Fast DD-classification of functional data

Karl Mosler · Pavlo Mozharovskyi

Abstract A fast nonparametric procedure for classifying functional data is introduced. It consists of a two-step transformation of the original data plus a classifier operating on a low-dimensional space. 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. This transformation yields a new notion of depth for functional data. Three alternative depth functions are employed for this, as well as two rules for the final classification in $[0, 1]^2$. The resulting classifier has to be cross-validated over a small range of parameters only, which is restricted by a Vapnik–Chervonenkis bound. The entire methodology does not involve smoothing techniques, is completely nonparametric and allows to achieve Bayes optimality under standard distributional settings. It is robust, efficiently computable, and has been implemented in an R environment. Applicability of the new approach is demonstrated by simulations as well as by a benchmark study.

Keywords Functional depth · Supervised learning · Central regions · Location-slope depth · DD-plot · Alpha-procedure

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 and Silverman (2005) and Ferraty and Vieu (2006) contain broad overviews of functional data analysis and the evolving field of classification. At the very beginning of the 21st century many classification approaches have been extended from multivariate to functional data: linear discriminant analysis (James and Hastie 2001), kernel-based classification (Ferraty and Vieu 2003), \( k \)-nearest-neighbours classifier (Biau et al. 2005), logistic regression (Leng and Müller 2006), neural networks (Ferré and Villa 2006), support vector machines (Rossi and Villa 2006). Transformation of functional data into a finite setting is done by using principal and independent (Huang and Zheng 2006) component analysis, principal coordinates (Hall et al. 2001), wavelets (Wang et al. 2007) or functions of very simple and interpretable structure (Tian and 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 for classification [see (Delaigle and 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 heuristic [as in (Ferraty et al. 2010)], a well classifying configuration of discretization points may be missed. (To see this, consider three discretization points which jointly discriminate well in \( \mathbb{R}^3 \) constructed from their evaluations, 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.

López-Pintado and Romo (2006), Cuevas et al. (2007), Cuesta-Albertos and Nieto-Reyes (2010) and Sguera et al. (2014) have introduced nonparametric notions of data depth for functional data classification [see also (Cuesta-Albertos et al. 2014)]. 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 and Romo 2006) of a function \( x \) in a class \( X \) of functions indicates the relative frequency of \( x \) 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 and 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 and Nieto-Reyes (2010) classify the Berkeley growth data (Tuddenham and Snyder 1954) by use of the random Tukey depth (Cuesta-Albertos and Nieto-Reyes 2008). Sguera et al. (2014) introduce a functional spatial depth and a kernelized version of it. Cuesta-Albertos et al. (2014) extensively study applications of the DD-plot to functional classification, employing a variety of functional depths.

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 function values. Second, several of these notions of functional depth break down in standard distributional settings, i.e., the depth functions vanish almost everywhere [see (Chakraborty and Chaudhuri 2014; Kuelbs and 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 (Lange et al. 2014a; Mozharovskyi et al. 2015)].

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 and Snyder 1954). It comprises the heights of 54 girls and 39 boys measured at 31 non-equally distant time points, see Fig. 1, left. This data was extensively studied in Ramsay and 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

![Fig. 1](image-url) Growth of 54 girls (red) and 39 boys (blue) (left); 35 observations of medflies fertility (right). The fat curve indicates a single member of each class. (Color figure online)
band depth López-Pintado and 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 and Nieto-Reyes 2010)]. Recently, Sguera et al. (2014) apply functional spatial depth to this data set, and Cuesta-Albertos et al. (2014) perform DD-plot classification.

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 34-th 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 Fig. 1, right, with the linearly interpolated evaluations). This task is taken from Müller and Stadtmüller (2005), who demonstrate that the problem is quite challenging and cannot be satisfactorily solved by means of functional linear models.

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 involves a new notion of functional data depth. It is completely non-parametric, oriented to work with raw and irregularly sampled functional data, and does not involve heavy computations. It extends the idea of the location-slope depth [see (Mosler and Polyakova 2012)], and employs the depth-to-depth classification technique proposed by Li et al. (2012), the $DD_\alpha$-classification of Lange et al. (2014a) and the depth-based $kNN$, first suggested by Vencálek (2011).

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 $L$ respectively $S$ equally sized subintervals.

We interpolate the functions linearly; 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 $L$ and $S$ by cross-validation of the final classification error, which is very quickly done. Thus, the loss of information caused by the finite-dimensional representation of data, which cannot be avoided, is measured by the final error of our classification procedure and minimized. This contrasts standard approaches using PCA projections or projections on spline bases, which penalize the loss of information with respect to a functional subspace but not to the ability of discrimination.

To construct the classification rule, the training classes in $(L, S)$-space are transformed to a depth-depth plot [$DD$-plot, see (Li et al. 2012)], which is a two-dimensional set of points, each indicating the depth of a data point regarding the two training
classes. Finally, the classes are separated on the DD-plot by either applying a $k$-nearest-neighbor ($k$NN) rule (Vencálek 2011) or the $\alpha$-procedure (Lange et al. 2014a).

The resulting functional data depth, called integral location-slope depth has a number of advantages. It is based on a linear mapping which does not introduce any spurious information and can preserve Bayes optimality under standard distributional assumptions. Already in the first step, the construction of the finite-dimensional space aims at the minimization of an overall goal function, viz. the misclassification rate. The simplicity of the mapping allows for its efficient computation.

The procedure has been implemented in the R-package dddalpha.

1.3 Overview

The rest of the paper is organized as follows: Section 2 reviews the use of data depth techniques in classifying finite-dimensional objects. Section 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 Sect. 4 we briefly introduce two alternative classifiers that operate on the DD-plot, a nearest-neighbor procedure and the $\alpha$-procedure. Section 5 treats some theoretical properties of the developed classifiers. Section 6 provides an approach to bound and select the dimension of the location-slope space. In Sect. 7 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 8 concludes. Implementation details and additional experimental results are collected in the Appendix.

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 and Serfling (2000) for definition and properties. Here we mention three depths, Mahalanobis, spatial and projection depth. These depths are everywhere positive. Hence they do not produce outsiders, that is, points having zero depth in both training classes. This no-outsider property appears to be essential in obtaining functional depths for classification.

For a point $y \in \mathbb{R}^d$ and a random vector $Y$ having an empirical distribution on $\{y_1, \ldots, y_n\}$ in $\mathbb{R}^d$ the Mahalanobis depth (Mahalanobis 1936) of $y$ w.r.t. $Y$ is defined as

$$D_{Mah}(y|Y) = \left(1 + (y - \mu_Y)'\Sigma_Y^{-1}(y - \mu_Y)\right)^{-1},$$

where $\mu_Y$ measures the location of $Y$, and $\Sigma_Y$ the scatter.

The affine invariant spatial depth (Vardi and Zhang 2000; Serfling 2002) of $y$ regarding $Y$ is defined as

$$D_{Sp}^Y(y|Y) = 1 - \left\|E_Y \left[u\left(\Sigma_Y^{-1/2}(y - Y)\right)\right] \right\|,$$
where $\| \cdot \|$ denotes the Euclidean norm, $v(w) = \|w\|^{-1}w$ for $w \neq 0$ and $v(0) = 0$, and $\Sigma_Y$ is the covariance matrix of $Y$. $D^{Mah}$ and $D^{Spt}$ can be efficiently computed in $\mathbb{R}^d$.

