Maierhofer, Thomas:

Classification of Functional Data Interpretable Ensemble Approaches

Masterarbeit, Sommersemester 2017

Fakultät für Mathematik, Informatik und Statistik

Ludwig-Maximilians-Universität München

https://doi.org/10.5282/ubm/epub.41008

Ludwig-Maximilians-Universität München
Faculty of Mathematics, Informatics, and Statistics

Classification of Functional Data

Interpretable Ensemble Approaches

Author:
Thomas Maierhofer

Supervisor:
Prof. Dr. Sonja Greven

Master's Thesis
in Statistics (MSc.)

July 21, 2017

Contents

Abstract

Classification of functional data is a fast growing area of research, driven by a need for methods to deal with the increasing availability of functional data. With the generation of functional data across diverse fields, research questions requiring classification of functional data naturally arise. A large proportion of models proposed in the literature are black box models. At best, these models allow an indirect interpretation by means of examining their predictions (for example, through partial prediction plots). This is unsatisfying in applications where not only the prediction, but also the underlying mechanism of the process are of interest. Further, interpretable models facilitate a plausibility check because a human is able to understand their inner workings.

The focus of this thesis is interpretable ensemble approaches to functional data classification. Ensemble methods tend to outperform individual models due to a more robust prediction, which is achieved by aggregating over several base models' predictions. Two different interpretable ensemble methods, the nearest neighbor ensemble and the random forest ensemble, are introduced and implemented into publicly available software. A user-friendly interface facilitates the application of these highly flexible modeling methods to new use cases. Both ensemble methods are based on combining multiple nearest neighbor estimators. The nearest neighbor ensemble works by creating a weighted mean of the predictions of its base models. The random forest ensemble creates nonlinear combinations of the base models' predictions by using the prediction probabilities for each class as explanatory variables. The random forest ensemble also allows multivariate covariables to be added into the ensemble next to the predictions of the base models. Due to the more complex structure of the random forest ensemble, interpretation of the model is less direct. While the base models' weights are interpreted in the nearest neighbor estimator, their variable importance is interpreted in the random forest.

Practicality of the proposed models and their software implementation is demonstrated using simulated data and real-world data from the UCR Time Series Classification Repository. A series of benchmark experiments show that the nearest neighbor ensemble does not generally outperform the best base model it contains or reference classification methods. In contrast, the random forest ensemble exhibits outstanding predictive performance in simulated and real-world data sets.

1 Introduction

Statistical methods for Functional Data Analysis (FDA) have become increasingly important in recent years. FDA is a field of statistics dealing with curves, surfaces and any other data with continuous support (Ramsay and Silverman, 2006). Applications span many disparate fields such as archeology, medicine, and engineering. Technical advances that allow measurements to be taken in a fully automated manner at high frequencies have resulted in vast amounts of data with functional features. New methods are required to cope with the peculiarities of this growing fund of data. Classifying functional data is a common problem across disciplines, for example using heart rhythms to classify pathological cardiovascular conditions (Bortolan and Willems, 1993), shape curves to determine the provenance of an artifact (Thomas, 1981), or pressure measurements to evaluate the efficiency of car assembly. Bagnall et al. (2016) provide an excellent overview of up to date methods for functional data classification using numerous data sets from a wide range of applications.

Functional data classification problems are differentiated from traditional classification problems because the attributes are entire functions, which are generally observed at a finite number of points. These functional features are ordered and may contain discriminatory information in their ordering (Wang et al., 2016). For these models with a high (theoretically infinite) number of covariables, sanity checks are particularly important to ensure sensible modeling of actual data structures. This thesis introduces two approaches to interpretable ensembles for functional data classification. The nearest neighbor ensemble, described in Fuchs et al. (2015), and the random forest ensemble.

The remainder of this thesis is structured as follows: Chapter 2 introduces the underlying concepts of classification, functional data, semimetrics, nearest neighbor estimators, nonparametric functional kernel estimators, and random forests. Building on this, Chapter 3 presents and theoretically compares the nearest neighbor ensemble from Fuchs et al. (2015) and the random forest ensemble. A brief overview of the novel software implementation for the classification methods is given in Chapter 4. This implementation is used for an empirical comparison of the nearest neighbor ensemble and the random forest ensemble on simulated and real data, the results of which are reported in Chapter 5. The thesis concludes in a final summary and discussion of the results in Chapter 6.

All analysis was conducted using the software package **R** (R Core Team, 2016). Figures were created using the **R**-package `ggplot2` (Wickham, 2009).

2 Background

This chapter introduces the main concepts of functional data (Section 2.1) and classification (Section 2.2) and gives application-oriented examples. Furthermore, semimetrics are defined and exemplified in Section 2.3. Building on these ideas, two classification methods for functional data are introduced, the nearest neighbor estimator for functional data (Section 2.4), and the nonparametric functional kernel

estimator (Section 2.5). Section 2.6 gives an informal introduction to classification trees and random forests. The concepts introduced in this chapter form the basis for the proposed modeling structures.

2.1 Functional Data

Ramsay and Silverman (2006) define functional data as the field of statistics dealing with curves, surfaces and all data with continuous support. A functional data observation is a function of this support. The characteristic feature is the continuity, meaning that the function can theoretically be evaluated at arbitrary points. There are numerous examples for such a continuous support like time, space, wavelengths, and angle. In real data, the underlying function is only evaluated at a discrete set of points. If observed on a regular grid, the finite set of observed points could be viewed as a multivariate data set, neglecting the information contained in the ordering of the data. The fact that the observed points are evaluations of one common function can be exploited to create better models.

This thesis uses the Berkley growth study (Tuddenham and Snyder, 1954) as an example for functional data classification. In this study, the height of 93 children (39 girls and 54 boys) was recorded from 0 to 18 years, see Figure 1. In addition to the original data, the first and second derivatives of the height are plotted. These carry the interpretation of the growth in cm/year and the acceleration in cm / year^2. The data set is contained in the **R**-package `fda` (Ramsay et al., 2014).

2.2 Classification

Classification deals with the assignment of new observations with unknown class to a set of predefined classes. A training data set, a set of observations with known class, is used to define a set of rules for new observations that map them onto the classes. Classification is useful in many fields of application. One example from medicine is the Pima Indians Diabetes Study Smith et al. (1988), in which participants are classified as healthy or diabetic based on a number of health indicators.

Transferring the concept of classification to functional data is straightforward. Instead of assigning observations with one or multiple characteristics to predefined classes, functions are assigned to predefined classes. The target variable to predict, the unknown class of an observation, stays the same for multivariate and functional data classification. Industrial applications for classification of functional data include the classification of cell based sensor chips (Fuchs et al., 2015) and detecting incorrectly assembled screws based on their angle-torque trajectory.

2.3 Semimetrics

This section introduces semimetrics as a distance measure and gives examples of useful semimetrics for functional data. Semimetrics are distance measures that define the similarity of two observations. Semimetrics are relevant to the problem at

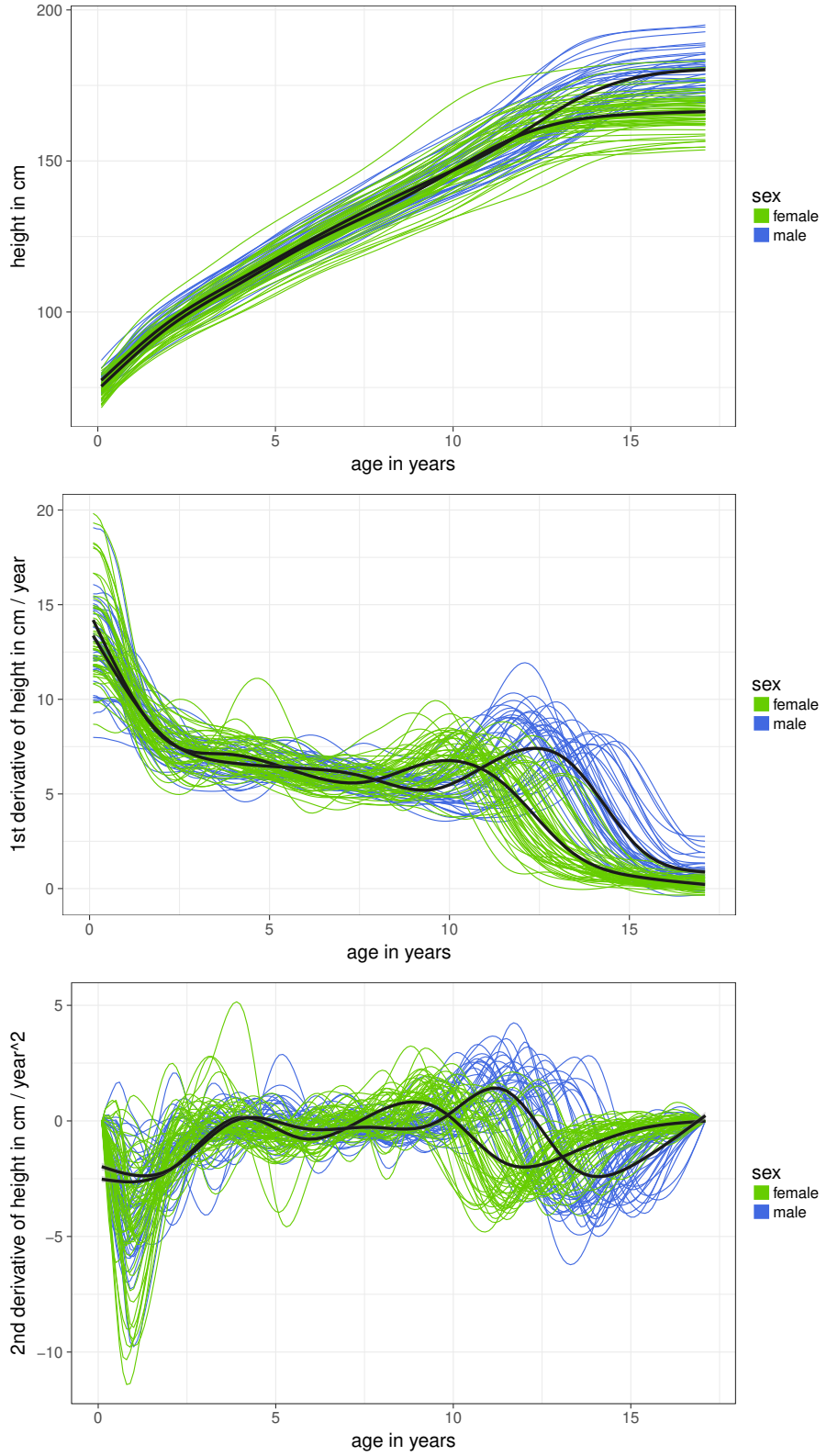

Figure 1: Illustration of the Berkley growth study. The growth curves of boys and girls are shown from 0 to 18 years.
Top: Original data; Middle: First order derivative; Bottom: Second order derivative.

hand, because they are used in nearest neighbor methods and functional kernel estimators for functional data classification. Because each semimetric captures different characteristics of the observations, using a variety of semimetrics makes it possible to optimally adapt the model to a particular classification problem. A function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a semimetric on a space $\mathcal{X}$ if:

- $\forall x \in \mathcal{X} : d(x, x) = 0$,
- $\forall x_1, x_2, x_3 \in \mathcal{X} : d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2)$

In contrast to metrics, semimetrics do not call for the additional property, that $\forall x_1, x_2 \in \mathcal{X} : d(x_1, x_2) = 0 => x_1 = x_2$. In this section we will look into semimetrics for one-dimensional regularly observed functional data, but many of them can easily be expanded to irregularly observed or multi-dimensional functional data.

Throughout the remainder of this thesis, $\mathcal{X}$ will denote the space of functions $x : \mathcal{T} \to \mathbb{R}$, for which the derivative of order a exists and is quadratically integrable, where $\mathcal{T}$ is an interval in $\mathbb{R}$. Using this notation, observations of functional data will be denoted by $x(t) \in \mathcal{X}$, with $t \in \mathcal{T}$.

Sections 2.3.1 through 2.3.9 introduce a variety of semimetrics. Each semimetric captures different features of functional data observations. The Euclidean distance (Section 2.3.1) and the Manhattan distance (Section 2.3.2) are $\mathcal{L}_p$ metrics that quantify the distance of entire functions. They are direct extensions of their multivariate counterparts. The global mean distance (Section 2.3.3) and the global minima/maxima distance (Section 2.3.4) measure the distance of the respective summary statistics of the curve. The relative areas distance (Section 2.3.5), the jump heights distance (Section 2.3.6), and the points of interest distance (Section 2.3.7) compare characteristics of the curves on predefined parts of the domain. Section 2.3.8 introduces the elastic semimetric, the amplitude distance, and the phase distance from the square root velocity framework as semimetrics that take warping into account. Another distance that is based on warping the functions is the dynamic time warping distance introduced in Section 2.3.9.

2.3.1 Euclidean Distance

The Euclidean distance of the derivatives of order a, denoted by $d^{(a)}_{\text{eucl}}(\cdot, \cdot)$, of two observations $x_1(t)$ und $x_2(t) \in \mathcal{X}$ is defined to be

$$d^{(a)}_{\text{eucl}}\big(x_1(t), x_2(t)\big) = \sqrt{\int_{\mathcal{T}} \Big(x_1^{(a)}(t) - x_2^{(a)}(t)\Big)^2 dt},$$

where $x^{(a)}$ denotes the derivative of order a of x. Note that for the special case of $a = 0$, $d^{(a)}_{\text{eucl}}(\cdot, \cdot)$ is a metric. For $a > 1$, $d^{(a)}_{\text{eucl}}(\cdot, \cdot)$ is only a semimetric, as the distance of two dissimilar observations $x_1(t), x_2(t)$ with $x_2(t) = x_1(t) + c$ equals 0. Additionally, the support $\mathcal{T}$ can also be restricted or weighted in order to create more concise semimetrics. This can be useful in applications, see Fuchs et al. (2015) for possible examples. For brevity of notation, the Euclidean norm $|| \cdot || = d^{(0)}_{eucl.}(\cdot, 0)$ is introduced, with 0 being the constant $\equiv 0$ function. This notation will be used throughout the remainder of this thesis.

2.3.2 Manhattan Distance

The Manhattan distance of the derivatives of order a, denoted by $d^{(a)}_{\text{eucl}}(\cdot,\cdot)$, of two observations $x_1(t)$ und $x_2(t) \in \mathcal{X}$ is defined to be

$$d^{(a)}_{\text{manh}}\big(x_1(t), x_2(t)\big) = \int_{\mathcal{T}} \left| x_1^{(a)}(t) - x_2^{(a)}(t) \right| dt,$$

where $|\cdot|$ denotes the absolute value function. This semimetric is similar to the Euclidean distance but less sensitive to large differences on small parts of the domain. Figure 2 visualizes the Manhattan distance for the average growth curves of boys and girls in the Berkley growth study. The Manhattan distance equates to the gray area in the figures.

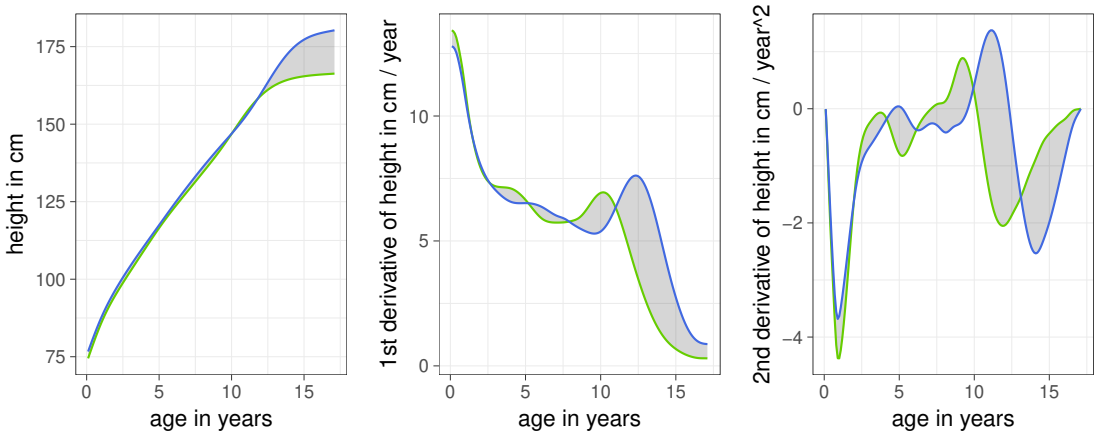

Figure 2: Manhattan distance of the mean growth curves and their derivatives for girls and boys in the Berkley growth study. The Manhattan distance is based on the vertical difference between the curves (depicted in gray).
Left: Age in years vs. height in cm; Middle: Age in years vs. growth in cm / year; Right: Age in years vs. acceleration in cm / year^2.

