.68	45.98	49.43	47.13	39.08	52.87	37.93	**36.78**	41.38
42	tips_DvsN	6.15	3.69	8.20	5.33	6.15	12.70	9.84	8.20	**3.28**	4.10	9.84
43	tips_MvsF	36.48	38.52	**32.38**	42.21	43.85	40.16	39.34	41.39	38.11	38.11	34.43
44	uscrime_SvsN	17.02	19.15	8.51	17.02	6.38	48.94	21.28	**2.13**	19.15	19.15	6.38
45	vertebral-column	15.81	17.42	15.81	17.10	18.06	21.94	23.55	19.03	**14.52**	15.16	16.45
46	veteran-lung-cancer	64.23	51.82	51.82	53.28	43.80	48.91	42.34	**40.15**	47.45	49.64	53.28
47	vowel_MvsF	0.10	0.71	**0.00**	1.41	1.92	11.82	0.00	2.02	0.51	0.51	12.63
48	wine_1vs2	**0.00**	0.77	6.15	1.54	2.31	5.38	2.31	3.08	1.54	1.54	1.54
49	wine_1vs3	**0.00**	**0.00**	11.21	**0.00**	0.93	0.93	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
50	wine_2vs3	0.84	**0.00**	23.53	1.68	2.52	5.04	4.20	5.04	**0.00**	**0.00**	8.40

Table 3.5: *Comparison of classification indicators: average classification error (ACE); average relative classification edge (ARCE) w.r.t. the* best traditional *and the* overall best *classifier for a classification task; frequencies of better performance than the* best traditional *(#≥trd) and the* overall best *(#≥bst) classifier.*

No.	Classifier	ACE	ARCE$_{\text{trd}}$	ARCE$_{\text{bst}}$	#≥trd	#≥bst
1	LDA	16.79	-2.95	-4.73	**0.52**	**0.32**
2	QDA	18.10	-3.99	-6.28	0.36	0.18
3	KNN	18.00	-3.57	-5.84	0.42	0.16
4	$DD\alpha$+RTD+LDA	16.58	-2.05	-4.37	0.38	0.22
5	$DD\alpha$+RTD+KNN	17.16	-2.82	-5.13	0.30	0.12
6	$DD\alpha$+RTD+Mah.d.(mom.)	19.52	-4.88	-7.27	0.20	0.08
7	$DD\alpha$+RTD+Mah.d.(MCD$_{\frac{3}{4}}$)	17.22	-2.37	-4.83	0.32	0.14
8	$DD\alpha$+RTD+SVM-s	17.38	-2.74	-5.18	0.38	0.28
9	$DD\alpha$+Mah.d.(mom.)	16.02	-1.05	-3.48	0.48	0.28
10	$DD\alpha$+Mah.d.(MCD$_{\frac{3}{4}}$)	17.43	-2.42	-4.79	0.42	0.28
11	$DD\alpha$+Spt.d.(mom.)	**15.79**	**-0.68**	**-3.15**	**0.52**	0.30
12	$DD\alpha$+Spt.d.(MCD$_{\frac{3}{4}}$)	17.63	-2.83	-5.24	0.30	0.12
13	$DD\alpha$+Prj.d.(10 000 dirs)	18.39	-4.09	-6.35	0.22	0.12
14	$DD\alpha$+Prj.d.(100 000 dirs)	17.51	-2.70	-5.12	0.36	0.16

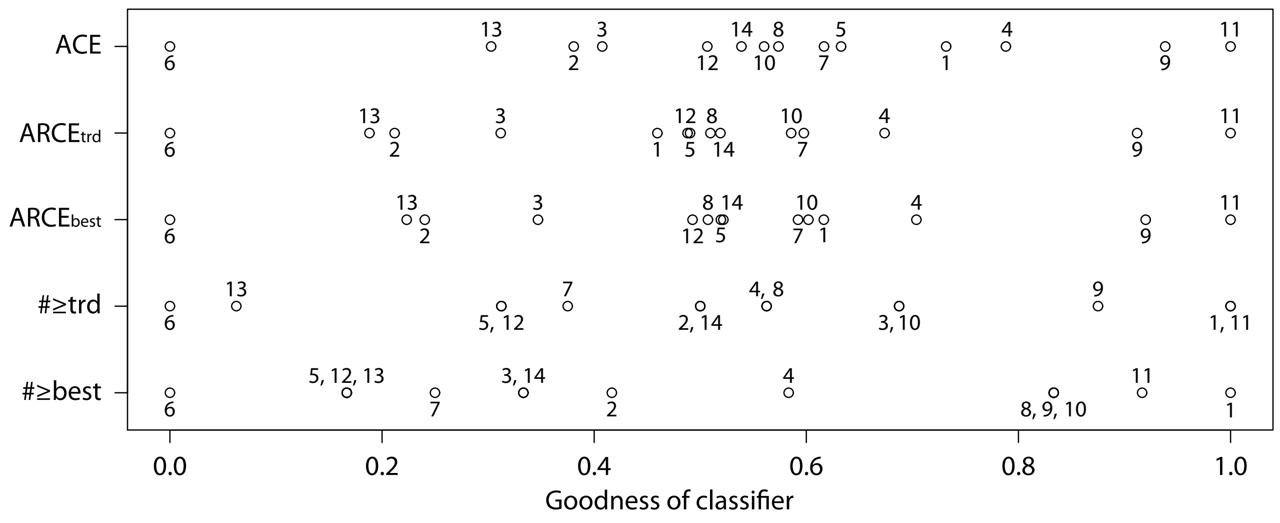

Figure 3.5: *Goodness of the 14 considered classifiers (abscissa) w.r.t. the five performance measures (ordinate), obtained after standardizing the measures. The classifiers are numerated as in Table 3.5.*

fraction of the training sample. In addition there are data sets (not that rarely encountered), where the outsider share amounts to 100% so that the pure $DD\alpha$-approach does not separate anything. This also explains the result obtained in Section 3.4.3 that the better classifiers are based on non-vanishing depths.

3.5.2 Dimension of the depth-feature space

Finally, we come to the question, how many depth features are needed on an average to satisfactorily separate the data. The $DD\alpha$-procedure uses the centered version of the α-procedure (Vasil'ev, 2003, Vasil'ev & Lange, 1998) to separate the classes in the depth space. As the α-procedure is a heuristic approach, it is of interest how close this separation rule comes to the optimal one. Such a point of view corresponds to the theoretical assumption that the depth transformation does not impair the separability of the data set. By its nature the resulting separating rule is similar to that proposed in Li *et al.* (2012), where the polynomial degree is chosen by cross-validating. Other possible approaches are regression depth (Christmann & Rousseeuw, 2001, Christmann *et al.*, 2002) or SVM (Christmann *et al.*, 2002, Vapnik, 1998). It is clear that in general the obtained separating hypersurface is not the one minimizing *EMR*, if more than two features are needed. But in which of the applications are they really needed?

Tables 3.6 and 3.7 exhibit the relative frequency of feature numbers, as they are selected by α-separation and leave-one-out cross-validation. Surely these tables cannot be regarded as histograms (since we would need some bootstrap for that), but it can be concluded that (on an average) in 99% of the cases two features are sufficient for the separating rule. Note that the two-feature separators may include depth features of the extended type. Such a two-feature rule clearly minimizes the *EMR* in the relevant plane. Of course it may be possible to find a hyperplane in a more extended space, delivering a smaller *EMR* than that obtained by the α-separation. On the other hand, no space extension is needed in around 78% of the cases. In these cases no polynomial products of depths are involved and the resulting separating rule is linear.

3.6 Conclusions

A fast classification procedure, the $DD\alpha$-procedure, has been introduced that is essentially nonparametric, robust, and computationally feasible for any dimension d of attributes. The $DD\alpha$-procedure is available in the R-package `ddalpha`. The $DD\alpha$-classifier is particularly robust for two reasons: first, as the classification is done by the α-procedure, which is *per se* robust; second, as the data are transformed into a low-dimensional compact space. Generally, two cases are to be distinguished with the depth transform: non-vanishing depth or depth vanishing beyond the convex hulls of the training classes. Non-vanishing depths, e.g. Mahalanobis or spatial, often induce a spurious symmetry and are intrinsically non-robust (though, can be robustified). The projection depth, which is non-vanishing, also produces ellipsis-like regions. It is robust, but computationally inefficient when $d \geq 3$. The last problem is faced by the Tukey depth, which best reflects the shape of the data. In place of the exact versions of projection and Tukey depth we employ their random versions by minimizing univariate depths in directions that are uniformly distributed on the sphere. A very large number of these directions is needed for the calculation of the projection depth, while for the random Tukey depth

Table 3.6: *Numbers of features selected by the α-separation; Part 1.*

No.	Dataset	2, %	3, %	≥ 4, %
1	baby	99.19	0.81	0.00
2	banknoten	100.00	0.00	0.00
3	biomed	100.00	0.00	0.00
4	blood-transfusion	91.31	1.34	7.35
5	breast-cancer-wisconsin	97.57	2.43	0.00
6	bupa	100.00	0.00	0.00
7	chemdiab_1vs2	100.00	0.00	0.00
8	chemdiab_1vs3	100.00	0.00	0.00
9	chemdiab_2vs3	100.00	0.00	0.00
10	cloud	100.00	0.00	0.00
11	crab_BvsO	100.00	0.00	0.00
12	crab_MvsF	100.00	0.00	0.00
13	crabB_MvsF	100.00	0.00	0.00
14	crabF_BvsO	100.00	0.00	0.00
15	crabM_BvsO	100.00	0.00	0.00
16	crabO_MvsF	100.00	0.00	0.00
17	cricket_CvsP	98.72	1.28	0.00
18	diabetes	99.61	0.13	0.26
19	ecoli_cpvsim	100.00	0.00	0.00
20	ecoli_cpvspp	100.00	0.00	0.00
21	ecoli_imvspp	100.00	0.00	0.00
22	gemsen_MvsF	99.85	0.07	0.07
23	glass	100.00	0.00	0.00
24	groessen_MvsF	99.57	0.00	0.43
25	haberman	100.00	0.00	0.00
	Average	99.08	0.65	0.27

the number of directions can be kept low, as there is a tradeoff between this number and the number of points being classified by their depth values.

On the other hand the random Tukey depth yields outsiders when classifying, i.e. points lying outside the convex hulls of all classes, which cannot be readily classified and need additional treatment. In real data applications the percentage of outsiders can be large (see the introduced measure of outsider proneness) and thus substantially influence the classification performance. Therefore, the choice of the treatment is important when applying the random Tukey depth. The treatments considered subject the outsiders either to linear discriminant analysis (LDA), classification according to k-nearest neighbors (KNN), maximum Mahalanobis depth classification based on moment or MCD estimates, or the newly introduced SVM-simplified procedure (*SVM-s*). The latter is very fast as it needs no tuning of a box-constraint; only the smallest separating γ has to be computed. Additional calculations (not included here, see also Lange *et al.* (2014a)) show, that regarding the other possible outsider treatments, the choices of number k in KNN as well as of the covering parameter in MCD do not much influence their performance.

Table 3.7: *Numbers of features selected by the α-separation; Part 2.*

No.	Dataset	2, %	3, %	$\geq$ 4, %
26	heart	100.00	0.00	0.00
27	hemophilia	100.00	0.00	0.00
28	indian-liver-patient_1vs2	99.48	0.52	0.00
29	indian-liver-patient_MvsF	98.96	1.04	0.00
30	iris_setosavsversicolor	100.00	0.00	0.00
31	iris_setosavsvirginica	100.00	0.00	0.00
32	iris_versicolorvsvirginica	100.00	0.00	0.00
33	irish-ed_MvsF	99.60	0.20	0.20
34	kidney	98.68	1.32	0.00
35	pima	93.00	5.50	1.50
36	plasma-retinol_MvsF	99.37	0.63	0.00
37	segmentation	93.48	4.85	1.67
38	socmob_IvsNI	99.57	0.35	0.09
39	socmob_WvsB	99.83	0.09	0.09
40	tae	93.38	6.62	0.00
41	tennis_MvsF	100.00	0.00	0.00
42	tips_DvsN	99.59	0.41	0.00
43	tips_MvsF	100.00	0.00	0.00
44	uscrime_SvsN	100.00	0.00	0.00
45	vertebral-column	98.39	1.61	0.00
46	veteran-lung-cancer	100.00	0.00	0.00
47	vowel_MvsF	94.75	3.43	1.82
48	wine_1vs2	100.00	0.00	0.00
49	wine_1vs3	100.00	0.00	0.00
50	wine_2vs3	100.00	0.00	0.00
	Average	99.08	0.65	0.27

Thus the $DD\alpha$-procedure needs practically no parameter tuning. The degree of the separating polynomial is chosen by cross-validation within the depth representation only, where the modified α-procedure, on each of the planar subspaces considered, has a quick sort complexity, $\mathrm{O}\big((n_1+n_2)\log(n_1+n_2)\big)$, and by that is very fast.

The above introduced variants of the $DD\alpha$-procedure are challenged by 50 binary classification problems that arise from a broad range of real data. The tasks are complicated by the presence of outliers and ties. As competitors of the $DD\alpha$-procedure three traditional classification methods (LDA, QDA, and KNN) are evaluated with the same data. Naturally, none of the classifiers is best at all tasks in terms of the average error rate, but each classifier is best at some of them. Our results also show that no single depth or outsider treatment dominates the others. Just for almost all data sets the classification of outsiders according to their maximum moment-Mahalanobis depth is outperformed by the same with the MCD-Mahalanobis depth. This can be explained by the outliers present.

Five goodness measures are introduced aggregating performance of the classifiers over the 50 classification tasks w.r.t. different aspects and allowing for direct comparison of the classifiers.

Clearly, classification performance greatly depends on the choice of the depth and, if needed, the outsider treatment. The measures point out two $DD\alpha$-classifiers (starting with the best): based on spatial and Mahalanobis depth (both using moment estimates). The rest of the classifiers show varying performance for different goodness measures, which is demonstrated by the goodness visualization.

The experience with real-data problems tells us further that, in most practical cases, the separation procedure in the depth space stops after a few steps. In most cases the subspace spanned by the depth features needed has dimension two, which points out the high stability of the separating rule.

The problems and solutions investigated in this study are also of interest in more general settings: The problem of coping with outsiders is common to any statistical procedure that is based on depth plots and involves a notion of depth vanishing outside the convex hull. Using the random Tukey depth as an efficient approximation of the Tukey depth and selecting the random directions has many applications. Available algorithms for exactly calculating the Tukey depth (Rousseeuw & Struyf, 1998, Liu & Zuo, 2014a) are computationally expensive, but can serve as a benchmark. Finally, the SVM-simplified method is introduced and appears as a simple and efficient way to avoid the computational burden of tuning the SVM.

Chapter 4

Fast DD-classification of functional data

4.1 Introduction

The problem of classifying objects that are represented by functional data arises in many fields of application like biology, biomechanics, medicine and economics; Ramsay & Silverman (2005) and Ferraty & Vieu (2006) contain broad overviews of functional data analysis and the evolving field of classification. The data are either genuine functional data or high-dimensional data representing functions at discretization points. At the very beginning of the 21st century many classification approaches have been extended from multivariate to functional data: linear discriminant analysis (James & Hastie, 2001), kernel-based classification (Ferraty & Vieu, 2003), k-nearest-neighbors classifier (Biau et al., 2005), logistic regression (Leng & Müller, 2006), neural networks (Ferré & Villa, 2006), support vector machines (Rossi & Villa, 2006). Transformation of functional data into a finite setting is done by using principal and independent (Huang & Zheng, 2006) component analysis, principal coordinates (Hall et al., 2001), wavelets (Wang *et al.*, 2007) or functions of very simple and interpretable structure (Tian & James, 2013), or some optimal subset of initially given evaluations (Ferraty *et al.*, 2010, Delaigle *et al.*, 2012).

Generally, functional data is projected onto a finite dimensional space in two ways: by either fitting some finite basis or using functional values at a set of discretization points. The first approach accounts for the entire functional support, and the basis components can often be well interpreted. However, the chosen basis is not *per se* best for classification purposes, E.g., principal component analysis (PCA) maximizes dispersion but does not minimize classification error. Moreover, higher order properties of the functions, which are regularly not incorporated, may carry information that is important in the classification phase; see Delaigle & Hall (2012) for discussion. The second approach appears to be natural as the finite-dimensional space is directly constructed from the observed values. But any selection of discretization points restricts the range of the values regarded, so that some classification-relevant information may be lost. Also, the data may be given at arguments of the functions that are neither the same nor equidistant nor enough frequent; then some interpolation is needed and interpolated data instead of the original one are analyzed. Another issue is the way the space is synthesized. If it is a heuristics (as in Ferraty *et al.* (2010)), a well classifying configuration of discretization points may be missed: To see this, consider three discretization points in $\mathbb{R}^3$ which jointly discriminate well, but which cannot be chosen subsequently because each of them has a relatively small discrimination power compared to some other available discretization points. To cope with this problem, Delaigle *et al.* (2012) consider (almost) all sets of discretization points that have a given cardinality; but this procedure involves an enormous computational burden, which restricts its practical application to rather small data sets.

Several authors López-Pintado & Romo (2006), Cuevas *et al.* (2007), Cuesta-Albertos & Nieto-Reyes (2010), Sguera *et al.* (2014) employ nonparametric notions of data depth for functional data classification. A data depth measures how close a given object is to an – implicitly given – center of a class of objects; that is, if objects are functions, how central a given function is in an empirical distribution of functions.

Specifically, the band depth (López-Pintado & Romo, 2006) of a function $\mathbf{x}$ in a class X of functions indicates the relative frequency of $\mathbf{x}$ lying in a band shaped by any J functions from X, where J is fixed. Cuevas *et al.* (2007) examine five functional depths for tasks of robust estimation and supervised classification: the *integrated depth* of Fraiman & Muniz (2001), which averages univariate depth values over the function's domain; the *h-mode depth*, employing a kernel; the *random projection depth*, taking the average of univariate depths in random directions; and the *double random projection depths* that include first derivatives and are based on bivariate depths in random directions. Cuesta-Albertos & Nieto-Reyes (2010) classify the Berkeley growth data (Tuddenham & Snyder, 1954) by use of the random Tukey depth (Cuesta-Albertos & Nieto-Reyes, 2008). Sguera *et al.* (2014) introduce a functional spatial depth and a kernelized version of it.

There are several problems connected with the depths mentioned above. First, besides double random projection depth, the functions are treated as multivariate data of infinite dimension. By this, the development of the functions, say in time, is not exploited; these depth notions are invariant with respect to an arbitrary rearrangement of the functions. Second, several of these notions of functional depth break down in standard distributional settings, i.e. the depth functions vanish almost everywhere; see Chakraborty & Chaudhuri (2014), Kuelbs & Zinn (2013). Eventually, the depth takes empirical zero values if the function's hyper-graph has no intersection with the hypo-graph of any of the sample functions or vice versa, which is the case for both half-graph and band depths and their modified versions, as well as for the integrated depth. If a function has zero depth with respect to each class it is mentioned as an *outsider*, because it cannot be classified immediately and requires an additional treatment (see Chapters 2 and 3).

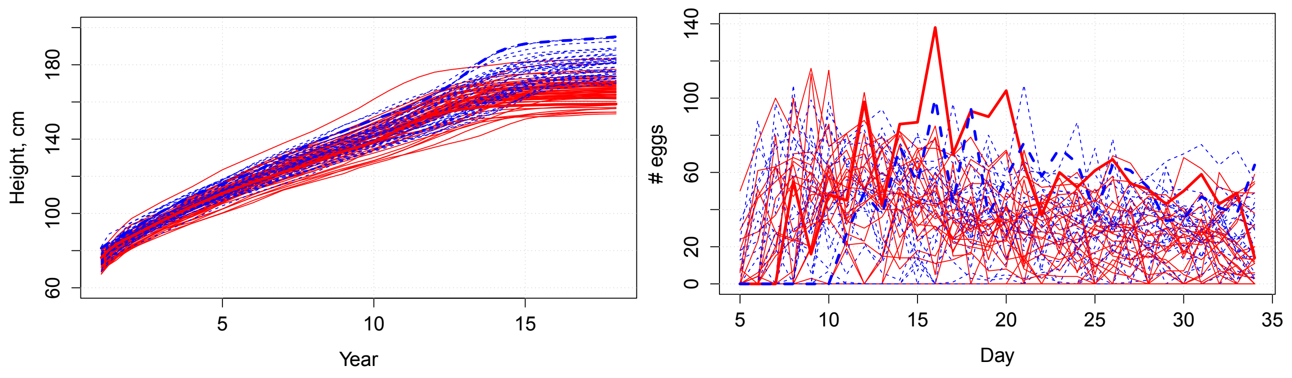

Figure 4.1: *Growth of 54 girls (solid) and 39 boys (dashed) (left); 35 observations of medflies fertility (right). The fat curve indicates a single member of each class.*

4.1.1 Two benchmark problems

Naturally, the performance of a classifier and its relative advantage over alternative classifiers depend on the actual problem to be solved. Therefore we start with two benchmark data settings, one fairly simple and the other one rather involved.

First, a popular benchmark set is the *growth data* of the Berkeley Growth Study (Tuddenham & Snyder, 1954). It comprises the heights of 54 girls and 39 boys measured at 31 non-equally distant time points, see Figure 4.1, left. This data was extensively studied in Ramsay & Silverman (2005). After having become a well-behaved classic it has been considered as a benchmark in many works on supervised and unsupervised classification, by some authors also in a data depth context. Based on the generalized band depth López-Pintado & Romo (2006) introduce a trimming and weighting of the data and classify them by their (trimmed) weighted average distance to observation or their distance to the trimmed mean. Cuevas *et al.* (2007) classify the *growth data* by their maximum depth, using the five above mentioned projection-based depths and kNN as a reference; see also Cuesta-Albertos & Nieto-Reyes (2010). Recently, Sguera *et al.* (2014) apply functional spatial depth to this data set.

Second, we consider a much more difficult task: classifying the *medflies data* of Carey *et al.* (1998), where 1000 30-day (starting from the fifth day) egg-laying patterns of Mediterranean fruit fly females are observed. The classification task is to explain longevity by productivity. For this a subset of 534 flies living at least 34 days is separated into two classes: 278 living 10 days or longer after the 34th day (long-lived), and 256 those have died within these 10 days (short-lived), which are to be distinguished using daily egg-laying sequences (see Figure 4.1 (right) with the linearly interpolated evaluations). This task is taken from Müller & Stadtmüller (2005), who demonstrate that the problem is quite challenging and cannot be satisfactorily solved by means of functional linear models.

4.1.2 The new approach

We shall introduce a new methodology for supervised functional classification covering the mentioned issues and validate it with the considered real data sets as well as with simulated data. Our approach is completely non-parametric, oriented to work with raw and irregularly sampled functional data, and does not involve heavy computations.

Clearly, as any statistical methodology for functional data, our classification procedure has to map the relevant features of the data to some finite dimensional setting. For this we map the functional data to a finite-dimensional location-slope space, where each function is represented by a vector consisting of integrals of its levels ('location') and first derivatives ('slope') over L resp. S equally sized subintervals. Thus, the location-slope space has dimension $L + S$. The functions are linearly interpolated; hence their levels are integrated as piecewise-linear functions, and the derivatives as piecewise constant ones. Then we classify the data within the location-slope space using a proper depth-based technique. We restrict $L + S$ by a Vapnik-Chervonenkis bound and determine it finally by cross-validation over the restricted set, which is very quickly done. The resulting functional data depth will be mentioned as the *integral location-slope depth*.

We suggest the following two depth-based classification techniques: (1) After a DD-transform (Li *et al.*, 2012), we apply the k-nearest neighbors (kNN) classifier in the depth-space, which has been first suggested by Vencálek (2011). This is a computationally tractable alternative to the polynomial separating rule used in Li *et al.* (2012) and yields the same asymptotic result. (2) Alternatively we employ the $DD\alpha$-classifier, which has a heuristic nature and is often much faster than the kNN approach.

The new approach is presented here for $q = 2$ classes, but it is not limited to this case. If $q > 2$, kNN is applied in the q-dimensional depth space without changes, and the $DD\alpha$-classifier is extended by either constructing q one-against-all or $\binom{q}{2}$ pair-wise classifiers in the depth space, and finally performing some aggregation in the classification phase; see also Chapter 2.