The projection depth (Zuo and Serfling 2000) of $y$ regarding $Y$ is given by

$$D^{Prj}(y|Y) = \inf_{u \in S^{d-1}} \left(1 + O^{Prj}(y|Y, u)\right)^{-1},$$

with

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

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 and Zuo 2014) 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:** Ghosh and Chaudhuri (2005) propose the maximum depth classifier, which assigns an object to the 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; their scale parameters need to be tuned (usually by some kind of cross-validation) over the whole learning process. For this, Ghosh and Chaudhuri (2005) combine the naive maximum-depth classifier with an estimate of the density. A similar approach is pursued with projection depth in Dutta and Ghosh (2012a) and $l_p$ depth in Dutta and Ghosh (2012b), 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. Li et al. (2012) 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 Lange
et al. (2014a) these problems are addressed via the \( \alpha \)-procedure, which is very fast and speeds up the learning phase enormously. Use of other procedures in the DD-plot is discussed in Cuesta-Albertos et al. (2014).

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 smooth at all points besides a finite set, and let \( \mathcal{F} \) be endowed with the supremum norm. 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 \([\text{see (Ramsay and 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 easily. E.g. the medflies data, as to the egg-laying process, is naturally discrete; see Fig. 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 (Sect. 3.1). Then an \( (L+S) \)-dimensional depth is applied to the transformed data yielding a DD-plot in the unit square (Sect. 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 (Sect. 4).

3.1 The location-slope transform

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

\[
\left[ \tilde{x}_i(t_{i1}), \tilde{x}_i(t_{i2}), \ldots, \tilde{x}_i(t_{ik_i}) \right].
\] (5)

Assume w.l.o.g. \( \min_i t_{i1} = 0 \) and let \( T = \max_i t_{ik_i} \). From \( \tilde{x}_i \) a function \( x_i : [0, T] \rightarrow \mathbb{R} \) is obtained as follows. Connect the points \( (t_{ij}, \tilde{x}_i(t_{ij})), j = 1, \ldots, k_i, \) with line segments and set \( x_i(t) = \tilde{x}_i(t_{i1}) \) when \( 0 \leq t < t_{i1} \) and \( x_i(t) = \tilde{x}_i(t_{ik_i}) \) when \( t_{ik_i} \leq t \leq T \). By this, the data become piecewise linear functions on \([0, T]\), and their first derivatives become piecewise constant ones, see Fig. 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 Fig. 2 (left). Thus, our location-slope space has dimension \( L + S \). It delivers the following transform,
Fig. 2 Example: Transformation of a single growth function into the integrated location-slope space with \( L = 0 \) and \( S = 2 \). Function(s) and first derivative(s) (left); corresponding two-dimensional space \((0, 2)\) (right). (Color figure online)

\[
\mathbf{x}_i \mapsto \mathbf{y}_i = \left[ \int_0^{T/L} \mathbf{x}_i(t)dt, \ldots, \int_0^{T/(L-1)/L} \mathbf{x}_i(t)dt, \int_0^{T/S} \mathbf{x}_i'(t)dt, \ldots, \int_0^{T/(S-1)/S} \mathbf{x}_i'(t)dt \right]
\]

(6)

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, \( L + S \geq 1 \). In case \( L = 0 \) or \( S = 0 \) the formula (6) is properly modified. Put together we obtain a composite transform \( \phi : \mathcal{F} \to \mathbb{R}^{L+S} \),

\[
\tilde{\mathbf{x}}_i \mapsto \left[ \tilde{\mathbf{x}}_i(t_{i1}), \ldots, \tilde{\mathbf{x}}_i(t_{ik}) \right] \mapsto \mathbf{x}_i \mapsto \mathbf{y}_i,
\]

(7)

which we call the location-slope (LS-) transform. Note that also Cuevas et al. (2007) and Cuesta-Albertos et al. (2014) include derivatives in their functional depths, but in a different way.

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 Fig. 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 2, left, exhibits the value (height) and first derivative (velocity) of a single interpolated function, which is then represented by the average slopes on the two half-intervals, yielding the rectangular point in Fig. 2(right).

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 certain standard distributional assumptions, the LS-transform preserves asymptotic Bayes optimality; see Sect. 5.

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 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. Generally, we have 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 Sect. 7 that our classifier performs similar with the three depths when the dimension is low. In higher dimensions the projection depth cannot 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 Sect. 6.

3.2 The DD-transform

Denote the location-slope-transformed training classes in \( \mathbb{R}^{L+S} \) by \( Y_0 = \{y_1, \ldots, y_m\} \) and \( Y_1 = \{y_{m+1}, \ldots, y_{m+n}\} \). The DD-plot is then

\[
Z = \left\{ z_i = (z_{i0}, z_{i1}) \left| z_{i0} = D^{L+S}(y_i|Y_0), \quad z_{i1} = D^{L+S}(y_i|Y_1), \quad i = 1, \ldots, m + n \right. \right\}.
\]

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 Fig. 3. Clearly, in this case, almost faultless separation is achievable by drawing a straight line through the origin.

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 Sect. 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 converges to the Bayes rule under standard elliptical models; see Sect. 4.1. Note that \( kNN \) needs to be cross-validated on the DD-plot, which is computationally expensive. Therefore, secondly, we employ the
$DD\alpha$-procedure, which is a very fast heuristic; see Sect. 4.2. Although its theoretical convergence is not guaranteed [see (Lange et al. 2014a)], the $DD\alpha$-classifier performs very well in applications.

### 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 (Sect. 5) that, if the underlying distribution of each class is strictly unimodal elliptical with the same radial density, the $kNN$ classifier, operating on the $DD$-plot, achieves asymptotically the optimal Bayes risk.

Given a function $x_0 \in \mathcal{F}$ to be classified, we represent $x_0$ (according to Sect. 3) as $y_0 \in \mathbb{R}^{L+S}$ and then as $z_0 = (D^{L+S}(y_0 | Y_0), D^{L+S}(y_0 | Y_1))$. According to $kNN$ on the $DD$-plot $x_0$ is classified as follows:

$$\text{class}(x_0) = I \left( \sum_{i=1}^{m} I(z_i \in R_{z_0}^{\beta(k)}) < \sum_{i=m+1}^{m+n} I(z_i \in R_{z_0}^{\beta(k)}) \right),$$

where $I(S)$ denotes the indicator function of a set $S$, and $R_{z_0}^{\beta(k)}$ is the neighborhood of $z_0$ defined by the $k$-closest observations. We use the $L_\infty$-distance on the $DD$-plot; thus $R_{z_0}^{\beta(k)}$ is the smallest rectangle centered at $z_0$ that contains $k$ training observations. In applications $k$ has to be chosen, usually by means of cross-validation.

### 4.2 $DD\alpha$-classification

The second classification approach is the $DD\alpha$-classifier, introduced in Lange et al. (2014a). It uses a projective-invariant method called the $\alpha$-procedure (Vasil’ev and Lange 1998), which is a heuristic classification procedure that iteratively decreases the empirical risk. The $DD\alpha$-classifier employs a modified version of the $\alpha$-procedure to construct a nonlinear separator in $[0, 1]^2$ that classifies the depth-represented data points. The construction is based on depth values and the products of depth values (= $D$-features) up to some degree $p$ that can be either chosen $a$ priori or determined by cross-validation. In a stepwise way, linear discrimination is performed in subspaces of the feature space. In each step a pair of $D$-features is replaced by a new $D$-feature as long as the average misclassification rate decreases and basic $D$-features are left to be replaced. For details, we refer to Lange et al. (2014a).

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 (Lange et al. 2014a)] and performs well for broad classes of simulated distributions and a wide variety of real data [see (Lange et al. 2014b; Mozharovskyi et al. 2015)]. The main advantage of the $DD\alpha$-classifier is its high training speed, as it contains the $\alpha$-procedure which, on the $DD$-plot, has the quick-sort complexity $O((m + n) \log(m +}$
and proves to be very fast. The separating polynomial is constructed by space extensions of the DD-plot (which is of low dimension) and cross-validation.