2.3.3 Global Mean Distance

The similarity of the global mean values between two observations x_1 and x_1 (or their derivatives of order a is denoted by

$$d^{(a)}_{\text{mean}}\big(x_1(t), x_2(t)\big) = \left| \int_{\mathcal{T}} x_1^{(a)}(t)\ dt - \int_{\mathcal{T}} x_2^{(a)}(t)\ dt \right|.$$

This semimetric can be used to quantify overall vertical shifts between two observations. Figure 3 visualizes the global mean distance together with the global maximum distance and the global minimum distance. The global mean distance between the mean function of the growth curves for boys and girls appears to be rather small.

2.3.4 Global Minima/Maxima Distance

The similarity between global minima/maxima of the derivatives of order a or the original curve for $a = 0$ is denoted by

$$\begin{aligned} d_a^{\min}\big(x_1(t), x_2(t)\big) &= \Big|\max_t\big(x_1^{(a)}(t)\big) - \max_t\big(x_2^{(a)}(t)\big)\Big| \\ d_a^{\max}\big(x_1(t), x_2(t)\big) &= \Big|\min_t\big(x_1^{(a)}(t)\big) - \min_t\big(x_2^{(a)}(t)\big)\Big|. \end{aligned}$$

This semimetric captures the similarity between the largest peak/valley of each observation for $a = 0$, the most extreme incline/decline for $a = 1$, and the strongest turnaround for $a = 2$. Note that the choice of representation and numerical derivation method can have a considerable impact on the extreme values of higher order derivatives. Figure 3 visualizes the global maximum distance and the global minimum distance together with the global mean distance for the Berkeley growth study. The mean height curves for boys and girls differ most in the global maximum distance on the original data and the global minimum distance on their second derivative.

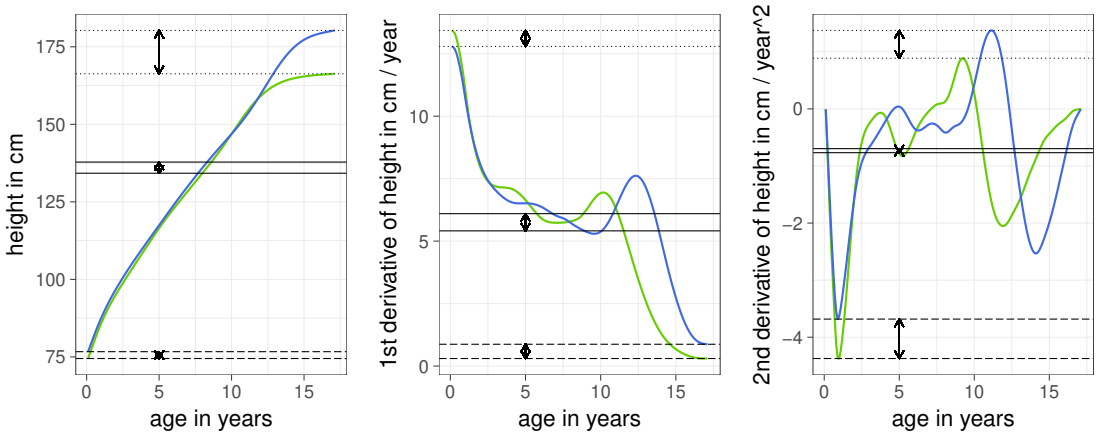

Figure 3: Global mean distance, global maximum distance and global minimum distance of the mean growth curves and their derivatives for girls and boys in the Berkley growth study. These semimetrics are based on the absolute difference in global mean (solid), global maximum (dotted), and global minimum (dashed), respectively. The length of the error equates to the value of the corresponding distance. Left: Age in years vs. height in cm; Middle: Age in years vs. growth in cm / year; Right: Age in years vs. acceleration in cm / year^2.

2.3.5 Relative Areas Distance

The difference of the relation between areas of two observations x_1 and x_1 (or their derivative of order a) on two parts of the domain $\mathcal{T}_1, \mathcal{T}_2 \subset \mathcal{T}$ is denoted by

$$d_{\text{relAreas}}^{(a)}\big(x_1(t), x_2(t)\big) = \left|\left|\frac{\int_{\mathcal{T}_1} x_1^{(a)}(t)\ dt}{\int_{\mathcal{T}_2} x_1^{(a)}(t)\ dt}\right| - \left|\frac{\int_{\mathcal{T}_1} x_1^{(a)}(t)\ dt}{\int_{\mathcal{T}_2} x_1^{(a)}(t)\ dt}\right|\right|.$$

This semimetric quantifies the similarity of the fraction between the means on partitions of the domain. The choice of partitions can be challenging in application.

2.3.6 Jump Heights Distance

The similarity between jump heights of two observations x_1 and x_1 (or their derivative of order a) on predefined points on the domain $t_n, t_m \in \mathcal{T}$ is denoted by

$$d_{\text{jump}}^{(a)}\big(x_1(t), x_2(t)\big) = \left|\left(x_1^{(a)}(t_n) - x_1^{(a)}(t_m)\right) - \left(x_2^{(a)}(t_n) - x_1^{(a)}(t_m)\right)\right|.$$

This semimetric is particularly useful if the application predetermines meaningful points t_n, t_m at which to compare the jump heights, e.g., the first and last observation of growth curves.

2.3.7 Points of Impact Distance

The differences of certain points of impacts $t_s, s = 1, \ldots, S$ of two observations x_1 and x_2 or their the derivatives of order a,

$$d_a^{\text{points}}\big(x_1(t), x_2(t)\big) \;=\; \frac{1}{S}\sum_{s=1}^{S}\left|x_1^{(a)}(t_s) - x_2^{(a)}(t_s)\right|,$$

are semimetrics. These are useful for applications where few and known points of the curves contain most of the information.

2.3.8 Semimetrics from Square Root Velocity Framework

This section introduces semimetrics for one-dimensional functional data based on the square root velocity (SRV) framework (Srivastava and Klassen, 2016), namely the elastic semimetric (which equates to the Fisher-Rao semimetric in the one dimensional case), the amplitude distance and the phase distance. These semimetrics differ vastly from all other semimetrics presented in Section 2.3 so far, as they use warping to quantify distances between functions.

Conceptual Introduction

The elastic semimetric measures the amount of bending and stretching needed to align two curves while being invariant to simultaneous warping of the curves. The variability between two one-dimensional functional data observations can be decomposed into the horizontal variability, which is measured by the phase distance, and the vertical variability, which is measured by the amplitude distance. The phase distance measures the amount of warping needed to align two functions. It quantifies the amount of stretching and compressing of the horizontal axis needed to minimize the vertical distance of the two functions. The amplitude distance corresponds to the distance between two functions once they are optimally aligned by warping. This distance neglects the amount of warping needed to align the functions

but quantifies the remaining vertical dissimilarity. By comparing the classification performance using the amplitude distance or the phase distance, conclusions can be drawn about whether the classes are more dissimilar in their horizontal or vertical axis.

Square Root Velocity Framework

Now, the SRV framework is formally introduced. Let $x_i(t)$ be a functional data observation that is absolutely continuous on $\mathcal{T}$. We define its square root velocity function to be

$$q_i(t) = \begin{cases} x_i^{(1)}(t)/\sqrt{|x_i^{(1)}(t)|}, & \text{if } |x_i^{(1)}(t)| \neq 0 \\ 0, & \text{otherwise,} \end{cases}$$

where $|\cdot|$ denotes the absolute value function. The square root velocity functions for the Berkley growth study data is depicted in Figure 4.

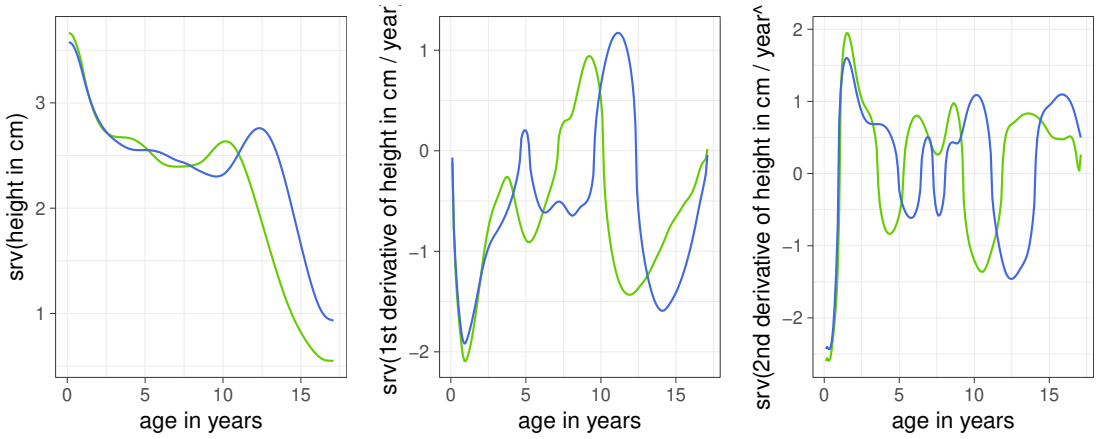

Figure 4: Square root velocity functions of the mean growth curves and their derivatives for girls and boys in the Berkley growth study.
Left: Age in years vs. square root velocity of height; Middle: Age in years vs. square root velocity of growth; Right: Age in years vs. square root velocity of acceleration.

Srivastava and Klassen (2016) show in Section 5.6.2 that the elastic semimetric of two observations $x_1(t), x_2(t)$ is equivalent to the Euclidean distance between their square root velocities $q_1(t), q_2(t)$

$$d_{el}\left(x_1(t), x_2(t)\right) = ||q_1 - q_2|| = \sqrt{\int_{\mathcal{T}} \left(q_1(t) - q_2(t)\right)^2 dt}, \tag{1}$$

for the special case that $a = \frac{1}{2}$ and $b = 1$, where the parameter a^2 penalizes the "bending" of the curve and b^2 the "stretching" needed to align the curves. Note that this still holds for multidimensional functions. Applying this result, we use Equation (1) as definition for the elastic semimetric. Additionally, they show in the same section that for one-dimensional functions the Fisher-Rao metric d_{FR} is equivalent to the Euclidean distance of the SRV functions and thus a special case of the elastic semimetric.

Warping Functions

For introducing the amplitude distance and the phase distance, the notion of warping functions is required. Warping functions map the support of a function monotonically back onto itself, but allow for distortion. Warping is used to stretch and compress a curve horizontally, usually to achieve a better alignment. Let Γ be the set of all warping functions $\gamma : \mathcal{T} \mapsto \mathcal{T}$, meaning all strictly monotonic increasing, differentiable functions on $\mathcal{T}$ with $\gamma(\min(\mathcal{T})) = \min(\mathcal{T})$ and $\gamma(\max(\mathcal{T})) = \max(\mathcal{T})$. Let (γ, q_i) denote the SRV function of the warped function $x_i \circ \gamma$ for $i = 1, 2$. The warping function that optimally aligns x_1 to x_2 is denoted by

$$\gamma^* = \underset{\gamma \in \Gamma}{\operatorname{argmin}}\ d_{el}(x_1 \circ \gamma, x_2) = \underset{\gamma \in \Gamma}{\operatorname{argmin}}\ ||(\gamma, q_1) - q_2||.$$

Thus $x_1 \circ \gamma^*$ and $x_2 \circ \mathrm{id}$, with id denoting the identity function, are optimally aligned. Let ψ^* denote the SRV function of γ^* and 1 (the constant $\equiv 1$ function) the SRV function of the identity function.

Amplitude Distance

Now the amplitude distance and the phase distance will be formally introduced. Both are semimetrics derived from the Square Root Velocity Framework, see Section 4.10 of Srivastava and Klassen (2016). For any two observations $x_1, x_2 \in \mathcal{X}$ and the corresponding SRV functions q_1, q_2, we define the amplitude distance d_{amp} to be:

$$\begin{aligned} d_{amp}(x_1, x_2) &= \inf_{\gamma_1, \gamma_2 \in \Gamma} (||(q_1, \gamma_1) - (q_2, \gamma_2)||) \\ &= ||(q_1, \gamma^*) - q_2|| \\ &= d_{el}(x_1 \circ \gamma^*, x_2). \end{aligned}$$

It is shown in Section 4.10.1 of Srivastava and Klassen (2016) that all properties of a semimetric are satisfied.

Phase Distance

Then, define the phase distance d_{ph} between x_1 and x_2 as

$$\begin{aligned} d_{ph}(x_1, x_2) &= \cos^{-1}\left(\langle \psi^*, 1 \rangle\right) \\ &= \cos^{-1}\left(\int_{\mathcal{T}} \psi^*(t) dt\right) \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner dot product. In Section 4.10.2 of Srivastava and Klassen (2016) it is proved that the phase distance satisfies all properties of a semimetric as well.

2.3.9 Dynamic Time Warping Distance

This section gives an introduction to dynamic time warping as a distance measure. The concept of dynamic time warping is to stretch and compress the functions horizontally (often along time, hence the name time warping), in order to minimize their vertical distance. This distance measure is useful in applications where the similarity of the functions, independent of variations in the horizontal dimension, is discriminatory. This makes it similar to the amplitude distance presented in the previous chapter.

For dynamic time warping, an optimal warping γ^* function between two functions is found that minimizes the distance between the warped functions x_1 and x_2,

$$\gamma^* = \underset{\gamma}{\operatorname{argmin}} ||x_1 \circ \gamma - x_2||.$$

Often a window constraint is applied to γ restricting the warping function around the identity function. Finding an optimal warping function tends to be time consuming, as the running time scales quadratically with the length of the support of the observations. The dynamic time warping distance is defined as

$$d_{DTW} = ||x_1 \circ \gamma^* - x_2||.$$

The dynamic time warping distance for two functions is visualized in Figure 5. It depicts the dynamic time warping distance of two functions next to the Euclidean distance of the same functions. The vertical lines connect the matching point. Note that dynamic time warping is not a semimetric, as it does not hold the triangle inequality. For the sake of completeness, it is still listed in this section.

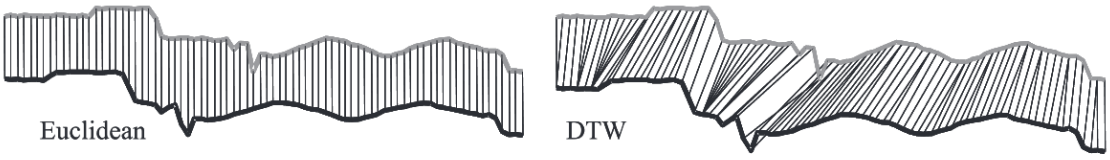

Figure 5: Euclidean distance and dynamic time warping distance between two functions. Vertical lines between the two functions connect the points that will be compared by the distance measure. (Source: Keogh and Ratanamahatana (2005))

2.4 Nearest Neighbor Estimator

The nearest neighbor estimator for functional data classification is the cornerstone of the nearest neighbor ensemble and the random forest ensemble presented in Chapter 3. The base models that are ensembled in these methods are nearest neighbor estimators.

Conceptual Introduction

The nearest neighbor estimator for functional data is a straightforward generalization of the multivariate nearest neighbor estimator. For a new observation, the distance to all observations in the training data is computed using a (semi-)metric. The unknown class of the new observation is assigned to the most prevalent class among the k nearest neighbors, where the k nearest neighbors are the observations in the training data with the smallest distance to the new observation. This procedure is illustrated in Figure 6. It visualizes a k nearest neighbor estimator for two-dimensional observations using the Euclidean distance.