We contrast our approach with several existing procedures applied to the data as they are represented in the location-slope space: a kNN classifier, three naive maximum-depth classifiers (Ghosh & Chaudhuri, 2005b) employing different depths, linear and quadratic discriminant analysis. Our space selection technique (incomplete cross-validation, restricted by a Vapnik-Cervonenkis bound) is compared, both in terms of error rates and computational time, with a full cross-validation as well as with the componentwise space synthesis method of Delaigle *et al.* (2012). We do this for all variants of the classifiers. For simulated data highly satisfactory results are obtained. Further, as we will see below in detail, our approach reaches substantially lower error rates for the *growth data* than those obtained in the literature. Even more important, we are able to give further insights regarding the *medflies'* classification problem, which could not be tackled so far.

4.1.3 Overview

The rest of the paper is organized as follows: Section 4.2 reviews the use of data depth techniques in classifying finite-dimensional objects. Section 4.3 presents the new two-step representation of functional data, first in a finite-dimensional Euclidean space (the location-slope space), and then in a depth-to-depth plot (DD-plot). In Section 4.4 we introduce two alternative classifiers that operate on the DD-plot, a nearest-neighbor procedure and an α-procedure. Section 4.5 provides an approach to bound and select the dimension of the location-slope space. In Section 4.6 our procedure is applied to simulated data and compared with other classifiers. Also, the above two real data problems are solved and the computational efficiency of our approach is discussed. Section 4.7 concludes. Theoretical properties of the developed methodology, implementation details and additional experimental results are collected in the Appendix.

4.2 Depth based approaches to classification in $\mathbb{R}^d$

For data in Euclidean space $\mathbb{R}^d$ many special depth notions have been proposed in the literature; see, e.g., Zuo & Serfling (2000) for definition and properties. Here we mention three depths, Mahalanobis, spatial and projection depth, which we will use later in defining notions of functional depth. These depths are everywhere positive. Hence they do not produce outsiders, that is, points having zero depth in both training classes. The no-outsider property appears to be essential for obtaining *nontrivial* functional depths.

For a point $\mathbf{y} \in \mathbb{R}^d$ and a random vector Y having an empirical distribution on $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ in $\mathbb{R}^d$ the *Mahalanobis depth* (Mahalanobis, 1936) of $\mathbf{y}$ w.r.t. Y is defined as

$$D^{Mah}(\mathbf{y}|Y) = \big(1 + (\mathbf{y} - \mu_Y)'\Sigma_Y^{-1}(\mathbf{y} - \mu_Y)\big)^{-1}, \tag{4.1}$$

where μ_Y measures the location of Y, and Σ_Y the scatter.

The *affine invariant spatial depth* (Vardi & Zhang, 2000, Serfling, 2002) of $\mathbf{y}$ regarding Y is defined as

$$D^{Spt}(\mathbf{y}|Y) = 1 - \|E_Y\left[v\big(\Sigma_Y^{-1/2}(\mathbf{y} - Y)\big)\right]\|, \tag{4.2}$$

where $v(\mathbf{w}) = \|\mathbf{w}\|^{-1}\mathbf{w}$ for $\mathbf{w} \neq \mathbf{0}$ and $v(\mathbf{0}) = \mathbf{0}$, and Σ_Y is the covariance matrix of Y. D^{Mah} and D^{Spt} can be efficiently computed in $\mathbb{R}^d$.

The *projection depth* (Zuo & Serfling, 2000) of $\mathbf{y}$ regarding Y is given by

$$D^{Prj}(\mathbf{y}|Y) = \inf_{\mathbf{u}\in S^{d-1}} \big(1 + O^{Prj}(\mathbf{y}|Y, \mathbf{u})\big)^{-1}, \tag{4.3}$$

with

$$O^{Prj}(\mathbf{y}|Y, \mathbf{u}) = \frac{|\mathbf{y}'\mathbf{u} - m(Y'\mathbf{u})|}{MAD(Y'\mathbf{u})}, \tag{4.4}$$

where m denotes the univariate median and MAD the median absolute deviation from the median. Exact computation of D^{Prj} is, in principle, possible (Liu & Zuo, 2014b) but practically infeasible when $d > 2$. Obviously, D^{Prj} is approximated from above by calculating the minimum of univariate projection depths in random directions. However, as D^{Prj} is piece-wise linear (and, hence, attains its maximum on the edges of the direction cones of constant linearity), a randomly chosen direction yields the exact depth value with probability zero. For a sufficient approximation one needs a huge number of directions, each of which involves the calculation of the median and MAD of a univariate distribution.

To classify objects in Euclidean space $\mathbb{R}^d$, the existing literature employs depth functions in principally two ways:

1. Classify the original data by their maximum depth in the training classes.
2. Transform the data by their depths into a low-dimensional depth space, and classify them within this space.

Ad 1: In Ghosh & Chaudhuri (2005a) the supervised learning task is restated in terms of data depth and this measure of centrality is adopted to the construction of classifiers. In Ghosh & Chaudhuri (2005b) the same authors propose the *maximum depth classifier*, which assigns an object to that class in which it has maximum depth. In its naive form this classifier yields a linear separation. Maximum-depth classifiers have a plug-in structure; thus their scale parameters need to be tuned (usually by some kind of cross-validation) over the whole learning process. For this, Ghosh & Chaudhuri (2005b) combine the naive maximum-depth classifier with an estimate of the density. A similar approach is pursued with projection depth in Dutta & Ghosh (2012) and l_p depth in Dutta & Ghosh (2011), yielding competitive classifiers.

Ad 2: Depth notions are also used to reduce the dimension of the data. Li *et al.* (2012) employ the DD-plot, which represents all objects by their depth in the two training classes, that is, by points in the unit square. (The same for q training classes in the q-dimensional unit cube.) To solve the classification task, some separation rule has to be constructed in the unit square. They minimize the *empirical risk*, that is the average classification error on the training classes, by smoothing it with a logistic sigmoid function and, by this, obtain a polynomial separating rule; they show that their approach (with Mahalanobis, projection and other depths) asymptotically achieves the optimal Bayes risk if the training classes are strictly unimodal elliptically distributed. However, in practice the choice of the smoothing constant and non-convex optimization, potentially with many local minima, encumber its application. In Chapter 2 these problems are addressed via the α-procedure, which is very fast and speeds up the learning phase enormously. With some depth notions, e.g. location (Tukey, 1975) and zonoid (Koshevoy & Mosler, 1997) depths, *outsiders* occur, as they vanish outside the convex hull of the distribution's support. In case of many outsiders the error rate of the DD-classifier can increase. This problem is discussed in Chapters 2 and 3, where several alternative treatments are proposed and compared. If a distribution has more than one mode, classical depth notions may be inappropriate as they refer to a single center. Multimodality of the underlying distributions is coped with by means of 'local' approaches (see, e.g., Paindaveine & Van Bever (2012), Dutta *et al.* (2012)), however at the price of onerous computations.

4.3 A new depth transform for functional data

Let $\mathcal{F}$ be the space of real functions, defined on a compact interval, which are continuous and almost everywhere smooth. The data may be given either as observations of complete functions in $\mathcal{F}$ or as functional values at some discretization points, in general neither equidistant nor common ones. If the functional data is given in discretized form, it is usually interpolated by splines of some order (see Ramsay & Silverman (2005)), so that sufficiently smooth functions are obtained. Here we use linear interpolation (that is splines of order 1) for the following reasons. Firstly, with linear interpolation, neither the functions nor their derivatives need to be smoothed. Thus almost any raw data can be handled. E.g., the *medflies data*, as to the egg-laying process, is naturally discrete; see Figure 4.1 (right). Secondly, higher order splines increase the computational load, especially when the number of knots or the smoothing parameter are to be determined as part of the task. Thirdly, splines of higher order may introduce spurious information.

We construct a depth transform as follows: In a first step, the relevant features of the functional data are extracted from $\mathcal{F}$ into a finite-dimensional Euclidean space $\mathbb{R}^{L+S}$, which we call *the location-slope space* (Section 4.3.1). Then an $(L+S)$-dimensional depth is applied to the transformed data yielding a *DD-plot* in the unit square (Section 4.3.2), which represents the two training classes. Finally the separation of the training classes as well as the classification of new data is done on the *DD*-plot (Section 4.4).

4.3.1 The location-slope transform

We consider two classes of functions in $\mathcal{F}$, $X_0 = \{\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_m\}$ and $X_1 = \{\tilde{\mathbf{x}}_{m+1}, \dots, \tilde{\mathbf{x}}_{m+n}\}$, which are given as measurements at ordered points $t_{i1} \le t_{i2} \le ... \le t_{ik_i}$, $i = 1, ..., m+n$,

$$[\tilde{\mathbf{x}}_i(t_{i1}), \tilde{\mathbf{x}}_i(t_{i2}), ..., \tilde{\mathbf{x}}_i(t_{ik_i})] \ .$$

Assume w.l.o.g. $\min_i t_{i1} = 0$ and notate $T = \max_i t_{ik_i}$. From $\tilde{\mathbf{x}}_i$ a function $\mathbf{x}_i : [0,T] \to \mathbb{R}$ is obtained as follows: connect the points $(t_{ij}, \tilde{\mathbf{x}}_i(t_{ij})), j = 1, \dots, k_i$, with line segments and set $\mathbf{x}_i(t) = \tilde{\mathbf{x}}_i(t_{i1})$ when $0 \le t \le t_{i1}$ and $\mathbf{x}_i(t) = \tilde{\mathbf{x}}_i(t_{ik_i})$ when $t_{ik_i} \le t \le T$. By this, the data become piecewise linear functions on $[0,T]$, and their first derivatives become piecewise constant ones, see Figure 4.2, left.

A finite-dimensional representation of the data is then constructed by integrating the interpolated functions over L subintervals (location) and their derivatives over S subintervals (slope), see Figure 4.2 (left). It delivers the following transform,

$$\begin{aligned} \mathbf{x}_i \quad \longmapsto \mathbf{y}_i \quad = \quad & \Big[\int_0^{T/L} \mathbf{x}_i(t)dt, \dots, \int_{T(L-1)/L}^{T} \mathbf{x}_i(t)dt, \\ & \int_0^{T/S} \mathbf{x}'_i(t)dt, \dots, \int_{T(S-1)/S}^{T} \mathbf{x}'_i(t)dt\Big] \, . \end{aligned} \tag{4.5}$$

That is, the $L+S$ average values and slopes constitute a point $\mathbf{y}_i$ in $\mathbb{R}^{L+S}$. Either L or S must be positive, $L + S \ge 1$. In case $L = 0$ or $S = 0$ the formula (4.5) is properly modified. Put together we obtain a composite transform $\phi : \mathcal{F} \to \mathbb{R}^{L+S}$,

$$\tilde{\mathbf{x}}_i \quad \mapsto \quad [\tilde{\mathbf{x}}_i(t_{i1}), \dots, \tilde{\mathbf{x}}_i(t_{ik_i})] \quad \mapsto \quad \mathbf{x}_i \quad \mapsto \quad \mathbf{y}_i \, , \tag{4.6}$$

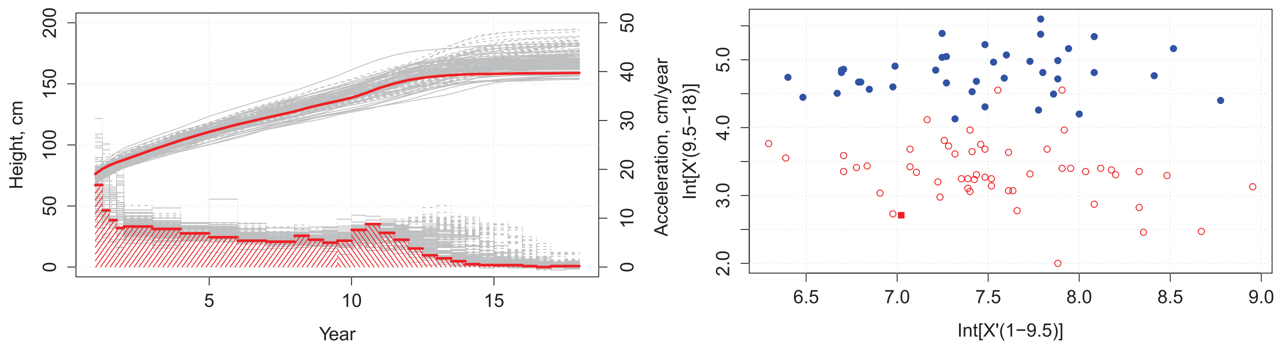

Figure 4.2: *Example: Transformation of a single growth function into the integral location-slope space with* $L = 0$ *and* $S = 2$. *Function(s) and first derivative(s) (left); corresponding two-dimensional space* $(0, 2)$ *(right).*

which we call the *location-slope (LS-) transform.*

For example, choose $L = 0$ and $S = 2$ for the *growth data.* Then they are mapped into the location-slope space $\mathbb{R}^{L+S} = \mathbb{R}^2$, which is shown in Figure 4.2 (right). Here the functions' first derivatives are averaged on two half-intervals. That is, for each function two integrals of the slope are obtained: the integral over $[1; 9.5]$ and that over $[9.5; 18]$. Here, the location is not incorporated at all. Figure 4.2, left, exhibits the value (height) and first derivative (acceleration) of a single interpolated function, which is then represented by the average slopes on the two half-intervals, yielding the rectangular point in Figure 4.2 (right).

Proposition 4.1. *The location-slope transform (4.6) is a weakly continuous functional* $\mathcal{F} \to \mathbb{R}^{L+S}$.

Note that the first of the sequential transforms is continuous in the weak topology of $\mathcal{F}$, and the others are continuous. Thus, the LS-transform is weakly continuous. Consequently, our procedure is stable against perturbations or contaminations of the data.

Further, given the data, L and S can be chosen large enough to reflect all relevant information about the functions. (Note that the computational load of the whole remaining procedure is linear in dimension $L + S$.) If L and S are properly chosen, under mild assumptions, the LS-transform preserves asymptotic Bayes optimality, thus allowing for procedures that achieve error rates close to minimum. This issue is treated in Appendix 1.

Naturally, only part of these $L + S$ intervals carries the information needed for separating the two classes. Delaigle *et al.* (2012) propose to determine a subset of points (not intervals) in $[0, T]$ based on which the training classes are optimally separated. However they do not provide a practical procedure to select these points; in applications they use cross-validation. Moreover, intervals whose information does not much contribute to the training phase may be important in the classification phase. So, we have generally no prior reason to weight the intervals differently. Therefore we use intervals of equal length, but possibly different ones for location and slope.

The question remains how many equally sized subintervals, L for location and S for slope, should be taken. We will see later in Section 5.4 that our classifier performs similar with the three depths when the dimension is low. In higher dimensions the projection depth cannot

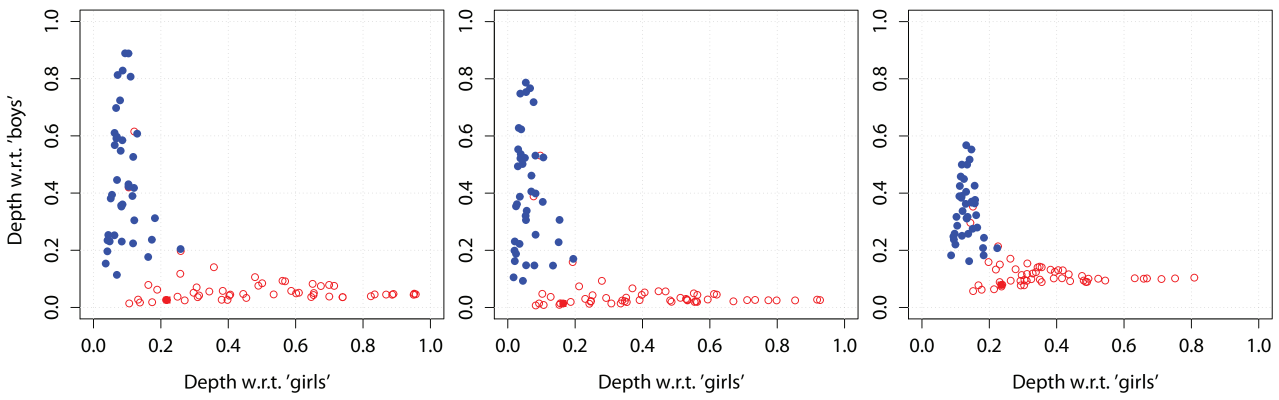

Figure 4.3: *DD-plots for* growth data *with* $(L, S) = (0, 2)$ *using Mahalanobis (left), spatial (middle) and projection (right) depths.*

be computed precisely enough, so that the classifier becomes worse. The performance is most influenced by the construction of the location-slope transform, that is, by the choice of the numbers L and S. We postpone this question to Section 4.5.

4.3.2 The DD-transform

Denote the location-slope-transformed training classes in $\mathbb{R}^{L+S}$ by $Y_0 = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ and $Y_1 = \{\mathbf{y}_{m+1}, \ldots, \mathbf{y}_{m+n}\}$. The DD-plot is then

$$Z = \{\mathbf{z}_i = (z_{i0}, z_{i1}) \,|\, z_{i0} = D^{L+S}(\mathbf{y}_i|Y_0), z_{i1} = D^{L+S}(\mathbf{y}_i|Y_1), i = 1, ..., m+n\}\,.$$

Here D^{L+S} is an $(L+S)$-dimensional depth. In particular, D^{L+S} may be D^{Mah}, D^{Spt} or D^{Prj}, each of which does not produce outsiders. The DD-plots of these three for *growth data*, taking $L = 0$ and $S = 2$, are pictured in Figure 4.3. Clearly, in this case, almost faultless separation is achievable by drawing a straight line through the origin. Note that in general any separating line in $[0, 1]^2$ should pass the origin, as a point having both depths $= 0$ cannot be readily classified.

4.4 DD-plot classification

The training phase of our procedure consists in the following: After the training classes have been mapped from $\mathcal{F}$ to the DD-plot as described in Section 4.3, a selector is determined in the DD-plot that separates the DD-transformed data. For the latter, we consider two classifiers operating on the DD-plot, and compare their performance. Firstly we propose the kNN classifier. It has the same asymptotic behavior as the polynomial rule suggested in Li *et al.* (2012), i.e. converges to the Bayes rule when the distributions are strictly unimodal elliptical; see Section 4.4.1. As kNN needs to be cross-validated over the entire learning process, it is computationally expensive. Therefore, secondly, we employ the $DD\alpha$-procedure, which is a very fast heuristic; see Section 4.4.2. Although its theoretical convergence is not guaranteed (see Section 2.4), the $DD\alpha$-classifier performs very well in applications.

4.4.1 kNN-classification on the DD-plot

When applied to multivariate data, (under mild assumptions) kNN is known to be a consistent classifier. By multiplying all distances with the inverse covariance matrix of the pooled data an affine invariant version is obtained. In our procedure we employ an affine-invariant kNN classifier on the DD-plot. It will be shown (Appendix 1) that, if the underlying distribution of each class is strictly unimodal elliptical, the kNN classifier, operating on the DD-plot, achieves asymptotical optimal Bayes risk.

Given a function $\mathbf{x}_0 \in \mathcal{F}$ to be classified, we represent $\mathbf{x}_0$ (according to Section 4.3) as $\mathbf{y}_0 \in \mathbb{R}^{L+S}$ and then as $\mathbf{z}_0 = \big(D^{L+S}(\mathbf{y}_0|Y_0), D^{L+S}(\mathbf{y}_0|Y_1)\big)$. According to kNN on the DD-plot $\mathbf{x}_0$ is classified as follows:

$$class(\mathbf{x}_0) = I\left(\sum_{i=1}^{m} I(\mathbf{z}_i \in R_{\mathbf{z}_0}^{\beta(k)}) < \sum_{i=m+1}^{m+n} I(\mathbf{z}_i \in R_{\mathbf{z}_0}^{\beta(k)})\right), \tag{4.7}$$

where $I(S)$ denotes the indicator function of a set S, and $R_{\mathbf{z}_0}^{\beta(k)}$ is the neighborhood of $\mathbf{z}_0$ defined by the k-closest observations, i.e. having the smallest L_∞ distances $\|\mathbf{z} - \mathbf{z}_0\|_\infty$. In other words, the k-neighborhood $R_{\mathbf{z}_0}^{\beta(k)}$ of $\mathbf{z}_0$ in the DD-plot is the smallest rectangle centered at $\mathbf{z}_0$ that contains k training observations. (Note that any norm on LS-space could be used.) In applications k has to be chosen, usually by means of cross-validation.

4.4.2 $DD\alpha$-classification

The second classification approach is the $DD\alpha$-classifier, introduced in Chapter 2. It uses a projective-invariant method called the α-procedure (Vasil'ev & Lange, 1998), which is a heuristic classification procedure that iteratively decreases the empirical risk. We employ three depths (Mahalanobis, spatial, and projection depths), which are positive on the whole $\mathbb{R}^{L+S}$ and thus do not produce outsiders. It is known that the $DD\alpha$-classifier is asymptotical Bayes-optimal for the location-shift model (see Theorem 2.1) and performs well for broad classes of simulated distributions and a wide variety of real data (see Sections 2.5–2.7 and Section 3.4). The main advantage of the $DD\alpha$-classifier is its high training speed, as it contains the α-procedure, which, on the DD-plot, has the quick-sort complexity $O\big((m+n)\log(m+n)\big)$ and proves to be very fast. The separating polynomial is constructed by space extensions of the DD-plot (which is of low dimension q) and cross-validation.

4.5 Choosing the dimensions L and S

Clearly, the performance of our classifier depends on the choice of L and S, which has still to be discussed. In what follows we assume that the data is given as functional values at a (usually large) number of discretization points. Let M denote the number of these points.

Delaigle *et al.* (2012) propose to perform the classification in a finite-dimensional space that is based on a subset of discretization points selected to minimize the average error. But these authors do not offer a construction rule for the task but rely on multi-layer leave-one-out cross-validation, which is very time-consuming. Having recognized this problem they suggest some time-saving modifications of the cross-validation procedure. Clearly, the computational load is then determined by the cross-validation scheme used. It naturally depends on the size of the data sample, factored with the training time of the finite-dimensional classifier, which may also include parameters to be cross-validated.

The Delaigle *et al.* (2012) approach (abbreviated *crossDHB* for short) suggests a straightforward procedure for our problem of constructing an LS-space: allow for a rich initial set of possible pairs (L, S) and then use cross-validation in selecting a pair that (on an average) performs best. The initial set shall consist of all pairs (L, S), say, satisfying $2 \le L + S \le M/2$. (Other upper bounds may be used, e.g. if $M/2$ exceeds the number of observations.) In addition, a dimension-reducing technique like PCA or factor analysis may be in place. But this cross-validation approach (*crossLS* for short), similar to the one of Delaigle *et al.* (2012), is still time-consuming; see Section 4.6.4 for computing times. The problem of selecting an appropriate pair $(L^\star, S^\star)$ remains challenging.

In deciding about L and S, we will consider the observed performance of the classifier as well as some kind of worst-case performance. Fortunately, the conservative error bound of Vapnik and Chervonenkis (Vapnik & Chervonenkis, 1974), see also Devroye *et al.* (1996), provides some guidance. The main idea is to measure how good a certain location-slope space can be at all, by referring to the *worst-case empirical risk of a linear classifier* on the training classes. Clearly, the class of linear rules is limited, but may be regarded as an approximation. Also, this limitation keeps the deviation of $\Delta\epsilon$ from empirical risk small and allows for its implicit comparison with the empirical risk ϵ itself. Moreover, in choosing the dimension of the location-slope space we may adjust for the required complexity so that finally the separation rule is close to a linear rule.

Here, *linear discriminant analysis* (LDA) is used for the linear classification. Although other approaches like perceptron, regression depth, support vector machine, or α-procedure can be employed instead, we choose LDA as it behaves very similar to the depth-based classifiers we use. In fact, LDA is the simplest discriminator of the plug-in type, that is, it delivers a linear separation rule and can be regarded as a Bayes classifier with a plugged-in density estimator.