5 Some theoretical properties of the classifiers

In this section, we treat optimality properties of classification procedures that operate on a DD-plot of LS-transformed functional data. Besides $k$NN-classification and $\alpha$-procedure they include linear and quadratic discriminant analysis as well as maximum depth classification, all applied to the DD-plotted data. The latter procedures will be included in the comparative study of Sect. 7.

First, in Sect. 5.1, it is shown that well-known properties regarding the Bayes optimality of classifiers for multivariate data carry over to LS-transformed functional data if the discretization points are fixed and $L$ is chosen large enough. Second, in Sect. 5.2, we introduce a sampling scheme that evaluates the functions at an increasing number of points, and establish Bayes optimality in the Gaussian case.

Let $\mathcal{F}$ be the space of real functions, defined on the finite interval $[0, T]$, which are continuous and piecewise smooth. Consider two stochastic processes, $G_0$ and $G_1$, on a common probability space $(\Omega, \mathcal{A}, P)$, whose paths are in $\mathcal{F}$ with probability 1. [For conditions on Gaussian processes that yield smooth paths, see e.g. Cambanis (1973).]

5.1 Fixed-discretization optimality

Let $\mathcal{T} = \{t_j | j = 1, \ldots, \ell \} \in [0, T]$ be a given finite set of discretization points, and for each $\tilde{v} \in \mathcal{F}$ let $v$ be its linear interpolation based on $\tilde{v}(t_1), \ldots, \tilde{v}(t_\ell)$, as described in Sect. 3. Denote the space of these linear interpolations by $\mathcal{F}(\mathcal{T})$. Then $\mathcal{F}(\mathcal{T})$ is a finite-dimensional linear space, which is isomorphic to $\mathbb{R}^\ell$. Consider two samples of independent random vectors in $\mathbb{R}^\ell$, $[\tilde{v}_i(t_1), \ldots, \tilde{v}_i(t_\ell)]$, distributed as $F_0$, $i = 1, \ldots, m$, and $[\tilde{w}_j(t_1), \ldots, \tilde{w}_j(t_\ell)]$, distributed as $F_1$, $j = 1, \ldots, n$, where $F_0$ and $F_1$ are the $\mathcal{T}$-marginal distributions of $G_0$ resp. $G_1$ in $\mathbb{R}^\ell$, and let the two samples be independent from each other.

Proposition 1 (Bayes optimality on $\mathcal{F}(\mathcal{T})$) Consider a class $\mathcal{C}$ of decision rules $\mathbb{R}^\ell \rightarrow \{0, 1\}$ and assume that $\mathcal{C}$ contains a sequence whose risk converges in probability to the Bayes risk. Then there exists a pair $(L^*, S^*)$ so that the same class of decision rules operating on LS-transformed data in $\mathbb{R}^{L+S}$ contains a sequence whose risk also converges in probability to the Bayes risk.

We say that a sequence of decision rules is Bayes optimal if their risk converges in probability to the Bayes risk. The proof of Proposition 1 is obvious: E.g. choose $S^* = 0$ and $L^* > T / \min_{i=1, \ldots, \ell-1} \{t_{i+1} - t_i\}$. Then the Bayes optimality trivially carries over from $\mathbb{R}^{L+S}$ to $\mathbb{R}^\ell$, hence to $\mathcal{F}(\mathcal{T})$. □

The Proposition allows for procedures that achieve error rates close to minimum. However note that it does not refer to Bayes optimality of classifying the underlying process, but just of classifying the $\ell$-dimensional marginal distributions corresponding to $\mathcal{T}$. In particular, if two processes have equal marginal distributions on $\mathcal{T}$, no discrimination is possible.
Corollary 1 (Optimality of QDA and LDA) Let $\mathcal{G}_0$ and $\mathcal{G}_1$ be Gaussian processes and the priors of class membership be equal, and let proper $L^*$ and $S^*$ be selected as above. The following rules operating on LS-transformed data are Bayes optimal, relative to the $T$-marginals:

(i) Quadratic discriminant analysis (QDA),
(ii) Linear discriminant analysis (LDA), if $\mathcal{G}_0$ and $\mathcal{G}_1$ have the same covariance function.

For a definition of LDA and QDA, [see, e.g. Devroye et al. (1996), Chap. 4.4]. Proof: As the processes are Gaussian, we obtain that

$$
(v_i(t_1), \ldots, v_i(t_k)) \sim \text{i.i.d.} \mathcal{N}(\mu_0, \Sigma_0), \quad i = 1, \ldots, m, \tag{10}
$$

$$
(w_j(t_1), \ldots, w_j(t_k)) \sim \text{i.i.d.} \mathcal{N}(\mu_1, \Sigma_1), \quad j = 1, \ldots, n. \tag{11}
$$

As by selecting $(L^*, S^*)$ the Bayes optimality is trivially preserved, the standard results of Fisher [see, again, Devroye et al. (1996)] apply; hence Corollary 1 holds.

The following result is taken from Lange et al. (2014a). Let $H$ be a hyperplane in $\mathbb{R}^\ell$, and let $\text{pr}_H$ denote the projection on $H$. A probability distribution $F_1$ is mentioned as the mirror image regarding $H$ of another probability distribution $F_0$ in $\mathbb{R}^\ell$ if for $X \sim F_0$ and $Y \sim F_1$ it holds: $Y - \text{pr}_H(Y)$ is distributed as $-(X - \text{pr}_H(X))$.

Proposition 2 (Lange et al. (2014a)) Let $F_0$ and $F_1$ be probability distributions in $\mathbb{R}^\ell$ having densities $f_0$ and $f_1$, and let $H$ be a hyperplane such that $F_1$ is the mirror image of $F_0$ with respect to $H$ and $f_0 \geq f_1$ in one of the half-spaces generated by $H$. Then, based on a 50:50 independent sample from $F_0$ and $F_1$, the DD-$\alpha$-procedure will asymptotically yield the linear separator that corresponds to the bisecting line of the DD-plot.

Notice that due to the mirror symmetry of the distributions in $\mathbb{R}^\ell$ the DD-plot is symmetrically distributed as well. Symmetry axis is the bisector, which is obviously the result of the $\alpha$-procedure when the sample is large enough. Then the DD-$\alpha$-rule corresponds to the Bayes rule. In particular, the requirements of the proposition are satisfied if $F_0$ and $F_1$ are mirror symmetric and unimodal.

A stochastic process $\{X_t\}$ is mentioned as a strictly unimodal elliptical process if its finite-dimensional marginals are elliptical with the same radial density $\varphi$, that is $(v_i(t_1), \ldots, v_i(t_k)) \sim \text{Ell}(a(t_1, \ldots, t_k), S(t_1, \ldots, t_k), \varphi)$, and $\varphi$ is strictly decreasing. Here, $a$ denotes the shift vector and $S$ the structural matrix.

Corollary 2 (Optimality of $DD_k$- and $DD_{\alpha}$-rules) Assume that the processes $\mathcal{G}_0$ and $\mathcal{G}_1$ are strictly unimodal elliptical and have the same radial density, and choose $L^*$ and $S^*$ as above. Then

(a) the following rules operating on LS-transformed data are Bayes optimal, relative to the $T$-marginals:

(i) $DD_k$-M,
(ii) $DD_k$-S,
(iii) $DD_k$-P,
as \( m, n, k \to \infty \) with \( \frac{k}{m} \to 0 \) and \( \frac{k}{n} \to 0 \).

(b) If, in addition, the structural matrices coincide, \( \mathbf{S}_0(T) = \mathbf{S}_1(T) \), and the priors of class membership are equal, then the following rules operating on LS-transformed data are Bayes optimal, relative to the \( T \)-marginals:

(iv) the maximum-depth rule,
(v) the \( DD\alpha\)-M rule,
(vi) the \( DD\alpha\)-S rule,
(vii) the \( DD\alpha\)-P rule,

as \( m, n \to \infty \).