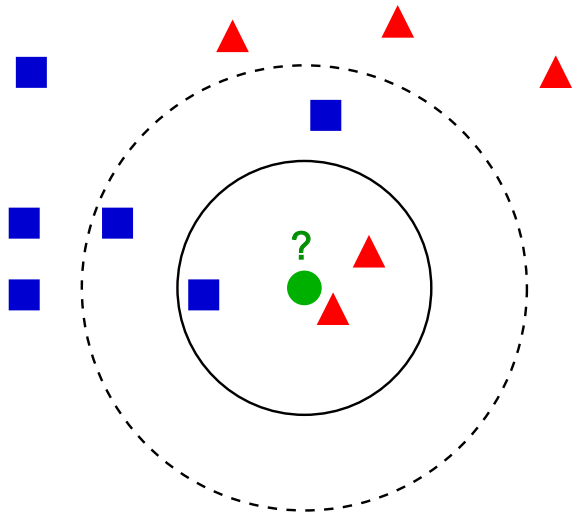

Figure 6: Illustration of k nearest neighbor classification for two-dimensional observations. For $k = 3$ nearest neighbors the class "red triangle" is predicted (with probablity $\frac{2}{3}$), for $k = 5$ nearest neighbors the class "blue square" is predicted (with probability $\frac{3}{5}$).
(Source: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)

Nearest Neighbor Estimator for Functional Data

Firstly, we look at how to create a nearest neighbor estimator for one semimetric $d(\cdot, \cdot)$ and one covariable $x(t)$. In order to create this nearest neighbor estimator, the observations in the training set $(y_i, x_i(t)), i = 1, \ldots, N$, are ordered according to their distance to the new observation $(y^*, x^*(t))$ with respect to the semimetric $d(\cdot, \cdot)$. This results in the ordered observations $(y_{(i)}, x_{(i)}(t)), i = 1, \ldots, N,$, for which

$$d(x^*(t), x_{(1)}(t)) \leq \ldots \leq d(x^*(t), x_{(k)}(t)) \leq \ldots \leq d(x^*(t), x_{(N)}(t))$$

holds. The neighborhood $\mathcal{N}(x^*(t))$ of the k nearest neighbors of $x^*(t)$ is defined as

$$\mathcal{N}(x^*(t)) = \{x_i(t) : d(x^*(t), x_i(t)) \leq d(x^*(t), x_{(k)}(t))\}.$$

The resulting nearest neighbor estimator for the probability of the new observation $x^*(t)$ being in class $g \in G$ is

$$\hat{\pi}_g = \frac{1}{k} \sum_{x_i(t) \in \mathcal{N}(x^*(t))} I(y_i = g). \tag{2}$$

The unknown y^* is estimated by selecting the class with highest probability,

$$\hat{y}^* = \underset{g \in G}{\operatorname{argmax}}(\hat{\pi}_g).$$

2.5 Nonparametric Functional Kernel Estimator

The nonparametric functional kernel estimator (Ferraty and Vieu, 2003) is a method for functional data classification that extends the principle of the nearest neighbor estimator. Instead of using a simple majority vote of the k nearest neighbors, a kernel estimator weights the classes of the closest observations in the training data.

Kernel Functions

A kernel function K is a non-negative real-valued integrable function that is normalized,

$$\int_{-\infty}^{+\infty} K(u)\, du = 1,$$

and symmetric,

$$K(u) = K(u)\, \forall u \in \mathbb{R}.$$

For every kernel function K, all functions $K^*(u) = hK(hu)$, where $h > 0$ are kernel functions as well. This allows the introduction of the bandwidth parameter h, which can be used to scale the kernel function appropriately to the data. A kernel function with bandwidth $h = 2$ is twice as wide as the same kernel function with bandwidth $h = 1$. In practice, the bandwidth h is often chosen using cross validation. However, it can be difficult to choose the best kernel function for a given dataset. A useful set of commonly used kernel functions is depicted in Figure 7.

Kernel Estimator for Functional Data

Let $(y_i, x_i(t)), i = 1, \ldots, N$, observations in the training data with known class $y_i \in 1, \ldots, G$, and $(y^*, x^*(t))$ a new observation to be classified. The distances between all curves are computed using a predetermined semimetric $d(\cdot, \cdot)$. The estimated probability $\hat{\pi}_{g,h}\big(x(t)\big)$ that an observation $x(t)$ belongs to class $g \in 1, \ldots, G$, given a fixed bandwidth h and a fixed positive kernel function $K(\cdot)$ is

$$\hat{\pi}_{g,h}\big(x(t)\big) = \frac{\sum_{i=1}^{N} I(y_i = g)\; K\Big(d\big(x(t), x_i(t)\big)/h\Big)}{\sum_{i=1}^{N} K\Big(d\big(x(t), x_i(t)\big)/h\Big)},$$

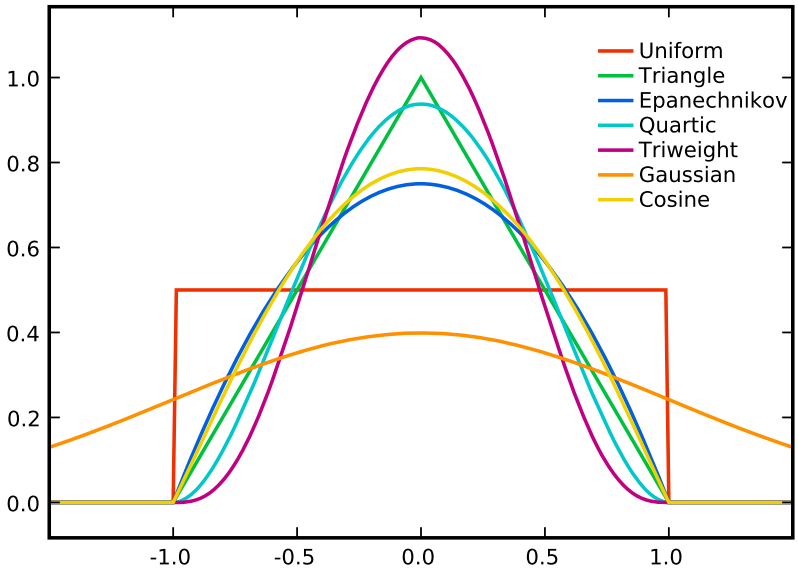

Figure 7: A selection of kernel functions with bandwidth $h = 1$. All kernel functions depicted are available for kernel estimators in the `classiFunc` package. (Source: `https://commons.wikimedia.org/wiki/File%3AKernels.svg`)

with $x_i(t), i = 1, \ldots, N$ the observations in the training data and the indicator function $I(g_1 = g_2) = 1 \Leftrightarrow g_1 = g_2$ and 0 otherwise. The kernel function determines the shape of the kernel and its width is determined by the bandwidth h. For predicting the class of a new observation, the class with the highest probability

$$\underset{g \in 1,\ldots,G}{\operatorname{argmax}} \hat{\pi}_{g,h}(x(t))$$

is selected.

2.6 Classification Trees and Random Forests

This section provides a conceptual introduction to classification trees (Breiman et al., 1984) and random forests (Breiman, 2001). Random Forests are one of the most commonly used machine learning methods because they display a high computational efficiency during training and prediction of new data while achieving a competitive prediction performance. Random Forests can be used for classification and regression problems, but we will focus on random forests for classification.

Classification Trees

A classification tree is created using an algorithm that recursively splits the data set into smaller and more homogeneous sub data sets, the nodes. The algorithm starts with the entire data set, the root node. Then, an optimal binary split of data set is computed, such that the observations in the two child nodes are maximally

heterogeneous. Those two child nodes are recursively split using the same procedure, until final nodes called leaves are reached. These leaves satisfy a stop criterion and will not be split any further. The resulting tree structure is depicted in the top left of Figure 8. There are four binary splits of the data set which result in five leaf nodes. The top right of Figure 8 is an alternative way of depicting the same classification tree. The axes represent the two covariables used in the tree. The background color equates to the proportions of the classes in the leaf nodes. The class of a new observation is predicted by running it through the tree until a leaf node is reached. Then the unknown class of the new observation is estimated by a majority vote of the classes for the training observations in this leaf node.

Random Forests

A random forest for classification is an ensemble of classification trees. The trees in a random forest are decorrelated by randomly choosing subsamples of covariables and observations to be used in each tree and binary split. To predict the class of a new observation, the random forest uses a majority vote of the classes predicted by its individual trees. The relevance of every explanatory variable for the model can be measured using variable importance. This measure uses the total decrease in node impurities from splitting on the variable, averaged over all classification trees in the random forest. The node impurity is measured by the Gini index (Gini, 1912).

A random forest cannot be depicted in one tree structure, as it generally consists of more than 200 individual trees. Instead, by aggregating over the predictions of the individual trees, every combination of the two covariables can be assigned to a class probability. This results in an analogous depiction of the random forest model (bottom of Figure 8) and the classification tree (top left of Figure 8).

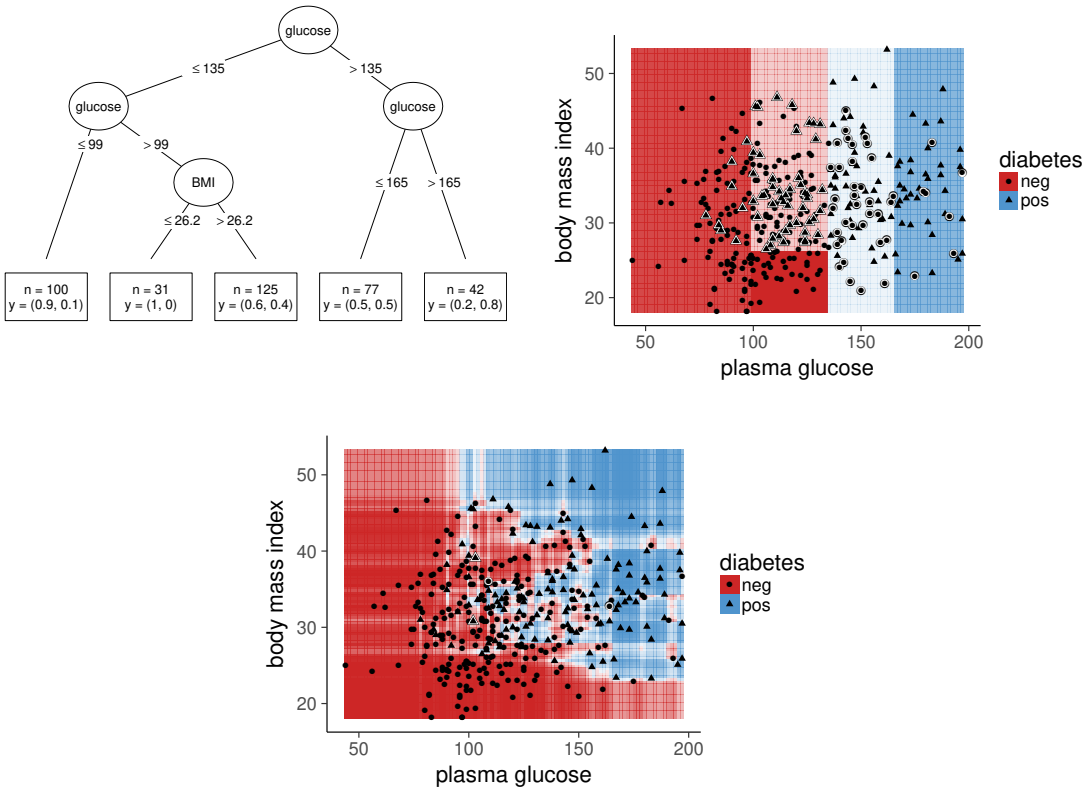

Figure 8: Classification tree for a subset of the Pima Indians Diabetes data set (`ftp.ics.uci.edu://pub/machine-learning-databases`). The body mass index and the plasma glucose are used to predict whether or not a person suffers from diabetes in a classification tree and a random forest.
Top left: Display of the classification tree in a tree structure; Top right: The classification tree in a planar display; Bottom: The random forest in a planar display. By combining several classification trees, "nonlinear" splits are achieved.

3 Ensembles for Functional Data Classification

This chapter introduces two interpretable approaches to functional data classification. The Nearest Neighbor Ensemble (Fuchs et al., 2015) is described in Section 3.1. It uses a weighted sum of the predictions of the underlying base estimators as an ensemble. Section 3.2 introduces an alternative ensemble based on the random forest model. The two modeling approaches are compared theoretically in Section 3.3. This section also gives a short outlook onto competing methods.

3.1 Nearest Neighbor Ensemble

Fuchs et al. (2015) introduces a novel approach for classification of functional data. It is based on an ensemble of prediction probabilities that were estimated using nearest neighbor classification. As an ensemble method a weighted sum is used.

The weights of the sum are estimated to minimize the Brier Score of the resulting ensemble classifier.

Conceptual Introduction

The basic idea of this approach is to compute numerous nearest neighbor estimations with different numbers of nearest neighbors and different semimetrics and combine them afterwards to an ensemble estimator using a weighted sum. In addition to the properties of a semimetric described in Section 2.3 the authors also ask that the semimetrics to be used in their nearest neighbor ensemble be symmetric, meaning $\forall x_1, x_2 \in \mathcal{X} : d(x_1, x_2) = d(x_2, x_1)$.

Ensembling Nearest Neighbor Estimators

The simple 1 nearest neighbor estimator presented in Section 2.4 is a special case of the nearest neighbor ensemble. There is only one estimator in the ensemble which receives weight 1. The generalization to a nearest neighbor ensemble is carried out by using p semimetrics instead of only one. For each semimetric the individual neighborhood $\mathcal{N}_l(x^*(t))$ is computed. The nearest neighbor estimator for each semimetric is defined analogously to (2) as

$$\hat{\pi}_{gl} = \frac{1}{k} \sum_{x_i(t) \in \mathcal{N}_l\left(x^*(t)\right)} I(y_i = g).$$

The ensemble estimator is defined as the weighted sum of the individual estimators, with

$$\hat{\pi}_g = \sum_{l=1}^{p} c_l \hat{\pi}_{gl}, \tag{3}$$

where the weights c_l are subject to the restrictions:

$$c_l \geq 0 \ \forall l \text{ and } \sum_{l=1}^{p} c_l = 1. \tag{4}$$

The unknown weights are chosen to minimize the Brier score of the resulting prediction. Instead of defining the ensemble estimator as a weighted sum of the base estimations, it could also be interpreted their weighted mean. For an ensemble consisting of two base estimators, Figure 9 shows the effect of different weights c_1 and c_2 onto the ensemble prediction. The ensemble estimation is always a linear combination of the base estimations. This is still valid for more than two base estimators

Brier Score

The Brier score Q is a measure how well a set of predictions $\hat{\pi}_{ig}$ fits the actual class affiliation of the predicted observations y_i. It is a proper scoring rule, which is

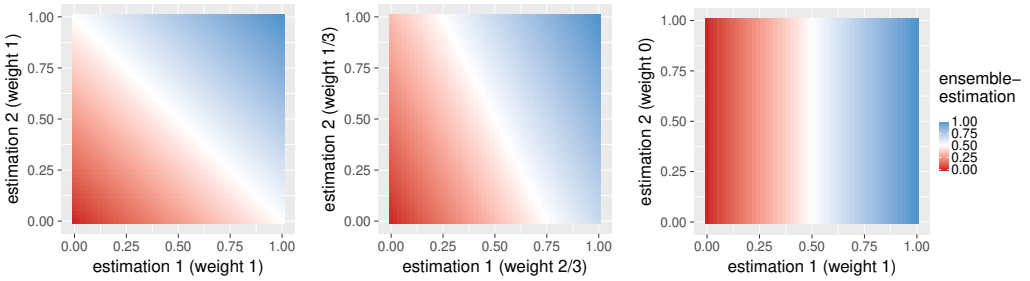

Figure 9: Effect of base estimation weightings on the ensemble estimation, for two base estimations and three sets of weights. The figure shows how the predicted ensemble estimation $\hat{\pi}_g$ depends on its base estimations $\hat{\pi}_{g1}$ and $\hat{\pi}_{g2}$.
Left: Equal weights of the estimators in the ensemble; Center: More weight on the first estimator; Right: All weight on the first estimator.

computed as the mean squared difference of the predictions and the truth,

$$Q = \frac{1}{N} \sum_{i=1}^{N} \sum_{g=1}^{G} (z_{ig} - \hat{\pi}_{ig})^2 \tag{5}$$

with $z_{ig} = 1$, if $y_i = g$, and $z_{ig} = 0$ for $y_i \neq g$. The Brier Score is used, because it is appropriately sensitive to changes in very small probabilities (Brier, 1950). An important result for the benchmark analysis in Chapter 5 is, that the Brier score for random guessing is 0.5 for a two class problem and 0.9 for a 10 class problem. Models performing worse than this upper bound are useless.

The Brier score of the ensemble can be rewritten in notation in dependence of the ensemble weights $\mathbf{c} = (c_1, \ldots, c_p)^T$ to be:

$$Q(\mathbf{c}) = \left(\underbrace{\mathbf{z}}_{N \cdot G \times 1} - \underbrace{\mathbf{P}}_{N \cdot G \times p} \underbrace{\mathbf{c}}_{p \times 1} \right)^T \left(\underbrace{\mathbf{z}}_{N \cdot G \times 1} - \underbrace{\mathbf{P}}_{N \cdot G \times p} \underbrace{\mathbf{c}}_{p \times 1} \right), \tag{6}$$

with $\mathbf{z} = (\mathbf{z_1} | \ldots | \mathbf{z_N})^T$, $\mathbf{z_i} = (z_{i1}, \ldots, z_{iG})^T$, $i = 1, \ldots, N$, $g = 1, \ldots, G$, and $\mathbf{P} = \left(\mathbf{P_1^T} | \ldots | \mathbf{P_N^T} \right)$, where

$$\mathbf{P_i} = \begin{bmatrix} \hat{\pi}_{11i} & \ldots & \hat{\pi}_{1pi} \\ \vdots & \ddots & \vdots \\ \hat{\pi}_{G1i} & \ldots & \hat{\pi}_{Gpi} \end{bmatrix}$$

contains the estimated probabilities π_{gli} for person i. These matrices contain the classes $g = 1, \ldots, G$, in their rows and all present combinations of semimetric, order derivative and number of nearest neighbors $l = 1, \ldots, p$ in their columns.