Given Y_0 and $Y_1 \in \mathbb{R}^{L+S}$, let $\mathcal{N}$ be the number of all different separations of $Y_0 \cup Y_1$ into two subsets, which is achieved by using any classification rule from some class of rules $\mathcal{L}$. Then, if a separation rule yields empirical risk ϵ, it will, with some reliability η, yield an error rate not worse then $\epsilon_{max} = \epsilon + \Delta\epsilon$. It holds (Theorem 5.1 in Vapnik & Chervonenkis (1974))

$$\Delta\epsilon = \sqrt{\frac{\ln \mathcal{N} - \ln \eta}{2(m+n)}}. \tag{4.8}$$

Now, let $\mathcal{L}$ be the class of linear rules. Unlike Vapnik & Chervonenkis (1974), we use a tighter bound on $\mathcal{N}$, which looks theoretically not as nice but is still computationally convenient. Let $C(N, d) = 2\sum_{k=0}^{d-1}\binom{N-1}{k}$ be the number of possible different separations of N points in $\mathbb{R}^d$,

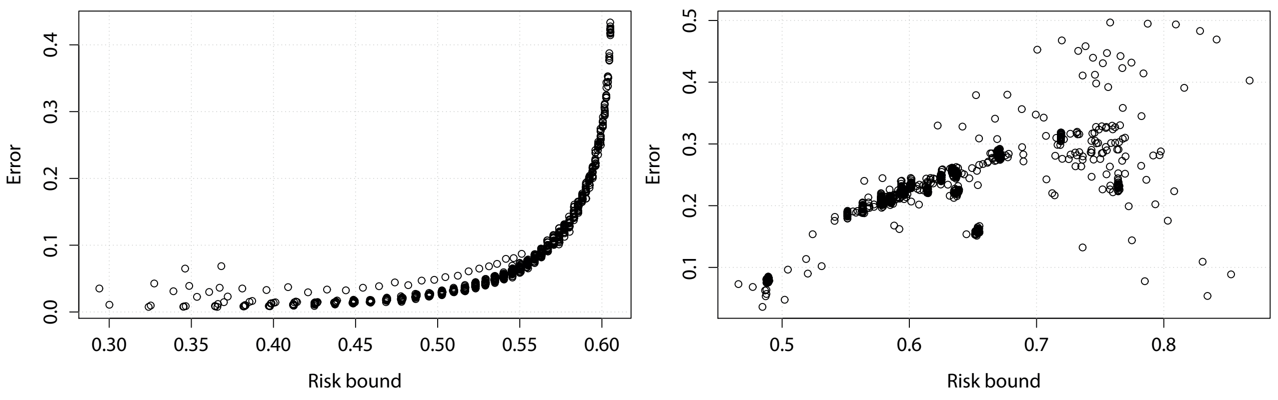

Figure 4.4: *Average misclassification error (ordinate) to maximal risk (abscissa) of the DDα-classifier based on Mahalanobis depth for Model 1 (left) and Model 2 (right) of Cuevas et al. (2007).*

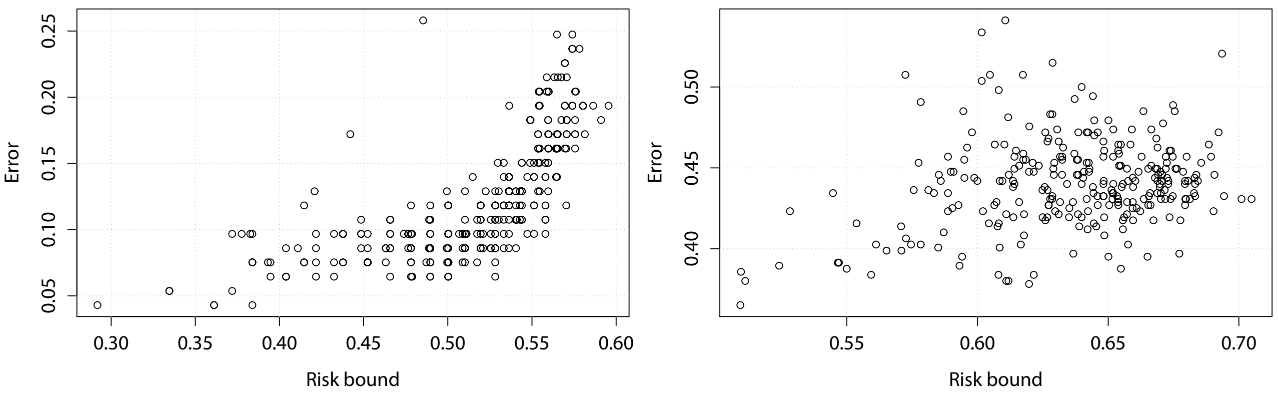

Figure 4.5: *Average misclassification rate (ordinate) to maximal risk (abscissa) of the DDα-classifier based on Mahalanobis depth for growth data (left) and medflies data (right).*

achievable by a hyperplane containing the origin. Avoiding the origin restriction by introducing an additional coordinate and with $\eta = \frac{1}{m+n}$ we obtain

$$\epsilon_{max} = \epsilon + \sqrt{\frac{\ln C(m+n, L+S+1) + \ln{(m+n)}}{2(m+n)}}. \tag{4.9}$$

Here, ϵ refers to empirical risk, while the second term penalizes the dimension, which balances fit and complexity.

To find out whether the proposed bound really helps in finding proper dimensions L and S, we first apply our approach to two simulated data settings of Cuevas *et al.* (2007), called "Model 1" and "Model 2" (both $M = 51$). The data generating process of Cuevas *et al.* (2007) is described in Section 4.6.2 below. We determine L and S, use the Mahalanobis depth to construct a DD-plot and apply the $DD\alpha$-classifier, which is abbreviated in the sequel as $DD\alpha$-M. For each pair (L, S) with $L + S \geq 2$ and $L + S \leq 26$, the risk bound ϵ_{max} and the average classification error (ACE) are calculated by averaging over 100 takes and plotted in Figure 4.4. Note that these patterns look similar when spatial or projection depth is used. Further, for the two benchmark data problems, we estimate ACE by means of leave-one-out cross-validation (see Figure 4.5). Here all (L, S)-pairs are considered with $L + S \geq 2$ and $L + S \leq 16$.

As expected, Figures 4.4 and 4.5 largely support the statement "the less ϵ_{max} the smaller the ACE". Although for the challenging *medflies data* (Figure 4.5, right) the plot remains a fuzzy scatter, it still guides us in configuring the location-slope space, as it is affirmed by our experimental results in Section 4.6.3 below. Computing ϵ_{max} involves a single calculation of the LDA-classifier (*viz.* to estimate ϵ), which is done in negligible time. Then, not taking the (L, S)-pair with the smallest ϵ_{max}, but cross-validating over a bunch of those, gives best results. This technique is employed here for space building. We abbreviate it as *VCcrossLS*.

4.6 Experimental comparisons

To evaluate the performance of the proposed methodology we compare it with several classifiers that operate either on the original data or in a location-slope space. After introducing those competitors (Section 4.6.1) we present a simulation study (Section 4.6.2) and a benchmark study (Section 4.6.3), including a discussion of computation loads (Section 4.6.4). Implementation details of the experiments are provided in Appendix 2.

4.6.1 Competitors

The new classifiers are compared with six classification rules: linear (LDA) and quadratic (QDA) discriminant analysis, k-nearest-neighbors (kNN) classification and the maximum-depth classification (employing the three depth notions), all operating in a properly chosen location-slope space that is constructed with the bounding technique *VCcrossLS* of Section 4.5. (One may argue that the ϵ_{max}-based choice of (L, S) is not generally suited for kNN, but it delivers comparable results in reasonable time, much faster than cross-validation over all (L, S)-pairs.) Also, the six classifiers mentioned above, together with the two new ones (with all three depths), are used when the dimension of the location-slope space is determined by non-restricted cross-validation *crossLS*. For further comparison, all 12 classifiers are applied in the finite-dimensional space constructed according to the methodology *crossDHB* of Delaigle *et al.* (2012).

LDA and QDA are calculated with classical moment estimates, and priors are estimated by the class portions in the training set. We include kNN in our competitors as it is Bayes-risk consistent in the finite-dimensional setting and generally performs very well in applications. The *kNN-classifier* is applied to the location-slope data in its affine invariant form. It is then defined as in (4.7), but with the Mahalanobis distance (determined from the pooled data) in place of the L_∞-distance. k is selected by cross-validation.

As further competitors we consider three *maximum depth classifiers*. They are defined as

$$class(\mathbf{x}) = \operatorname*{argmax}_i \pi_i D(\mathbf{y}|Y_i)\,, \tag{4.10}$$

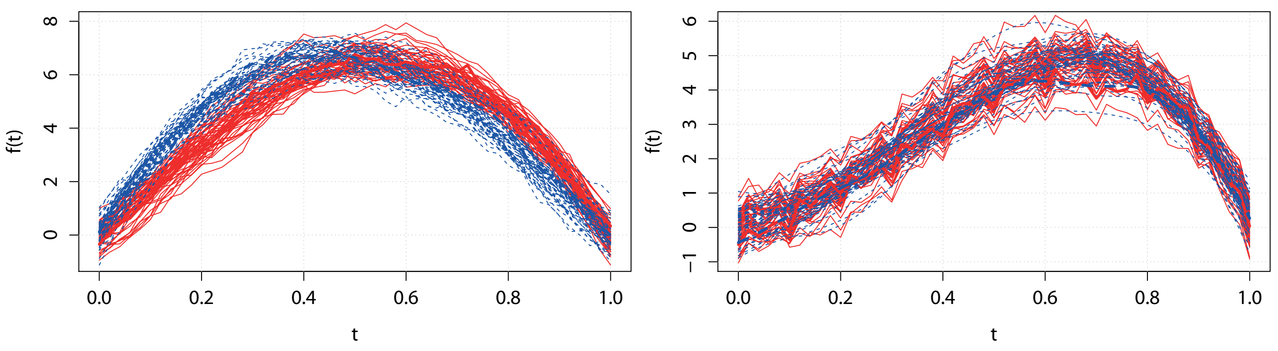

Figure 4.6: *Synthetic data: Model 1 (left) and Model 2 (right) of Cuevas et al. (2007).*

with D being either Mahalanobis depth D^{Mah} (4.1), or spatial depth D^{Spt} (4.2), or projection depth D^{Prj} (4.3). π_i denotes the prior probability for class i. The priors are estimated by the class portions in the training set. This classifier is asymptotically optimal regarding Bayes risk if the data comes from an elliptical location-shift model with known priors. For technical and implementation details the reader is referred to Appendix 2.

4.6.2 Simulation settings and results

Next we explore our methodology by applying it to the simulation setting of Cuevas *et al.* (2007). Their data are generated from two models, each having two classes. The first model is *Model 1*:

$$\begin{aligned} X_0 &= \{\mathbf{x}^0(t)|\mathbf{x}^0(t) = 30(1-t)t^{1.2} + u(t)\}\,, \\ X_1 &= \{\mathbf{x}^1(t)|\mathbf{x}^1(t) = 30(1-t)^{1.2}t + u(t)\}\,, \end{aligned}$$

where $u(t)$ is a Gaussian process with $E[u(t)] = 0$ and $Cov[u(s), u(t)] = 0.2e^{-\frac{1}{0.3}|s-t|}$, discretized at 51 equally distant points on $[0, 1]$ ($M = 51$), see Figure 4.6 (left) for illustration. The functions are smooth and differ in mean only, which makes the classification task rather simple.

We take 100 observations (50 from each class) for training and 100 (50 from each class) for evaluating the performance of the classifiers. Training and classification are repeated 100 times to get stable results. Figure 4.7 (left) presents boxplots (over 100 takes) of error rates of twelve classifiers applied after mapping the data to properly constructed finite-dimensional spaces. The top panel refers to a location-slope space, where L and S are selected by Vapnik-Chervonenkis restricted cross-validation (*VCcrossLS*), the middle panel to a location-slope space whose dimensions are determined by mere, unrestricted cross-validation (*crossLS*), the bottom panel to the finite-dimensional argument subspace constructed by the componentwise method *crossDHB* of (Delaigle *et al.*, 2012). The classifiers are: linear (LDA) and quadratic (QDA) discriminant analysis, k-nearest neighbors classifier (kNN), maximum depth classifier with Mahalanobis (MD-M), spatial (MD-S) and projection (MD-P) depth, DD-plot classifier with kNN rule based on L_∞ distance (DDk-M, DDk-S, DDk-P), and $DD\alpha$-classifier ($DD\alpha$-M, $DD\alpha$-S, $DD\alpha$-P), both with the three depths, correspondingly. The last approach (*cross-*

DHB) has not been combined with the projection depth for two reasons: First, performing the necessary cross-validations with the projection depth becomes computationally infeasible; for computational times see Table 4.14 in Appendix 3. Second, the quality of approximation of the projection depth differs between the tries, and this instability is possibly misleading when choosing the optimal argument subspace, thus yielding rather high error rates; compare, e.g., the classification errors for *growth data*, Table 4.1 in Section 4.6.3.

As expected, all DD-plot-based classifiers, applied in a properly chosen location-slope space, show highly satisfactory performance, which is in line with the best result of Cuevas *et al.* (2007). The location-slope spaces that have been selected for the different classifiers, *viz.* the corresponding (L, S)-pairs, do not much differ; see Table 4.3 in Appendix 3. The picture remains the same when the location-slope space is chosen using unrestricted cross-validation *crossLS*, which yields no substantial improvement.

On the other hand, classifiers operating in an optimal argument subspace (*crossDHB*) are outperformed by those employed in the location-slope space (*crossLS, VCcrossLS*), although their error rates are still low; see Figure 4.7 (left). A plausible explanation could be that differences between the classes at each single argument point are not significant enough, and a bunch of them has to be taken for reasonable discrimination. But the sequential character of the procedure (as discussed in the Introduction and in Appendix 2) prevents from choosing higher dimensions. Most often the dimensions two or three are chosen; see Table 4.11 in Appendix 3. Our procedure appears to be better, as by integrating more information is extracted from the functions, so that they become distinguishable.

Next we consider an example, where averaging comes out to be rather a disadvantage. *Model 2* of Cuevas *et al.* (2007) looks as follows:

$$X_0 = \{\mathbf{x}^0(t)|\mathbf{x}^0(t) = 30(1-t)t^2 + 0.5|sin(20\pi t)| + u(t)\}$$

with $u(t)$ and $M = 51$ as before. X_1 is an 8-knot spline approximation of X_0. See Figure 4.6 (right) for illustration.

The corresponding boxplots of the error rates are depicted in Figure 4.7 (right). The results for individual classifiers are different. When the location-slope space is chosen using Vapnik-Chervonenkis bound (*VCcrossLS*), LDA, kNN and all maximum depth classifiers perform poorly, while the DDk-classifiers with Mahalanobis and spatial depths perform better. The $DD\alpha$-classifiers perform comparably. The last two lack efficiency when employed with projection depth; as seen from Table 4.4 in Appendix 3, the efficient location-slope spaces are of substantially higher dimension (8 and more). Thus, the larger error rates with projection depth are explained by the insufficient number of random directions used in approximating this depth. Also, with projection depth, different to Mahalanobis and spatial depth, less efficient (L, S)-pairs are preferred; see Tables 4.4 and 4.8 in Appendix 3.

Choosing the location-slope space by unrestricted cross-validation (*crossLS*) does not change a lot. The error rates obtained with this location-slope space are larger than those obtained with the synthesized space (*crossDHB*), although with QDA and DD-plot-based

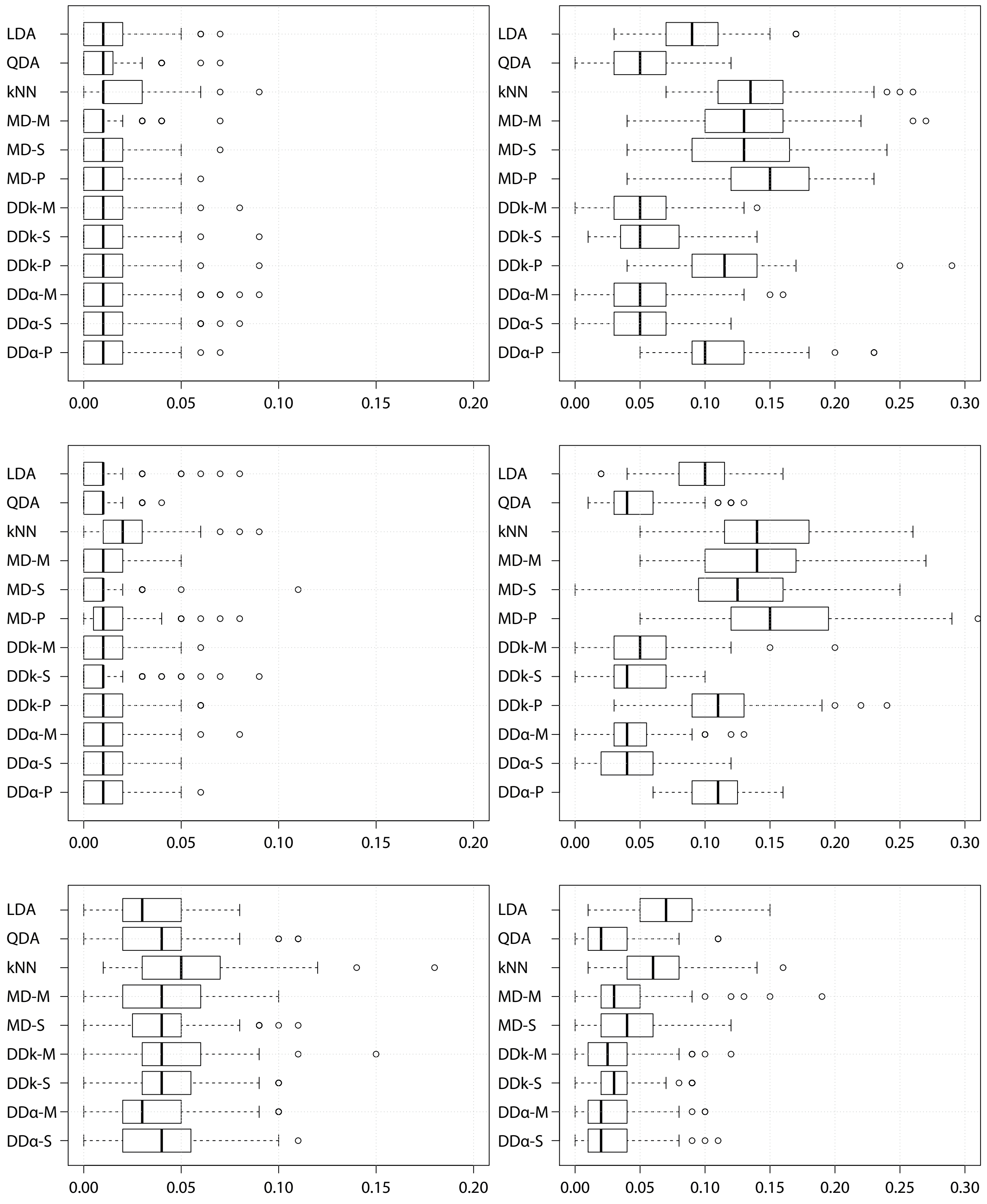

Figure 4.7: *Boxplots of error rates for Model 1 (left) and Model 2 (right), using Vapnik-Chervonenkis bound (VCcrossLS, top), cross-validation (crossLS, middle) and componentwise method (crossDHB, bottom).*

classifiers they stay reasonable. In Model 2, taking several extreme points would be enough for distinguishing the classes, and the finite-dimensional spaces have most often dimension four, and sometimes three. (Note, that all *DD*-plot-based classifiers regard these dimensions

Table 4.1: *Error rates (in %) when selecting a proper location-slope dimension.*

Data set	growth data			medflies data	
Classifier	VCcrossLS	crossLS	crossDHB	VCcrossLS	crossLS
LDA	4.3	4.3	3.23	39.33	41.01
QDA	4.3	5.38	7.53	38.58	42.7
kNN	3.23	3.23	4.3	41.39	44.76
MD-M	5.38	5.38	5.38	38.95	40.64
MD-S	7.53	7.53	7.53	38.2	38.01
MD-P	8.6	7.53	9.68	44.01	43.82
DDk-M	5.38	5.38	5.38	43.26	41.39
DDk-S	4.3	4.3	7.53	39.33	41.95
DDk-P	5.38	3.23	8.6	42.32	41.57
$DD\alpha$-M	5.38	5.38	3.23	37.83	38.58
$DD\alpha$-S	5.38	5.38	6.45	38.95	40.26
$DD\alpha$-P	6.45	6.45	6.45	35.02	35.96

as sufficient, and, together with QDA, deliver best results.) On the other hand, the classifiers operating in some location-slope space select efficient dimension 8 and higher, which is also seen from Tables 4.4 and 4.8.

4.6.3 Comparisons on benchmark data

Now we come back to the two benchmark problems given in the Introduction. The *growth data* have been already analyzed by several authors. López-Pintado & Romo (2006) achieve a best classification error of **16.13** % when classifying with the L_1 distance from the trimmed mean and trimming is based on the generalized band depth with trimming parameter $\alpha = 0.2$. Cuesta-Albertos & Nieto-Reyes (2010) use an average distance weighting with the random Tukey depth and get classification error **13.68** %. Cuevas *et al.* (2007) obtain mean error rate of **9.04** % when using the double random projection depth and **4.04** % with kNN, dividing the sample into 70 training and 23 testing observations over 70 tries. Sguera *et al.* (2014) get error rates of **3.45** % when classifying using kernelized functional spatial depth and choosing kernel parameter by means of cross-validation.

Table 4.1 (columns *growth data*) provides errors, estimated by leave-one-out cross-validation, of different classification techniques. In this, either a proper location-slope space is based on Vapnik-Chervonenkis bound (column *VCcrossLS*), on unrestricted cross-validation (*crossLS*), or an optimal argument subset is chosen by the componentwise technique of Delaigle *et al.* (2012) (*crossDHB*). Note that with *VCcrossLS* classification by kNN is best. It achieves error rate **3.23** %, which means here that only three observations are misclassified. Both DD-plot-based classifiers perform well with all three depths, while the maximum depth classifiers MD-S and MD-P perform worse.

The Vapnik-Chervonenkis restricted cross-validation *VCcrossLS* seems to perform not much worse than the unrestricted cross-validation *crossLS*, and it mostly outperforms the componentwise approach *crossDHB*. The latter is particularly bad when the projection depth is involved. Tables 4.5, 4.9 and 4.13 in Appendix 3 exhibit how often various (L, S)-pairs and dimensions of optimal argument subspace are chosen.

In general, all three space-constructing techniques allow for very low error rates, producing as little as three misclassifications if the classifier is properly chosen. The DD-classifiers on LS-spaces yield at most six misclassifications.

Compared to López-Pintado & Romo (2006) and Cuesta-Albertos & Nieto-Reyes (2010), the better performance of our classifiers appears to be due to the inclusion of average slopes. Observe that the acceleration period starting with approximately nine years discriminates particularly well between girls and boys; see Figure 4.2. Note that also the componentwise method prefers (location) values from this interval.

A much more involved real task is the classification of the *medflies data*. In Müller & Stadtmüller (2005) these data are analyzed by generalized functional linear models. The authors employ logit regression and semi-parametric quasi-likelihood regression; they get errors of **41.76** % and **41.2** %, respectively, also estimated by leave-one-out cross-validation.

We apply all classifiers to the same data in properly constructed location-slope spaces. With our procedure we are able to point out the differential between long- and short-lived flies. Especially, with the $DD\alpha$-classifier based on projection depth an error of **35.02** % is obtained (see Table 4.1, columns captioned *medflies data* for the errors). The role of the derivatives in building the location-slope space is emphasized in Tables 4.6 and 4.10 in Appendix 3, which show the frequencies at which the various (L, S)-pairs are selected. *crossLS* is outperformed in most of the cases. We were not able to compare the componentwise approach *crossDHB* as the computational load is too high. On an average, $DD\alpha$-classifiers perform very satisfactory. LDA and QDA with Vapnik-Chervonenkis bound, and maximum-depth classifiers with Mahalanobis and spatial depth (MD-M, MD-S), also deliver reasonable errors.

Note that, in configuring the location-slope space with *crossLS*, lower errors could be obtained by using finer (e.g. leave-one-out) cross-validations. To make componentwise classification feasible and the comparison fair, we have used only 10-fold cross-validation in all procedures besides kNN. (k in kNN and DD-plot-based kNN has been determined by leave-one-out cross-validation.) For exact implementation details the reader is referred to Appendix 2.

4.6.4 Computational loads

Most methods of functional classification tend to be time consuming because of their needs for preprocessing, smoothing and parameter-tuning, but the literature on such methods usually does not discuss computation times. Nevertheless this is an important practical issue. Our procedure comes out to be particularly efficient due to three main reasons: Firstly, an eventual cross-validation is restricted to very few iterations. Secondly, the depth space, where the final classification is done, has low dimension, which equals the number of classes. Thirdly, the linear interpolation requires no preprocessing or smoothing of the data.