Proof Parts (i) to (iii) follow from Proposition 1 above and Theorem 3.5 in Vencálek (2011). Parts (v) to (vii) of the Corollary 2 are a consequence of Proposition 1 and Proposition 2. Part (iv) is deduced from Proposition 1 and Ghosh and Chaudhuri (2005), 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.

The above optimality results regard rules that operate on a \( DD \)-plot of LS-transformed data. For comparison, we will also apply the \( k\)NN-rule directly to LS-transformed data.

Corollary 3 (Optimality of \( k\)NN) The \( k\)NN-classifier applied to LS-transformed data (where \( L^* \) and \( S^* \) are selected as above) is Bayes optimal, relative to the \( T \)-marginals.

Proof This follows from Proposition 1 and the universal consistency of the \( k\)NN rule; see Devroye et al. (1996, Chap. 11).

5.2 Functional optimality

Next let us consider the asymptotic behavior of our procedure regarding the functions themselves. The above results refer to Bayes optimality with respect to the \( \ell \)-dimensional marginal distributions of the two processes that correspond to a fixed set \( T \) of \( \ell \) discretization points. Obviously they allow no inference on the whole functions.

In order to classify asymptotically with respect to the processes, we introduce a sequential sampling scheme that evaluates the functions at an increasing number of discretization points. For this, a slightly different notation will be needed.

For each \( k \in \mathbb{N} \) consider a set of discretization points, \( \mathcal{T}_k = \{ t_{k,1}, t_{k,2}, \ldots, t_{k,\ell_k} \} \), such that \( \mathcal{T}_k \subset \mathcal{T}_{k+1}, t_{k,0} = 0, t_{k,\ell_k} = T, t_{k,j+1} > t_{k,j} \) for all \( j \), and

\[
\max_{j=0,\ldots,\ell_k} |t_{k,j+1} - t_{k,j}| \to 0 \quad \text{when } k \to \infty. \tag{12}
\]

For \( k \in \mathbb{N} \) assume that \( \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \ldots, \tilde{\mathbf{v}}_k \) are i.i.d. sampled paths from \( \tilde{\mathbf{v}} \sim \mathcal{G}_0 \), and that each \( \tilde{\mathbf{v}}_j \) is observed at times \( t_{k,1}, \ldots, t_{k,\ell_k} \), \( j = 1, 2, \ldots, k \). Similarly assume that \( \tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \ldots, \tilde{\mathbf{w}}_k \) are i.i.d. from \( \tilde{\mathbf{w}} \sim \mathcal{G}_1 \), each \( \tilde{\mathbf{w}}_j \) being observed at times \( t_{k,1}, \ldots, t_{k,\ell_k} \), and that the two samples are independent of each other.
In this setting, we apply our classification procedure with \( L \) and \( S \) depending on \( k \). By cross-validation we select an optimal \( L_k \) from the set

\[
\mathcal{M}_k = \left\{ L \in \mathbb{N} | L \leq \frac{T}{\min_j (t_{k,j+1} - t_{k,j})} + 1 \right\}.
\]

For \( i = 1, 2, \ldots, k \), let \( v_i^k \) be the linearly interpolated function that is supported at \((t_{k,1}, v_i(t_{k,1})) \), \ldots , \((t_{k,\ell_k}, v_i(t_{k,\ell_k}))\), and define \( w_i^k \) similarly. \( \mathcal{G}_{0,k} \) denotes the empirical distribution on \( v_1^k \), \ldots , \( v_k^k \), and \( \mathcal{G}_{1,k} \) that on \( w_1^k \), \ldots , \( w_k^k \). Then, obviously, the classification based on the \( LS \)-transformation, with \( L_k \) being selected from \( \mathcal{M}_k \) via cross validation, is optimal regarding the empirical distributions \( \mathcal{G}_{0,k} \) and \( \mathcal{G}_{1,k} \).

Now we can extend Corollary 1 to the case of process classification:

**Theorem 1** (Bayes optimality with respect to Gaussian processes) Let \( \mathcal{G}_0 \) and \( \mathcal{G}_1 \) be Gaussian processes that have piecewise smooth paths, and let the priors of class membership be equal. Under the above sequential sampling scheme and if \( L_k \) is selected from \( \mathcal{M}_k \), the following rules operating on the \( LS \)-transformed data are almost surely Bayes optimal regarding the processes \( \mathcal{G}_0 \) and \( \mathcal{G}_1 \):

(i) quadratic discriminant analysis (QDA),
(ii) linear discriminant analysis (LDA), if \( \mathcal{G}_0 \) and \( \mathcal{G}_1 \) have the same covariance function.

**Proof** As the processes are Gaussian, we obtain that

\[
(v_i^k(t_{k,1}), \ldots , v_i^k(t_{k,\ell_k})) \sim (i.i.d.) N(\mu_{0,k}, \Sigma_{0,k}), \quad i = 1, \ldots , k, \tag{13}
\]

\[
(w_i^k(t_{k,1}), \ldots , w_i^k(t_{k,\ell_k})) \sim (i.i.d.) N(\mu_{1,k}, \Sigma_{1,k}), \quad i = 1, \ldots , k, \tag{14}
\]

where \( \mu_{j,k} \in \mathbb{R}^{\ell_k} \), and the \( \Sigma_{j,k} \) are \((\ell_k \times \ell_k)\)-covariance matrices, \( j = 0, 1 \). Observe that the optimal (i.e., crossvalidated) error rate converges to the probability of error. If \( L_k \) is selected from \( \mathcal{M}_k \), then, as in Corollary 1, the Bayes risk is achieved with respect to the empirical distributions. Theorem 1(ii) of Nagy and Gijbels and Hlubinka (2015) says that under the above sampling scheme it holds

\[
P[\mathcal{G}_{0,k} \rightarrow \mathcal{G}_0 \text{ weakly}] = 1 \quad \text{and} \quad P[\mathcal{G}_{1,k} \rightarrow \mathcal{G}_1 \text{ weakly}] = 1. \tag{15}
\]

As for every \( k \) the Bayes risk is achieved with respect to \( \mathcal{G}_{0,k} \) and \( \mathcal{G}_{1,k} \), (15) implies that, in the limit \((k \rightarrow \infty)\), the Bayes risk is almost surely achieved with respect to the processes \( \mathcal{G}_0 \) and \( \mathcal{G}_1 \).

\( \square \)

6 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 \(2 \leq L + S \leq M/2\). 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 Sect. 7.4 for computing times. The problem of selecting an appropriate pair \((L^*, S^*)\) 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 and 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 \(\Delta\epsilon\) from empirical risk small and allows for its implicit comparison with the empirical risk \(\epsilon\) 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 \(\alpha\)-procedure can be employed instead, we use LDA as it is fast and linear. Below we derive the Vapnik–Chervonenkis conservative bound for our particular case; this is similar to Theorem 5.1 in Vapnik and Chervonenkis (1974). Here we use another bound on the cardinality of the class of separating rules [see also (Devroye et al. 1996)].