Estimation of Ensemble Weights

The π_{gli} are estimated by leave-one-out cross validation, as otherwise every observation would be its own nearest neighbor in the training data set. Other types of cross

validation can be used as well. Minimizing (6) with respect to the ensemble weights $\mathbf{c}$, $\min_c Q(\mathbf{c})$, under restrictions (4) implies a lasso-type penalty on the estimated weights $\hat{c} = \operatorname{argmin}_c Q(\mathbf{c})$. This results in some weights c_l being set to 0, eliminating combinations of semimetrics, order of derivatives and number of nearest neighbors in the ensemble. This allows a more sparse ensemble, an easier interpretation and a faster prediction of new test data. By implementing this optimization problem as a quadratic program, the ensemble weights can be computed efficiently using the **R**-package `limSolve` (Soetaert et al., 2009).

Ensemble Interpretation

In the nearest neighbor ensemble the weights c_l can be interpreted. These coefficients quantify the importance the base model in the ensemble, allowing informative conclusions about the model and the data. High ensemble weights on a semimetric are indicative of a high discriminatory power of the feature captured by this semimetric. A weight of 0 means that a semimetric does not contribute enough to the prediction to be included into the ensemble. When the same semimetric is included into the model with a different number of nearest neighbors, this can be interpreted similarly to a nonparametric kernel estimation of this semimetric. Different numbers of nearest neighbors of the same semimetric can be seen as a kind of non parametric kernel function. An exemplary interpretation is given in Figure 10, which shows the weights of the base models in the nearest neighbor ensemble for the Berkley growth study. The global maximum distance and the amplitude distance of the first derivative are detected as the most informative semimetrics for this task. The large weight on the maximum distance seem surprising, as the maxima of the mean curves of the first derivative are quite similar, see Figure 3, but looking at the individual observations in Figure 1 shows that the maximum of the first derivative is informative about the sex for a lot of children.

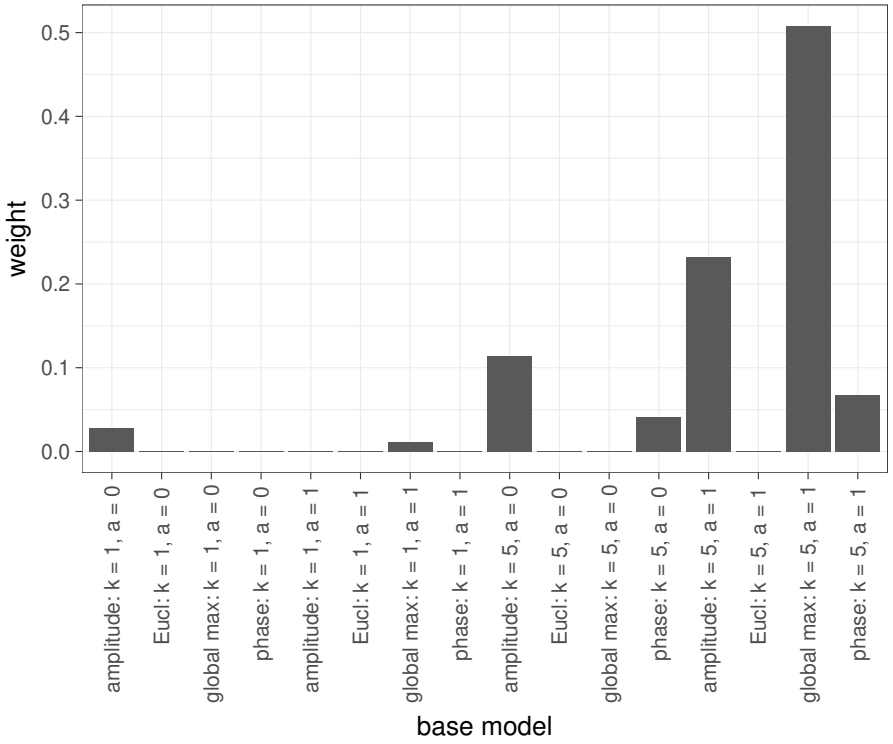

Figure 10: Weights of the base models in the nearest neighbor ensemble for the data of the Berkley growth study.

3.2 Random Forest Ensemble

This section introduces the random forest ensemble as an approach to classification of functional data. Instead of using a weighted sum of the individual nearest neighbor estimates, a nonlinear combination is used by applying a random forest. Goldsmith and Scheipl (2014) use a similar ensemble for a scalar-on-function regression.

Conceptual Introduction

The random forest model uses the predictions of the nearest neighbor estimators $\hat{\pi}_{gl}$ as predictor variables and the known class membership of the observations in the training data as a target variable. This allows more complex interactions of the nearest neighbor estimations and thus, the distances of the underlying semimetrics. Figure 9 shows how the weighted sum is restricted to linear combinations of the underlying predictions. Using a random forest expands on this by allowing nonlinear separations, see Figure 8 at the bottom.

A natural expansion is to not only use the nearest neighbor estimations as predictor variables in the random forest but also (non-functional) variables. These could be innately non-functional covariables from the data set or features generated from a functional covariable by applying an information compression procedure (e.g., functional principal component analysis (see Chapter 8 in Ramsay and Silverman (2006)), spline representation (Fahrmeir et al., 2013)), Fourier transformation

(Fourier, 1822) or computing meaningful features (e.g., location and value of the maxima/minima).

Ensemble Interpretation

In the weighted sum ensemble the weights c_l of the nearest neighbor estimations are interpreted. In the random forest ensemble the variable importance of the nearest neighbor estimations can be interpreted analogously. Analogously, the variable importance of the base models carries over directly onto the importance of the semimetric used in this base model in comparison with the semimetrics of the other base models. Figure 11 shows the relative variable importance of the base models in the random forest ensemble for the Berkley growth study. In contrast to the nearest neighbor ensemble, the random forest ensemble selects the original data and the first order derivative to be most informative in combination with the Euclidean distance for this classification task.

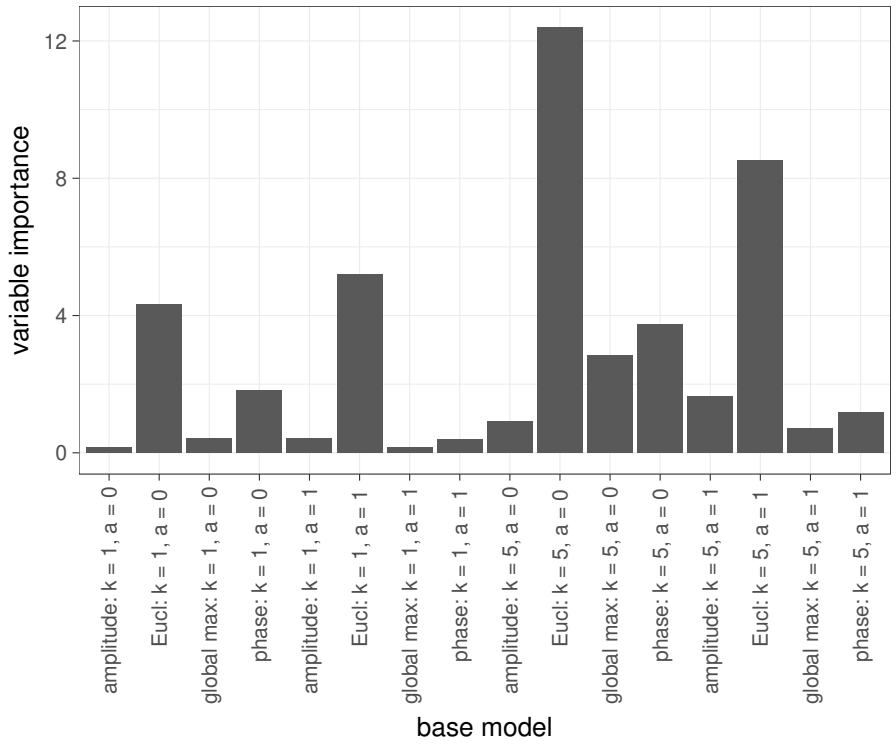

Figure 11: Variable importance of the base models in the random forest ensemble for the data of the Berkley growth study.

3.3 Theoretical Comparison and Competing Methods

This section highlights the most important commonalities and differences of the nearest neighbor ensemble and the random forest ensemble. Subsequently the methods are compared to established functional data classification methods.

Theoretical Comparison

Both the nearest neighbor ensemble and the random forest ensemble are interpretable classification methods for functional data. This means that they allow insights into how the prediction is carried out. Both methods are ensemble methods, that base their prediction on the predictions of base models. Fuchs et al. (2015) suggest to use exclusively nearest neighbor estimators as base models, but their ensemble is in no means limited to this class of base models. Any other type of functional data classification method can be used in the nearest neighbor ensemble, e.g., functional kernel estimators. In fact, the nearest neighbor ensemble is not even limited to classifying functional data. There is no reason not to use multivariate classification methods as base models. This allows the use of nearest neighbor ensembles for classification of multivariate data. The implementation of the nearest neighbor ensemble into software makes these additional features available, see Chapter 4 for a detailed description of the software implementation. The same is true for the random forest ensemble. It can be used as an ensemble method for functional data as well as multivariate data and any type of base model.

Competing Methods

In general, time series classification methods can be used on discretized functional data and functional data classification methods can be used for functionalized time series. A recent overview of time series classification methods is given in Bagnall et al. (2016). A strict separation between the methods used in either field of statistics is consequently pointless. Many methods for functional data classification are based on preceding dimension reduction procedures which allow the use of an arbitrary multivariate data classification algorithm by breaking the curse of (theoretically) infinite dimensionality. Examples for dimension reduction methods are principal component analysis (Besse et al., 1997), spline approximation (Eilers and Marx, 1996), shapelet transformation (e.g., Ye and Keogh (2011)), and computing summary statistics like mean and slope on sub-intervals of the support (e.g., Rodríguez et al. (2005)). The output of these procedures can then be used in any multivariate classification methods like logistic regression models (Fahrmeir et al., 1996) or in machine learning methods like decision trees, random forests (Breiman, 2001), and support vector machines (Suykens and Vandewalle, 1999). In a wider sense the ensembles introduced in this thesis can be seen as another case of this approach to functional data classification. By ignoring the fact that the base models already deliver individual predictions by themselves, and viewing the estimated class affiliations merely as multivariate covariables, the base models act as a dimension reduction technique. The second step of ensembling the base learners using either a weighted sum or a random forest, is nothing more than fitting a model that uses the predictions of the base estimators as multivariate covariables.

4 Software Implementation

This chapter outlines the implementation of the ensemble estimators, proposed in Chapter 3, in the software package **R**. The core functionality was implemented in the new **R**-package `classiFunc` (Maierhofer, 2017). The `mlr`-package (Bischl et al., 2016) is extended to embed the `classiFunc`-package and offer the proposed ensembles.

4.1 Creation of R-package `classiFunc`

The most important semimetrics for functional data and the nearest neighbor classification algorithm are reimplemented more efficiently in the new **R**-package `classiFunc` (Maierhofer, 2017). This easily extendable package also includes all semimetrics proposed in Fuchs et al. (2015) (see Section 2.3, e.g., short Euclidean distance, global minima/maxima distance, relative areas distance), semimetrics based on the SRV framework (amplitude distance and phase distance, see Section 2.3.8), all semimetrics from the **R**-package `proxy` (Meyer and Buchta, 2017), and semimetrics based on dynamic time warping (Giorgino et al., 2009). The re-implementation of the nearest neighbor algorithm in the `classiFunc`-package is approximately 40 times faster than the implementation of the `fda.usc`-package (Febrero-Bande and Oviedo de la Fuente, 2012) for small data sets. For large data sets, the increased scalability of the new implementation offers even larger improvements in speed over the existing implementation of nearest neighbor classification for functional data in **R**. This is enabled by both a more efficient calculation of the semimetrics and more efficient implementation of the nearest neighbor algorithm. For a hands-on introduction to the newly implemented software, see the vignette of the `classiFunc` package in Appendix B or on CRAN `https://cran.r-project.org/web/packages/classiFunc/vignettes/classiFunc.html`, where an extensive documentation is provided. The latest development version of the package is available on github (`https://github.com/maierhofert/classiFunc`). Please report all bugs that escaped the intense test routines there.

From its release on CRAN on May 29th 2017 until July 19th 2017 the `classiFunc`-package has been downloaded by 243 individual users from 90 countries. An overview of daily downloads is depicted in Figure 12. Users who downloaded the development version from github are not included. The considerable number of people who are interested in software for functional data classification speaks for the relevance of the topic.

4.2 Extension of R-package `mlr`

The existing machine learning framework of the package `mlr` is used to provide a comprehensive and flexible implementation of the nearest neighbor and random forest ensembles proposed in Chapter 3.

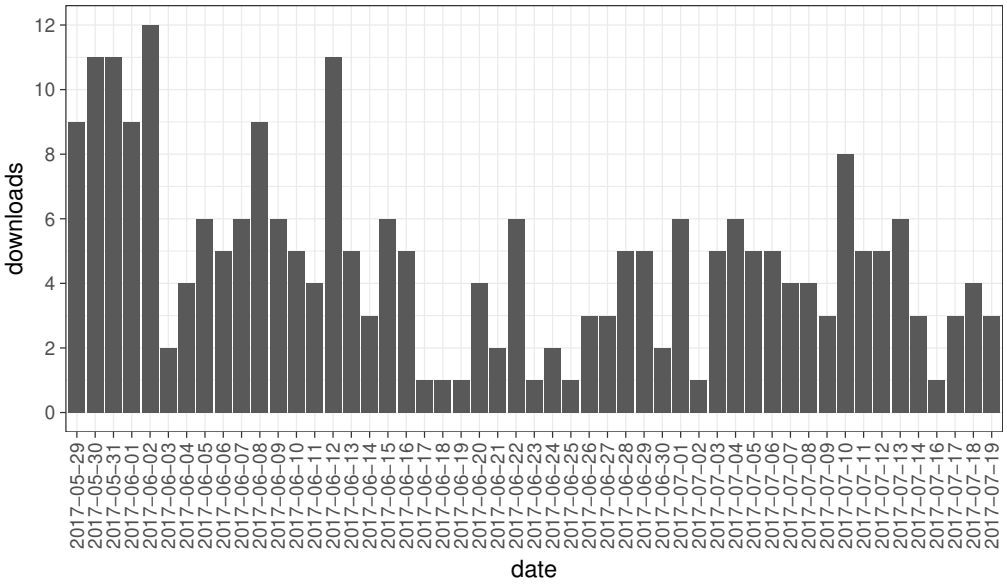

Figure 12: Daily number of downloads from unique users of the `classiFunc`-package between May 29th 2017 and July 19th 2017. (Data source: `http://cran-logs.rstudio.com/`)

Introduction to `mlr`

The user-friendly interface offered by `mlr` makes it easy to apply the proposed modeling structures to new data sets, even for beginners in **R**. A wide range of models are offered that are constantly updated and extended. It is straightforward to apply and compare multiple models using the robust and reproducible framework for conducting benchmark analysis. These attributes have made `mlr` a hugely popular package, see Figure 13. For an introduction to `mlr` see the tutorial web page `http://mlr-org.github.io/mlr-tutorial/`. Currently, `mlr` is being extended to contain methods for functional data. The github branch for functional data analysis is not yet merged into the official `mlr` master branch. So, the software created for this thesis will be part of the first official CRAN release with the new functionality. For the initial release functional covariables must be regular and can not contain missing values.

Implementation of Ensembles

By embedding the methods from the `classiFunc`-package and the `fda.usc`-package (Febrero-Bande and Oviedo de la Fuente, 2012) as new learners into `mlr`, both implementations for nearest neighbor estimators can be employed and compared in a uniform framework.