To illustrate this we briefly present the durations of both training and classification phases for the two real data sets and the twelve classification techniques in Table 4.2. (Classification time of a single object is reported in parentheses below.) As the times depend on implementation and platform used, we also report (in square brackets) the numbers of cross-validations done, as this measure is independent of the eventual classification technique. The training times have been obtained as the average over all runs of leave-one-out cross-validation (thus using 92, respectively 533, observations for *growth* and *medflies data*). This comes very close to the time needed to train with the entire data set, as the difference of one observation is negligible. The classification times in Table 2 have been obtained in the same way, i.e. averaging the classification of each single observation over all runs of the leave-one-out cross-validation. The same holds for the number of cross-validating iterations. For the componentwise classifiers the averages have been replaced by the medians for the following reason. The sequential character of the procedure causes an exponential increase of time with each iteration (in some range; see Appendix 2 for implementation details). Therefore, occasionally the total computation time can be outlying. (In our study, once the time exceeded four hours, *viz.* when classifying *growth data* with the DD-plot-based kNN-classifier and projection depth, which required 20450 iterations to cross-validate.) On the other hand, when employing faster classifiers (which usually require stronger assumption on the data) the training phase can take less than two minutes. (This has been pointed by Delaigle *et al.* (2012) as well.)

With *growth data* training times are substantially higher when choosing an (L, S)-pair by unrestricted cross-validation than when the Vapnik-Chervonenkis bound is employed. Though, for the fast maximum depth classifier (with Mahalanobis or spatial depth) computation times appear quite comparable. Application of the componentwise method causes an enormous increase in time (as well as in the number of cross-validations needed). For *medflies data*, as expected, *VCcrossLS* is faster than *crossLS*. We were not able to perform the leave-one-out cross-validation estimation for the componentwise method for this data set, because of its excessive computational load.

See also Table 4.14 for the same experiment regarding simulated data. Here, the projection-depth-based classifiers have not been implemented at all, as they need too much computation time.

4.7 Conclusions

An efficient nonparametric procedure has been introduced for binary classification of functional data. It extends to $q > 2$ classes in the usual way. The procedure consists of a two-step transformation of the original data plus a classifier operating on the unit square. The functional data are first mapped into a finite-dimensional location-slope space and then transformed by a multivariate depth function into the DD-plot, which is a subset of the unit square. Three alternative depth functions are employed for this, as well as two rules for the final classification on the q-dimensional unit cube.

Table 4.2: *Average (median for componentwise classification=crossDHB) training and classification (in parentheses) times (in seconds), and numbers of cross-validations performed (in square brackets), estimated by leave-one-out cross-validation.*

Data set	growth data			medflies data	
Classifier	VCcrossLS	crossLS	crossDHB	VCcrossLS	crossLS
LDA	2.73 (0.002) [5.39]	12.42 (0.0019) [150]	150.78 (0.002) [3110]	9.87 (0.0021) [5]	29.36 (0.0023) [150]
QDA	2.73 (0.0017) [5.39]	12.23 (0.0017) [150]	48.45 (0.0019) [1020]	9.77 (0.0016) [5]	29.12 (0.0027) [150]
kNN	2.91 (0.001) [5.39]	26.38 (0.001) [150]	213.36 (0.0011) [2576]	23.87 (0.0024) [5]	814.46 (0.0028) [150]
MD-M	2.49 (0.0009) [5.39]	2.77 (0.0009) [150]	65.82 (0.0004) [6920]	9.46 (0.0007) [5]	9.86 (0.0007) [150]
MD-S	2.63 (0.0016) [5.39]	5.47 (0.0017) [150]	168.14 (0.0017) [6940]	10.04 (0.0017) [5]	35.89 (0.0018) [150]
MD-P	4.51 (0.0398) [5.39]	78.32 (0.0481) [150]	1795.86 (0.0392) [4699]	19.86 (0.2247) [5]	439.03 (0.2233) [150]
DDk-M	3.06 (0.0012) [5.39]	17.31 (0.0011) [150]	299.3 (0.0012) [2856]	20.53 (0.002) [5]	335.48 (0.002) [150]
DDk-S	3.61 (0.0022) [5.39]	36.31 (0.0021) [150]	631.97 (0.0019) [3135]	24.13 (0.003) [5]	551.36 (0.0033) [150]
DDk-P	6.65 (0.0308) [5.39]	143.42 (0.0308) [150]	3103.57 (0.03) [4115]	41.56 (0.1995) [5]	1145.16 (0.1987) [150]
$DD\alpha$-M	3.42 (0.0009) [5.39]	24.7 (0.0009) [150]	182.03 (0.0011) [1020]	14.98 (0.001) [5]	174.56 (0.001) [150]
$DD\alpha$-S	4 (0.0018) [5.39]	42.49 (0.0017) [150]	860.14 (0.0017) [3135]	18.71 (0.0018) [5]	392.61 (0.0019) [150]
$DD\alpha$-P	7.02 (0.0306) [5.39]	154.24 (0.0309) [150]	2598.38 (0.0298) [3135]	36.04 (0.2) [5]	983.61 (0.1995) [150]

Our procedure outperforms existing approaches on simulated as well as on real benchmark data sets. The results of the DD-plot-based kNN and the $DD\alpha$-procedure are generally good, although, occasionally (cf. Model 2) they are slightly outperformed by the componentwise classification method of Delaigle *et al.* (2012).

As the raw data are linearly interpolated neither information is lost nor spurious one is added. The core of our procedure is the data-dependent construction of the location-slope space. Its dimension $L + S$ is bounded by a Vapnik-Chervonenkis bound. The subsequent depth transformation into the unit hypercube makes the procedure rather robust since the final classification is done on a low-dimensional compact set.

Our use of statistical data depth functions demonstrates the variety of their application and opens new prospects when considering the proposed location-slope space. To reflect the dynamic structure of functional data, the construction of this space, in a natural way, takes levels together with derivatives into account. As it has been shown, efficient information extraction is done via piece-wise averaging of the functional data in its raw form, while the changing of the functions with their argument is reflected by their average slopes.

The finite-dimensional space has to be constructed in a way that respects the important intervals and includes most information. Here, equally spaced intervals are used that cover the entire domain but have possibly different numbers for location and slope. This gives sufficient freedom in configuring the location-slope space. Note, that in view of the simulations as well as the benchmark results, choosing a particular depth function is of limited relevance only. While, depending on given data, different intervals are differently relevant, location and slope may differ in information content as well. The set of reasonable location-slope spaces is enormously reduced by application of the Vapnik-Chervonenkis bound, and the selection is done by fast cross-validation over a very small set. The obtained finite-dimensional space can be augmented by coordinates reflecting additional information on the data, that may be available. Obviously, higher order derivatives can be involved, too. But obtaining those requires smooth extrapolation, which affords additional computations and produces possibly spurious information.

In future research comparisons with existing functional classification techniques as well as the use of other finite-dimensional classifiers on the DD-plot are needed. Refined data-dependent procedures, which size the relevant intervals and leave out irrelevant ones, may be developed to configure the location-slope space. However such refinements will possibly conflict with the efficiency and generality of the present approach.

4.8 Appendix

4.8.1 Appendix 1 – Bayes optimality

Denote by $\mathcal{F}$ the space of real functions, defined on the finite interval $[0, T]$, which are continuous and almost everywhere smooth.

Theorem 4.1. *(LS-transform preserves Bayes optimality).*
Assume that $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{m+n}$ are independently sampled from two stochastic processes that have a.s. paths in $\mathcal{F}$, $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_m \sim \mathfrak{G}_0$ and $\tilde{\mathbf{x}}_{m+1}, \dots, \tilde{\mathbf{x}}_{m+n} \sim \mathfrak{G}_1$. Let $\mathcal{T} = \{t_j | j = 1, ..., \ell\} \in [0, T]$ be some finite set of discretization points, and for each $\tilde{\mathbf{x}} \in \mathcal{F}$ let $\mathbf{x}$ be its linear interpolation based on $\tilde{\mathbf{x}}(t_1), \dots, \tilde{\mathbf{x}}(t_\ell)$, as described in Section 4.3, and construct its transformation to a proper LS-space. Consider a class $\mathcal{C}$ of decision rules $\mathbb{R}^\ell \to \{0, 1\}$ and assume that $\mathcal{C}$ contains

a sequence converging to a Bayes rule. Then there exists a pair (L, S) so that the same class of decision rules operating on the location-slope-transformed data in $\mathbb{R}^{L+S}$ contains a sequence converging to a Bayes rule as well.

The proof is obvious: We have a sample X_0 of independent random vectors $[\mathbf{x}_i(t_1), \dots, \mathbf{x}_i(t_\ell)]$ in $\mathbb{R}^\ell$, all distributed as F_0, $i = 1, \dots, m$, and a second sample X_1, independent from the first, consisting of independent random vectors $[\mathbf{x}_i(t_1), \dots, \mathbf{x}_i(t_\ell)]$ in $\mathbb{R}^\ell$, all distributed as F_1, $i = m+1, \dots, m+n$. Choose $S = 0$ and $L > T/\min_{i=1,\dots,n-1}\{t_{i+1} - t_i\}$. Then the location-slope transform is continuous, linear and injective, hence preserves information.

Consequently, all properties of the above employed classifiers $\mathbb{R}^\ell \to \{0,1\}$ regarding Bayes optimality carry over to our whole procedure if the discretization points are fixed and L is chosen large enough. However note that Theorem 4.1 does not refer to Bayes optimality of classifying the underlying process, but just of classifying the ℓ-dimensional marginal distribution corresponding to $\mathcal{T}$. (In particular, if two processes have equal marginal distributions on $\mathcal{T}$, no discrimination is possible.)

Corollary 4.1. *If $\mathfrak{G}_0$ and $\mathfrak{G}_1$ are Gaussian processes and the priors of class membership are equal, then optimal Bayes risk is achieved by the following rules operating on the LS-transformed data (L being large enough):*

(i) *quadratic discriminant analysis (QDA),*

(ii) *linear discriminant analysis (LDA), if $\mathfrak{G}_0$ and $\mathfrak{G}_1$ have the same covariance function.*

As the processes are Gaussian, we obtain that

$$\begin{aligned}
\mathbf{x}_i(t_1), \dots, \mathbf{x}_i(t_\ell) &\sim (i.i.d.) N(\mu_\mathbf{0}, \mathbf{\Sigma_0}), \quad i = 1, \dots, m, \\
\mathbf{x}_i(t_1), \dots, \mathbf{x}_i(t_\ell) &\sim (i.i.d.) N(\mu_\mathbf{1}, \mathbf{\Sigma_1}), \quad i = m+1, \dots, m+n.
\end{aligned}$$

If L is large enough, our LS transformation preserves all information and thus the standard results of Fisher (see, e.g., Devroye *et al.* (1996), Ch. 4.4) apply; hence Corollary 4.1 holds. The following proposition is taken from Chapter 2.

Proposition 4.2 (Theorem 2.1)**.** *Let F and G be probability distributions in $\mathbb{R}^d$ having densities f and g, and let H be a hyperplane such that G is the mirror image of F with respect to H and $f \geq g$ in one of the half-spaces generated by H. Then based on a 50:50 independent sample from F and G the DDα-procedure will asymptotically yield the linear separator that corresponds to the bisecting line of the DD-plot.*

Due to the mirror symmetry of the distributions in $\mathbb{R}^d$ the DD-plot is symmetric as well. Symmetry axis is the bisector, which is obviously the result of the α-procedure when the sample is large enough. This rule corresponds the Bayes rule. In particular, the requirements of the proposition are satisfied if F and G are mirror symmetric and unimodal.

A stochastic process is mentioned as a *strictly unimodal elliptical process* if all its finite-dimensional marginals are elliptical with the same strictly decreasing radial density.

Corollary 4.2. *Assume that the processes $\mathfrak{G}_0$ and $\mathfrak{G}_1$ are strictly unimodal elliptical and have the same radial density and the same structural matrices, $\mathbf{\Sigma}_0(\mathcal{T}) = \mathbf{\Sigma}_1(\mathcal{T})$ for all $\mathcal{T}$. If priors of class membership are equal (and L is large enough), then the risk of*

(i) *the maximum-depth rule,*

(ii) *the DDα-x rules, $x \in \{M, S, P\}$,*

achieves asymptotically the optimal Bayes risk.

Part (ii) of Corollary 4.2 follows from Theorem 4.1 and Proposition 4.2, part (i) from Theorem 4.1 and Ghosh & Chaudhuri (2005b), who demonstrate that the maximum-depth rule is asymptotically equivalent to the Bayes rule if the two distributions have the same prior probabilities and are elliptical in $\mathbb{R}^\ell$ with only a location difference.

Corollary 4.3. *Assume that the processes $\mathfrak{G}_0$ and $\mathfrak{G}_1$ are strictly unimodal elliptical and have the same radial density. Then the above procedures DDk-x, $x \in \{M, S, P\}$, (with large enough L) are asymptotically Bayes-risk efficient, that is, if $m, n, k \to \infty$, $\frac{k}{m} \to 0$ and $\frac{k}{n} \to 0$, then the risk of each of the three rules DDk-M, DDk-S, and DDk-P converges in probability to the optimal Bayes risk.*

This follows from the above Theorem 4.1 and Theorem 3.5 by Vencálek (2011).

Corollary 4.4. *The above kNN-classifier (with large enough L) is asymptotically Bayes-risk efficient.*

This follows from Theorem 4.1 and the universal consistency of the *kNN* rule; see Devroye *et al.* (1996), Ch. 11.

4.8.2 Appendix 2 – Implementation details

In calculating the depths, μ_Y and Σ_Y for the *Mahalanobis depth* have been determined by the usual moment estimates and similarly, Σ_Y for the *spatial depth.* The *projection depth* has been approximated by drawing 1 000 directions from the uniform distribution on the unit sphere. Clearly, the number of directions needed for satisfactory approximation depends on the dimension of the space. Observe that for higher-dimensional problems 1 000 directions are not enough, which becomes apparent from the analysis of Model 2 in Section 4.6.2. There the location-slope spaces chosen have dimension eight and higher; see also Tables 4.4 and 4.8 in Appendix 3. On the other hand, calculating the projection depth even in one dimension costs something. Computing 1 000 directions to approximate the projection depth takes substantially more time than computing the exact Mahalanobis or spatial depths (see Tables 4.2 and 4.14 in Appendix 3).

LDA and *QDA* are used with classical moment estimates, and priors are estimated by the class portions in the training set. The *kNN-classifier* is applied to location-slope data in its affine invariant form, based on the covariance matrix of the pooled classes. For time reasons, its parameter k is determined by leave-one-out cross-validation over a reduced range, *viz.* $k \in \{1, \ldots, \max\{\min\{10(m+n)^{1/d} + 1, m+n-1\}, 2\}\}$. The α-procedure separating the

DD-plot uses polynomial space extensions with maximum degree three; the latter is selected by cross-validation. To keep the training speed of the depth-based kNN-classifier comparable with that of the $DD\alpha$-classifier, we also determine k by leave-one-out cross-validation on a reduced range of $k \in \{1, \dots, \max\{\min\{10\sqrt{m+n}+1, (m+n)/2\}, 2\}\}$.

Due to *linear interpolation*, the levels are integrated as piecewise-linear functions, and the derivatives as piecewise constant ones. If the dimension of the location-slope space is too large (in particular for inverting the covariance matrix, as it can be the case in Model 2), PCA is used to reduce the dimension. Then ϵ_{max} is estimated and all further computations are performed in the subspace of principal components having positive loadings.

To *construct the location-slope space*, firstly all pairs (L, S) satisfying $2 \leq L + S \leq M/2$ are considered. ($M/2$ amounts to 26 for the synthetic and to 16 for the real data sets.) For each (L, S) the data are transformed to $\mathbb{R}^{L+S}$, and the Vapnik-Chervonenkis bound ϵ_{max} is calculated. Then those five pairs are selected that have smallest ϵ_{max}. Here, tied values of ϵ_{max} are taken into account as well, with the consequence that on an average slightly more than five pairs are selected; see the *growth data* in Table 4.2 and both synthetic models in Table 4.14 of Appendix 3. Finally, among these the best (L, S)-pair is chosen by means of cross-validation. Note that the goal of this cross-validation is not to actually choose a best location-slope dimension but rather to get rid of obviously misleading (L, S)-pairs, which may yield relatively small values of ϵ_{max}. This is seen from Figures 4.4 and 4.5. When determining an optimal (L, S)-pair by *crossLS*, the same set of (L, S)-pairs is considered as with *VCcrossLS*.

In implementing the *componentwise method* of finite-dimensional space synthesis (*crossDHB*) we have followed Delaigle *et al.* (2012) with slight modifications. The original approach of Delaigle *et al.* (2012) is combined with the sequential approach of Ferraty *et al.* (2010). Initially, a grid of equally (Δt) distanced discretization points is built. Then a sequence of finite-dimensional spaces is synthesized by adding points of the grid step by step. We start with all pairs of discretization points that have at least distance $2\Delta t$. (Note that Delaigle *et al.* (2012) start with single points instead of pairs.) The best of them is chosen by cross-validation. Then step by step features are added: In each step, that point that has best discrimination power (again, in the sense of cross-validation) when added to the already constructed set is chosen as a new feature. The resulting set of points is used to construct a neighborhood of combinations to be further considered. As a neighborhood we use twenty $2\Delta t$-distanced points in the second step, and ten in the third; from the fourth step on the sequential approach is applied only.

All our *cross-validations* are ten-fold, except the leave-one-out cross-validations in determining k with both kNN-classifiers. Of course, partitioning the sample into ten parts only may depreciate our approach against a more comprehensive leave-one-out cross-validation. We have chosen it to keep computation times of the *crossDHB* approach (Delaigle *et al.*, 2012) in practical limits and also to make the comparison of approaches equitable throughout our study.

The calculations have been implemented in an *R-environment*, based on the R-package "ddalpha" (Chapter 3), with speed critical parts written in C++. The R-code implementing our methodology as well as that performing the experiments can be obtained upon request from the authors. In all experiments, one kernel of the processor Core i7-2600 (3.4 GHz) having enough physical memory has been used. Thus, regarding the methodology of Delaigle *et al.* (2012) our implementation differs from their original one and, due to its module-based structure, may result in larger computation times. For this reason we provide the number of cross-validations performed; see Tables 4.2 and 4.14 of Appendix 3. The comparison appears to be fair, as we always use ten-fold cross-validation together with an identical set of classification rules in the finite-dimensional spaces.

4.8.3 Appendix 3 – Additional tables

Table 4.3: *Frequency (in %) of selected location-slope dimensions using the Vapnik-Chervonenkis bound; Model 1.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(2,1)	65	64	55	58	63	54	65	63	47	49	51	50
(3,1)	15	23	29	34	23	29	16	24	34	29	26	31
(2,0)	14	7	7	3	9	11	7	3	5	11	12	10
(2,2)	1	2	5	0	1	3	4	2	6	3	1	6
(3,0)	2	1	4	1	1	1	2	1	3	4	5	3
Others	3	3	0	4	3	2	6	7	5	4	5	0

Table 4.4: *Frequency (in %) of selected location-slope dimensions using the Vapnik-Chervonenkis bound; Model 2.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(5,4)	41	51	27	39	31	0	40	37	0	37	37	1
(0,8)	19	23	24	31	40	20	13	12	30	16	10	23
(1,8)	4	4	13	9	10	23	8	6	18	3	4	21
(2,8)	8	8	3	1	0	7	15	14	4	12	8	4
(4,8)	1	4	5	1	1	6	6	9	4	8	6	6
(3,8)	7	4	3	0	1	8	0	3	6	7	5	4
(6,4)	2	2	4	3	3	0	7	6	0	8	12	0
(6,8)	3	2	1	1	0	4	1	1	6	0	3	4
(5,8)	2	0	1	0	1	3	2	3	7	0	4	2
(10,8)	2	1	3	2	1	5	0	0	5	0	2	2
(7,8)	2	0	0	0	0	0	2	0	6	3	0	5
(12,8)	1	0	1	1	0	5	1	1	3	0	0	3
(18,8)	1	0	0	0	0	2	0	0	0	0	0	5
Others	7	1	15	12	12	17	5	8	11	6	9	20

Table 4.5: *Frequency (in %) of selected location-slope dimensions using the Vapnik-Chervonenkis bound; growth data.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(0,2)	87.1	94.62	64.52	46.24	45.16	23.66	27.96	3.23	41.94	72.04	66.67	36.56
(1,2)	12.9	5.38	35.48	6.45	19.35	0	17.20	89.25	32.26	24.73	26.88	39.78
(0,3)	0	0	0	24.73	4.30	68.82	27.96	2.15	3.23	0	0	1.08
(2,2)	0	0	0	21.51	30.11	1.08	1.08	3.23	11.83	1.08	0	17.20
(1,3)	0	0	0	0	1.08	1.08	23.66	2.15	9.68	0	5.38	4.30
(4,0)	0	0	0	1.08	0	0	2.15	0	1.08	2.15	1.08	1.08
(0,4)	0	0	0	0	0	5.38	0	0	0	0	0	0

Table 4.6: *Frequency (in %) of selected location-slope dimensions using the Vapnik-Chervonenkis bound; medflies data.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(1,1)	1.87	0	3.18	0.75	93.45	80.34	11.42	8.05	70.04	74.91	87.64	98.13
(2,1)	84.27	0	63.48	0	0	0	64.42	79.96	0.19	24.53	12.17	0
(0,2)	9.18	100	32.58	99.25	6.55	18.54	23.97	10.11	27.34	0.56	0.19	1.31
(1,2)	1.12	0	0	0	0	1.12	0.19	0.19	2.43	0	0	0.56
(3,1)	3.56	0	0.75	0	0	0	0	1.69	0	0	0	0

Table 4.7: *Frequency (in %) of selected location-slope dimensions using cross-validation; Model 1.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(2,1)	47	49	34	43	41	31	45	42	35	50	42	31
(3,1)	14	21	17	11	22	15	13	14	14	17	12	15
(4,1)	8	1	7	6	4	6	1	7	3	3	7	8
(2,0)	6	2	3	5	5	1	5	1	5	3	4	4
(2,2)	0	1	9	1	2	4	6	5	1	5	4	2
(5,1)	1	0	2	6	4	3	2	3	2	1	1	3
(2,3)	2	1	6	2	4	0	1	4	0	2	3	0
(3,2)	0	4	5	3	1	3	3	2	2	0	2	0
(3,0)	1	0	2	0	0	1	3	0	2	1	1	1
Others	21	21	15	23	17	36	21	22	36	18	24	36

Table 4.8: *Frequency (in %) of selected location-slope dimensions using cross-validation; Model 2.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(5,4)	38	56	27	42	35	0	41	47	0	44	36	0
(0,8)	17	6	14	21	33	6	5	2	11	3	5	14
(6,4)	6	7	11	3	5	0	11	13	0	10	17	0
(10,0)	0	11	0	0	0	3	17	15	0	20	16	0
(2,8)	4	8	4	2	0	8	6	14	1	11	11	8
(1,8)	7	2	6	9	9	10	2	1	11	1	1	10
(3,8)	7	1	9	0	0	8	3	2	8	1	0	7
(4,8)	1	1	2	0	0	9	5	1	8	1	8	6
(5,8)	1	0	2	1	0	9	0	0	4	1	0	9
(6,8)	1	0	3	1	2	8	0	0	3	0	1	6
(7,8)	0	0	1	0	1	5	0	0	8	0	0	6
(9,0)	0	6	0	0	0	0	5	2	0	2	2	0
(5,7)	2	0	2	7	5	0	0	0	0	0	0	0
(16,8)	0	0	1	0	0	5	0	0	5	0	0	4
(12,8)	1	0	1	0	0	1	0	0	7	0	0	4
(8,8)	3	0	1	0	0	2	0	0	6	0	0	1
(17,8)	0	0	0	0	0	2	0	0	3	0	0	8
(13,8)	0	0	1	0	0	3	0	0	2	0	0	6
(10,8)	2	0	0	0	0	0	0	0	6	0	0	3
(18,8)	0	0	0	0	0	5	0	0	3	0	0	1
Others	10	2	15	14	10	16	5	3	14	6	3	7

Table 4.9: *Frequency (in %) of selected location-slope dimensions using cross-validation; growth data.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(0,2)	87.1	93.55	64.52	44.09	44.09	0	25.81	3.23	19.35	68.82	60.22	3.23
(1,2)	12.9	5.38	35.48	6.45	19.35	0	17.20	70.97	12.90	24.73	25.81	17.20
(0,3)	0	0	0	21.51	4.30	29.03	26.88	2.15	1.08	0	0	1.08
(2,2)	0	0	0	11.83	29.03	1.08	1.08	2.15	8.60	1.08	0	7.53
(4,0)	0	0	0	13.98	2.15	0	5.38	1.08	4.30	5.38	10.75	5.38
(1,3)	0	0	0	0	1.08	0	21.51	2.15	5.38	0	2.15	0
(4,2)	0	0	0	0	0	0	0	7.53	1.08	0	0	9.68
(2,3)	0	0	0	0	0	1.08	2.15	5.38	1.08	0	0	1.08
(4,5)	0	0	0	0	0	0	0	0	2.15	0	0	7.53
(4,6)	0	0	0	0	0	0	0	0	0	0	0	6.45
(7,1)	0	0	0	0	0	5.38	0	0	0	0	0	0
Others	0	1.07	0	2.14	0	63.43	0	5.36	44.08	0	1.07	40.84