Consider the LS-transformed data \(Y_0\) and \(Y_1 \subset \mathbb{R}^{L+S}\), as two samples of random vectors that are distributed as \(\mathcal{G}^{L+S}_0\) and \(\mathcal{G}^{L+S}_1\), respectively. Let \(\mathcal{L}\) be a class of separation rules \(r : \mathbb{R}^{L+S} \mapsto \{0, 1\}\), where two different rules separate \(Y_0 \cup Y_1\) into two different subsets. Then each \(r\) produces an expected classification error,

\[
\mathcal{E} (r, \mathcal{G}^{L+S}_0, \mathcal{G}^{L+S}_1) = \pi_0 \int_{\mathbb{R}^{L+S}} I(r(y) = 1) d\mathcal{G}^{L+S}_0(y) + \pi_1 \int_{\mathbb{R}^{L+S}} I(r(y) = 0) d\mathcal{G}^{L+S}_0(y), \tag{16}
\]

with \(\pi_0 + \pi_1 = 1\).
with \( \pi_0 \) and \( \pi_1 \) being prior probabilities of \( \mathcal{G}_0^{L+S} \) and \( \mathcal{G}_1^{L+S} \), and \( I(\cdot) \) being an indicator function. As mentioned above, \( \mathcal{E}(r, \mathcal{G}_0^{L+S}, \mathcal{G}_1^{L+S}) \) can be consistently estimated by leave-one-out cross-validation from \( Y_0 \) and \( Y_1 \). However, this estimate comes at substantial computational cost. On the other hand, based on the empirical risk \( \epsilon \), the expectation \( \mathcal{E}(r, \mathcal{G}_0^{L+S}, \mathcal{G}_1^{L+S}) \) can be bounded from above. The empirical risk is given by

\[
\epsilon(r, Y_0, Y_1) = \frac{1}{m + n} \left( \sum_{i=1}^{m} I(r(y_i) = 1) + \sum_{i=m+1}^{m+n} I(r(y_i) = 0) \right). \tag{17}
\]

As discussed before, we constrain \( \mathcal{L} \) to be the class of linear rules. Then the number of possible rules equals the number \#\( \mathcal{L} \) of different separations of \( Y_0 \cup Y_1 \) by hyperplanes \( \subset \mathbb{R}^{L+S} \). Note that \( C(N, d) = 2 \sum_{k=0}^{d-1} \binom{N-1}{k} \) is the number of possible different separations of \( N \) points in \( \mathbb{R}^d \) that can be achieved by hyperplanes containing the origin. Consequently, it holds \#\( \mathcal{L} \leq C(m + n, L + S + 1) \), and equality is attained if \( Y_0 \cup Y_1 \) are in general position. Applying Hoeffding’s (1963) inequality, the probability that the highest (over the class \( \mathcal{L} \)) excess of the classification error over the empirical risk will not exceed some positive value \( \Delta \epsilon \) can be bounded from above:

\[
P \left( \sup_{r \in \mathcal{L}} \mathcal{E}(r, \mathcal{G}_0^{L+S}, \mathcal{G}_1^{L+S}) - \epsilon(r, Y_0, Y_1) > \Delta \epsilon \right) \leq C(m + n, L + S + 1) e^{-2(m+n)(\Delta \epsilon)^2}. \tag{18}
\]

This allows to formulate the following proposition:

**Proposition 3** For the class of linear separating rules in \( \mathbb{R}^{L+S} \), it holds with probability \( 1 - \eta \):

\[
\mathcal{E}(r, \mathcal{G}_0^{L+S}, \mathcal{G}_1^{L+S}) \leq \epsilon(r, Y_0, Y_1) + \sqrt{\frac{\ln C(m + n, L + S + 1) - \ln \eta}{2(m + n)}}. \tag{19}
\]

**Proof** Requiring \( P \left( \sup_{r \in \mathcal{L}} \mathcal{E}(r, \mathcal{G}_0^{L+S}, \mathcal{G}_1^{L+S}) - \epsilon(r, Y_0, Y_1) > \Delta \epsilon \right) \leq \eta \) yields \( \eta = C(m + n, L + S + 1) e^{-2(m+n)(\Delta \epsilon)^2} \). Solving the last w.r.t. \( \Delta \epsilon \) gives \( \Delta \epsilon = \sqrt{\frac{\ln C(m+n,L+S+1)-\ln \eta}{2(m+n)}} \). Then, with probability \( 1 - \eta \), Eq. (19) follows immediately. \( \square \)

The probability \( \eta \) is fixed to some small constant. Here we use \( \eta = \frac{1}{m+n} \), given the sample size. The goodness of a certain choice of \( L \) and \( S \) is then measured by

\[
\epsilon_{\text{max}} = \epsilon + \sqrt{\frac{\ln \left( (m + n)C(m + n, L + S + 1) \right)}{2(m + n)}}. \tag{20}
\]

We restrict our choice of \( L \) and \( S \) to those that make \( \epsilon_{\text{max}} \) small. An intuitive justification is the following: \( \epsilon \) refers to empirical risk, while the second term penalizes the dimension, which balances fit and complexity.
Fast DD-classification of functional data

Fig. 4 Average misclassification error (ordinate) to maximal risk (abscissa) of the $DD\alpha$-classifier based on Mahalanobis depth for Model 1 (left) and Model 2 (right) of Cuevas et al. (2007)

Fig. 5 Average misclassification rate (ordinate) to maximal risk (abscissa) of the $DD\alpha$-classifier based on Mahalanobis depth for growth data (left) and medflies data (right)

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 Sect. 7.2. 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 $\epsilon_{\text{max}}$ and the average classification error (ACE) are calculated by averaging over 100 takes and plotted in Fig. 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 Fig. 5). Here all $(L, S)$-pairs are considered with $L + S \geq 2$ and $L + S \leq 16$.

As expected, Figs. 4 and 5 largely support the statement “the less $\epsilon_{\text{max}}$ the smaller the ACE”. Although for the challenging medflies data (Fig. 5, right) the plot remains a fuzzy scatter, a substantial part of points with small error have low values of $\epsilon_{\text{max}}$. [Even more, the $(L, S)$-pair with the smallest $\epsilon_{\text{max}}$ actually corresponds to the smallest error, see the point in the lower left corner.] Thus $\epsilon_{\text{max}}$ still guides us in configuring the location-slope space, as it is affirmed by our experimental results in Sect. 7.3. Computing $\epsilon_{\text{max}}$ involves a single calculation of the LDA-classifier (viz. to estimate $\epsilon$), which is done in negligible time. Then, the $(L, S)$-pair with the smallest $\epsilon_{\text{max}}$ can be picked. We restrict attention to a few $(L, S)$-pairs that deliver smallest values of $\epsilon_{\text{max}}$, and choose the best one by cross-validating over this tiny range. This new technique is employed here for space building. We abbreviate it as $\text{VCcrossLS}$. 
7 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 (Sect. 7.1) we present a simulation study (Sect. 7.2) and a benchmark study (Sect. 7.3), including a discussion of computation loads (Sect. 7.4). Implementation details of the experiments are provided in Appendix 1.

7.1 Competitors

The new classifiers are compared with four classification rules: linear (LDA) and quadratic (QDA) discriminant analysis, \(k\)-nearest-neighbors (\(k\)NN) 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 \(V_{CcrossLS}\) of Sect. 6. [One may argue that the \(\epsilon_{max}\)-based choice of \((L, S)\) is not generally suited for \(k\)NN, but it delivers comparable results in reasonable time, much faster than cross-validation over all \((L, S)\)-pairs.] Also, the four 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 \(k\)NN in our competitors as it is Bayes-risk consistent in the finite-dimensional setting and generally performs very well in applications. The \(k\)NN-classifier is applied to the location-slope data in its affine invariant form. It is then defined as in (9), but with the Mahalanobis distance (determined from the pooled data) in place of the \(L_\infty\)-distance. \(k\) is selected by cross-validation.

As further competitors we consider three \textit{maximum depth classifiers}. They are defined as

\[
\text{class}(x) = \arg \max_i \pi_i D(y|Y_i),
\]

with \(D\) being either Mahalanobis depth \(D^{Mah}\), or spatial depth \(D^{Spt}\), or projection depth \(D^{Prj}\). \(\pi_i\) denotes the prior probability for class \(i\). The priors are estimated by the class portions in the training set. This classifier is Bayes optimal if the data comes from an elliptical location-shift model with known priors. For technical and implementation details the reader is referred to Appendix 1.