The ensemble proposed in Fuchs et al. (2015), see Section 3.1, is implemented as a stacked learner in `mlr`. This makes it available as an ensemble method for arbitrary base models (e.g., nearest neighbor estimators, kernel estimators, classification trees and support vector machines) and arbitrary classification problems,

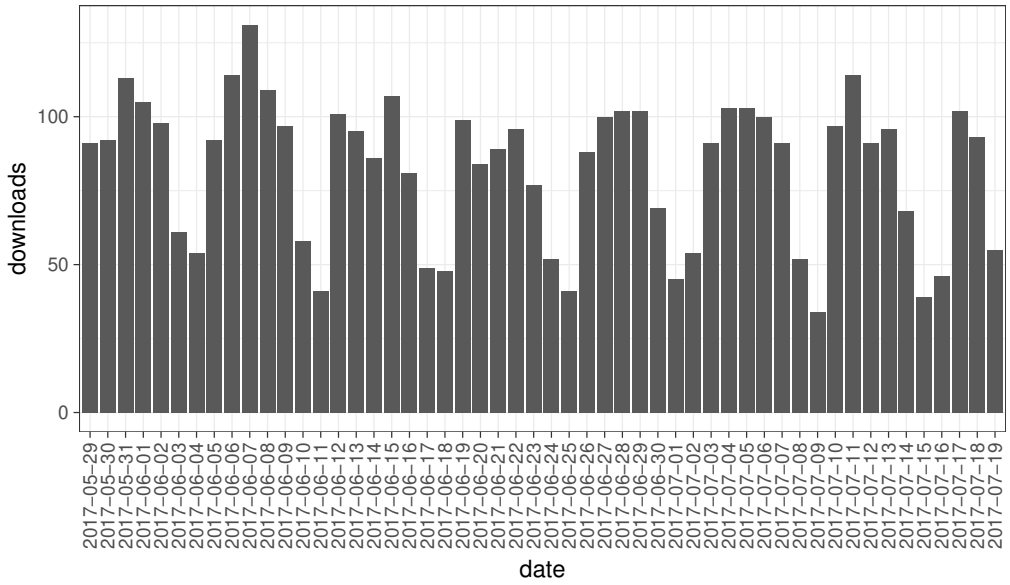

Figure 13: Daily number of downloads of the `mlr`-package in the same time window as the `classiFunc`-package in Figure 12. (Data source: `http://cran-logs.rstudio.com/`)

for functional as well as for non-functional classification problems. This flexibility is the primary motivation for choosing `mlr` as a container for the nearest neighbor ensemble. The random forest is already available as an ensemble method in `mlr` so the implementation of the random forest ensemble for functional data only requires combining the newly implemented base learners for functional data with the already existing structure for ensembling learners.

This thesis contributes a small part to functional data methods in `mlr` by implementing functional data classification. With the official release of functional data analysis in `mlr`, dimension reduction methods for functional data provided in another branch can be added into the random forest ensemble. The popularity and utilization of the `classiFunc`-package as a background package for functional data classification in `mlr` can be expected to increase.

5 Empirical Comparison of Classification Ensembles

This chapter reports the result of the systematic comparison of the functional data classification methods described in Chapter 3. A benchmark analysis is conducted analogously to Bagnall et al. (2016) using the Brier Score (5) to measure performance. All data used in this chapter is either simulated or has been previously analyzed multiple times. Conducting a benchmark analysis on these data sets is useful nonetheless, as the aim of the benchmark analysis is to find underlying structures and mechanisms in the proposed ensemble models. This helps to decide which classification method to use in future applications.

Section 5.1 introduces the Friedman test and the corresponding Nemenyi post-hoc test (Demšar, 2006). These tests are used to test for statistical significance of the differences in classification quality. Section 5.2 lists the models that are compared in the benchmark analysis. The benchmark analysis was run for simulated and real-world data. Section 5.3 describes the approach to simulating data and reports the respective results of the benchmark analysis. The results for the real-world data sets are reported in Section 5.4. More detailed benchmarking results in the form of boxplots showing the performance for every learner in the folds of the cross validation can be found in the digital supplement; see Appendix A for an overview of the contents of the digital supplement.

Main Goals

The central questions to be answered by the benchmark analysis are:

1. Which of the ensemble methods proposed in Chapter 3 performs better given the same base models?
2. How does their performance compare with reference models?
3. How good are the ensemble methods at detecting useful semimetrics? Does their performance change when useless semimetrics are added?
4. Are the ensemble methods proposed in Chapter 3 better than the best individual model they contain? This question can be subdivided:
   (a) Does an ensemble of classifiers using the same semimetric d but different numbers of nearest neighbors k outperform a classifier with the same semimetric d and an optimally chosen k, or simply $k = 1$?
   (b) Does an ensemble of classifiers using the same number of nearest neighbors k (e.g., $k = 1$) and the same semimetric d but different order derivatives a outperform a classifier with optimally chosen order derivative a, or simply $a = 0$?
   (c) Does an ensemble of classifiers using a fixed number of nearest neighbors k (e.g., $k = 1$) but different semimetrics d outperform a classifier with the same number of nearest neighbors k and an optimally chosen semimetric?
   (d) Does an ensemble of classifiers using different numbers of nearest neighbors k, different semimetrics d and different order derivatives a outperform a classifier with optimally chosen number of nearest neighbors k, semimetric d and derivative of order a?

The main goals are answered using the simulated data and real-world data sets from the UCR Time Series Classification Repository. Within the simulation study, only the random splines data set is used evaluate the main goals, which is simulated to not contain any warping. This is useful in order to separate effects of the ensemble method from effects of the semimetric/data combination.

5.1 Statistical Inference for Benchmark Experiments

This section gives an introduction to statistical inference in benchmark experiments. Benchmark experiments are useful for comparing multiple models over numerous data sets. By means of cross validation, the performance of every model can be estimated for every data set. To carry out the benchmark analysis and test for significance, the framework of the **R**-package `mlr` is used.

Friedman Test

Demšar (2006) and Bagnall et al. (2016) suggest using the Friedman test (Friedman, 1937, 1940) for evaluating benchmark analyses. The Friedman test is a nonparametric equivalent of the repeated-measures ANOVA. This test compares the performance of the models on the scale of their rank as opposed to the numeric value of the performance measure. This makes sense, as for most benchmark experiments the assumptions of a repeated-measures ANOVA (particularly the gaussian distribution of the residuals and the random intercepts of the models) do not hold. If they were met, the Friedman test would have lower power in comparison to the repeated-measures ANOVA. The Friedman test ranks the algorithms for each data set separately, with the best performing algorithm receiving rank 1, the second best rank 2, and so forth. In the case of ties, average ranks are assigned. Then the test statistic is computed as follows: Let r_{ji} be the rank of the model j, $j = 1, \ldots, J$ on data set i, $i = 1, \ldots, N$. The Friedman test compares the average ranks R_j of the algorithms j, $R_j = \frac{1}{N} \sum_i^N r_{ij}$. Under the null hypothesis "All the algorithms are equivalent," the average ranks R_j should be equal and the Friedman statistic

$$\chi_F^2 = \frac{12N}{J(J+1)} \left[\sum_{j=1}^{J} R_j^2 - \frac{J(J+1)^2}{4} \right]$$

is χ^2 distributed with $J - 1$ degrees of freedom, when N and J are big enough. Demšar (2006) give $N > 10$ and $J > 5$ as a rule of thumb for when this approximation holds. This is the case for this benchmark analysis.

Nemenyi Post-Hoc Test

If the null hypothesis of the Friedman test is rejected, the Nemenyi post-hoc test (Nemenyi, 1962) can be applied. The Nemenyi test is the counterpart to the Tukey test for ANOVA and is used to compare all classifiers to each other. The performance of two classifiers is different on an α level of significance if their average ranks differ by at least the critical difference

$$\text{CD} = q_\alpha \sqrt{\frac{J(J+1)}{6N}}.$$

The critical values of q_α are based on the Studentized range statistic divided by $\sqrt{2}$. They are tabulated in Demšar (2006) or can be computed in **R** using the `qtukey` function from the `stats` package (R Core Team, 2016).

Model ID	Description	Main Goals
Eucl: k 1; nderiv 0	1 nearest neighbor estimator on original data with Euclidean distance	2
Eucl-Kernel: h CV-opt	Nonparametric functional kernel estimator on original data with CV-optimal bandwidth	2
dtw: k 1; nderiv 0	1 nearest neighbor estimator on original data with dynamic time warping distance	2

Table 1: Model ID, description and corresponding main goals of the benchmark analysis for the reference models.

5.2 Models in Benchmark Experiment

A benchmark analysis compares models over different data sets. The aim is to find underlying structures, which make it possible to predict relative model performance. This section describes which models were used to gain insights into the underlying patterns. This is an intricate problem, as a relatively small number of models had to be chosen due to limitations in time and computational power. Despite this challenge, a careful selection of the models makes it possible for this benchmark analysis to robustly characterize the most important aspects of the investigated model classes. The benchmark experiments conducted in this thesis compare the same set of 17 models, 3 of which are reference models included to put the classification performance of the other 12 models into perspective. The results of these other models will be used to find answers to the main research questions posed at the beginning of this chapter.

Reference Models

An overview of the reference models evaluated in the benchmark analysis is given in Table 1. As reference classification methods, 1 nearest neighbor estimators with dynamic time warping (DTW) and the Euclidean distance as a distance measure are used. They are included in the nearest neighbor ensemble as the special case where only one nearest neighbor estimator is used, which will automatically receive weight 1 in the ensemble. The nonparametric functional kernel estimator proposed in Ferraty and Vieu (2003) will be used as another benchmark classification method. A Gaussian kernel with optimal bandwidth is used. The bandwidth h is selected from the interval $[10^{-1}, 10^{4}]$ using a 5 fold cross validation.

Evaluated Models

An overview of the models evaluated in the benchmark analysis is given in Table 2. The models evaluated in the benchmark analysis are designed to find answers the research questions raised at the beginning of this chapter. The nearest neighbor ensembles are designed to systematically evaluate the gain of ensembling models with

a different number of nearest neighbors k, order derivative a or semimetric d as opposed to determining these parameters using cross validation. The parameters were limited to the set of $k = \{1, 5, 9, 13\}$, $a = \{0, 1, 2\}$ and $d = \{d_{\text{eucl}}, d_{\text{manh}}, d_{\text{min}}, d_{\text{max}}\}$. The full nearest neighbor ensemble, meaning the nearest neighbor ensemble containing the nearest neighbor models with all possible combinations of k and a, is further compared to a random forest ensemble containing the same base models to assess their effectiveness for a small number of (mostly) useful base models. In order to determine which of the ensembles is better in filtering out useless base models, "noisy" ensembles were created by adding 100 base models without any predictive value to the ensemble. The amplitude and phase distance were used in nearest neighbor estimators for an exploratory valuation of their predictive power.

Color Scheme

Throughout this thesis a uniform color scheme is used for graphical representations of the models, see for example the legend of Figure 17. Reference models are shown in dark tones, the 1 nearest neighbor estimator with the Euclidean distance in black, the kernel estimator in light gray, and the 1 nearest neighbor estimator with the dynamic time warping distance in brown. Other 1 nearest neighbor estimators with warping distance are shown in similar sand tones, yellow for the amplitude distance and gold for the phase distance. The CV optimal 1 nearest neighbor estimators always have the same color as their responding nearest neighbor ensembles, but a few shades lighter. Ensembling over different k nearest neighbors is depicted in red; ensembling over different derivatives of order a is depicted in blue; ensembling over different semimetrics d is depicted in coral pink; and ensembling over k and a and d is depicted in purple. All random forest ensembles are shown in green: light green for the ensemble exclusively based on the base models and dark green for the ensemble also using the discretized values of the functional observation as further explanatory variables. The "noisy" ensembles, meaning ensembles with uninformative base models added to them are paler in color: pale green for the random forest ensemble and pale purple for the nearest neighbor ensemble.

5.3 Benchmark Experiments on Simulated Data

A first benchmark analysis was conducted on simulated data, in order to observe the behavior of the models for artificial and well-behaved data sets. Note that for this case statistical tests for classification performance of the different models are meaningless, as the data they are based on is completely artificial and can be tweaked to achieve (basically) every possible outcome. Instead, the data simulation procedure and its hyperparameters are linked to the classification performance using graphical methods.

This section is divided into two parts: The simulation of functional data which does not contain any warping in Section 5.3.1, and the simulation of warped functional data in Sections 5.3.2 through Section 5.3.4. The data without warping is used to give answers to the main research questions posed at the beginning of this chapter concerning the properties of the competing ensemble methods. Warped

Model ID	Description	Main Goal
Eucl-ensemble: k 1, 5, 9, 13; nderiv 0	Ensemble of 1, 5, 9 and 13 nearest neighbor estimators on original data with Euclidean distance	4a
Eucl-ensemble: k 1; nderiv 0, 1, 2	Ensemble of 1 nearest neighbor estimators on derivative of order 0, 1, 2 with Euclidean distance	4b
Semimet-ensemble: k 1; nderiv 0	Ensemble of 1 nearest neighbor estimators on original Euclidean distance with Euclidean distance, Manhattan distance, global minima distance and global maxima distance	4c
Semimet-ensemble: k 1, 5, 9, 13; nderiv 0, 1, 2	Ensemble of 1, 5, 9 and 13 nearest neighbor estimator on derivative of order 0, 1, 2 with Euclidean distance, Manhattan distance, global minima distance and global maxima distance	1, 3, 4d
Eucl: opt k; nderiv 0	k nearest neighbor estimator on original data with Euclidean distance, where k is chosen from $\{1, 3, 5, 7\}$ using CV	4a
Eucl: k 1; opt nderiv	1 nearest neighbor estimator on derivative of order a with Euclidean distance, where a is chosen from $\{0, 1, 2\}$ using CV	4b
Opt Semimetric: k 1; nderiv 0	1 nearest neighbor estimator on original data with semimetric d, where d is chosen from $\{d_{\text{eucl}}, d_{\text{manh}}, d_{\text{min}}, d_{\text{max}}\}$ using CV	4c
Opt Semimetric: opt k; opt nderiv	k nearest neighbor estimator on derivative of order a with semimetric d, where k is chosen from $\{1, 3, 5, 7\}$ and a is chosen from $\{0, 1, 2\}$ and d is chosen from $\{d_{\text{eucl}}, d_{\text{manh}}, d_{\text{min}}, d_{\text{max}}\}$ using CV	4d
rf-ensemble: k 1, 5, 9, 13; nderiv 0, 1, 2; no feat	Same base learners as 'Semimet-ensemble: k 1, 5, 9, 13; nderiv 0, 1, 2', but with random forest ensemble	1, 3
rf-ensemble: k 1, 5, 9, 13; nderiv 0, 1, 2; use feat	Same base learners as 'Semimet-ensemble: k 1, 5, 9, 13; nderiv 0, 1, 2', but with random forest ensemble using the observation points of the functional covariable as multivariate features	1
nn-ensemble: noisy	Same as 'Eucl-ensemble: k 1, 5, 9, 13; nderiv 0, 1, 2' with additional 100 random base learners	3
rf-ensemble: noisy	Same as 'Eucl-rf: k 1, 5, 9, 13; nderiv 0, 1, 2; no feat' with additional 100 random base learners	3
amplitude: k 1; nderiv 0	1 nearest neighbor estimator on original data with amplitude distance	-
phase: k 1; nderiv 0	1 nearest neighbor estimator on original data with phase distance	-

Table 2: Model ID, description and corresponding main goals of the models evaluated in the benchmark analysis.

functions are simulated in order to get a deeper understanding of the comparison between methods involving warping and methods that do not include warping. Here only the subset of relevant models is included in the benchmark analysis.

Four different types of data sets were created using different data generation processes. For the random splines data set (Section 5.3.1), class-specific spline coefficients are randomly sampled to create spline functions. This data set does not contain any warping. In the warped trigonometric data set (Section 5.3.2) cosine functions with class-specific amplitude and phase are sampled. In the warped bimodal data set (Section 5.3.3), random class-specific bimodal functional observations are created, which are warped by applying class-specific random warping functions. For the warped splines data set (Section 5.3.4), random class-specific warping functions are used to integrate warping into the data generation process of the random splines data.

5.3.1 Random Splines Data

The random splines data set is generated by sampling class-specific random splines. The splines can be created to be more or less similar within and between the classes, which allows the creation of easier and more difficult classification tasks. This data set does not contain any warping and will be used to assess the main goals of the benchmark analysis.

Conceptual Introduction to B-Splines

B-Splines can be used to approximate smooth functions and curves. The idea of splines is to represent a complex function $f(t)$ through a weighted sum of K simple basis functions B_j, $j = 1, \ldots, K$,

$$f(t) = \sum_{i}^{K} \omega_j B_j(t).$$

These simple basis functions are depicted in Figure 14 for cubic B-Splines. Usually the weights of the basis function are chosen to minimize the distance of the resulting B-Spline to a function or curve from which only noisy observations are available. This thesis turns this concept around. By randomly sampling the spline coefficients ω_j of the basis functions $B_j(t)$, the resulting B-Spline $f(t)$ will be a random function.