Table 4.10: *Frequency (in %) of selected location-slope dimensions using cross-validation; medflies data.*

				Max.depth			*DD-kNN*			*DDα*		
(L,S)	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
(1,1)	0	0	0.75	0.56	89.89	77.72	0.37	0	64.98	71.54	79.59	98.31
(0,2)	7.68	0	14.23	91.39	4.12	19.66	1.87	0	24.34	0.37	0.19	1.69
(2,1)	57.87	0	47.00	0	0	0	0	1.87	0	0	0	0
(0,6)	0	0	4.68	0	0	0	32.02	14.23	5.24	0.37	7.49	0
(2,1)	0	0	0	0	0	0	6.55	0	0	22.28	10.86	0
(7,5)	30.90	0	0	0	0	0	0	0	0	0	0	0
(2,6)	0	0	21.16	0	0	0	0.19	0.75	0	0	0	0
(14,2)	0	18.35	0	0	0	0	0.75	2.06	0	0	0.19	0
(2,12)	0	9.36	0	0	0	0	3.75	3.75	0	0	0	0
(13,3)	0	12.55	0	0	0	0	0.94	1.69	0	0.56	0	0
(1,12)	0	0	0	0	0	0	12.17	3.37	0.19	0	0	0
(3,9)	0	0.37	0	0	0	0	7.49	6.74	0	0	0	0
(7,6)	0	3.93	0	0	0	0	3.18	7.30	0	0	0	0
(11,2)	0	8.24	0	0	0	0	3.56	1.50	0	0.56	0	0
(0,5)	0	0	0	7.49	5.99	0	0	0.19	0	0	0	0
(3,12)	0	0	0	0	0	0	4.49	8.61	0	0	0	0
(4,3)	0	0	0.19	0	0	0	0.56	11.80	0	0	0	0
(7,3)	0	10.49	0	0	0	0	0.94	0	0	0	0	0
(4,6)	0	10.86	0	0	0	0	0	0.19	0	0	0	0
(11,5)	0	6.55	0	0	0	0	0.19	1.50	0	0	0	0
(4,10)	0	0	0	0	0	0	2.81	5.06	0	0	0	0
(15,0)	0	6.74	0	0	0	0	0.37	0.19	0	0	0	0
(1,3)	0	0	0.19	0	0	0	0	6.74	0.19	0	0	0
Others	3.55	12.56	11.80	0.56	0	2.62	17.80	22.46	5.06	5.44	0.56	0

Table 4.11: *Frequency (in %) of selected dimensions using componentwise method; Model 1.*

				Max.depth		*DD*-kNN		*DD*α	
dim	LDA	QDA	kNN	Mah.	Spt.	Mah.	Spt.	Mah.	Spt.
2	50	55	45	45	50	46	47	50	64
3	41	38	46	46	43	48	40	42	27
4	9	7	9	9	7	6	13	8	9

Table 4.12: *Frequency (in %) of selected dimensions using componentwise method; Model 2.*

				Max.depth		*DD*-kNN		*DD*α	
dim	LDA	QDA	kNN	Mah.	Spt.	Mah.	Spt.	Mah.	Spt.
3	11	14	14	8	14	20	13	17	18
4	89	86	51	54	60	79	85	83	81
5	0	0	26	35	23	1	2	0	1
6	0	0	9	3	2	0	0	0	0
7	0	0	0	0	1	0	0	0	0

Table 4.13: *Frequency (in %) of selected dimensions using componentwise method; growth data.*

				Max.depth			*DD*-kNN			*DD*α		
dim	LDA	QDA	kNN	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.	Mah.	Spt.	Prj.
2	100.00	92.47	45.16	12.90	3.23	41.94	88.17	50.54	39.78	100.00	96.77	60.22
3	0	6.45	30.11	62.37	60.22	52.69	7.53	44.09	53.76	0	2.15	38.71
4	0	1.08	24.73	24.73	36.56	5.38	4.30	5.38	6.45	0	1.08	1.08

Table 4.14: *Average (median for componentwise classification=crossDHB) training and classification (in parentheses) times (in seconds), and numbers of cross-validations performed (in square brackets), over 100 tries.*

Data set	Model 1			Model 2		
Classifier	VCcrossLS	crossLS	crossDHB	VCcrossLS	crossLS	crossDHB
LDA	7.61 (0.0011) [5.55]	40.59 (0.0011) [375]	547.5 (0.001) [10759.5]	6.83 (0.0012) [8.58]	30.3 (0.0012) [375]	392.16 (0.001) [7648.5]
QDA	7.59 (0.0011) [5.53]	39.44 (0.0011) [375]	503.5 (0.001) [9628]	6.74 (0.0011) [8.32]	29.72 (0.0011) [375]	411.23 (0.001) [8009]
kNN	7.86 (0.0012) [5.27]	133.77 (0.0012) [375]	1263.76 (0.0011) [12996]	7.63 (0.0013) [8.8]	63.16 (0.0014) [375]	489.18 (0.0011) [4886]
MD-M	7.38 (0.0011) [5.46]	7.53 (0.0011) [375]	101.05 (0.001) [10344.5]	6.22 (0.0011) [9.05]	7.11 (0.0011) [375]	58.45 (0.001) [5884.5]
MD-S	7.48 (0.0012) [5.51]	16.98 (0.0012) [375]	259.75 (0.0011) [10113]	6.38 (0.0012) [8.35]	14.36 (0.0012) [375]	169.54 (0.0011) [6518.5]
MD-P	9.58 (0.0017) [5.56]	245.52 (0.0018) [375]	—	10.76 (0.002) [8.96]	195.61 (0.0021) [375]	—
DDk-M	7.99 (0.0012) [5.49]	50.69 (0.0013) [375]	1421.2 (0.0011) [11909.5]	7.26 (0.0013) [9.1]	48.95 (0.0013) [375]	601.2 (0.001) [5491]
DDk-S	8.63 (0.0013) [5.44]	119.84 (0.0014) [375]	2493.52 (0.0012) [10873]	8.71 (0.0014) [9.68]	102.75 (0.0014) [375]	1296.86 (0.0012) [5715.5]
DDk-P	12.16 (0.0016) [5.62]	453.28 (0.0016) [375]	—	14.42 (0.0017) [8.06]	383.15 (0.0018) [375]	—
$DD\alpha$-M	8.13 (0.0012) [5.57]	34.55 (0.0012) [375]	1866.99 (0.001) [10113]	7.93 (0.0012) [9.43]	76.58 (0.0012) [375]	995.68 (0.001) [5182]
$DD\alpha$-S	8.68 (0.0012) [5.51]	104.71 (0.0013) [375]	2840.91 (0.0011) [9774]	9.06 (0.0013) [8.99]	128.62 (0.0013) [375]	1860.12 (0.0012) [6124]
$DD\alpha$-P	12.19 (0.0015) [5.45]	466.76 (0.0016) [375]	—	16.02 (0.0017) [8.92]	410.66 (0.0017) [375]	—

Chapter 5

Exact computation of the Tukey depth

5.1 Introduction

Determining the representativeness of a point within a bunch of data or a probability measure has recently become a desirable task in multivariate analysis. It finds applications in different domains of economics, biology, geography, medicine, cosmology and many others. A statistical data depth is a function determining how centrally a point is located in a data cloud. The upper-level sets it generates – trimmed regions – are set-valued statistics. They trim data w.r.t. degree of centrality. One of the most important depth notions is the Tukey (=halfspace, location) depth. The Tukey depth of a point $\mathbf{z}$ w.r.t. a data cloud X, $D_{Tukey}(\mathbf{z}|X)$, further simply $D(\mathbf{z}|X)$, is defined as the smallest fraction of X to be cut off by a hyperplane through $\mathbf{z}$ so that the remaining points lie in an open halfspace not containing $\mathbf{z}$. For a population distributed with probability measure P the Tukey depth of a point $\mathbf{z} \in \mathbb{R}^d$ is defined as:

$$D(\mathbf{z}|P) = \inf\{P(H) \,|\, H \text{ closed half-space, } \mathbf{z} \in H\}. \tag{5.1}$$

For a data set $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$ it holds:

$$D(\mathbf{z}|X) = \frac{1}{n} \min_{\mathbf{r}\in S^{d-1}} \#\{i \,|\, \mathbf{x}_i'\mathbf{r} \geq \mathbf{z}'\mathbf{r},\, \mathbf{x}_i \in X\}. \tag{5.2}$$

The Tukey depth possesses many desirable properties: it is affine invariant, tends to zero at infinity, is monotone on rays from any deepest point, quasiconcave and upper semicontinuous. By that it satisfies all the postulates imposed on a depth function (Zuo & Serfling, 2000, Dyckerhoff, 2004, Mosler, 2013). If P is absolutely continuous, the Tukey depth is a continuous function of $\mathbf{z}$ achieving maximum value of $\frac{1}{2}$, for angularly symmetric distributions at the center of symmetry (Zuo & Serfling, 2000). If P has no Lebesgue density, the Tukey depth is a discrete function of $\mathbf{z}$ and can have a non-unique maximum. By definition, its empirical version vanishes beyond the convex hull of the data. The Tukey depth determines uniquely an empirical distribution (Struyf & Rousseeuw, 1999, Koshevoy, 2002), taking a finite number of values in the interval from 0 (for the points lying outside the convex hull of the data) to $\frac{1}{2}$, increasing by a multiple of $\frac{1}{n}$. Naturally, it has attractive breakdown properties and converges for a sample from P almost surely to the depth w.r.t. P (Donoho & Gasko, 1992). The Tukey depth has a direct connection to such concepts as regression depth of Rousseeuw & Hubert (1999) and the separating hyperplane delivering smallest empirical risk in supervised classification; it can be extended to functional settings (López-Pintado & Romo, 2011, Claeskens *et al.*, 2014).

The exact calculation of the Tukey depth and its trimmed regions is a computationally challenging task. For this reason, great part of the literature on the Tukey depth concerns its computational aspects. An overview of the most important sources follows in Section 5.1.1. Computing the Tukey depth exactly is a problem of non-polynomial (NP) complexity (Johnson & Preparata, 1978); thus a reasonable approximation can be of use. Some important works on this topic are briefly regarded in Section 5.1.2.

5.1.1 Previous approaches to calculation of the Tukey depth

1975 John W. Tukey introduced a new concept of data picturing, which later crystalized into the Tukey depth (Tukey, 1975). A similar mechanism of cutting by hyperplanes (lines in the two-dimensional case) has also been used for bivariate sign test by Hodges (1955). During the last several decades, a variety of attempts have been taken to compute the Tukey depth and its trimmed regions.

Rousseeuw & Ruts (1996) pioneered in calculating the Tukey depth exactly for bivariate data clouds and constructing its contours (Ruts & Rousseeuw (1996a,b), algorithm ISODEPTH) exploiting the idea of the circular sequence (Edelsbrunner, 1987). Here, the depth of a point is computed with complexity $O(n \log n)$, and a single depth region is constructed with complexity $O(n^2 \log n)$, both essentially determined by the complexity of the QUICK-SORT procedure. Johnson *et al.* (1998) suggest to account for a small subset of points only when constructing the first k depth contours, which yields a better complexity for small k (algorithm FDC). The Tukey depth describes a data cloud by a finite number of depth contours. Miller *et al.* (2003) compute all these with complexity $O(n^2)$, by which the depth of a single point can be calculated with complexity $O(\log^2 n)$ afterwards.

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

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

An interesting issue of depth maintenance when points continuously appear, is handled by Burr *et al.* (2011) for bivariate depth and contours.

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

Also, based on a similar idea, using (similar to Hallin *et al.* (2010)) breadth-first search algorithm to cover $\mathbb{R}^d$, and QHULL to define the direction cones, Liu & Zuo (2014a) compute the Tukey depth exactly. Currently, this seems to be the only known implementation of the Tukey depth computation that works for $d > 3$.

5.1.2 Approximation of the Tukey depth

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

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

The latter work of Chen *et al.* (2013) determines the number of tries needed to achieve required precision exploiting the third approximation method of Rousseeuw & Struyf (1998). The authors also present its generalization by projecting $\mathbf{z}$ and X onto affine spaces of dimension > 1. The approximated depth values are exact in most of the experiments and the approximation errors achieved are never larger than $\frac{2}{n}$.

5.1.3 Proposal

In this work we propose and treat theoretically and practically two algorithms for computing the Tukey depth exactly. First, an algorithm based on application of linear programming only is presented in Section 5.2. It exploits the same idea as Liu & Zuo (2014a) do, but the application of the simplex method and binary coding of the direction cones allow for simplification and acceleration. Then, Section 5.3 suggests an exact, theoretically grounded version of the approximating algorithm used by Rousseeuw & Struyf (1998) and Chen *et al.* (2013). This new version is faster than the first algorithm, thanks to its combinatorial nature. Finer speed differences between the presented algorithms become evident after the experimental study in Section 5.4. Section 5.5 summarizes.

5.2 Computing Tukey depth via linear programming

In this section we present an algorithm for exact calculation of the Tukey depth, that is based on linear programming. Essentially the algorithm proceeds similar to Liu & Zuo (2014a), i.e. it minimizes the univariate depth over all possible permutations of points in the projection onto a univariate direction. All directions yielding the same permutation are gathered in a direction (polyhedral) cone. To determine this cone, Liu & Zuo (2014a) use the vertex-facet enumeration and the QHULL algorithm. We employ exclusively linear programming, coding the direction cones by binary sequences. This increases the speed, and reduces the RAM usage by utilizing the special spread structure of the breadth-first search algorithm. It is also interesting that the algorithm exploits the connection between the Tukey depth and supervised classification in an elegant way.

First, in Section 5.2.1, under the assumption of general position of $\{\mathbf{z}\} \cup X$, we introduce some convenient notation, and give the main theoretical results allowing for linear programming. This is followed by the suggestion of a simplification of the breadth-first search algorithm and time- and RAM-saving lemmas. Section 5.2.2 successively weakens the general position assumption. The algorithm itself is given in Section 5.2.3.

5.2.1 Theoretical background of the algorithm

Given a data sample $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^d$, $d < n$, and a point $\mathbf{z} \in \mathbb{R}^d$, the Tukey depth of $\mathbf{z}$ w.r.t. X shall be calculated. Here, in Section 5.2.1, we assume w.l.o.g. that $\mathbf{z} = \mathbf{0}$ and that $\{\mathbf{z}\} \cup X$ are in general position, i.e., every subset of $k+1$ points spans a subspace of dimension $\min\{k, d\}$. Violation of these assumptions can be compensated by a slight perturbation of the data and a location shift. The Tukey depth is discrete, so such a perturbation can be harmful, as only a small shift of one point can move the border of a depth-trimmed region and thus change the depth value of $\mathbf{z}$ in a non-continuous way. Before performing such a perturbation, we suggest to first check whether $\mathbf{z} \in \text{conv}(X)$ (if not, $D(\mathbf{z}|X) = 0$), and only then calculate the depth of $\mathbf{z}$ using perturbed data. When having enough points in X and specially treating the zero-depth case as above, possible perturbation damage is negligible. Further, in Section 5.2.2,

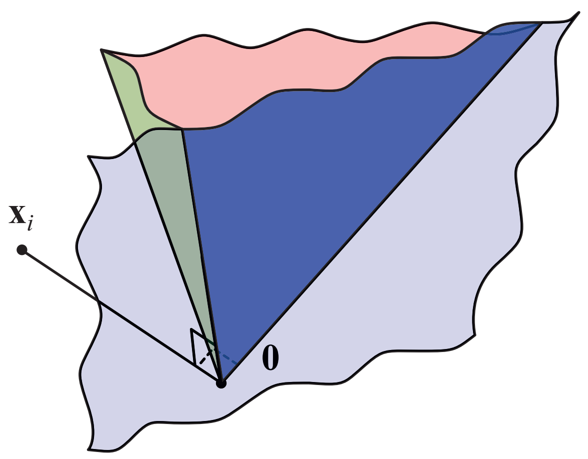

Figure 5.1: *A cone's facet defined by a point.*

we propose a set of modifications for the algorithm to deal with the situation of non-general position.

Consider a direction, i.e., a point on the unit sphere $\mathbf{r} \in S^{d-1}$. It yields an ordered sequence, a permutation π_r on $\mathcal{N} = \{1, ..., n\}$, such that $\mathbf{x}'_{\pi_r(1)}\mathbf{r} \leq \mathbf{x}'_{\pi_r(2)}\mathbf{r} \leq ... \leq \mathbf{x}'_{\pi_r(n)}\mathbf{r}$. If the data are in general position a vector $\mathbf{r}$ can be found such that all inequalities hold strictly $\mathbf{x}'_{\pi_r(1)}\mathbf{r} < \mathbf{x}'_{\pi_r(2)}\mathbf{r} < ... < \mathbf{x}'_{\pi_r(n)}\mathbf{r}$, and $\mathbf{x}'_{\pi_r(i)}\mathbf{r} \neq 0$, $i = 1, ..., n$. Then such $\mathbf{r}$ splits X into two disjoint subsets (by its normal hyperplane $H_\mathbf{r}$ through $\mathbf{0}$ yielding two open halfspaces $H_\mathbf{r}^+$ and $H_\mathbf{r}^-$ in $\mathbb{R}^d$), $X_\mathbf{r}^+$ and $X_\mathbf{r}^-$, containing points with positive, respectively negative, projections on all $\mathbf{r}$ giving the strictly ordered and non-zero projection. Let us call the closure of the set of all $\lambda\mathbf{r}, \lambda \geq 0$, maintaining the same $X_\mathbf{r}^+$ and $X_\mathbf{r}^-$, a *direction cone* C (yielding X_C^+ and X_C^- respectively). This is because its form constitutes an infinite polyhedral cone with the apex in the origin. The entire $\mathbb{R}^d$ is then filled by the set of all direction cones, say $\mathcal{C}(X)$, while each cone $C \in \mathcal{C}(X)$ defines some portion of the sample. Denote this portion $D_C(\mathbf{0}|X) = \frac{1}{n}\min\{\sharp X_C^+, \sharp X_C^-\}$ ($\sharp$ stands for the set's cardinality), then the Tukey depth is $D(\mathbf{0}|X) = \min_{C\in\mathcal{C}(X)} D_C(\mathbf{0}|X)$. Below $\mathcal{C}(X)$ will be mentioned as *cone segmentation.*

The further task is then to go through all such cones and to find the one(s) delivering the smallest $\frac{1}{n}\min\{\sharp X_C^+, \sharp X_C^-\}$, i.e., the Tukey depth. Starting with Mosler *et al.* (2009), when constructing depth regions (see also Paindaveine & Šiman (2012a,b), Bazovkin & Mosler (2012)) and calculating depths (see Liu & Zuo (2014a,b)), the usual way to proceed is: (1) choose an arbitrary direction cone and – using (3) breath-first search – (2) move from each directional cone to the neighbors, covering the entire $\mathbb{R}^d$. Doing that a check whether a direction cone has already been considered is needed, so (4) 'all proceeded direction cones (facets)' are saved, in a binary search tree – a structure maintaining fast search.

Ad (1), the task is trivial: a direction $\mathbf{r} \in S^{d-1}$ maintaining the ordering with strict inequalities $\mathbf{x}'_{\pi_r(1)}\mathbf{r} < \mathbf{x}'_{\pi_r(2)}\mathbf{r} < ... < \mathbf{x}'_{\pi_r(n)}\mathbf{r}$ and no projection coinciding with $\mathbf{z}'\mathbf{r} = 0$ has to be generated. When drawing $\mathbf{r}$ randomly, the theoretical probability of this event $= 1$. As in practice draw concerns only a finite number of digits, it can (though rarely) happen that one needs more than one draw.

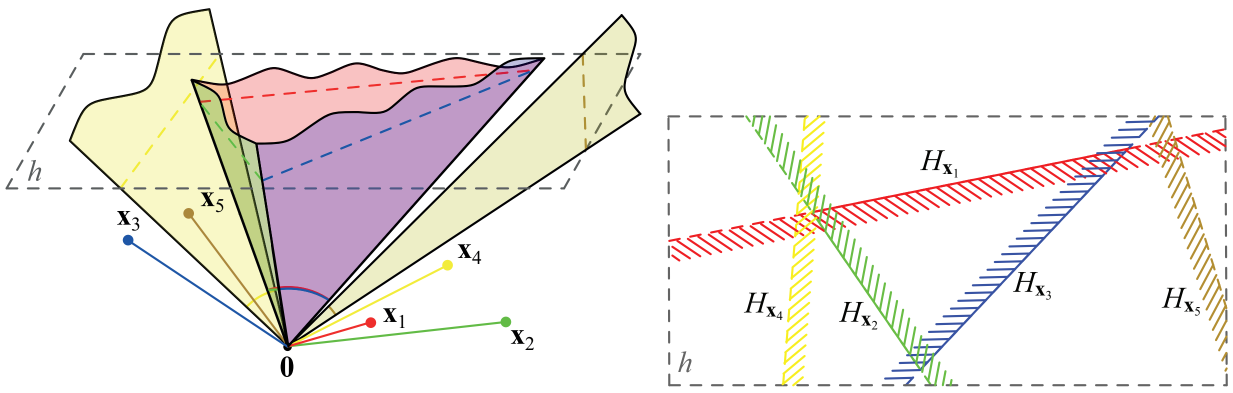

Figure 5.2: *A direction cone in $\mathbb{R}^3$ defined by the points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, halfspaces formed by $\mathbf{x}_4$ and $\mathbf{x}_5$ are not directly involved (left); arbitrary cutting hyperplane h visualizing how the hyperplanes are involved (right).*

Identification of adjacent cones

Ad (2), identifying neighboring direction cones (2a) and transition to each of them if new (2b) is to be done. Let us take a closer look at the direction cone. Two different cones C_1 and C_2 differ in their corresponding sets $(X^+_{C_1}, X^-_{C_1})$ and $(X^+_{C_2}, X^-_{C_2})$. So, if a point $\mathbf{r}$ on S^{d-1} moves from one direction cone to another, projections of one or more points on $\mathbf{r}$ 'migrate passing the origin', i.e., change the sign. Let C_1 and C_2 be two cones, such that a direct (i.e., not crossing other cones) rotational movement of $\mathbf{r}$ from C_1 to C_2 (and vice-versa) is possible. That means that C_1 and C_2 have an intersection of affine dimension between 1 and $d-1$. If a transition of $\mathbf{r}$ from C_1 to C_2 involves changing the halfspace (from $H^+_{\mathbf{r}}$ to $H^-_{\mathbf{r}}$ or vice versa) by one point $\in X$ only (correspondingly changing the sign of the projection on $\mathbf{r}$), then C_1 and C_2 intersect in affine dimension $d-1$. This intersection constitutes the cones' common facet. We call these two cones *neighboring cones*.

So, the transition of a single point $\mathbf{x}_i \in X$ from $X^+_{C_1}$ to $X^-_{C_2}$ means traversing from C_1 to a neighboring cone C_2 through a facet, and thus the facet is defined by this point $\mathbf{x}_i$, see Figure 5.1. Naturally, given a cone C, any facet of C lies in a hyperplane, normal to the line, connecting a point $\in X$ with $\mathbf{z} = \mathbf{0}$, as it is shown in Figure 5.1, but not each point $\in X$ generates a facet of C, see Figure 5.2. A direction cone C is defined by the intersection of closed halfspaces $\{\mathbf{y}|\mathbf{y}'(\mathbf{x}-\mathbf{z}) \geq 0, \mathbf{x} \in X^+_C\}$ and $\{\mathbf{y}|\mathbf{y}'(\mathbf{x}-\mathbf{z}) \leq 0, \mathbf{x} \in X^-_C\}$. Hyperplanes directly involved in the intersection (generated by points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$) contain the cone's facets and those outside (generated by points $\mathbf{x}_4, \mathbf{x}_5$) do not (see Figure 5.2). Thus, given a direction cone, a natural question is which points $\in X$ define its facets, and which do not. This is summarized in Theorem 5.1.