7.2 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 \textit{Model 1}:
\[ X_0 = \{ x^0(t) | x^0(t) = 30(1 - t)t^{1.2} + u(t) \}, \]
\[ X_1 = \{ x^1(t) | x^1(t) = 30(1 - t)^{1.2}t + u(t) \}, \]

where \( u(t) \) is a Gaussian process with \( E[u(t)] = 0 \) and \( Cov[u(s), u(t)] = 0.2e^{-|s-t|^2/0.3} \), discretized at 51 equally distant points on \([0, 1]\) (\( M = 51 \)), see Fig. 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 7 (left) presents boxplots (over 100 takes) of error rates of twelve classifiers applied after transforming 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 \( L_\infty \) 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 (\( crossDHB \)) 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 14 in Appendix 2. 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 1 in Sect. 7.3.

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 3 in Appendix 2. The picture remains the same when the location-slope space is chosen using unrestricted cross-validation \( crossLS \), which yields no substantial improvement.
Fig. 7 Boxplots of error rates for Model 1 (left) and Model 2 (right), using Vapnik–Chervonenkis bound (top), cross-validation (middle) and componentwise method (bottom)

On the other hand, classifiers operating in an optimal argument subspace (cross-DHB) are outperformed by those employed in the location-slope space (crossLS, VCCrossLS), although their error rates are still low; see Fig. 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 1) prevents from choosing higher dimensions. Most often the dimensions two or three are chosen; see Table 11 in Appendix 2. Our procedure appears to be better, as by integrating more information is extracted from the functions, so that they become distinguishable.
Table 1  Error rates (in %) when selecting a proper location-slope dimension

| Data set | Growth data | Medflies data |
|----------|-------------|---------------|
|          | 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   |
| DDα-M    | 5.38        | 5.38          | 3.23          | 37.83     | 38.58   |
| DDα-S    | 5.38        | 5.38          | 6.45          | 38.95     | 40.26   |
| DDα-P    | 6.45        | 6.45          | 6.45          | 35.02     | 35.96   |

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

\[
X_0 = \left\{ x^0(t) \mid x^0(t) = 30(1 - t)t^2 + 0.5 \sin(20\pi t) + u(t) \right\}
\]

with \( u(t) \) and \( M = 51 \) as before. \( X_1 \) is an 8-knot spline approximation of \( X_0 \). See Fig. 6(right) for illustration.

The corresponding boxplots of the error rates are depicted in Fig. 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 DDα-classifiers perform comparably. The last two lack efficiency when employed with projection depth; as seen from Table 4 in Appendix 2, 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 and 8 in Appendix 2.

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 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 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 and 8 in Appendix 2.
7.3 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 and 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 and 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 employing the LDA-classifier on a $DD$-plot that is based on the $h$-mode depth applied to both location and slope. 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 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. It coincides with a best cross-validation result of Bafilo and Cuevas (2008). 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 5, 9 and 13 in Appendix 2 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 and Romo (2006) and Cuesta-Albertos and 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 Fig. 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 and 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 improve 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 1, columns captioned *medflies data* for the
errors). The role of the derivatives in building the location-slope space is emphasized in Tables 6 and 10 in Appendix 2, which show the frequencies at which the various $(L, S)$-pairs are selected. $\text{crossLS}$ is outperformed in most of the cases. We were not able to compare the componentwise approach $\text{crossDHB}$ as the computational load is too high. On an average, $\text{DD\alpha}$-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 $\text{crossLS}$, 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 $k\text{NN}$. ($k$ in $k\text{NN}$ and $\text{DD}$-plot-based $k\text{NN}$ has been determined by leave-one-out cross-validation.) For exact implementation details the reader is referred to Appendix 1.

### 7.4 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 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 ($\text{crossDHB}$) 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 1 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 $\text{DD}$-plot-based $k\text{NN}$-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 2 min. [This has been pointed by Delaigle et al. (2012) as well.]
Table 2  Average (median for componentwise classification=crossDHB) training and classification (in parentheses) times (in s), and numbers of cross-validations performed (in square brackets), estimated by leave-one-out cross-validation

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

8 Conclusions

An efficient nonparametric procedure has been introduced for binary classification of functional data. 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 \([0, 1]^2\).

Our procedure either outperforms or matches 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, (cf. Model 2) they are slightly outperformed by the componentwise classification method of Delaigle et al. (2012).

As the raw data are linearly interpolated, no spurious information is added. The core of our procedure is the new data-dependent construction of the location-slope space. We bound its dimension \(L + S\) by a Vapnik–Chervonenkis bound. The subsequent depth transformation into the unit square 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. While our procedure is designed for the analysis of functions that are piecewise smooth, it can be also used for functions that are just piecewise continuous (by simply fixing the parameter $S$ at 0). Obviously, higher order derivatives can be involved, too. But obtaining those requires smooth extrapolation, which affords additional computations and produces possibly spurious information.

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 (Lange et al. 2014a)].

Our space selection technique (incomplete cross-validation, restricted by a Vapnik–Chervonenkis 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.

In future research comparisons with other 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.

Acknowledgements We thank Dominik Liebl for his critical comments on an earlier version of the manuscript, as well as Ondrej Vencalek and Aurore Delaigle for their helpful remarks. The reading and suggestions of two referees are also gratefully acknowledged.

Appendix 1: Implementation details

In calculating the depths, $\mu_Y$ and $\Sigma_Y$ for the Mahalanobis depth have been determined by the usual moment estimates and similarly, $\Sigma_Y$ for the spatial depth. The projection depth has been approximated by drawing 1000 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 1000 directions are not enough, which becomes apparent from the analysis of Model 2 in Sect. 7.2. There the location-slope spaces chosen have dimension eight and higher; see also Tables 4 and 8 in Appendix 2. 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 2 and 14 in Appendix 2).

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

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 \(\epsilon_{\text{max}}\) 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 \(\epsilon_{\text{max}}\) is calculated. Then those five pairs are selected that have smallest \(\epsilon_{\text{max}}\). Here, tied values of \(\epsilon_{\text{max}}\) 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 2 and both synthetic models in Table 14 of Appendix 2. 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 \(\epsilon_{\text{max}}\). This is seen from Figs. 4 and 5. When determining an optimal \((L, S)\)-pair by \(\text{crossLS}\), the same set of \((L, S)\)-pairs is considered as with \(\text{VCcrossLS}\).

In implementing the componentwise method of finite-dimensional space synthesis (\(\text{crossDHB}\)) 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 \((\Delta t)\) 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 tenfold, except the leave-one-out cross-validations in determining \(k\) with both \(k\text{NN}\)-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 \(\text{crossDHB}\) 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 “\text{ddalpha}” (Mozharovskyi et al. 2015), 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 2 and 14 of Appendix 2. 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.