Data Generation

Every observation is represented by a spline, which in turn is defined by the spline coefficients/weights ω_j of its basis functions. The data is simulated in a two step procedure. The first step is to sample one set of spline coefficients per class at random. These will be the underlying "true" splines for each class. In the second step, random deviations from these spline coefficients are sampled for every observation. All spline coefficients are drawn from a normal distribution with expectation zero. With this procedure, an arbitrary number of observations can be sampled from an

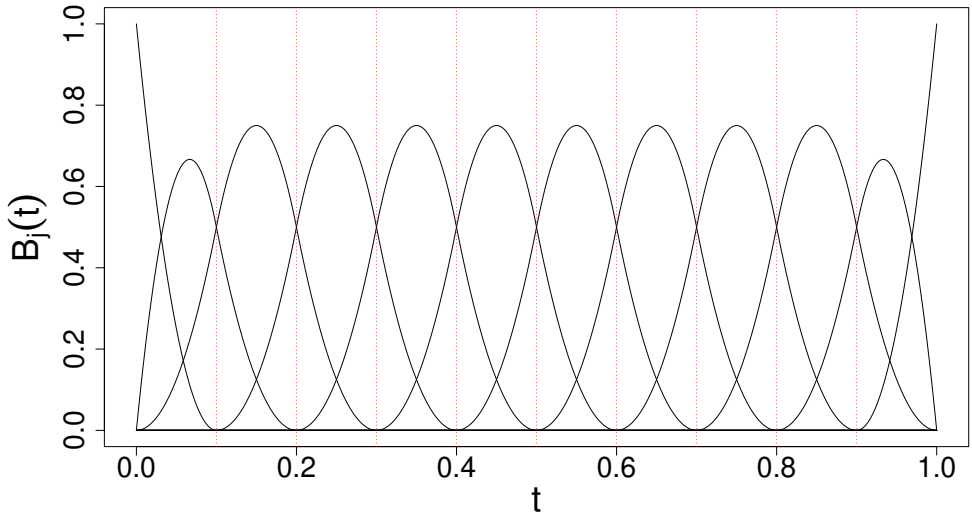

Figure 14: Basis functions for cubic B-splines.
(Source: Fahrmeir et al. (2013))

arbitrary number of classes. The hyperparameters that define this procedure are (with considered parameter sets in curly braces):

1. `nclass` {2, 10}: the number of classes.

2. `var_between_class` {1}: the variance of the spline coefficients that are sampled in the first step. A higher variance will lead to more dissimilar classes and thus an easier classification problem.

3. `var_within_class` {0.5, 2}: the variance of the spline coefficients that are sampled in the second step. A higher variance will lead to more dissimilar observations within the classes and thus a harder classification problem.

Note that in this simulation setup the hyperparameters `var_between_class` and `var_within_class` can be compressed into one, as only their ratio is relevant. Multiplying them simultaneously with a constant factor leads to a rescaling of the resulting data, which is irrelevant for all methods at hand. Hence the hyperparameter `var_between_class` can be discarded by setting it to 1. For the benchmark analysis all possible combinations of the hyperparameters were used to create data sets. Figure 15 shows simulated random splines data for `nclass` = 2, `nobs` = 20 and `var_within_class` = 0.5.

Results

All reference models (Table 1) and evaluated models (Table 2) are compared in a benchmark analysis on the random splines data sets using 10-fold cross validation. An in-depth result of the benchmark analysis for these simulated data sets is shown

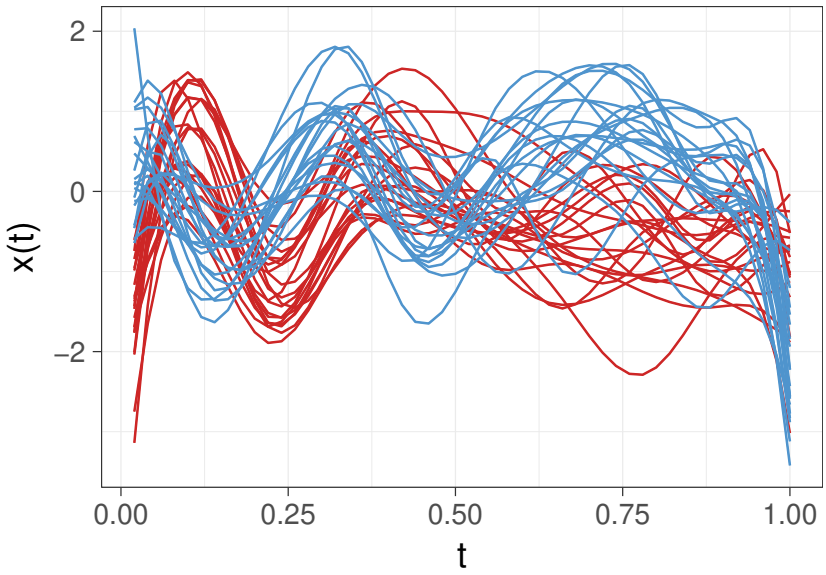

Figure 15: The random splines data for `nclass` = 2, `nobs` = 20 and `var_within_class` = 0.5.

in Figure 16. It consists of one subplot per data set showing the performance of every model as a boxplot. The individual boxplots consist of the Brier Score achieved by this model in the 10 folds of the cross validation. As expected, the predictive performance of all models is better, the smaller the variance within the curves (`vwc`), the larger the sample size (number of observations per class, `nobs`) and the fewer classes (`ncl`) there are in the simulated data set. It is reassuring to see that for "easier" classification tasks, all models do well and for "harder" classification tasks, all models do worse.

An aggregated overview of the results of the benchmark analysis is shown in Figure 17, mapping out the ranks of the models for every simulated data set. In almost all cases, the random forest models (green dots) show the best performance. The only exception is an easy data set with a low number of observations (`ncl = 2`, `nobs = 20`, `vwc = 0.5`), where most models achieve almost perfect prediction. The full CV-optimal estimator (light purple dot), the k CV-optimal estimator (light red dot), the nearest neighbor ensembles ensembling over k (dark red dot) and a (dark blue dot), and the functional kernel estimator also perform well. The 1 nearest neighbor estimator with the Euclidean distance (black dot), the estimator choosing the derivative of order a (light blue dot) or the semimetric d (light coral pink dot) CV-optimal, the full nearest neighbor ensemble (dark purple dot), its noisy counterpart (pale purple dot), and the ensemble over different semimetrics d (dark coral pink dot) tend to perform worse. All models based on warping distances (sand colored dots) consistently perform badly for these simulated data sets. This is not surprising as the data is simulated to not contain any informative warping.

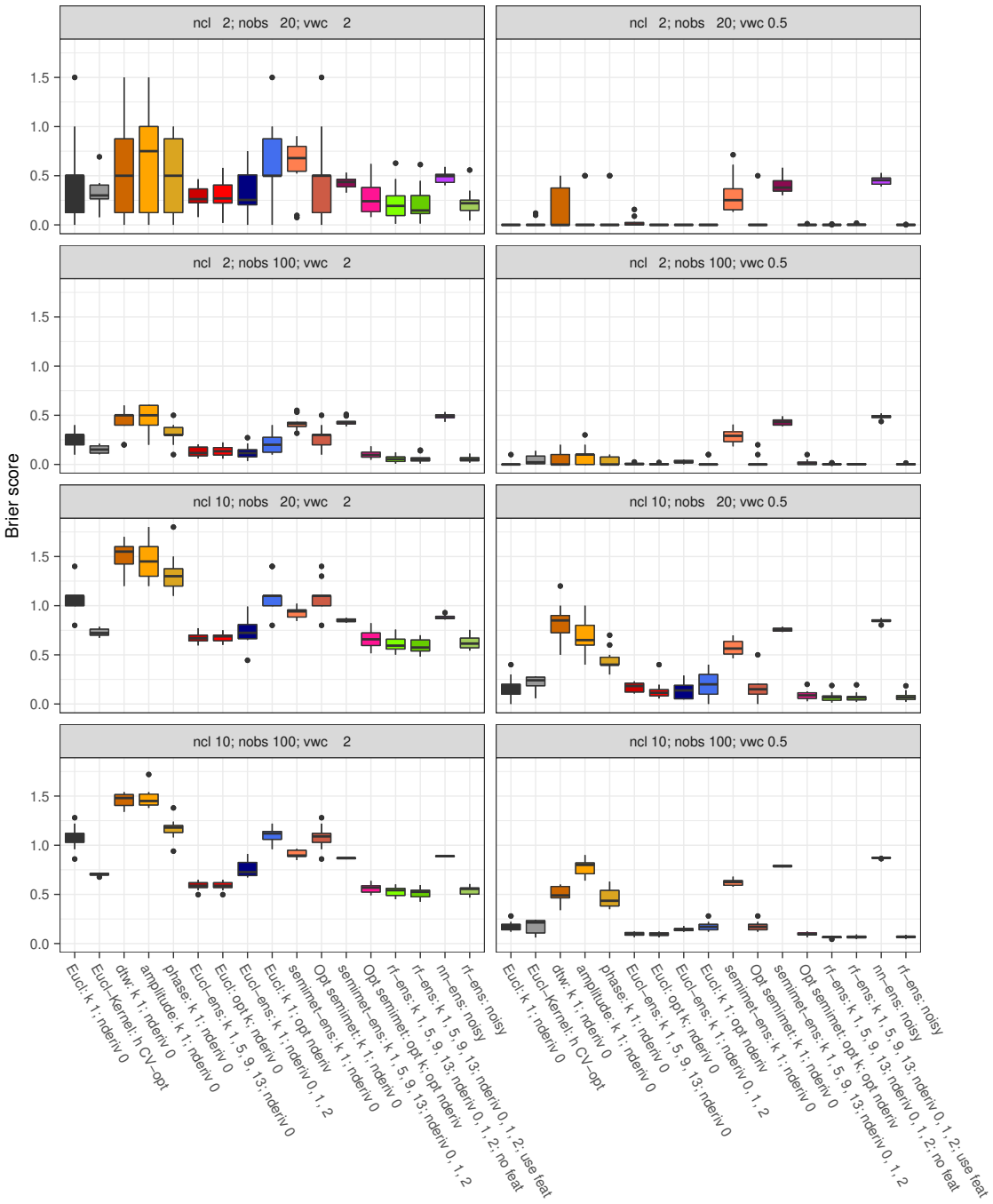

Figure 16: Result of the benchmark analysis for the random splines data set. Every subplot shows the performance of every model as a boxplot for one random splines data set. The individual boxplots depict the Brier score achieved by this model in the 10 folds of the cross validation.
Upper half: `ncl = 2`, number of classes = 2; Lower half: `ncl = 10` (more difficult); Row 1 & 3: `nobs = 20`, number of observations per class = 20; Row 2 & 4: `nobs = 100` (easier); Left column: `vwc = 2`, variance of the spline coefficients within the observations of the same class is 2 times the variance of the splines coefficients between the class centers; Right column: `vwc = 0.5` (easier)

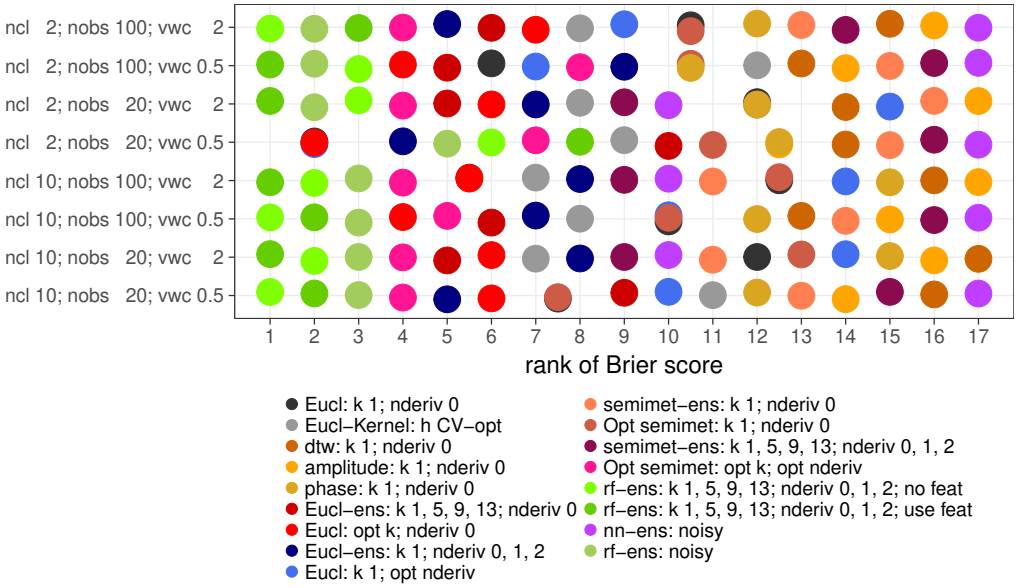

Figure 17: Result of the benchmark analysis for the random splines data set. Depicted is the classification task vs. the rank of the average Brier score achieved by the models. Shared ranks are overplotted. This is a condensed depiction of the information shown in Figure 16.

Assessment of Main Goals

Based on these results, the research questions concerning the properties of the ensemble methods can be answered regarding the simulated data:

1. The random forest ensemble performs better than the nearest neighbor ensemble given the same base models.

2. Both ensemble methods tend to outperform reference classification models. For the nearest neighbor ensemble this statement has to be limited to ensembles containing mostly useful base models.

3. The random forest model is better at detecting (and filtering out) uninformative base models than the nearest neighbor ensemble.

4. A nearest neighbor ensemble method as proposed in Chapter 3.1 does generally not outperform the best individual model contained. This can be subdivided into:

   (a) An ensemble of classifiers using the same semimetric d but different numbers of nearest neighbors k does not outperform a classifier with the same semimetric d and an optimally chosen k. They generally seem to perform equally well.

   (b) An ensemble of classifiers using the same numbers of nearest neighbors k (e.g., $k = 1$) and the same semimetric d but different order derivatives

a outperforms a classifier with the same number of nearest neighbors k and optimally chosen order derivative a.

(c) An ensemble of classifiers using a fixed number of nearest neighbors k (e.g., $k = 1$) but different semimetrics d is outperformed by a classifier with the same number of nearest neighbors k and an optimally chosen semimetric.

(d) An ensemble of classifiers using different numbers of nearest neighbors k, different semimetrics d and different order derivatives a is outperformed by a classifier with optimally chosen number of nearest neighbors k, semimetric d and derivative of order a. This should be considered carefully, as this might change depending on the ratio of informative semimetric/order derivative combinations in the base models.

5.3.2 Warped Trigonometric Data

The idea behind the warped trigonometric data sets is to create class specific functions that differ in their amplitude and phase. For this purpose, cosine functions with class-specific amplitude and phase are sampled. By adding random noise to the amplitude and phase of the individual observations of the classes, the underlying amplitude and phase differences of the classes are blurred.

Data Generation

Every observation $x_{gi}(t)$, $i = 1, \ldots, N_g$, observations per class, $g = 1, \ldots, G$, classes, is represented by a cosine function, which is defined by its amplitude α_{gi} and its phase ϕ_{gi},

$$x_{gi}(t) = \alpha_{gi} \cos(\phi_{gi} + t),$$

where $\alpha_{gi} \sim \mathcal{N}(\alpha_g, \sigma_\alpha)$ and $\phi_{gi} \sim \mathcal{U}(\phi_g, \phi_g + \delta_\phi)$. The parameters α_g and ϕ_g are set to $\alpha_g = 1 + g/G$ and $\phi_g = 2\pi \cdot g/G$. Observations from the same class are drawn from the same distribution, as the parameters α_g and ϕ_g are class specific.

With this procedure an arbitrary number of observations can be sampled from an arbitrary number of classes. The hyperparameters that define this procedure are (with considered parameter sets in curly braces):

1. `nclass` {2, 10} : G, the number of classes.

2. `nobs` {100}: N_g, the number of observations per class. This can theoretically vary from class to class to create an unbalanced data set. In this thesis all simulated data is balanced.

3. `max.phase.dif` {1, 1.5}: δ_ϕ/π, the range of the uniform distribution where the phase shift is sampled from.

4. `vamp` {0.5, 2}: σ_α, the variance of the amplitude within the classes.

Figure 18 shows the warped trigonometric data for `nclass` = 2, `nobs` = 100, `max.phase.dif` = 1 and `vamp` = 0.5.