Theorem 5.1. *Given a finite set of points $X \in \mathbb{R}^d$, assume that $\{\mathbf{0}\} \cup X$ are in general position, and let C be a direction cone. Also, for a point $\mathbf{x} \in X$ let $X_{H_{\mathbf{x}}}$ be the orthogonal projection of X onto the hyperplane $H_{\mathbf{x}}$ normal to $\mathbf{x}$, and $X^+_{H_{\mathbf{x}},C}$ and $X^-_{H_{\mathbf{x}},C}$ be the two subsets of $X_{H_{\mathbf{x}}} \setminus \{\mathbf{0}\}$ corresponding to X^+_C and X^-_C, respectively. Then:*

(i) $H_{\mathbf{x}}$ contains a facet of C if and only if $X^+_{H_{\mathbf{x}},C}$ and $X^-_{H_{\mathbf{x}},C}$ are linearly strictly separable through $\mathbf{0}$, i.e., can be separated by a (d-2)-hyperplane $\subset H_{\mathbf{x}}$ containing the origin and no points from $X_{H_{\mathbf{x}}} \setminus \{\mathbf{0}\}$,

(ii) if $\mathbf{r} \in S^{d-1}$ moves from C to a neighboring direction cone, there exists exactly one point $\mathbf{x} \in X$, which defines the facet, so that the projection of $\mathbf{x}$ on the line through $\mathbf{r}$ changes sign.

Proof. (i) "$\Longrightarrow$": If $\mathbf{x} \in X$ defines a facet of C, then $H_{\mathbf{x}}$ contains this facet, and thus there should exist some direction $\mathbf{r} \in S^{d-1} \cap H_{\mathbf{x}}$ such that X^+_C and X^-_C projected onto $\mathbf{r}$ maintain their signs, except for the single point $\mathbf{x}$ being projected into $\mathbf{0}$. So, these projections of $X^+_C \setminus \{\mathbf{x}\}$ and X^-_C (or X^+_C and $X^-_C \setminus \{\mathbf{x}\}$) are separated in $H_{\mathbf{x}}$ by the hyperplane normal to $\mathbf{r}$ through $\mathbf{0}$.

"$\Longleftarrow$": Strict linear separability of $X^+_{H_{\mathbf{x}},C}$ and $X^-_{H_{\mathbf{x}},C}$ through $\mathbf{0}$ means that there exists some $\mathbf{r} \in S^{d-1} \cap H_{\mathbf{x}}$, such that $\mathbf{x}'\mathbf{r} > 0 \ \forall \ \mathbf{x} \in X^+_{H_{\mathbf{x}},C}$ and $\mathbf{x}'\mathbf{r} < 0 \ \forall \ \mathbf{x} \in X^-_{H_{\mathbf{x}},C}$. Then a slight infinitesimal rotation of $\mathbf{r}$ towards (and inside of) the cone does not cause projections to change sign, and thus maintains X^+_C and X^-_C.

(ii) Let $\mathbf{x} \in X$ define a common facet of C_1 and C_2. It follows from (i) that, when $\mathbf{r}$ moves from C_1 to C_2, the projection of $\mathbf{x}$ on $\mathbf{r}$ changes sign. The question is: which points (potentially) can also change sign? We first distinguish the special case that points lie on the line through $\mathbf{0}$ and $\mathbf{x}$. Generally, we consider the case that points from $X^+_{H_{\mathbf{x}},C}$ and $X^-_{H_{\mathbf{x}},C}$ lie in the $(d-2)$-dimensional space separating them, i.e., when no $(d-2)$-hyperplane containing $\mathbf{0}$ but not containing any points from them can separate $X^+_{H_{\mathbf{x}},C}$ from $X^-_{H_{\mathbf{x}},C}$.

In the first case, at least three points lie on one line, so the assumption of general position of $\{\mathbf{0}\} \cup X$ is violated. In the second case, if in a $(d-1)$-dimensional linear subspace (namely $H_{\mathbf{x}}$) two sets can only be separated by a $(d-2)$-hyperplane containing $\mathbf{0}$ and also points from these sets, then at least d points from $X_{H_{\mathbf{x}}} \setminus \{\mathbf{0}\}$ lie in this separating $(d-2)$-hyperplane. But this means that, together with $\mathbf{x}$, $d+1$ points $\in X$ should lie in a $(d-1)$-hyperplane. This violates as well the general position assumption of X.

□

Optimization of the breadth-first search algorithm

Having addressed (2a), (2b) can be drawn trivially from Theorem 5.1. From the first part, one can easily find out which points define the cone's facets. Then, following the second part, moving the direction $\mathbf{r}$ to the neighboring cone traversing their common facet means changing the sign of the projection on $\mathbf{r}$ of the point which defines this facet.

Ad (3), we use the results from above to describe the *breadth-first search* algorithm: generate an initial direction cone (ad (1)) and move to the neighboring cones (ad (2)), calculating the depth in each of them, till the entire $\mathbb{R}^d$ is covered. Note, that these operations are general for many depth-calculating algorithms (Liu & Zuo (2014a,b)), algorithms constructing trimmed regions (Mosler *et al.* (2009), Paindaveine & Šiman (2012a,b), Bazovkin & Mosler (2012)), as well as arbitrary algorithms exploiting cone segmentation of $\mathbb{R}^d$.

Algorithm 5.1. *The breadth-first search algorithm on a cone segmentation of $\mathbb{R}^d$ proceeds as in the following steps:*

a) Draw an initial cone and store it in a queue.

b) Pop one cone from the head of the queue, process it, remember it, and for each of its neighboring cones do:

  c) If the cone has not been processed till now push it into the tail of the queue.

d) If the queue is not empty, go to Step b.

Further, let us introduce the notion of the cone's *generation*, a number given to each cone (in Step c) when it is processed. The initially drawn cone (in Step a) is given the initial number, say 1. The generation can be thought of as the 'depth' of the current searching path of the algorithm. When covering a cone segmentation, only a few generations have to be remembered (i.e., held in the memory).

In algorithms of the combinatorial nature, the number of the direction cones to be processed usually depends exponentially on n and d, so that even for moderately n and d it can be enormously large. All these processed cones (or their facets) are usually to be stored (in the RAM), to check whether the current one has already been processed. The next observation and the following lemma give some insights how to reduce the number of the cones to be stored. The observation 5.1 is applicable to most algorithms on the convex polytopes including Mosler *et al.* (2009), Paindaveine & Šiman (2012a,b), Bazovkin & Mosler (2012), Liu & Zuo (2014a,b). It follows from the stepwise nature of the algorithm and the symmetry of the cone-neighborhood relationship.

Observation 5.1. *When covering a cone segmentation of $\mathbb{R}^d$ with Algorithm 5.1 starting at an arbitrary initial cone, for processing cones of the i-generation, only cones of the $(i-1)$-, i- and $(i+1)$-generation have to be remembered.*

While on starting (low) generations the number of the cones from one generation to another grows rapidly, on close to 'equatorial' generations (these basically constitute the segmentation) the increase is much less. It amounts to 5-15% compared to the previous generation, when n and d are moderate. Also, though the store-search structure for the cones is a binary tree, the computational time for search can be saved either, especially when the search is frequently performed.

Ad (4): When calculating the Tukey depth under the general position assumption of $\{\mathbf{z}\} \cup X$, one can step much further in this direction, than Observation 5.1 does. First, to simplify further presentation, let us code the cones. As mentioned above, the interior of each cone C maintains the disjoint division of X into X_C^+ and X_C^- according to the signs in X's projection onto any of its directions, and thus is uniquely defined by this division. So, binary identifiers for the cones can be used: a cone is coded by a binary sequence ("0" and "1" say), where each bit is responsible for a point $\in X$ w.r.t. some initial arbitrary ordering that is kept constant during the entire procedure. Then, points belonging to X_C^+ are coded by "1", those belonging to X_C^- by "0".

After coding the initial cone this way, other cones can be coded either the same way, or by another binary sequence identifying whether a point has changed the sign ("1") or not ("0").

Then any cone's code can be obtained as the code of the initial cone with those bits inverted that have been switched on in this last sequence. This leads to Lemma 5.1.

Lemma 5.1. *Let us start Algorithm 5.1 with an arbitrary initial cone, and in stage b regard only neighboring cones defined by points which have not changed their sign in the projection onto* $\mathbf{r}$ *yet. Then in processing cones of the i-th generation, only cones of the i-th generation have to be remembered to seek for neighbors and of the* $(i{+}1)$*-th generation to check for newness.*

Proof. If the cones defined by already processed points, i.e. those having changed their sign in the projection, are not considered, then only cones of the $(i+1)$-th generation can be taken into account, and no cones of the $(i-1)$-th or i-th generation. Then one can go through all the cones of the i-th generation, and add those newly found from the $(i+1)$-th generation to the queue. □

Theorem 5.1 (ii) and Lemma 5.1 lead to Lemma 5.2. Note that $\lfloor u \rfloor$ stands for the largest integer $\leq u$.

Lemma 5.2. *When starting Algorithm 5.1 with an arbitrary initial cone, and in stage b regard only neighboring cones defined by points which have not changed their sign in the projection onto* $\mathbf{r}$ *yet, only* $\lfloor \frac{n+2}{2} \rfloor$ *generations have to be considered.*

Proof. From Theorem 5.1 (ii), each point may define a cone's facet, changing its sign in projections on all directions of the neighboring cone. If, following Lemma 5.1, on each new step only not yet considered points are taken into account, then in each new generation exactly one point more has its sign on projection changed (compared to the initial cone). Then the maximum generation (if the initial cone is the 1st generation) will be $(n+1)$-th generation.

Each cone has its mirror-copy cone, where projections of X on all directions have exactly opposite signs; these cones need not be considered, of course. Then, if n is odd, exactly $\frac{n+1}{2}$ generations have to be considered, if n is even, $\lfloor \frac{n+1}{2} \rfloor + 1$ generations have to be considered, as the mirror-copy cones of the 'equatorial' (having number $\lfloor \frac{n+1}{2} \rfloor + 1$) generation also belong to the equatorial generation. Thus, at most $\lfloor \frac{n+2}{2} \rfloor$ generations have to be regarded. □

5.2.2 Weakening assumptions

Here we sequentially weaken the assumption that $\{\mathbf{z}\} \cup X$ are in general position. We do it in three steps:

a) allow point(s) $\in X$ to exactly coincide with $\mathbf{z}$;

b) additionally, allow > 1 point(s) $\in X$ to lie in $\{\mathbf{z} + \lambda(\mathbf{x}_i - \mathbf{z}) : \lambda \in \mathbb{R}\}$ for some $\mathbf{x}_i \in X$;

c) allow an arbitrary position of $\{\mathbf{z}\} \cup X$.

Ad (a), assume that the subset $X_{\mathbf{z}} = \{\mathbf{x} | \mathbf{x} = \mathbf{z}, \mathbf{x} \in X\}$ is not empty. After removing $X_{\mathbf{z}}$ from X, $\{\mathbf{z}\} \cup (X \setminus X_{\mathbf{z}})$ is in general position, and thus all the results of Section 5.2.1 are valid. Then $D(\mathbf{z}|X \setminus X_{\mathbf{z}})$ is calculated under the general position assumption of $\{\mathbf{z}\} \cup (X \setminus X_{\mathbf{z}})$, and $D(\mathbf{z}|X)$ is obtained as $\frac{D(\mathbf{z}|X \setminus X_{\mathbf{z}})(n - \sharp X_{\mathbf{z}}) + \sharp X_{\mathbf{z}}}{n}$.

Ad (b), let $X_{\mathbf{z}}^i$ be the set $X \cap \{\mathbf{z} + \lambda(\mathbf{x}_i - \mathbf{z}) : \lambda \in \mathbb{R}\}$ for some $\mathbf{x}_i \in X$. All $X_{\mathbf{z}}^i$, $i = 1, ..., n$ are singletons in the case when the general position assumption of $\{\mathbf{z}\} \cup X$ is satisfied. If

$\sharp X^i_{\mathbf{z}} > 1$ for at least one i, a facet can be defined by each $X^i_{\mathbf{z}}$, and the results of Section 5.2.1 should be reconsidered.

First, part (ii) of Theorem 5.1 affects the entire set $X^i_{\mathbf{z}}$, defining the corresponding hyperplane as described in part (i) of the Theorem (the change in proof is straightforward), i.e. all points $\in X^i_{\mathbf{z}}$ should change sign in the projection when traversing a facet of two cones. Lemma 5.1 is not valid any more, as when moving from i-generation cones through facets defined by $X^i_{\mathbf{z}}$ with $\sharp X^i_{\mathbf{z}} > 1$ the breadth-first search algorithm can 'jump forward', changing sign of more than one point at once, and thus can reach an already processed neighbor. Lemma 5.2 holds, although some extra cones (e.g., mirror but not equatorial ones) can be processed.

Ad (c), points from $X^+_{H_{\mathbf{x}},C}$ or $X^-_{H_{\mathbf{x}},C}$ can lie on the – separating them – $(d-2)$-plane containing the origin. Regarding part (ii) of Theorem 5.1, all these should change sign in the corresponding projection when moving to a neighboring cone (the change of proof is straightforward). The remaining stays as above.

5.2.3 Algorithm based on linear programming

For the sake of simplicity in this section we assume the general position of $\{\mathbf{z}\} \cup X$, $\mathbf{z} = \mathbf{0}$, and thus calculate $D(\mathbf{0}|X)$. Further, one can also act as in the beginning of Section 5.2.1, or extend the algorithm given below for arbitrarily positioned data as described in Section 5.2.2.

Basically, Algorithm 5.2 is the application of the breadth-first-search algorithm (Algorithm 5.1) for searching over the direction cones covering the entire $\mathbb{R}^d$. We will need some notation. As above, let C be the binary sequence of length n coding a direction cone containing some interior direction $\mathbf{r} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$: $C_j = 1\ \forall\ j \in \{j|\mathbf{r}'\mathbf{x}_j > 0, j = 1,...,n\}$ and $C_j = 0\ \forall\ j \in \{j|\mathbf{r}'\mathbf{x}_j < 0, j = 1,...,n\}$, where C_j denotes the j-th bit of C. Also, let I_j be the zero-filled binary sequence with the j-th bit set to "1", $\oplus$ denote the binary 'exclusive disjunction'="XOR" operation and $\sum C$ be the number of "1"s in C (Hamming distance between C and the zero-filled binary sequence).

Algorithm 5.2. Input: $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^d$, $d < n$, $\{\mathbf{0}\} \cup X$ *in general position.*

```
1. Initialization: calculate $X_{H_{\mathbf{x}_i}}, i = 1,...,n$, set $D = n$, draw $\mathbf{r} \in S^{d-1}$ maintaining strict
   order of $X$'s projection and let $C_0$ be its binary code, initialize a queue $\mathcal{C}_{topical}$ containing
   $C_0$ only and an empty searchable storage (e.g., binary tree) $\mathcal{C}_{future}$.
2. For $i = 1 : \lfloor\frac{n+2}{2}\rfloor$ do:
   (a) pop $C$ = head of $\mathcal{C}_{topical}$, $D = \min\{D, \sum C, n - \sum C\}$,
   (b) if $i = \lfloor\frac{n+2}{2}\rfloor$, then go to Step 2d,
   (c) for $j = 1 : n$ do:
       - if (i) $(C \oplus C_0)_j = 0$, and (ii) $\exists\ \mathbf{r} \in S^{d-1} \cap H_{\mathbf{x}_j}$ such that $\mathbf{r}'\mathbf{x} > 0\ \forall\ \mathbf{x} \in X^+_{H_{\mathbf{x}_j},C}$
         and $\mathbf{r}'\mathbf{x} < 0\ \forall\ \mathbf{x} \in X^-_{H_{\mathbf{x}_j},C}$, and (iii) $C \oplus I_j \notin \mathcal{C}_{future}$, then add $C \oplus I_j$ to $\mathcal{C}_{future}$,
   (d) if $\mathcal{C}_{topical} \neq \varnothing$, then go to Step 2a, else $\mathcal{C}_{topical} = \mathcal{C}_{future}$, $\mathcal{C}_{future} = \varnothing$.
3. Return: $D/n$.
```

Nontrivial is the check of condition (ii) in Step 2c, i.e. to find out whether there $\exists\ \mathbf{r} \in S^{d-1} \cap H_{\mathbf{x}_j}$ such that $\mathbf{r}'\mathbf{x} > 0\ \forall\ \mathbf{x} \in X^+_{H_{\mathbf{x}_j},C}$ and $\mathbf{r}'\mathbf{x} < 0\ \forall\ \mathbf{x} \in X^-_{H_{\mathbf{x}_j},C}$. This is the same as to check whether two sets $X^+_{H_{\mathbf{x}_j},C}$ and $X^-_{H_{\mathbf{x}_j},C}$ are linearly separable via the origin, i.e. separable by a $(d-2)$-hyperplane $\in H_{\mathbf{x}_j}$ containing $\mathbf{0}$. In fact this corresponds to determining whether $\mathbf{0} \in \operatorname{conv}(X^+_{H_{\mathbf{x}_j},C} \cup -X^-_{H_{\mathbf{x}_j},C})$ (or $-X^+_{H_{\mathbf{x}_j},C} \cup X^-_{H_{\mathbf{x}_j},C}$). If it does, the sets $X^+_{H_{\mathbf{x}_j},C}$ and $X^-_{H_{\mathbf{x}_j},C}$ are not linearly separable, the direction $\mathbf{r} \in S^{d-1} \cap H_{\mathbf{x}_j}$ separating the projections of $X^+_{H_{\mathbf{x}_j},C}$ and $X^-_{H_{\mathbf{x}_j},C}$ on it by 0 cannot be found, and thus $\mathbf{x}_j$ generates no facet of the direction cone.

Let $Y = X^+_{H_{\mathbf{x}_j},C} \cup -X^-_{H_{\mathbf{x}_j},C}$ for some $C \in \mathcal{C}(\mathbf{0}, X)$ and some $j \in \{1, ..., n\}$, represented as an $(n-1) \times (d-1)$ matrix $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_{n-1})'$ with rows being the points $\in H_{\mathbf{x}_j}$. The task from above narrows down to finding a feasible solution satisfying the constraints:

$$\begin{aligned} \mathbf{Y}'\Lambda &= \mathbf{0}_{d-1}, \\ \Lambda'\mathbf{1}_{n-1} &= 1, \\ \Lambda &\geq \mathbf{0}_{n-1}, \end{aligned}$$

with $\Lambda = (\lambda_1, ..., \lambda_{n-1})'$ and $\mathbf{0}_k$ ($\mathbf{1}_k$) being a vector-column of k zeros (ones). This is what is done in the first phase of the usual simplex algorithm.

If the assumption of the general position of $\{\mathbf{0}\} \cup X$ is violated, it can happen that for some $C \in \mathcal{C}(\mathbf{0}, X)$ and $j \in \{1, ..., n\}$, $\mathbf{0}$ lies on a facet of $\operatorname{conv}(X^+_{H_{\mathbf{x}_j},C} \cup -X^-_{H_{\mathbf{x}_j},C})$ that has affine dimension $d-2$ or lower. Then, all the points generating this (sub)facet define the cone's facet (at once), and their projections all change sign when $\mathbf{r}$ traverses this cone's facet.

5.3 Computing Tukey depth via combinatorial approach

In this section, a combinatorial approach is presented. It accounts for a sufficient set of point combinations to determine the Tukey depth, and is an exact version of the third approximation method proposed by Rousseeuw & Struyf (1998) and the first approximation method of Chen *et al.* (2013).

First, some combinatorial issues are regarded in Section 5.3.1. Then in Section 5.3.2 the algorithm is presented and the main theoretical results of its sufficiency are proven.

5.3.1 Intuition behind the approach

The Tukey depth is the smallest fraction of X lying in a closed halfspace with $\mathbf{z}$ on its boundary such that the remaining open halfspace does not contain $\mathbf{z}$, i.e. the portion of points lying on the dividing hyperplane or in the outer open halfspace. For simplicity, assume that $\mathbf{z} = \mathbf{0}$ and $\{\mathbf{0}\} \cup X$ are in general position. Then a hyperplane containing $\mathbf{0}$ and no points $\in X$ with a normal vector $\mathbf{r} \in S^{d-1}$, together with its perturbations in some range of angles, maintains certain dichotomy of points $X^+_{\mathbf{r}} = \{\mathbf{x}|\mathbf{x}'\mathbf{r} > 0\}$ and $X^-_{\mathbf{r}} = \{\mathbf{x}|\mathbf{x}'\mathbf{r} < 0\}$ that corresponds to division of X by positive and negative halfspaces (while $X^0_{\mathbf{r}} = \{\mathbf{x}|\mathbf{x}'\mathbf{r} = 0\} = \varnothing$). Taking $X^+_{\mathbf{r}}$

as a candidate for a separating halfspace determining the Tukey depth, it can be then obtained as $D(\mathbf{0}|X) = \min_{\mathbf{r}\in S^{d-1}} \#X_{\mathbf{r}}^{+}$.

W.r.t. $\mathbf{r}$, this is an optimization problem which is neither convex nor linear or continuous. For its exact solution all possible variants are to be considered. Computationally that means to regard all possible dichotomies $(X_{\mathbf{r}}^{+}, X_{\mathbf{r}}^{-})$, i.e. all reachable permutations of projection of X onto $\mathbf{r}$. This is what is eventually done in the exact algorithms for the Tukey depth. Each $(X_{\mathbf{r}}^{+}, X_{\mathbf{r}}^{-})$ is a distinct separation of X by the hyperplane containing $\mathbf{0}$. The number of such separations, and thus the number of variants to be regarded by the algorithm, under the assumption of general position of $\{\mathbf{0}\} \cup X$, is determined by $n = \#X$ and d only, independently of the position of the points $\in X$: $C(n,d) = 2\sum_{i=0}^{d-1}\binom{n-1}{i}$. If general position of $\{\mathbf{0}\} \cup X$ is not assured, $C(n,d)$ is the upper bound on the number of achievable separations. Clearly, a pair of such (mirror) dichotomies can be regarded at once by taking $\min\{X_{\mathbf{r}}^{+}, X_{\mathbf{r}}^{-}\}$ as a candidate for determining the Tukey depth, so only half of this number has to be processed. This amounts to the complexity $O(n^{d-1})$.

In the algorithms of Liu & Zuo (2014a) and in Section 5.2, the first-breadth search over S^{d-1} is used to fully address all such dichotomies. The idea of the combinatorial algorithm is: as all the combinations have to be regarded anyways, why not go through all of them chaotically, i.e. using some order that has nothing to do with any space ordering, but is of combinatorial nature? This brings an enormous acceleration and requires minimum memory, see Section 5.4 for computation times. The approach is especially efficient when the relation n/d is small, as in this case no high-dimensional memory-consuming structures have to be created.

5.3.2 Combinatorial algorithm

The following algorithm calculates the exact Tukey depth of $\mathbf{0} \in \mathbb{R}^d$ w.r.t. $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$ under the assumption that $X \cap \{\mathbf{0}\} = \varnothing$ and X has affine dimension d. If the point to calculate the depth for $\neq \mathbf{0}$, X can be shifted correspondingly. If the affine dimension of X is $< d$, X should be projected onto the space of its dimension. Then, the depth is calculated in this lower-dimensional space. Note, that the assumption of general position is not required here, and points coinciding with $\mathbf{0}$ can be separated from X before Step 2 of the algorithm, and their number should be added to the result afterwards. ϵ is a precision constant that has to be chosen, and $\mathbf{x}_{\mathbf{r}_c}^{\perp}$ denotes the projection of $\mathbf{x}$ onto the hyperplane through $\mathbf{0}$ normal to $\mathbf{r}_c$.

Algorithm 5.3. Input: *$X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$ with $X \cap \{\mathbf{0}\} = \varnothing$, X has affine dimension d.*

```
1. If $d = 1$, then compute $n_{min} = n \cdot D^1(\mathbf{0}|X)$, **return:** $n_{min}$, else $n_{min} = n$.
2. $C = \{j_1, ..., j_{d-1} | j_1, ..., j_{d-1}$ distinct, $j_1, ..., j_{d-1} \in \{1, ..., n\}\}$.
3. For each $c \in C$ do:
    (a) if affine dimension of $\{\mathbf{x}_{j_1}, ..., \mathbf{x}_{j_{d-1}}, \mathbf{0}\} < d - 1$, then go to next iteration,
```

(b) *compute* $\mathbf{r}_c \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ *normal to the hyperplane* $H_c \ni \{\mathbf{x}_{j_1}, ..., \mathbf{x}_{j_{d-1}}, \mathbf{0}\}$,

(c) $P = \{p|p = \mathbf{x}_i'\mathbf{r}_c, i = 1, ..., n\}$, $P^+ = \{p|p > \epsilon, p \in P\}$, $P^- = \{p|p < -\epsilon, p \in P\}$, $P^0 = \{p||p| \leq \epsilon, p \in P\}$,

(d) $\tilde{n}_{min} = \min\{\#P^+, \#P^-\}$,

(e) *if* $\#P^0 > d-1$, *then* $X_{\mathbf{r}_c}^{\perp} = \{\mathbf{x}_{\mathbf{r}_c}^{\perp}||\mathbf{x}'\mathbf{r}_c| \leq \epsilon, \mathbf{x} \in X\} \in \mathbb{R}^{d-1}$, $\tilde{n}_{min} = \tilde{n}_{min} +$ *Algorithm 5.3*($X_{\mathbf{r}_c}^{\perp}$),

(f) *if* $\tilde{n}_{min} < n_{min}$, *then* $n_{min} = \tilde{n}_{min}$.