**Appendix 2: Additional tables**

See Tables 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14

**Table 3** Frequency (in %) of selected location-slope dimensions using the Vapnik–Chervonenkis bound; Model 1

| (L, S) | LDA | QDA | kNN | Max.depth | DD-kNN | DDα |
|--------|-----|-----|-----|-----------|--------|-----|
|        |     |     |     | 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** Frequency (in %) of selected location-slope dimensions using the Vapnik–Chervonenkis bound; Model 2

| (L, S) | LDA | QDA | kNN | Max.depth | DD-kNN | DDα |
|--------|-----|-----|-----|-----------|--------|-----|
|        |     |     |     | 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   | 0   | 5   | 1   | 1   | 3   | 0   | 0   | 3   |
| (18,8) | 1   | 0   | 0   | 0   | 2   | 0   | 0   | 0   | 0   | 0   | 0   | 5   |
| Others | 7   | 1   | 15  | 12  | 12  | 17  | 5   | 8   | 11  | 6   | 9   | 20  |
Table 5  Frequency (in %) of location-slope dimensions chosen using the Vapnik–Chervonenkis bound; growth data

| (L, S) | LDA | QDA | kNN | Max. depth | DD-kNN | DDα |
|--------|-----|-----|-----|------------|--------|-----|
|        |     |     |     | Mah. | Spt. | Prj. | 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   | 1.08 | 0    | 0    | 2.15  | 0    | 1.08  | 2.15 | 1.08 | 1.08 |
| (4,0)  | 0   | 0   | 0   | 5.38 | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |

Table 6  Frequency (in %) of location-slope dimensions chosen using the Vapnik–Chervonenkis bound; medflies data

| (L, S) | LDA | QDA | kNN | Max. depth | DD-kNN | DDα |
|--------|-----|-----|-----|------------|--------|-----|
|        |     |     |     | 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    | 1.69 | 0    | 0    | 0    | 0    | 0    |

Table 7  Frequency (in %) of selected location-slope dimensions using cross-validation; Model 1

| (L, S) | LDA | QDA | kNN | Max. depth | DD-kNN | DDα |
|--------|-----|-----|-----|------------|--------|-----|
|        |     |     |     | 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 8  Frequency (in %) of selected location-slope dimensions using cross-validation; Model 2

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

Table 9  Frequency (in %) of location-slope dimensions chosen using cross-validation; growth data

| (L, S) | LDA | QDA | kNN | Max.depth | DD-kNN | DDα |
|--------|-----|-----|-----|-----------|--------|-----|
|        | 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 |
| (1,2)  | 12.9 | 5.38 | 35.48 | 6.45 | 19.35 | 0   | 17.20 | 70.97 | 12.90 |
| (0,3)  | 0   | 0   | 0   | 21.51 | 4.30 | 29.03 | 26.88 | 2.15 | 1.08 |
| (2,2)  | 0   | 0   | 0   | 11.83 | 29.03 | 1.08 | 1.08 | 2.15 | 8.60 |
| (4,0)  | 0   | 0   | 0   | 13.98 | 2.15 | 0   | 5.38 | 1.08 | 4.30 |
| (1,3)  | 0   | 0   | 0   | 0   | 1.08 | 0   | 21.51 | 2.15 | 5.38 |
| (4,2)  | 0   | 0   | 0   | 0   | 0   | 0   | 0    | 7.53 | 1.08 |
| (2,3)  | 0   | 0   | 0   | 0   | 0   | 1.08 | 2.15 | 5.38 | 1.08 |
| (4,5)  | 0   | 0   | 0   | 0   | 0   | 0   | 0    | 2.15 | 0   |
| (4,6)  | 0   | 0   | 0   | 0   | 0   | 0   | 0    | 0    | 6.45 |
| (7,1)  | 0   | 0   | 0   | 0   | 0   | 5.38 | 0    | 0    | 0   |
| Others | 10  | 2.14 | 63.43 | 5.36 | 44.08 | 0   | 1.07 | 40.84 | 0   |
### Table 10  Frequency (in %) of location-slope dimensions chosen using cross-validation; medflies data

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

### Table 11  Frequency (in %) of selected dimensions using componentwise method; Model 1

| $(L, S)$ | LDA | QDA | $kNN$ | Max. depth | $DD-kNN$ | $DDa$ |
|----------|-----|-----|-------|------------|----------|-------|
|          | Mah. | Spt. | Mah. | Spt. | Mah. | Spt. | Mah. | Spt. |
| 2        | 50   | 55   | 45   | 45   | 50   | 46   | 47   | 50   |
| 3        | 41   | 38   | 46   | 46   | 43   | 48   | 40   | 42   |
| 4        | 9    | 7    | 9    | 9    | 7    | 6    | 13   | 8    |
Table 12 Frequency (in %) of selected dimensions using componentwise method; Model 2

| Dim | LDA | QDA | kNN | Max.depth | DD-kNN | DDα |
|-----|-----|-----|-----|-----------|--------|-----|
|     |     |     |     |           | 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         | 0      | 0   | 0   | 0   | 0   |

Table 13 Frequency (in %) of selected dimensions using componentwise method; growth data

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

Table 14 Average (median for componentwise classification = crossDHB) training and classification (in parentheses) times (in s), and numbers of cross-validations performed (in square brackets), over 100 tries

| Data set | Model 1 | Model 2 |
|----------|---------|---------|
|          | VCcrossLS | crossLS | crossDHB | VCcrossLS | crossLS | crossDHB |
| LDA      | 7.61     | 40.59   | 547.5    | 6.83      | 30.3    | 392.16   |
|          | (0.0011) | (0.0011) | (0.001) | (0.0012) | (0.0012) | (0.001) |
|          | 5.55     | 375     | 10759.5  | 8.58      | 375     | 7648.5   |
| QDA      | 7.59     | 39.44   | 503.5    | 6.74      | 29.72   | 411.23   |
|          | (0.0011) | (0.0011) | (0.001) | (0.0011) | (0.0011) | (0.001) |
|          | 5.53     | 375     | 9628     | 8.32      | 375     | 8009     |
| kNN      | 7.86     | 133.77  | 1263.76  | 7.63      | 63.16   | 489.18   |
|          | (0.0012) | (0.0012) | (0.001) | (0.0013) | (0.0014) | (0.001) |
|          | 5.27     | 375     | 12996    | 8.8       | 375     | 4886     |
| MD-M     | 7.38     | 7.53    | 101.05   | 6.22      | 7.11    | 58.45    |
|          | (0.0011) | (0.0011) | (0.001) | (0.0011) | (0.0011) | (0.001) |
|          | 5.46     | 375     | 10344.5  | 9.05      | 375     | 5884.5   |
| MD-S     | 7.48     | 16.98   | 259.75   | 6.38      | 14.36   | 169.54   |
|          | (0.0012) | (0.0012) | (0.001) | (0.0012) | (0.0012) | (0.0011) |
|          | 5.51     | 375     | 10113    | 8.35      | 375     | 6518.5   |
| MD-P     | 9.58     | 245.52  | –        | 10.76     | 195.61  | –        |
|          | (0.0017) | (0.0018) | (0.002) | (0.0021) | (0.0021) | –        |
|          | 5.56     | 375     | 8.96     | 375       |         |          |
Table 14 continued

| Data set | Model 1 | | | Model 2 | | |
|---|---|---|---|---|---|---|
| | VCCrossLS | crossLS | crossDHB | VCCrossLS | crossLS | crossDHB |
| **DDk-M** | 7.99 | 50.69 | 1421.2 | 7.26 | 48.95 | 601.2 |
| | (0.0012) | (0.0013) | (0.0011) | (0.0013) | (0.0013) | (0.001) |
| | [5.49] | [375] | [11909.5] | [9.1] | [375] | [5491] |
| **DDk-S** | 8.63 | 119.84 | 2493.52 | 8.71 | 102.75 | 1296.86 |
| | (0.0013) | (0.0014) | (0.0012) | (0.0014) | (0.0014) | (0.0012) |
| | [5.44] | [375] | [10873] | [9.68] | [375] | [5715.5] |
| **DDk-P** | 12.16 | 453.28 | – | 14.42 | 383.15 | – |
| | (0.0016) | (0.0016) | – | (0.0017) | (0.0018) | – |
| | [5.62] | [375] | – | [8.06] | [375] | – |
| **DDα-M** | 8.13 | 34.55 | 1866.99 | 7.93 | 76.58 | 995.68 |
| | (0.0012) | (0.0012) | (0.001) | (0.0012) | (0.0012) | (0.001) |
| | [5.57] | [375] | [10113] | [9.43] | [375] | [5182] |
| **DDα-S** | 8.68 | 104.71 | 2840.91 | 9.06 | 128.62 | 1860.12 |
| | (0.0012) | (0.0013) | (0.0011) | (0.0013) | (0.0013) | (0.0012) |
| | [5.51] | [375] | [9774] | [8.99] | [375] | [6124] |
| **DDα-P** | 12.19 | 466.76 | – | 16.02 | 410.66 | – |
| | (0.0015) | (0.0016) | – | (0.0017) | (0.0017) | – |
| | [5.45] | [375] | – | [8.92] | [375] | – |