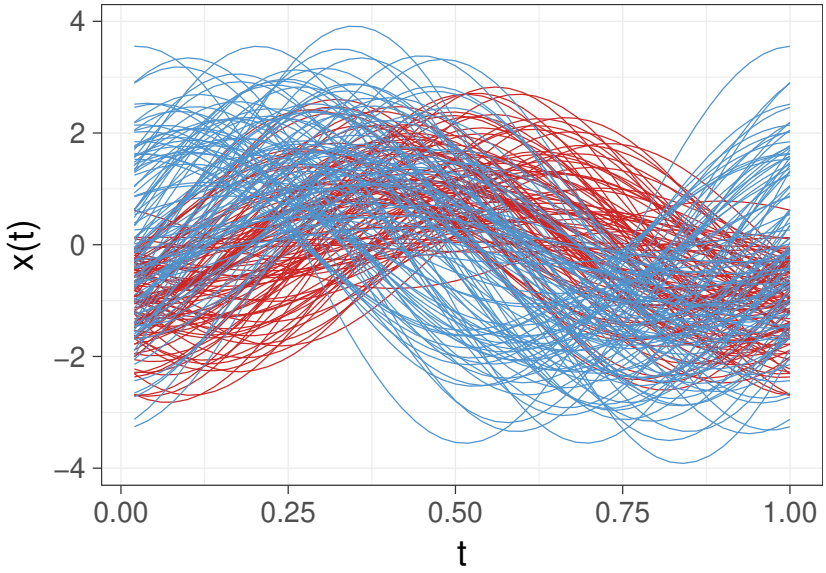

Figure 18: The warped trigonometric data for `nclass` = 2, `nobs` = 100, `max.phase.dif` = 1 and `vamp` = 0.5. The class of the observations is color-coded in red and blue.

Results

The results of the benchmark analysis are depicted in Figure 19, showing the rank of each model for every data set. Only models relevant to analyzing the difference between methods including warping and methods not including warping are evaluated in the benchmark experiments for warped data, see Table 3. These models are the nearest neighbor estimators for the Euclidean distance and the three warping distances (dynamic time warping distance, amplitude distance and phase distance). The nonparametric functional kernel estimator is included as an additional reference model. For the two class problems, the distances based on warping do not outperform the Euclidean distance. This might be the case because the difference in the phase of the two classes is too large. For the ten class problem, the phase distance greatly outperforms the Euclidean distance. The warping functions needed to align these observations seem to be informative. The amplitude distance slightly outperforms the Euclidean distance for the case of ten classes as well, but the actual difference between the average Brier scores is rather small (see the detailed results in the digital supplement). Unexpectedly, the kernel estimator using the Euclidean distance outperforms the 1 nearest neighbor estimators using warping distances. It seems that the advantage of using a kernel estimator over a nearest neighbor estimator outweighs the use of sophisticated semimetrics in this case.

Model ID	Description
Eucl: k 1; nderiv 0	1 nearest neighbor estimator on original data with Euclidean distance
Eucl-Kernel: h CV-opt	Nonparametric functional kernel estimator on original data with CV-optimal bandwidth
dtw: k 1; nderiv 0	1 nearest neighbor estimator on original data with dynamic time warping distance
amplitude: k 1; nderiv 0	1 nearest neighbor estimator on original data with amplitude distance
phase: k 1; nderiv 0	1 nearest neighbor estimator on original data with phase distance

Table 3: Model ID and description for the models evaluated in the benchmark experiments for simulated warped functional data.

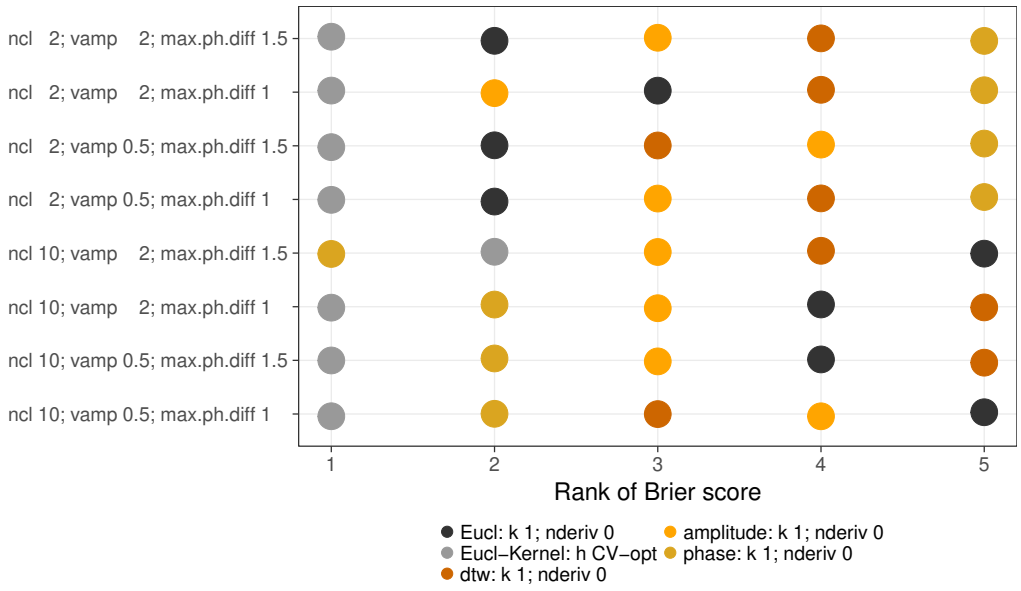

Figure 19: Ranks of the learners in the benchmark analysis for the warped trigonometric data set.

5.3.3 Warped Bimodal Data

In the warped bimodal data sets (Section 5.3.3) random class-specific bimodal functional observations are created, which are warped by class-specific random warping functions. The procedure follows Example 8.1 from Srivastava and Klassen (2016). This data generation process allows the simulation of data where the amplitude or the amount of warping of the data is informative or completely random.

Data Generation

Every observation $x_{gi}(t) = f_{gi}\big(\gamma_{gi}(t)\big)$, $i = 1, \ldots, N_g$, $g = 1, \ldots, G$, is a bimodal function with random location and height of its peaks. The observations are sampled as bimodal functions

$$f_{gi}(t) = z_{1gi} \exp\big(-(t-1.5)^2/2\big) + z_{2gi} \exp\big(-(t+1.5)^2/2\big),$$

where z_{1gi}, $z_{2gi} \sim \mathcal{N}(\mu_g, \sigma_z)$, with $\mu_g = 1 + \delta_g \cdot g$. These functions are warped according to a random warping function

$$\gamma_{gi}(t) = 6\frac{\exp\big(a_{gi}(t+3)/6\big) - 1}{\exp\big(a_{gi}\big) - 1} - 3,$$

where $a_{gi} \sim \mathcal{N}(a_g, \sigma_a)$, where $a_g = \delta_a \cdot \big(-(G+1)/2 + g\big)$. The parameters a_g are set to be symmetric around 0, as $\gamma_{gi}(t) = t$ for $a_{gi} = 0$.

This procedure allows the sampling of an arbitrary number of observations from an arbitrary number of classes. The hyperparameters defining this procedure are (with considered parameter sets in curly braces):

1. `nclass` $\{2, 10\}$: G, the number of classes.
2. `nobs` $\{100\}$: N_g, the number of observations per class.
3. `cl.z.diff` $\{0, 0.1\}$: δ_z, the average vertical difference between observations of neighboring classes g, $g+1$. If set to 0, the differences in the amplitude of the data are completely random.
4. `cl.a.diff` $\{0, 1\}$: δ_a, the average distance between the warping parameters a_g of neighboring classes. If set to 0, the differences in warping of the data are completely random.
5. `obs.z.var` $\{0.01\}$: σ_z, the variance of the vertical difference between observations of neighboring classes g, $g+1$.
6. `obs.a.var` $\{0.01\}$: σ_a, the variance of the average distance between the warping parameters a_g of neighboring classes.

Figure 20 shows the warped bimodal data for `nclass` $= 2$, `nobs` $= 100$, `cl.z.diff` $= \{0, 0.1\}$ and `cl.a.diff` $= \{0, 1\}$. The data belonging to different parameter combinations is shown in a grid. In the top left of Figure 20, the observations do not

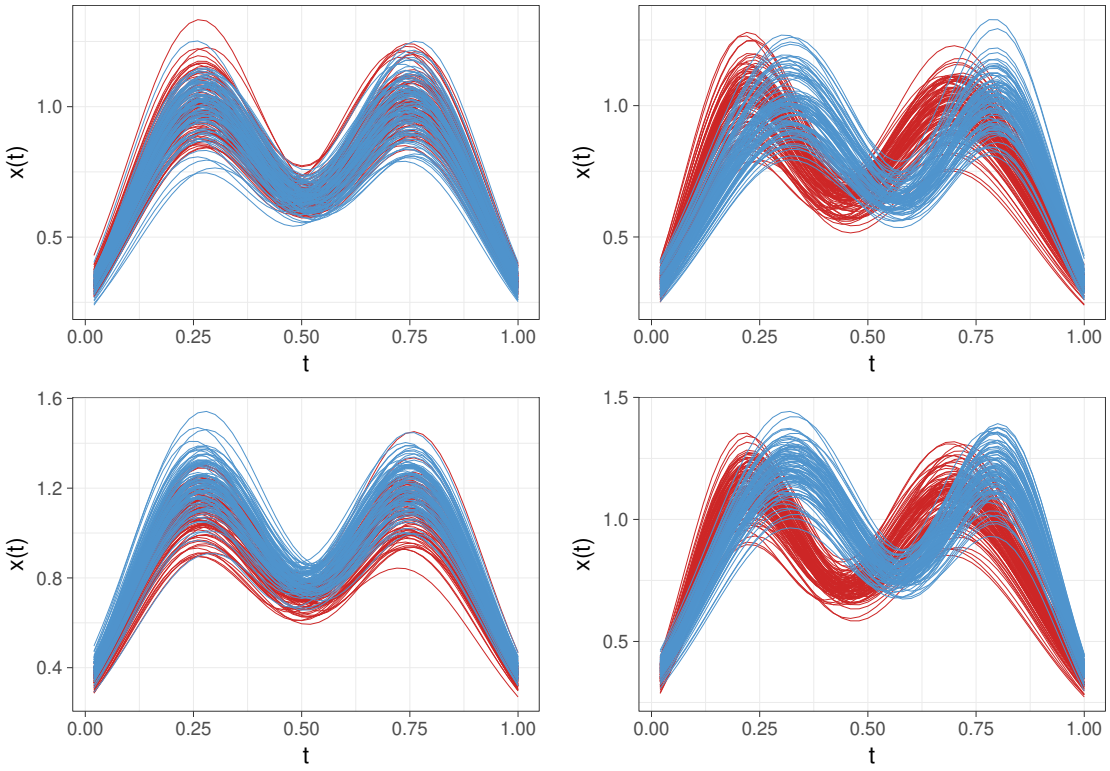

Figure 20: The warped bimodal data for `nclass` = 2, `nobs` = 100, and Top left: `cl.z.diff` = 0 and `cl.a.diff` = 0; Top right: `cl.z.diff` = 0 and `cl.a.diff` = 1; Bottom left: `cl.z.diff` = 0.1 and `cl.a.diff` = 0; Bottom right: `cl.z.diff` = 0.1 and `cl.a.diff` = 1.

contain any information about their class, as `cl.z.diff` = 0 and `cl.a.diff` = 0. All observations are sampled from the same distribution, so all variation is completely random. The bottom right of Figure 20 shows the warped bimodal data for `cl.z.diff` = 0.1 and `cl.a.diff` = 1. The observations in the classes differ in amplitude as well as in their warping functions.

Results

The results of the benchmark analysis are depicted in Figure 21, showing the rank of each learner for every data set. For this set of simulated data, the 1 nearest neighbor estimator using the Euclidean distance outperforms the warping distances in all cases where the observations contain information about the class. The only exception is the data set with 10 classes and a small difference in amplitude, where the warping distances perform slightly better. It has to be noted that, for this data set, only the kernel estimator outperforms the threshold set by random guessing (Brier Score = 0.9, see digital supplement).

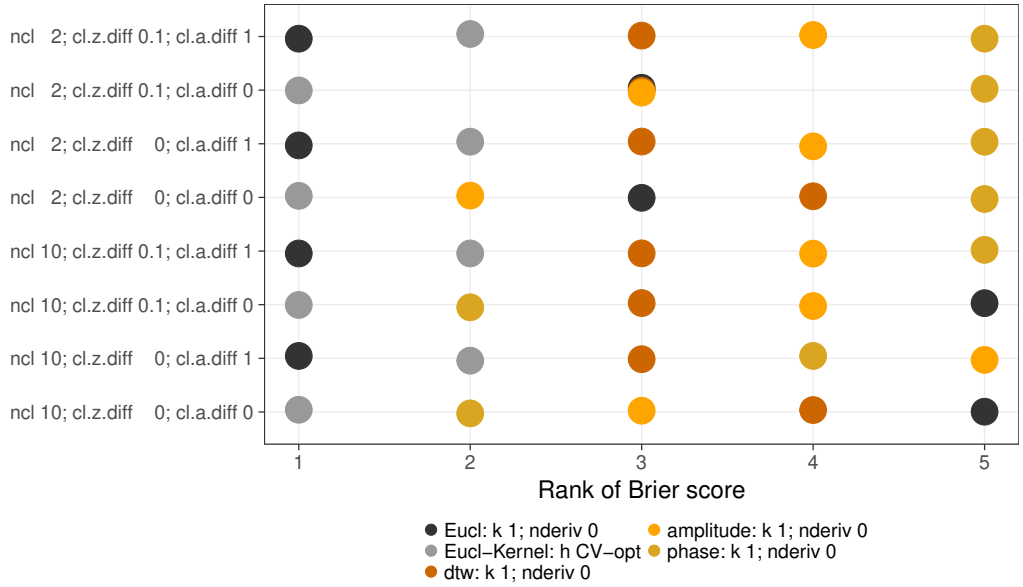

Figure 21: Ranks of the learners in the benchmark analysis for the warped bimodal data sets.

5.3.4 Warped Splines Data

In the warped splines data sets (Section 5.3.4), random class-specific warping functions are used warp data generated by the same process as the random splines data.

Data Generation

The data consists of random splines $f(t)$ that are drawn to have expectation t. This is useful to still have effects of different warping functions if the variance of the spline coefficients is very small. The warping functions γ_{gi} are sampled using the following procedure: The support $\mathcal{T} = [0, 1]$ is separated in M intervals of equal length $[0, t_1 = 1/M], [t_1, t_2 = 2/M], \ldots, [t_{M-1}, t_M = 1]$, on which the warping function is piece-wise linear. At the endpoints of these intervals, the warping function γ_{gi} takes the values

$$\gamma_{gi}(t_j) = \frac{1}{\sum_{m=1}^{M} \epsilon_{gm}(t_m)} \sum_{m=1}^{j} \epsilon_{gm}(t_m),$$

where $\epsilon_{gm}(t_m) \sim \text{Exp}(\lambda_g(t_m))$, with time dependent rate $\lambda_g(t) = 0.1 + \phi \cdot g^{2(t-0.5)}$. For $g = 1$ the rate $\lambda_1(t)$ is constant and the warping function is the identity function in expectation. The higher g is, the higher the expectation function of γ_{gi}, as the expectation of the $\epsilon_{gm}(t_m)$ is more dissimilar for small and large t_m. With increasing g, the warping functions of observations of class g and class $g + 1$ are more similar.

With this procedure, an arbitrary number of observations from an arbitrary number of classes can be sampled. The hyperparameters that define this procedure are (with considered parameter sets in curly braces):

1. `nclass` $\{2, 10\}$: G, the number of classes.

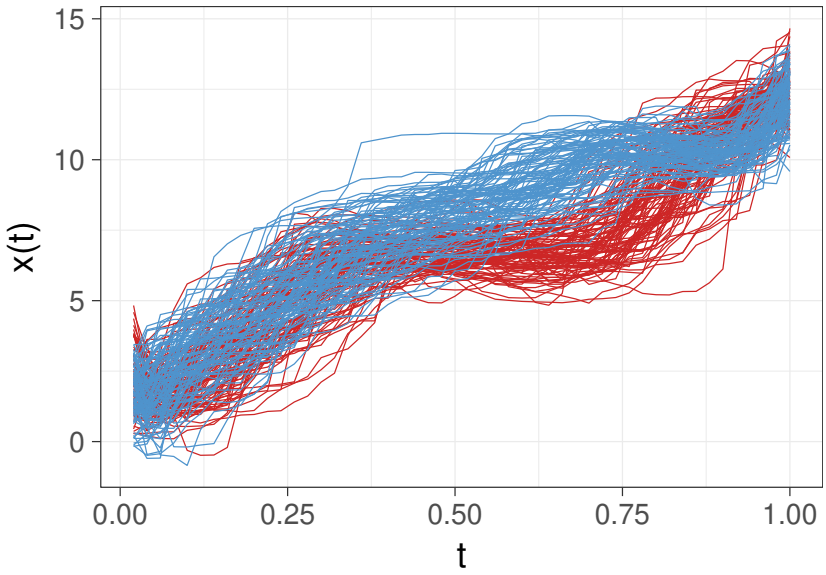

Figure 22: The warped splines for `nclass` = 2, `nobs` = 100, `var_between_class` = 1 and `phase_factor` = 1

2. `nobs` {100}: N_g, the number of observations per class.
3. `var_between_class` {0, 1}: the variance of the spline coefficients sampled for the class centers. If 0, the class centers are the same across all classes.
4. `var_within_class` {1}: the variance of the spline coefficients within the classes.
5. `phase_factor` {0, 1}: ϕ, a binary indicator for un-/informative warping.