4. **Return:** n_{min}.

To prove the correctness of the algorithm we will need several lemmas.

Lemma 5.3. *Given* $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$ *with* $X \cap \{\mathbf{0}\} = \varnothing$ *and* $\mathbf{r} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, *let* $X_{\mathbf{r}}^+ = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} > 0\}$, $X_{\mathbf{r}}^- = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} < 0\}$ *and* $X_{\mathbf{r}}^0 = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} = 0\}$ *be disjoint subsets of X lying in the positive and negative open half-spaces generated by the linear subspace normal to* $\mathbf{r}$, *and in the hyperplane itself, correspondingly. Then a slight perturbation of* $\mathbf{r}$ *can be found yielding* $\tilde{\mathbf{r}}$ *with* $X_{\mathbf{r}}^+ \subseteq X_{\tilde{\mathbf{r}}}^+$, $X_{\mathbf{r}}^- \subseteq X_{\tilde{\mathbf{r}}}^-$ *and* $X_{\tilde{\mathbf{r}}}^0 = \varnothing$.

Proof. We prove the existence of such $\tilde{\mathbf{r}}$ by construction.

Without loss of generality we consider the projection of X and $H_{\mathbf{r}} = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} = 0\}$ onto S^{d-1}. The projection of X will be just n points, name them $X_{S^{d-1}}$, and that of $H_{\mathbf{r}}$ will be some $S_{\mathbf{r}}^{d-2} = H_{\mathbf{r}} \cap S^{d-1}$. Now, the task is to find some $S_{\tilde{\mathbf{r}}}^{d-2}$ with $X_{S^{d-1}} \cap S_{\tilde{\mathbf{r}}}^{d-2} = \varnothing$.

Given any $S_{\mathbf{r}}^{d-2}$ and some $(d-2)$-dimensional rotation axis containing $S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3} \in S_{\mathbf{r}}^{d-2}$ with $S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3} \cap X_{S^{d-1}} = \varnothing$, one can always slightly rotate $S_{\mathbf{r}}^{d-2}$ around $S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3}$ in a 2-dimensional plane orthogonal to $S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3}$, obtaining $S_{\tilde{\mathbf{r}}}^{d-2}$ such that $X_{S^{d-1}} \cap S_{\tilde{\mathbf{r}}}^{d-2} = \varnothing$ holds. ($S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3} = S_{\mathbf{r}}^{d-2} \cap S_{\tilde{\mathbf{r}}}^{d-2}$ defines the axis around which a slight rotation of $S_{\mathbf{r}}^{d-2}$ yields $S_{\tilde{\mathbf{r}}}^{d-2}$.) Now it suffices to construct this $S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3}$ satisfying $X_{S^{d-1}} \cap S_{\mathbf{r},\tilde{\mathbf{r}}}^{d-3} = \varnothing$. But this task is identical to the one being solved, just in dimension lowered by one.

If $d = 3$, the task degenerates to finding a line l with $\{\mathbf{0}\} \in l$, intersecting a unit circle S^{d-2} on the unit sphere S^{d-1} satisfying $X_{S^{d-1}} \cap l = \varnothing$, which is always doable. Further, recursively, any $d > 3$ will be consequently reduced to $d = 3$. For $d = 2$ the construction is trivial as well, and for $d = 1$ of no need. ☐

Straightforward corollary.

Corollary 5.1. *Under the conditions of Lemma 5.3 it holds:* $\min\{\#X_{\tilde{\mathbf{r}}}^+, \#X_{\tilde{\mathbf{r}}}^-\} \leq \min\{\#(X_{\mathbf{r}}^+ \cup X_{\mathbf{r}}^0), \#(X_{\mathbf{r}}^- \cup X_{\mathbf{r}}^0)\}$.

Lemma 5.4. *Given* $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$, *assume that* $X \cap \{\mathbf{0}\} = \varnothing$, *X has affine dimension d. Let a vector* $\mathbf{r} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ *generate the normal hyperplane H with* $\mathbf{0} \in H$ *and* $H \cap X = \varnothing$, *and denote* $X_{\mathbf{r}}^+ = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} > 0\}$, $X_{\mathbf{r}}^- = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} < 0\}$ *and* $X_{\mathbf{r}}^0 = \{\mathbf{x} \in \mathbb{R}^d|\mathbf{x}'\mathbf{r} = 0\} = \varnothing$. *Then a slight perturbation of* $\mathbf{r}$ *can be found yielding* $\tilde{\mathbf{r}}$ *and the corresponding normal zero-hyperplane* $\tilde{H}$ *with* $\#X_{\tilde{\mathbf{r}}}^0 \geq d-1$, $X_{\tilde{\mathbf{r}}}^+ \subseteq X_{\mathbf{r}}^+$ *and* $X_{\tilde{\mathbf{r}}}^- \subseteq X_{\mathbf{r}}^-$, *i.e. H can be rotated to some* $\tilde{H}$ *containing* $\mathbf{0}$ *and at least* $d-1$ *points* $\in X$, *so that no point* $\in X$ *changes the side of the hyperplane.*

Proof. We prove the existence of such $\tilde{\mathbf{r}}$ by construction.

Set $H_0 = H$. We rotate H_0 consequently laying it on $d-1$ points $\in X$. Define $A = \{\mathbf{0}\}$, which is the axis set. Choose an arbitrary $(d-2)$-dimensional subspace O_1 with $A \in O_1$.

Rotate H_0 in the plane orthogonal to O_1, till it 'meets' (at least) one of the points $\in X \setminus H_0$, obtain H_1. Append A with one of the new points $X \cap (H_1 \setminus H_0)$. Continue, till (after step $d-1$) A contains $d-1$ points $\in X$ (and $\{\mathbf{0}\}$), i.e. $H_{d-1} = \tilde{H}$, $\#(X \cap \tilde{H}) \geq d-1$. This is ensured by the affine dimension d of X. $\square$

Based on the last lemma one can derive the following corollary.

Corollary 5.2. *Under the conditions of Lemma 5.3, if additionally* $\{\mathbf{0}\} \cup X$ *are in general position, it holds:*
$\min\{\#X_{\mathbf{r}}^+, \#X_{\mathbf{r}}^-\} = \min_{\tilde{\mathbf{r}} \in \widetilde{\mathbf{R}}}\{\#X_{\tilde{\mathbf{r}}}^+, \#X_{\tilde{\mathbf{r}}}^-\}$,
where $\widetilde{\mathbf{R}} = \{\hat{\mathbf{r}} | X_{\mathbf{r}}^+ \subseteq X_{\hat{\mathbf{r}}}^+, X_{\mathbf{r}}^- \subseteq X_{\hat{\mathbf{r}}}^-, \hat{\mathbf{r}} \in \mathbb{R}^d \setminus \{\mathbf{0}\}\}$.

Now we can state and prove the main result.

Theorem 5.2. *Given* $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$, *assume* $X \cap \{\mathbf{0}\} = \varnothing$ *and that* X *has affine dimension* d. *Then Algorithm 5.3 returns* $nD(\mathbf{0}|X)$.

Proof. Let $\mathbf{r} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ be an arbitrary direction, let $X_{\mathbf{r}}^+ = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x}'\mathbf{r} > 0\}$, $X_{\mathbf{r}}^- = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x}'\mathbf{r} < 0\}$ and $X_{\mathbf{r}}^0 = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x}'\mathbf{r} = 0\}$ be disjoint subsets of X lying in the positive and negative open half-spaces generated by the zero-hyperplane normal to $\mathbf{r}$, and in the hyperplane itself, correspondingly. Then, according to Lemma 5.3, a slight perturbation of $\mathbf{r}$ can be found yielding $\tilde{\mathbf{r}}$ with $X_{\mathbf{r}}^+ \subseteq X_{\tilde{\mathbf{r}}}^+$, $X_{\mathbf{r}}^- \subseteq X_{\tilde{\mathbf{r}}}^-$ and $X_{\tilde{\mathbf{r}}}^0 = \varnothing$. Consider a set of all such dichotomies $S_X = \{s | s = (X_{\tilde{\mathbf{r}}}^+, X_{\tilde{\mathbf{r}}}^-)\}$, each characterized by $\tilde{\mathbf{r}}$, and by $n_{min}^s = \min\{\#X_{\tilde{\mathbf{r}}}^+, \#X_{\tilde{\mathbf{r}}}^-\}$. From Corollary 5.1, $\tilde{\mathbf{r}}$ is at least as good a candidate for the Tukey depth as $\mathbf{r}$, because points lying in the closed half-space are accounted for, too. Then, it suffices to show that Algorithm 5.3 regards all $s \in S_X$, i.e. that Algorithm 5.3 ensures $n_{min} \leq n_{min}^s \forall s \in S_X$.

Consider some $s = (X_{\tilde{\mathbf{r}}}^+, X_{\tilde{\mathbf{r}}}^-) \in S_X$. According to Lemma 5.4, a rotation of $\tilde{\mathbf{r}}$ to $\hat{\mathbf{r}}$, and corresponding normal hyperplane $H_{\hat{\mathbf{r}}}$ containing at least $d-1$ points $\in X$, can be found yielding $X_{\hat{\mathbf{r}}}^+ = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x}'\hat{\mathbf{r}} > 0\}$, $X_{\hat{\mathbf{r}}}^- = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x}'\hat{\mathbf{r}} < 0\}$ and $X_{\hat{\mathbf{r}}}^0 = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x}'\hat{\mathbf{r}} = 0\}$. As $\#X_{\hat{\mathbf{r}}}^0 \geq d-1$, all such $H_{\hat{\mathbf{r}}}$ are regarded by Algorithm 5.3 at least once.

We have $X_{\hat{\mathbf{r}}}^+ \subseteq X_{\tilde{\mathbf{r}}}^+$, $X_{\hat{\mathbf{r}}}^- \subseteq X_{\tilde{\mathbf{r}}}^-$, and let $\tilde{n}_{min} = \min\{\#X_{\hat{\mathbf{r}}}^+, \#X_{\hat{\mathbf{r}}}^-\}$. Then $\tilde{n}_{min}$ should be augmented by $\tilde{n}_{min}^+$, the smallest number of points $\in X_{\hat{\mathbf{r}}}^0$ one can get on one side – namely this yielding $\tilde{n}_{min}$ – of $H_{\hat{\mathbf{r}}}$ by slight perturbation of $\hat{\mathbf{r}}$. $\tilde{n}_{min} + \tilde{n}_{min}^+ \leq n_{min}^s$, because $\tilde{n}_{min} + \tilde{n}_{min}^+$ is the smallest number of points one can get on one side of the hyperplane preserving $X_{\hat{\mathbf{r}}}^+ \subseteq X_{\tilde{\mathbf{r}}}^+$ and $X_{\hat{\mathbf{r}}}^- \subseteq X_{\tilde{\mathbf{r}}}^-$, i.e. essentially almost the same hyperplane, just slightly rotated.

If $X \cup \{\mathbf{0}\}$ are in general position then $\#X_{\hat{\mathbf{r}}}^0 = d-1$, and the entire $X_{\hat{\mathbf{r}}}^0$ can occur on the side of a slightly rotated hyperplane $H_{\hat{\mathbf{r}}}$, that is opposite to the $\tilde{n}_{min}$-side, and thus $\tilde{n}_{min}^+ = 0$. If $X \cup \{\mathbf{0}\}$ is not in general position, $\#X_{\hat{\mathbf{r}}}^0$ can be $> d-1$, then $\tilde{n}_{min}^+ = D(\mathbf{0}|X_{\hat{\mathbf{r}}}^0)$. In this case Algorithm 5.3 proceeds recursively till in each considered, possibly lower-dimensional, hyperplane points are in general position or the trivial case of $D^1(\mathbf{0}|\cdot)$ is reached. $\square$

Based on Corollary 5.2, and due to the fact that, if $\{\mathbf{0}\} \cup X$ are in general position the condition on Step 3e is never true, one can derive the following corollary:

Corollary 5.3. *Under conditions of Theorem 5.2, if additionally* $\{\mathbf{0}\} \cup X$ *are in general position, no recursion is needed in Algorithm 5.3.*

From Theorem 5.2 it is clear that it is sufficient to regard all the hyperplanes based on $\mathbf{0}$ and $d-1$ points from X. There are $\binom{n}{d-1}$ such hyperplanes. For each of them the normal vector has to be found, which amounts to complexity $O(d^3)$. Then the data should be projected on it, with complexity $O(nd)$. This gives the complexity of the algorithm: $O\big(n^{d-1}(d^3 + nd)\big)$.

Table 5.1: *Computational times of the algorithm of Liu & Zuo (2014a) when calculating the Tukey depth of the origin w.r.t. a sample of cardinality n in $\mathbb{R}^3$ drawn from the standard normal distribution, in seconds.*

$d\backslash n$	40	80	160	320	640	1280	2560
3	0.935	3.675	15.493	74.148	449.614	3681.818	42116.390

5.4 Experiments

One of the most important characteristics of any algorithm is its execution time. Here we compare the time consumption of both presented algorithms, and of that designed by Liu & Zuo (2014a). The experiment is set similar to Liu & Zuo (2014a): the Tukey depth of $\mathbf{z} = \mathbf{0}$ w.r.t. a d-variate sample of cardinality n drawn from the standard normal distribution is computed. Liu & Zuo (2014a) also suggest to calculate the depth of other points. Here it is not done for the following reason. If $\mathbf{z} \in \text{conv}(X)$, all the dichotomies are to be regarded anyways. Thus, as long as the data are in general position, independent of where $\mathbf{z}$ is located, the same number of algorithmic steps has to be performed. If $\mathbf{z} \notin \text{conv}(X)$, its depth $= 0$, and this can be assured by means of linear programming as described in Section 5.2.3. Also, for time saving reasons, different to Liu & Zuo (2014a), we conduct only 10 executions of the algorithm, as, to our experience, deviation of the execution time is negligible, also for the above mentioned reasons. On the other hand, an experiment for one pair (n, d) and for one algorithm never lasts longer than 24 hours (i.e., e.g., for $n = 40$ and $d = 13$ only two executions of the combinatorial algorithm have been performed). We use processor Core i7-2600 (3.4 GHz) having enough physical memory.

In Table 5.1 the execution times (in seconds) for $d = 3$ and $n = 40$, 80, 160, 320, 640, 1280, 2560 of the algorithm of Liu & Zuo (2014a) are reported. They are obtained using Matlab source code kindly sent by Prof. Yijun Zuo and Dr. Xiaohui Liu. Table 5.2 presents execution times (in seconds) of the linear algorithm from Section 5.2 (top) and of the combinatorial algorithm from Section 5.3 (bottom). The sign "—" in the Tables means that the depth is not computable at least once in 24 hours. Similarly this is not possible for $d = 3$ and $n \geq 5120$ with the algorithm of Liu & Zuo (2014a), see Table 5.1 (but also with the proposed linear algorithm, see Table 5.2). One can conclude that both presented algorithms are faster than the algorithm of Liu & Zuo (2014a). The combinatorial algorithm is very efficient, especially in higher dimensions. Note that for moderate n the growth of computational time decreases with increasing dimension, which can be explained by the dominating combinatorial term in the algorithmic complexity.

5.5 Concluding remarks

The paper presents two algorithms for computing the Tukey depth by finding a global minimum over a finite range of variants. The task of computing the Tukey depth is NP-complete while all separations of X by hyperplanes through $\mathbf{z}$ are regarded. The algorithms reflect two different

Table 5.2: *Computational times of the two proposed algorithms when calculating the Tukey depth of the origin w.r.t. a sample of cardinality n in $\mathbb{R}^d$ drawn from the standard normal distribution, in seconds. Time for the linear algorithm is the upper number, for the combinatorial algorithm it is the lower one.*

$d\backslash n$	40	80	160	320	640	1280	2560	5120	10240	20480
3	0.028	0.228	1.888	17.371	174.744	1789.335	18436.420	—	—	—
	0.002	0.005	0.019	0.119	0.902	7.405	73.496	512.250	4124.849	32936.300
4	0.403	7.022	119.010	2035.505	35924.400	—	—	—	—	—
	0.011	0.109	1.213	15.607	229.467	3627.886	60243.800	—	—	—
5	4.752	174.848	5974.114	—	—	—	—	—	—	—
	0.125	2.619	59.017	1502.102	42874.300	—	—	—	—	—
6	48.170	3877.931	—	—	—	—	—	—	—	—
	1.159	51.062	2337.476	—	—	—	—	—	—	—
7	368.112	67897.800	—	—	—	—	—	—	—	—
	8.625	789.074	71564.600	—	—	—	—	—	—	—
8	2441.332	—	—	—	—	—	—	—	—	—
	51.127	10358.740	—	—	—	—	—	—	—	—
9	12703.730	—	—	—	—	—	—	—	—	—
	261.044	—	—	—	—	—	—	—	—	—
10	58767.700	—	—	—	—	—	—	—	—	—
	1156.536	—	—	—	—	—	—	—	—	—
11	—	—	—	—	—	—	—	—	—	—
	4262.319	—	—	—	—	—	—	—	—	—
12	—	—	—	—	—	—	—	—	—	—
	13547.800	—	—	—	—	—	—	—	—	—
13	—	—	—	—	—	—	—	—	—	—
	38890.400	—	—	—	—	—	—	—	—	—

views at the depth's calculation. First of them, the one based on the linear programming, follows the traditions of cone segmentation of a finite-dimensional space and regards candidate hyperplanes for the Tukey depth according to a first-breadth order on the unit sphere. It employs the initial idea of Liu & Zuo (2014a) by identifying a facet using linear programming, and by exploiting the fact that each point $\in X$ changes the halfspace only once during the entire execution of the breadth-first search algorithm. This yields a substantial acceleration. The second algorithm makes use of the combinatorial nature of the depth regarding all candidates chaotically, and by this it is fast.

Both algorithms require neither a general position assumption on the data, nor their perturbation. Their straightforward modifications allow to account for weighted observations. The linear algorithm saves memory by storing only two layers of the direction cones in RAM, and the combinatorial one needs no huge storing structures at all. Because the hyperplane candidates for the Tukey depth can be looked through independently, the combinatorial algorithm can be parallelized in a very efficient way.

Both algorithms presented here can be modified to solve related tasks, such as computing regression depth (Rousseeuw & Hubert, 1999) or finding a linear classification rule separating two training classes with a minimal number of errors (=empirical risk), e.g. on the plane in the α-procedure (Vasil'ev, 2003). Ghosh & Chaudhuri (2005a) investigate the connection between Tukey (also regression) depth and binary supervised classification. In a different way, either of the algorithms can be used for finding a hyperplane through a fixed point minimizing empirical risk. When adding an artificial coordinate equaling zero for all observations and letting the hyperplane go through $(\mathbf{0}'_d, 1)'$, say, its $(d-1)$-dimensional trace corresponds to the risk minimizing separation. After removing erroneous points, an optimal margin classifier (Boser *et al.*, 1992) can be applied to find the optimal separation hyperplane. High parallelization abilities allow for finding the separating hyperplane that minimizes the empirical risk exactly, while by a (say, polynomial) extension of the space nonlinear classification rules may be involved.

The ideas considered in this paper can be applied to a wider range of tasks. Thus, the way of covering the space by a breadth-first search algorithm can be applied to many tasks involving a cone segmentation. Application of the linear programming, used here, can be a good alternative to the QHULL algorithm (used by Hallin *et al.* (2010), Paindaveine & Šiman (2012a,b), Liu & Zuo (2014a,b)), while it allows to check whether a single point is a vertex of the convex hull of a data cloud. For instance, in the linear algorithm, close to the equatorial generations, after filtering already seen points and unreachable neighbors (of the same generation), the number of points to be checked is halved. The approaches can be tried to be extended for computing other depths of combinatorial nature and possessing the weak projection property (Dyckerhoff, 2004), which is common for many depths.

Chapter 6

Outlook

This dissertation covers depth-based classification in Chapters 2 through 4 and exact calculation of location depth in Chapter 5. Of course, many problems remain unsolved, and the presented methods may be extended in several directions.

The key constituent of the $DD\alpha$-classifier, which has been developed and investigated in Chapters 2 through 4, is the α-separator, an adaptation of the α-procedure to the DD-plot. Due to the fact that it is reasonable for the separating line in the DD-plot to contain the origin, the time complexity of the α-separator in each DD-plane equals the time complexity of quick sorting, and thus amounts to $O\big((m+n)\log(m+n)\big)$, i.e. it is very fast. It also inherits desirable robustness from the α-procedure. The α-procedure, introduced by Vasil'ev (1991), was designed to construct a small-dimensional space of highly efficient – in sense of supervised classification – features, see Vasil'ev (2003). It is an inductive heuristic based on the principles of reduction theory. During the last years similar ideas have been used, e.g., by Ferraty *et al.* (2010) for choosing an optimal subset of arguments in functional classification and Croux *et al.* (2013) to search for robust principle components. Different to the α-separator, and in absence of the depth transform, the α-procedure is not quick, and in some cases can be computationally quite involved. The reasons are: the hyperplane separating the DD-plot is anchored at a fixed point, the origin, while this is not the case with the α-procedure; the dimension of the initial space is usually > 2, in some cases polynomial extension is needed to fit the nonlinearity of the separating rule. Theoretical properties of the α-procedure also constitute an area of further research and are not less intriguing. Still to investigate is the Bayes optimality for continuous and empirical distributions and precise cases and reasons for its violation.

Statistical depth functions find further applications in unsupervised classification. Jörnsten *et al.* (2002) introduce the *relative L_1-depth* (ReD), which is the difference between the depths of an observation w.r.t. the own cluster and the highest among the others. The authors suggest a clustering method based on the multivariate L_1-median and use ReD to determine the number of clusters. Jörnsten (2004) combines ReD with the *silhouette width*, i.e. the normalized difference between the distance to the center of the closest among competing clusters and that of the own. Application of this method together with a depth notion of local nature may make it possible to allocate nonconvex clusters. The extension of clustering approaches to functional data constitutes a rapidly developing field of study, see, e.g., Ferraty & Romain

(2010), chapter by Baíllo, Cuevas and Fraiman for a survey. Using functional data depths, clustering can be extended to infinite settings. On the other hand, a distance based clustering algorithm, *partition around medoids* (PAM) say, can be applied by exploiting appropriately defined distances between functions. For example, the Hausdorff distance between the functions' hypographs (Cuevas *et al.*, 2013) is a promising measure of closeness; robustness and computational issues are to be solved though. Using it in PAM together with the data depth for determining the number of clusters, similar to Jörnsten *et al.* (2002), seems to be a reasonable approach. In such a way the geometry of functions can be captured, and a notion of functional depth, chosen and (or) adjusted for specific application, can make the data tell the number of clusters.

In Chapter 4 a methodology of DD-plot-based classification of functional data is presented and its efficiency is explored in a comparative study. The data sets used, though covering a range of different problems connected with supervised functional classification, have a rather demonstrative character. To distinguish more special application areas for classifiers of this kind, further research could start from real problems for which the presented methodology has a substantial edge over competitors. Besides that, our notion of functional data depth, which can be adjusted for a particular application, needs additional theoretical development and practical investigation. In particular, derivatives of higher order, possibly needing less nodes, may be considered to extend deepness of consideration of functional dynamics with the argument. Attention should also be payed to finding alternative ways of defining the numbers of intervals to integrate over, and to size these. This information can be obtained from the functions' paths, based on functions' variability or dispersion say, but also taking into account application's peculiarity.