References

Bailló A, Cuevas A (2008) Supervised functional classification: a theoretical remark and some comparisons. arXiv:0806.2831v1 [stat.ML]

Biau G, Bunea F, Wegkamp MH (2005) Functional classification in Hilbert spaces. IEEE Trans Inf Theory 51:2163–2172

Cambanis S (1973) On some continuity and differentiability properties of paths of Gaussian processes. J Multivar Anal 3:420–434

Carey JR, Liedo P, Müller H-G, Wang J-L, Chiu J-M (1998) Relationship of age patterns of fecundity to mortality, longevity, and lifetime reproduction in a large cohort of Mediterranean fruit fly females. J Gerontol 53A:B245–B251

Chakraborty A, Chaudhuri P (2014) On data depth in infinite dimensional spaces. Ann Inst Stat Math 66:303–324

Cuesta- Albertos JA, Febrero-Bande M, Oviedo de la Fuente M (2015) The DDG-classifier in the functional setting. arXiv:1501.00372 [stat.ME]

Cuesta-Albertos JA, Nieto-Reyes A (2008) The random Tukey depth. Comput Stat Data Anal 52:4979–4988

Cuesta-Albertos JA, Nieto-Reyes A (2010) Functional classification and the random Tukey depth. Practical issues. In: Borgelt C, Rodríguez GG, Trutschnig W, Lubiano MA, Angeles Gil M, Grzegorzewski P, Hryniewicz O (eds) Combining soft computing and statistical methods in data analysis. Springer, Berlin/Heidelberg, pp 123–130

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

Delaigle A, Hall P (2012) Achieving near-perfect classification for functional data. J R Stat Soc 74:267–286

Delaigle A, Hall P, Bathia N (2012) Componentwise classification and clustering of functional data. Biometrika 99:299–313
Devroye L, Györfi L, Lugosi G (1996) A probabilistic theory of pattern recognition. Springer, New York
Dutta S, Ghosh AK (2012a) On robust classification using projection depth. Ann Inst Stat Math 64:657–676
Dutta S, Ghosh AK (2012b) On classification based on $L_p$ depth with an adaptive choice of $p$. Technical Report Number R5/2011, Statistics and Mathematics Unit. Indian Statistical Institute, Kolkata
Ferraty F, Hall P, Vieu P (2010) Most-predictive design points for functional data predictors. Biometrika 94:807–824
Ferraty F, Vieu P (2003) Curves discrimination: a nonparametric functional approach. Comput Stat Data Anal 44:161–173
Ferraty F, Vieu P (2006) Nonparametric functional data analysis. Springer, New York
Ferré L, Villa N (2006) Multi-layer perceptron with functional inputs: an inverse regression approach. Scand J Stat 33:807–823
Fraiman R, Muniz G (2001) Trimmed means for functional data. TEST 10:419–440
Ghosh AK, Chaudhuri P (2005) On maximum depth and related classifiers. Scand J Stat 32:327–350
Hall P, Poskitt D, Presnell B (2001) A functional data-analytic approach to signal discrimination. Technometrics 43:1–9
Hoeffding W (1963) Probability inequalities for sums of bounded random variables. J Am Stat Assoc 58:13–30
Huang D-S, Zheng C-H (2006) Independent component analysis-based penalized discriminant method for tumor classification using gene expression data. Bioinformatics 22:1855–1862
James G, Hastie T (2001) Functional linear discriminant analysis for irregularly sampled curves. J R Stat Soc Ser B 63:533–550
Kuelbs J, Zinn J (2013) Concerns with functional depth. Lat Am J Probab Math Stat 10:831–855
Lange T, Mosler K, Mozharovskyi P (2014a) Fast nonparametric classification based on data depth. Stat Pap 55:49–69
Lange T, Mosler K, Mozharovskyi P (2014b). $DD_\alpha$-classification of asymmetric and fat-tailed data. In: Spiliopoulou M, Schmidt-Thieme L, Janning R (eds) Data analysis, machine learning and knowledge discovery. Springer, Berlin, pp 71–78
Leng XY, Müller H-G (2006) Classification using functional data analysis for temporal gene expression data. Bioinformatics 22:68–76
Li J, Cuesta-Albertos JA, Liu RY (2012) $D_D$-classifier: nonparametric classification procedure based on $DD$-plot. J Am Stat Assoc 107:737–753
Liu X, Zuo Y (2014) Computing projection depth and its associated estimators. Stat Comput 24:51–63
López-Pintado S, Romo J (2006) Depth-based classification for functional data. In: Liu R, Serfling R, Souvaine D (eds) Data depth: robust multivariate analysis. American Mathematical Society, Computational Geometry and Applications, pp 103–120
Mahalanobis P (1936) On the generalized distance in statistics. Proc Natl Acad India 12:49–55
Mosler K, Polyakova Y (2012) General notions of depth for functional data. arXiv:1208.1981v1 [stat.ME]
Mozharovskyi P, Mosler K, Lange T (2015) Classifying real-world data with the $DD_\alpha$-procedure. Adv Data Anal Classif 9:287–314
Müller H-G, Stadtmüller U (2005) Generalized functional linear models. Ann Stat 33:774–805
Nagy S, Gijbels I, Hlubinka D (2015) Weak convergence of discretely observed functional data with applications. J Multivar Anal. doi: 10.1016/j.jmva.2015.06.006
Ramsay JO, Silverman BW (2005) Functional data analysis. Springer series in statistics, 2nd edn. Springer, Berlin
Rossi F, Villa N (2006) Support vector machine for functional data classification. Neurocomputing 69:730–742
Serfling R (2002) A depth function and a scale curve based on spatial quantiles. In: Y Dodge (ed) Statistics and data analysis based on $L_1$-norm and related methods. Birkhäuser, pp 25–38
Sguera C, Galeano P, Lillo RE (2014) Spatial depth-based classification for functional data. TEST 23:725–750
Tian ST, James G (2013) Interpretable dimensionality reduction for classifying functional data. Comput Stat Data Anal 57:282–296
Tuddenham R, Snyder M (1954) Physical growth of California boys and girls from birth to eighteen years. University of California Press, Berkeley
Vapnik VN, Chervonenkis A (1974) Teorija raspoznavanija obrazov (statisticheskie problemy obuchenija) (The theory of pattern recognition (statistical learning problems), in Russian). Nauka, Moscow
Vardi Y, Zhang CH (2000) The multivariate $L_1$-median and associated data depth. Proc Natl Acad Sci USA 97:1423–1426
Vasil’ev VI, Lange T (1998) The duality principle in learning for pattern recognition (in Russian). Kibern i Vytschislit’elnaya Tech 121:7–16
Vencálek (2011) Weighted data depth and depth based discrimination. Doctoral thesis. Charles University, Prague
Wang XH, Ray S, Mallick BK (2007) Bayesian curve classification using wavelets. J Am Stat Assoc 102:962–973
Zuo YJ, Serfling R (2000) General notions of statistical depth function. Ann Stat 28:461–482