Chapter 5 suggests two algorithms for exact computation of the location depth, exploiting different principles. The first principle consists in the cone segmentation of the space and subsequent application of the breadth-first search algorithm. It can be employed to calculate any depth which – for each direction cone – possesses a known analytical form. This can also be used for construction of depth-trimmed regions. The examples are: zonoid-depth trimmed regions (Mosler *et al.*, 2009), location-depth trimmed regions (Hallin *et al.*, 2010, Paindaveine & Šiman, 2012a,b), location depth (Liu & Zuo, 2014a), projection depth and its regions (Liu & Zuo, 2014b). The principle can be applied to compute further depths and their central regions, too. The second principle, regarding all combinations of some kind, can be modified to compute another depth of combinatorial nature, say simplicial depth. Calculating location depth in a $(d+1)$-dimensional space may be modified to find the smallest number of errors by a linear separation of two classes in $\mathbb{R}^d$. After removing the errors, an optimal margin classifier (Boser *et al.* (1992), or the generalized portrait algorithm of Vapnik & Chervonenkis (1974)) can find the hyperplane itself in polynomial time. Thus, these algorithms can be used to find the optimal risk-minimizing hyperplane in classification algorithms, for instance, in the two- (or more)- dimensional linear spaces iterated by the projective invariant method α-procedure (Vasil'ev, 2003).

Based on the idea of Rousseeuw & Struyf (1998), Rainer Dyckerhoff developed an algorithm computing the location depth in any dimension. This proceeds recursively projecting data on the lower-dimensional subspaces till the two-dimensional space is achieved, and then applying the algorithm of Rousseeuw & Ruts (1996) for the bivariate location depth, which amounts to the complexity $O(n^{d-1}\log n)$ for a sample of size n in $\mathbb{R}^d$. (Recall, the complexity of the combinatorial algorithm is $O\big(n^{d-1}(d^3 + nd)\big)$.) Latest comparative studies have shown, that this algorithm is particularly efficient when $n >> d$, while for moderate ratio $\frac{n}{d}$, vice versa, the combinatorial algorithm demonstrates high computational speed. This points onto developing a hybrid algorithm, choosing one of the two depending on n and d. The ongoing work by Dyckerhoff & Mozharovskyi (2014) regards these issues.

Some depths, having attractive properties, such as robustness or the ability to reflect asymmetricities of a distribution, etc., cause substantial computational burden. The examples are location or projection depth. Dyckerhoff (2004) introduced the *weak projection property*: a depth possessing this equals the minimum univariate depth over projections onto lines. Depths satisfying it, e.g. Mahalanobis, location, projection depth, can be approximated from above by minimizing univariate depths in projections on randomly drawn directions. Cuesta-Albertos & Nieto-Reyes (2008) investigate this for the location depth, using uniform distribution on the unit sphere for the random directions. Here the so-called random Tukey depth is applied in Section 2.7 and in Chapter 3. Further questions arise: In which way should the directions be generated? How many of them are needed? How can their depths be distributed? Instead of the random search, data depth satisfying the weak projection property can also be approximated by means of nonconvex minimization on the unit sphere. The fact that for many depth notions computation time increases rapidly with the number of observations, gives rise to another idea. The depth may be computed exactly in the same space for a random subsample of smaller cardinality. The decision about the exact depth can then be made based on the obtained distribution of the subsamples' depths. The advantages of this approach are that the depth is approximated via exactly computed depths and the values obtained during the approximation can lie both above and below the real depth.

Bibliography

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

Asuncion, A. and Newman, D. (2007). *UCI machine learning repository* `http://archive.ics.uci.edu/ml`. School of Information and Computer Science, University of California, Irvine CA.

Barnett, V. (1976). The ordering of multivariate data. *Journal of the Royal Statistical Society. Series A: Statistics in Society* **139** 318–355.

Bazovkin, P. and Mosler, K. (2012). An exact algorithm for weighted-mean trimmed regions in any dimension. *Journal of Statistical Software* **47**.

Van Bever, G. (2013). *Contributions to Nonparametric and Semiparametric Inference Based on Statistical Depth. PhD-thesis.* Université Libre de Bruxelles.

Biau, G., Bunea, F. and Wegkamp, M.H. (2005). Functional classification in Hilbert spaces. *IEEE Transactions on Information Theory* **51** 2163–2172.

Biau, G., Devroye, L., Dujmovic, V. and Krzyzak, A. (2012). An affine invariant k-nearest neighbor regression estimate. *Journal of Multivariate Analysis* **112** 24–34.

Biblarz, T.J. and Raftery, A.E. (1993). The effects of family disruption on social mobility. *American Sociological Review* **58** 97–109.

De Boor, C. (2001). *A Practical Guide to Splines (Revised Edition).* Springer-Verlag, New York.

Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In: Haussler, D. (Ed.), *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, ACM Press, New York, 144–152.

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

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

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

Carey, J.R., Liedo, P., Müller, H.-G., Wang, J.-L. and Chiou, J.-M. (1998). Relationship of age patterns of fecundity to mortality, longevity, and lifetime reproduction in a large cohort of Mediterranean fruit fly females. *Journal of Gerontology. Series A: Biological Sciences* **53A** 245–251.

Cascos, I. (2007). The expected convex hull trimmed regions of a sample. *Computational Statistics* **22** 557–569.

Cascos, I. (2010). Data depth: multivariate statistics and geometry. In: Kendall, W., Molchanov, I. (Eds.), *New Perspectives in Stochastic Geometry*, Oxford University Press, Oxford, 398–423.

Chakraborty, A. and Chaudhuri, P. (2014). The deepest point for distributions in infinite dimensional spaces. *Statistical Methodology* **20** 27–39.

Chakraborty, A. and Chaudhuri, P. (2014). On data depth in infinite dimensional spaces. *Annals of the Institute of Statistical Mathematics* **66** 303–324.

Chaudhuri, P. (1996). On a geometric notion of quantiles for multivariate data. *Journal of the Americal Statistical Association* **91** 862–872.

Chaudhuri, P. and Sengupta, D. (1993). Sign tests in multidimension: inference based on the geometry of the data cloud. *Journal of the Americal Statistical Association* **88** 1363–1370.

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

Christmann, A., Fischer, P. and 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.

Christmann, A. and Rousseeuw, P.J. (2001). Measuring overlap in binary regression. *Computational Statistics and Data Analysis* **37** 65–75.

Claeskens, G., Hubert, M., Slaets, L. and Vakili K. (2014). Multivariate functional halfspace depth. *Journal of the American Statistical Association* **109** 411–423.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning* **20** 273–297.

Cox, L.H., Johnson, M.M. and Kafadar, K. (1982). Exposition of statistical graphics technology. In: American Statistical Association, *Proceedings of the Statistical Computing Section*, Cambridge University Press, London, 55–56.

Croux, C., Filzmoser, P. and Fritz, H. (2013). Robust sparse principal component analysis. *Technometrics* **55** 202–214.

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

Cuesta-Albertos, J.A. and Nieto-Reyes, A. (2010). Functional classification and the random Tukey depth. Practical issues. In: Borgelt, C., Rodríguez, G.G., Trutschnig, W., Lubiano, M.A., Angeles Gil, M., Grzegorzewski, P., Hryniewicz, O. (Eds.), *Combining Soft Computing and Statistical Methods in Data Analysis*, Springer-Verlag, Berlin Heidelberg, 123–130.

Cuevas, A., Cholaquidis, A. and Fraiman, R. (2013). Some geometric tools in functional data analysis. Conference Talk at *Computational and Methodological Statistics (ERCIM 2013)*, 14–16 December 2013.

Cuevas, A., Febrero, M. and Fraiman, R. (2007). Robust estimation and classification for functional data via projection-based depth notions. *Computational Statistics* **22** 481–496.

Cui, X., Lin, L. and Yang, G. (2008). An extended projection data depth and its aplications to discrimination. *Communications in Statistics – Theory and Methods* **37** 2276–2290.

Delaigle, A. and Hall, P. (2012). Achieving near-perfect classification for functional data. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* **74** 267–286.

Delaigle, A., Hall, P. and Bathia, N. (2012). Componentwise classification and clustering of functional data. *Biometrika* **99** 299–313.

Devroye, L., Györfi, L. and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag, New York.

Dhar, S.S. and Chaudhuri, P. (2011). On the statistical efficiency of robust estimators of multivariate location. *Statistical Methodology* **8** 113–128.

Donoho, D.L. (1982). *Breakdown Properties of Multivariate Location Estimators. PhD-thesis.* Harvard University.

Donoho, D.L. and Gasko, M. (1987). Multivariate generalization of the median and trimmed mean I. *Technical Report No.133*, Department of Statistics, University of California, Berkeley CA.

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

Dutta, S. and Ghosh, A.K. (2011). On classification based on L_p depth with an adaptive choice of p. *Technical Report No. R5/2011*, Statistics and Mathematics Unit, Indian Statistical Institute.

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

Dutta, S., Chaudhuri, P. and Ghosh, A.K. (2012). Classification using localized spatial depth with multiple localization. *Mimeo.*

Dutta, S., Ghosh, A.K. and Chaudhuri, P. (2011). Some intriguing properties of Tkey's half-space depth. *Bernoulli* **17** 1420–1434.

Dyckerhoff, R. (2000). Computing zonoid trimmed regions of bivariate data sets. In: Bethlehem, J., Van Der Heijden, P. (Eds.), *Proceedings in Computational Statistics*, COMPSTAT 2000, Physica-Verlag, Heidelberg, 295–300.

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

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

Dyckerhoff, R. and Mosler, K. (2011). Weighted-mean trimming of multivariate data. *Journal of Multivariate Analysis* **102** 405–421.

Dyckerhoff, R. and Mosler, K. (2012). Weighted-mean regions of a probability distribution. *Statistics and Probability Letters* **82** 318–325.

Dyckerhoff, R. and Mozharovskyi, P. (2014). Exact computation of the halfspace depth. *In Preparation.*

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

Ferraty, F., Hall, P. and Vieu, P. (2010). Most-predictive design points for functional data predictors. *Biometrika* **94** 807–824.

Ferraty, F. and Romain, Y. (2010). *The Oxford Handbook of Functional Data Analysis.* Oxford University Press, Oxford.

Ferraty, F. and Vieu, P. (2003). Curves discrimination: a nonparametric functional approach. *Computational Statistics and Data Analysis* **44** 161–173.

Ferraty, F. and Vieu, P. (2006). *Nonparametric Functional Data Analysis.* Springer-Verlag, New York.

Ferré, L. and Villa, N. (2006). Multi-layer perceptron with functional inputs: an inverse regression approach. *Scandinavian Journal of Statistics* **33** 807–823.

Finch, S. and Hueter, I. (2004). Random convex hulls: a variance revisited. *Advances in Applied Probability* **36** 981–986.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics* **7** 179–188.

Fix, E. and Hodjes, M. (1951). Discriminatory analysis. Nonparametric discrimination: consistency properties. *Technical Report 21-49-004 4*, USAF School of Aviation Medicine, Randolph Field, Texas.

Flury, B. and Riedwyl, H. (1988). *Multivariate Statistics: A Practical Approach.* Chapman and Hall, New York.

Frahm, G. (2004). *Generalized Elliptical Distributions: Theory and Applications. PhD-thesis.* University of Cologne.

Fraiman, R. and Muniz, G. (2001). Trimmed means for functional data. *TEST* **10** 419–440.

Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute* **15** 246–263.

Genton, M.G. (2001). Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research* **2** 299–312.

Ghosh, A.K. and Chaudhuri, P. (2005a). On data depth and distribution free discriminant analysis using separating surfaces. *Bernoulli* **11** 1–27.

Ghosh, A.K. and Chaudhuri, P. (2005b). On maximum depth and related classifiers. *Scandinavian Journal of Statistics* **32** 327–350.

Greaney, V. and Kelleghan, T. (1984). *Equality of Opportunity in Irish Schools: A Longitudinal Study of 500 Students.* The Educational Company, Dublin.

Green, P.J. and Silverman, B.W. (1979). Constructing the convex hull of a set of points in the plane. *The Computer Journal* **22** 262–266.

Habemma, J.D.F., Hermans, J. and Van Den Broek, K. (1974). A stepwise discriminant analysis program using density estimation. In: Bruckmann, G. (Ed.), *Proceedings in Computational statistics*, COMPSTAT 1974, Physica-Verlag, Heidelberg, 101–110.

Hall, P., Poskitt, D. and Presnell, B. (2001). A functional data-analytic approach to signal discrimination. *Technometrics* **43** 1–9.

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

Hand, D.J., Daly, F., Lunn, A.D., McConway, K.J. and Ostrowski, E. (1994). *A Handbook of Small Data Sets.* Chapman and Hall, London.

Hastie, T., Tibshirani, R. and Friedman, J.H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Second Edition).* Springer-Verlag, New York.

Hayford, J.F. (1902). What is the center of an area, or the center of a population? *Publications of the American Statistical Association* **8** 47–58.

Haykin, S. (2009). *Neural Networks: A Comprehensive Foundation (Third Edition).* Pearson, Upper Saddle River NJ.

Hoberg, R. (2002). *Cluster Analysis, Classification and Data Depth. PhD-thesis (In German).* University of Cologne.

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

Höffgen, K.U., Simon, H.-U. and Van Horn, K.S. (1995). Robust trainability of single neurons. *Journal of Computer and System Sciences* **50** 114–125.

Huang, D.-S. and Zheng, C.-H. (2006). Independent component analysis-based penalized discriminant method for tumor classification using gene expression data. *Bioinformatics* **22** 1855–1862.

Hubert, M. and Van Driessen, K. (2004). Fast and robust discriminant analysis. *Computational Statistics and Data Analysis* **45** 301–320.

James, G. and Hastie, T. (2001). Functional linear discriminant analysis for irregularly sampled curves. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* **63** 533–550.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge MA, 169–184.

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

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

Jörnsten, R. (2004). Clustering and classification based on the L_1 data depth. *Journal of Multivariate Analysis* **90** 67–89.

Jörnsten, R., Vardi, Y. and Zhang, C.-H. (2002). A robust clustering method and visualization tool based on data depth. In: Dodge, Y. (Ed.), *Statistical Data Analysis Based on the L_1-Norm and Related Methods*, Birkhäuser-Verlag, Basel, 353–366.

Kalbfleisch, J.D. and Prentice, R.L. (1980). *The Statistical Analysis of Failure Time Data*. Wiley, New York.

Koltchinskii, V.I. (1997). M-estimation, convexity and quantiles. *The Annals of Statistics* **25** 435–477.

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

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

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

Kuelbs, J. and Zinn, J. (2013). Concerns with functional depth. *Latin American Journal of Probability and Mathematical Statistics* **10** 831–855.

Lange, T., Mosler, K. and Mozharovskyi, P. (2014a). Fast nonparametric classification based on data depth. *Statistical Papers* **55** 49–69.

Lange, T., Mosler, K. and Mozharovskyi, P. (2014b). *DDα*-classification of asymmetric and fat-tailed data. In: Spiliopoulou, M., Schmidt-Thieme, L., Janning, R. (Eds.), *Data Analysis, Machine Learning and Knowledge Discovery*, Springer-Verlag, Berlin Heidelberg, 71–78.

Lange, T. and Mozharovskyi, P. (2014). The alpha-procedure – a nonparametric invariant method for automatic classification of d-dimensional objects. In: Spiliopoulou, M., Schmidt-Thieme, L., Janning, R. (Eds.), *Data Analysis, Machine Learning and Knowledge Discovery*, Springer-Verlag, Berlin Heidelberg, 79–86.

Lange, T., Mozharovskyi, P. and Barath, G. (2011). Two approaches for solving tasks of pattern recognition and reconstruction of functional dependencies. *XIV International Conference on Applied Stochastic Models and Data Analysis (ASMDA 2011)*, 7–10 June 2011.

Leng, X.Y. and Müller, H.-G. (2006). Classification using functional data analysis for temporal gene expression data. *Bioinformatics* **22** 68–76.

Li, J., Cuesta-Albertos, J.A. and Liu, R.Y. (2012). *DD*-classifier: nonparametric classification procedure based on *DD*-plot. *Journal of the American Statistical Association* **107** 737–753.

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

Liu, R.Y. (1992). Data depth and multivariate rank tests. In: Dodge, Y. (Ed.), *L_1-Statistical Analysis and Related Methods*, Elsevier, Amsterdam, 279–294.

Liu, R.Y., Parelius, J. and Singh, K. (1999). Multivariate analysis by the data depth: descriptive statistics and inference. *Annals of Statistics* **27** 783–858.

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

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

Liu, X.H., Zuo, Y.J. and Wang, Z.Z. (2013). Exactly computing bivariate projection depth median and contours. *Computational Statistics and Data Analysis* **60** 1–11.

López-Pintado, S. and Romo, J. (2006). Depth-based classification for functional data. In: Liu, R., Serfling, R., Souvaine, D. (Eds.), *DIMACS. Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, American Mathematical Society, Providence RI, 103–120.

López-Pintado, S. and Romo, J. (2009). On the concepth of depth for functional data. *Journal of the American Statistical Association* **104** 718–734.

López-Pintado, S. and Romo, J. (2011). A half-region depth for functional data. *Computational Statistics and Data Analysis* **55** 1679–1695.

Lopuhaä, H.P. and Rousseeuw, P.J. (1991). Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals of Statistics* **19** 229–248.

Luntz, A.L. and Brailovsky, V.L. (1969). On estimation of characters obtained in statistical procedure of recognition (In Russian). *Technicheskaya Kibernetika* **3** 48–52.

Mahalanobis, P. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Science of India* **12** 49–55.

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5** 115–133.

McGilchrist, C.A. and Aisbett, C.W. (1991). Regression with frailty in survival analysis. *Biometrics* **47** 461–466.

Miller, A.J., Shaw, D.E., Veitch, L.G. and Smith, E.J. (1979). Analyzing the results of a cloud-seeding experiment in Tasmania. *Communications in Statistics – Theory and Methods* **8** 1017–1047.

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

Mizera, I. (2002). On depth and deep points: a calculus. *The Annals of Statistics* **30** 1681–1736.

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

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

Mosler, K. and Hoberg, R. (2006). Data analysis and classification with the zonoid depth. In: Liu, R., Serfling, R., Souvaine, D. (Eds.), *DIMACS. Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, American Mathematical Society, Providence RI, 49–59.

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

Mosler, K. and Polyakova, Y. (2012). General notions of depth for functional data. `arXiv:1208.1981v1 [stat.ME]`.

Mozharovskyi, P., Mosler, K. and Lange, T. (2014). Classifying real-world data with the *DD*α-procedure. *Advances in Data Analysis and Classification*, to appear.

Müller, H.-G. and Stadtmüller, U. (2005). Generalized functional linear models. *The Annals of Statistics* **33** 774–805.

Nierenberg, D.W., Stukel, T.A., Baron, J.A., Dain, B.J. and Greenberg, E.R. (1989). Determinants of plasma levels of beta-carotene and retinol. *American Journal of Epidemiology* **130** 511–521.

Niinimaa, A. and Oja, H. (1999). Multivariate median. In: Kotz, S., Johnson, N.L., Read, C.P. (Eds.), *Encyclopedia of Statistical Sciences (Volume 3)*, Wiley, New York, 497–505.

Novikoff, A.B.J. (1962). On convergence proofs for perceptrons. In: Fox, J. (Ed.), *Proceeding of the Symposium on the Mathematical Theory of Automata*, Polytechnic Institute of Brooklyn, Brooklyn NY, 615–622.

Oja, H. (1983). Descriptive statistics for multivariate distributions. *Statistics and Probability Letters* **1** 327–332.

Oja, H. (2013). Multivariate median. In: Becker, C., Fried, R., Kuhnt, S. (Eds.), *Robustness and Complex Data Structures. Festschrift in Honour of Ursula Gather*, Springer-Verlag, Berlin Heidelberg, 3–15.

Osuna, E., Freund, R. and Girosi, F. (1997). An improved training algorithm for support vector machines. In: Principe, J., Gile, L., Morgan, N., Wilson, E. (Eds.), *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, IEEE, New York, 276–285.

Paindaveine, D. and Van Bever, G. (2012). Nonparametrically consistent depth-based classifiers. *Bernoulli*, to appear.

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

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

Platt, J. (1999). Fast training of support vector machines using sequantial minimal optimization. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge MA, 185–208.

Ramsay, J.O. and Silverman, B.W. (2005). *Functional Data Analysis (Second Edition).* Springer-Verlag, New York.

Reaven, G.M. and Miller, R.G. (1979). An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia* **16** 17–24.

Revuz, D. and Yor, M. (1999). *Continuous Martingales and Brownian Motion (Third Edition).* Springer-Verlag, New York.

Ripley, B.D. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge.

Romanazzi, M. (2009). Data depth, random simlices and multivariate dispersion. *Statistics and Probability Letters* **79** 1473–1479.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* **65** 386–408.

Rossi, F. and Villa, N. (2006). Support vector machine for functional data classification. *Neurocomputing* **69** 730–742.

Rousseeuw, P.J. and Van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics* **41** 212–223.

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

Rousseeuw, P.J. and Leroy, A.M. (1987). *Robust Regression and Outlier Detection.* Wiley, New York.

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

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

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

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

Serfling, R. (2002). A depth function and a scale curve based on spatial quantiles. In: Dodge, Y. (Ed.), *Statistical Data Analysis Based on the L_1-Norm and Related Methods*, Birkhäuser-Verlag, Basel, 25–38.

Serfling, R. (2006). Depth functions in nonparametric multivariate inference. In: Liu, R., Serfling, R., Souvaine, D. (Eds.), *DIMACS. Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, American Mathematical Society, Providence RI, 1–16.

Sguera, C., Galeano, P. and Lillo, R. (2014). Spatial depth-based classification for functional data. *TEST*, to appear.

Singh, K. (1991). A notion of majority depth. *Technical Report*, Department of Statistics, Rutgers University.

Small, C.G. (1990). A survey of multidimensional medians. *International Statistical Review* **58** 263–277.

Stahel, W.A. (1981). *Robust Estimation: Infinitesimal Optimality and Covariance Matrix Estimators. PhD-thesis (in German).* Swiss Federal Institute of Technology in Zurich.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines.* Springer-Verlag, New York.

Stone, C.J. (1977). Consistent nonparametric regression. *The Annals of Statistics* **5** 595–645.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* **36** 111–147.

Struyf, A. and Rousseeuw, P.J. (1999). Halfspace depth and regression depth characterize the empirical distribution. *Journal of Multivariate Analysis* **69** 135–153.

Tian, S.T. and James, G. (2013). Interpretable dimensionality reduction for classification with functional data. *Computational Statistics and Data Analysis* **57** 282–296.

Tuddenham, R.D. and Snyder, M.M. (1954). Physical growth of California boys and girls from birth to eighteen years. *University of California Publications in Child Development* **1** 183–364.

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

Turney, P. (1993). Robust classification with context-sensitive features. In: Chung, P.W.H., Lovegrove, G., Ali, M. (Eds.), *Proceedings ofthe Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Gordon and Breach Science Publishers, Philadelphia, 268–276.

Vapnik, V.N. (1998). *Statistical Learning Theory.* Wiley, New York.

Vapnik, V.N. and Chervonenkis, A.Ya. (1974). *Theory of Pattern Recognition (In Russian).* Nauka, Moscow.

Vapnik, V. and Lerner, A. (1963). Pattern recognition using generalized portrait method (In Russian). *Avtomatika i Telemechanika* **24** 774–780.

Vardi, Y. and Zhang, C.H. (2000). The multivariate L_1-median and associated data depth. *Proceedings of the National Academy of Sciences of the United States of America* **97** 1423–1426.

Vasil'ev, V.I. (1991). The reduction principle in pattern recognition learning (PRL) problem. *Pattern Recognition and Image Analysis* **1** 23–32.

Vasil'ev, V.I. (2003). The reduction principle in problems of revealing regularities I. *Cybernetics and Systems Analysis* **39** 686–694.

Vasil'ev, V.I. and Lange, T.I. (1998). The duality principle in learning for pattern recognition (In Russian). *Kibernetika i Vytschislitelnaya Technika* **121** 7–16.

Vencálek, O. (2011). *Weighted Data Depth and Depth Based Discrimination. Doctoral thesis.* Charles University in Prague.

Wand, M.P. and Jones, M.C. (1995). *Kernel Smoothing.* Chapman and Hall, London.

Wang, X.H., Ray, S. and Mallick, B.K. (2007). Bayesian curve classification using wavelets. *Journal of the American Statistical Association* **102** 962–973.

Wolberg, W.H. and Mangasarian, O.L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences of the United States of America* **87** 9193–9196.

Yeh, I.-C., Yang, K.-J. and Ting, T.-M. (2009). Knowledge discovery on RFM model using Bernoulli sequence. *Expert Systems with Applications* **36** 5866–5871.

Zuo, Y.J. and Lai, S.Y. (2011). Exact computation of bivariate projection depth and the Stahel-Donoho estimator. *Computational Statistics and Data Analysis* **55** 1173–1179.

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