Figure 22 shows the warped splines for `nclass` = 2, `nobs` = 100, `var_between_class` = 1 and `phase_factor` = 1.

Results

The results of the benchmark analysis are depicted in Figure 23, showing the rank of each learner for every data set. The 1 nearest neighbor estimator using the Euclidean distance outperforms the estimators using warping distances if the observations contain any information about their class. For the only data set where this is not the case (`ncl = 2, vbc = 1, ph.f = 0`), it performs basically equally well to the amplitude distance (see digital supplement). The kernel estimator vastly outperforms the 1 nearest neighbor estimators.

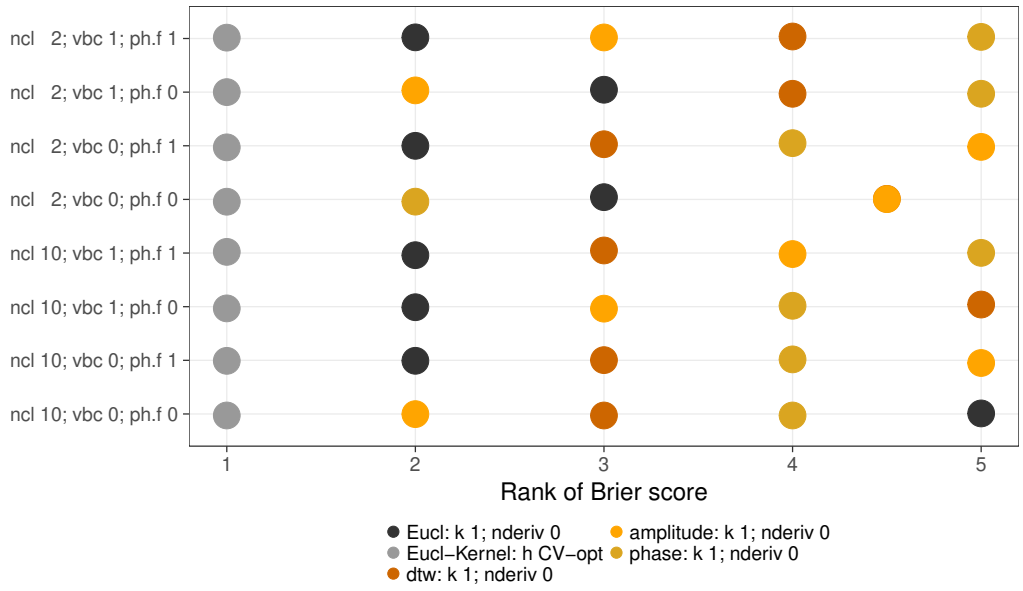

Figure 23: Ranks of the learners in the benchmark analysis for the warped splines data set.

5.4 Benchmark Experiment on Real-world Data

In addition to using simulated data in order to verify theoretical characteristics of the methods, real data sets are used in order to compare the performance of the methods in real-world problems. The data sets are taken from the UCR Time Series Classification Repository (Chen et al., 2015). The same set of models is used as in Section 5.3 (see Table 1 and Table 2). This benchmark experiment will be used to answer the main research questions.

Results

For the benchmark analysis, 17 models are compared over 34 data sets in a 10-fold cross validation. The 34 smallest data sets (number of observations times length per observation) were used in order to reduce the run-time. The total run-time of this benchmark is about 800 CPU hours. For this large number of data sets, a graphical display showing summarized information is provided in Figure 25. Individual boxplots for the performance of every model for every data set can be found in the digital supplement. In addition to showing individual ranks of the model performance per data set, the distribution of the models per rank are visualized in Figure 24. For more than two thirds of the data sets, the random forest ensembles obtain the first and/or the second rank in prediction performance. The nearest neighbor estimator based on the phase distance shows the best performance for a few data sets, but also the worst performance for many others, indicating that the semimetric should be carefully chosen for the individual classification task at hand. The other warping based models also show a wide range of ranks, supporting this theory. A data set where the warping distances greatly outperform the Euclidean distance is the

"Beetle-Fly" data set, see Figure 26. This data set encodes the outlines of 20 beetles and 20 flies as MPEG-7 shape descriptors. It is plausible that warping distances lead to a good classification in this application, as warping may superimpose the same anatomical structures of individual insects. The kernel based model and the nearest neighbor ensembles show performance in the middle ranks. The random forest ensemble appears to be rather unaffected by adding useless semimetrics, whereas the prediction of the nearest neighbor estimator worsens considerably.

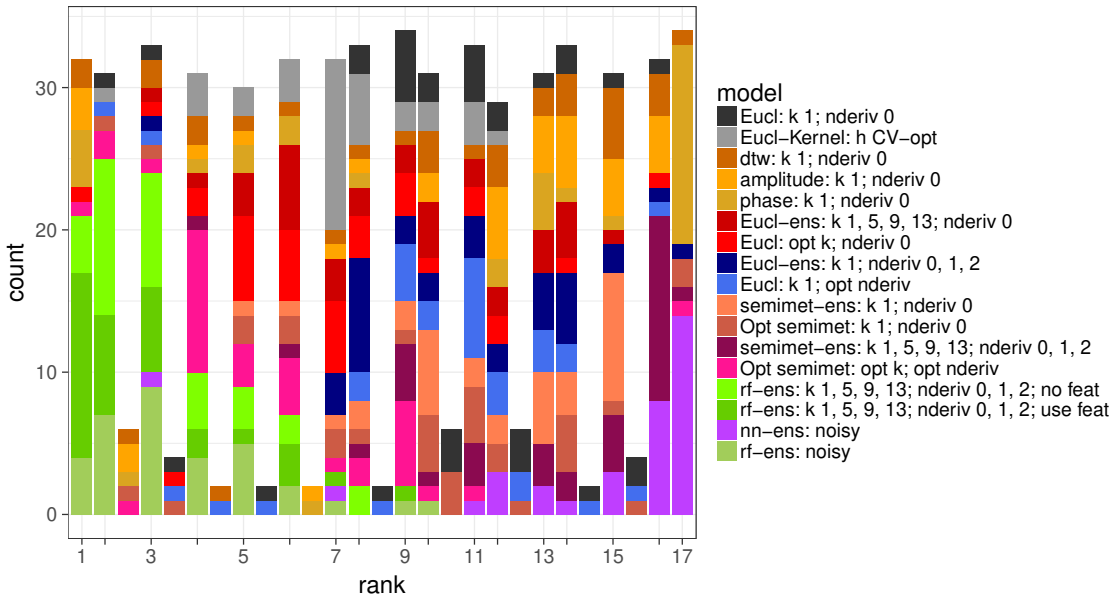

Figure 24: Result of the benchmark analysis for the real-world data sets from the UCR Time Series Classification Repository. For every rank on the x-axis, the distribution of the models is plotted. The more often a model achieves a high rank, the better the model.

Figure 25: Result of the benchmark analysis for the real-world data sets from the UCR Time Series Classification Repository. Depicted is the classification task vs. the ranks of the models.

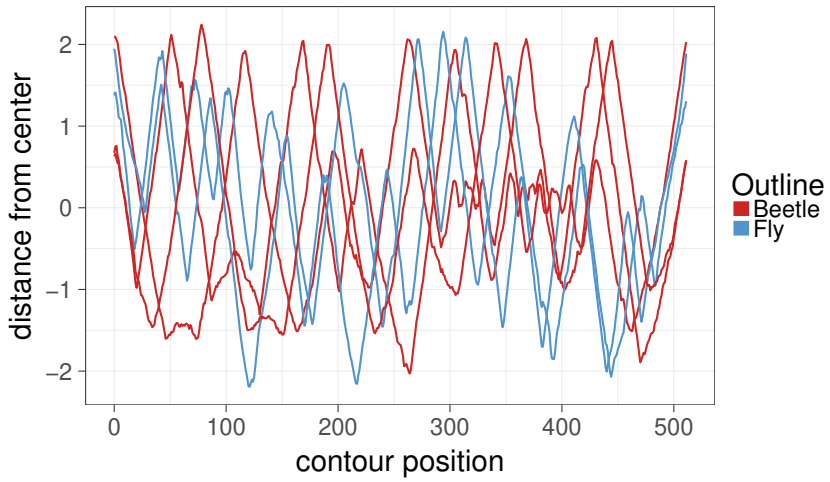

Figure 26: Random sample of 5 observations from the "Beetle-Fly" data set. (Source: `http://www.dabi.temple.edu/šhape/MPEG7/dataset.html`)

Statistical Inference

These findings are supported by the Friedman test and the Nemenyi post-hoc test (see Section 5.1). With the Friedman test showing significant results (see digital supplement), the Nemenyi post-hoc test was calculated for the ranks the models achieved for each data sets in the benchmark analysis. The critical difference diagrams in Figure 27 and Figure 28 are a visual representation of the Nemenyi post-hoc test. Critical difference diagrams show the average rank for every model on the x-axis. The critical difference, meaning the minimal difference in average rank of two models for them to be significantly different, is plotted on top. The gray bars at the bottom connect groups of models that are not significantly different in their ranks. Figure 27 shows that choosing the number of nearest neighbors k, the order of derivative a, and the semimetric d using cross validation outperforms ensembling over these parameters on average. All CV optimal nearest neighbor estimators outperform their respective nearest neighbor ensemble. In the case of ensembling over a, k, and d, the difference in average rank is significant. Figure 28 shows that the random forest ensemble significantly outperforms the nearest neighbor estimator and reference models. Even the nearest neighbor estimator with CV optimal number of nearest neighbors k, derivative of order a, and semimetric d, which is the best of all nearest neighbor ensembles, is vastly outperformed.

Figure 29 shows that the nearest neighbor estimators do not outperform the reference models significantly, if at all. The amplitude and phase distance perform slightly worse than the Euclidean distance. The kernel estimator using the Euclidean distance is on average one rank better that the nearest neighbor ensemble using different numbers of nearest neighbors k. This is an interesting finding, as ensembling over different nearest neighbors k is very similar to a bandwidth optimized kernel estimator.

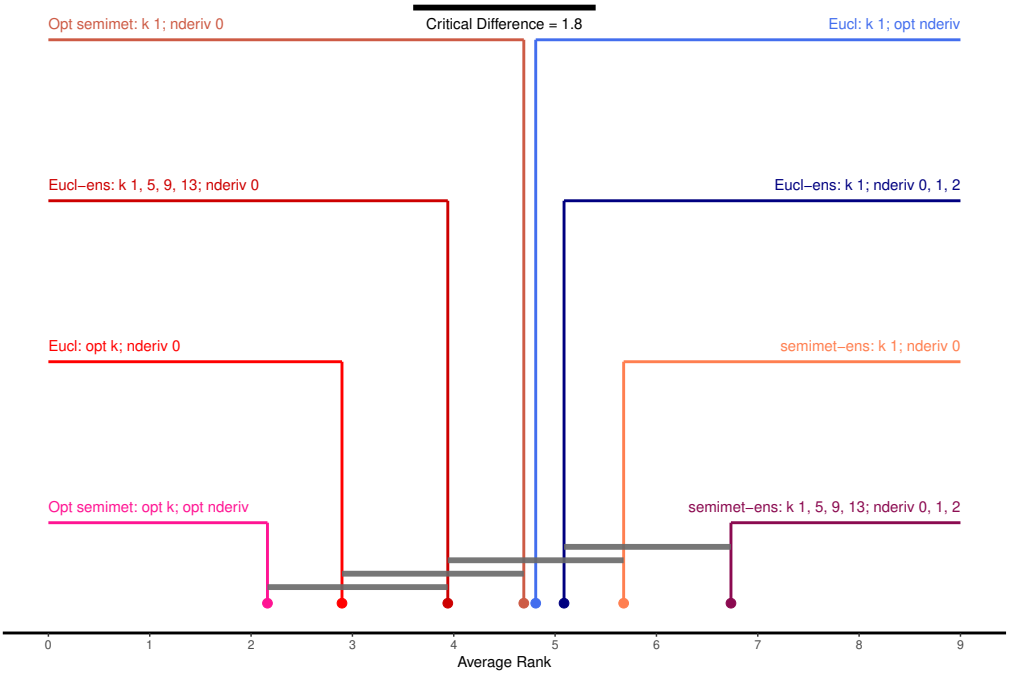

Figure 27: Critical difference diagram ($\alpha = 0.05$) for the benchmark analysis for the real-world data sets from the UCR Time Series Classification Repository. Depicted are nearest neighbor ensembles and their respective one nearest neighbor estimator with CV optimal parameters.

Assessment of Main Goals

Based on these results, the main research questions can be answered regarding the real-world data:

1. The random forest ensemble performs better than the nearest neighbor ensemble given the same base models.

2. The random forest ensemble tends to outperform reference classification models. For the nearest neighbor ensemble this is not necessarily the case.

3. The random forest model is better at detecting (and expelling) useless base models than the nearest neighbor ensemble. It proves to be much more robust to adding uninformative base models into the ensemble.

4. A nearest neighbor ensemble method as proposed in Chapter 3.1 is generally outperformed by the best individual model contained. This is true for ensembling over different numbers of nearest neighbors k, different derivatives of order a, and different semimetrics d. The difference is strikingly large when a large number of rather uninformative base models are added into the ensemble or chosen from using cross validation.

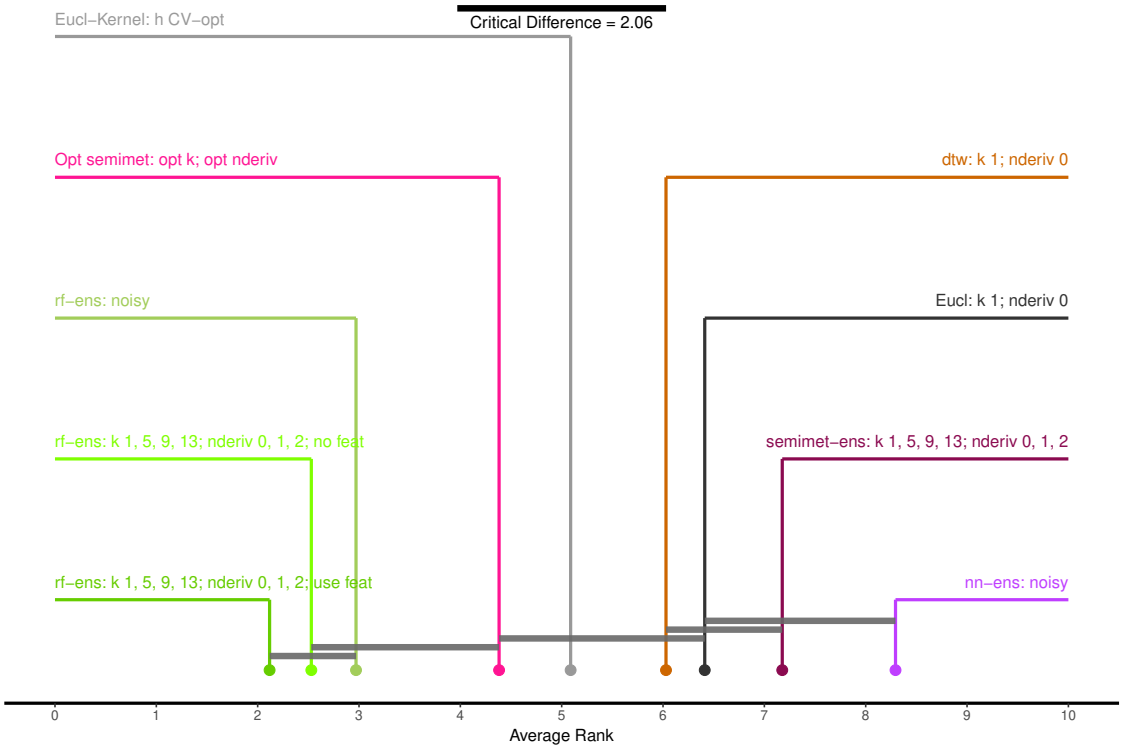

Figure 28: Critical difference diagram ($\alpha = 0.05$) for the benchmark analysis for the real-world data sets. Depicted are the full nearest neighbor ensemble, the random forest ensemble, their noisy counterparts, their CV optimal counterpart, and the reference models.

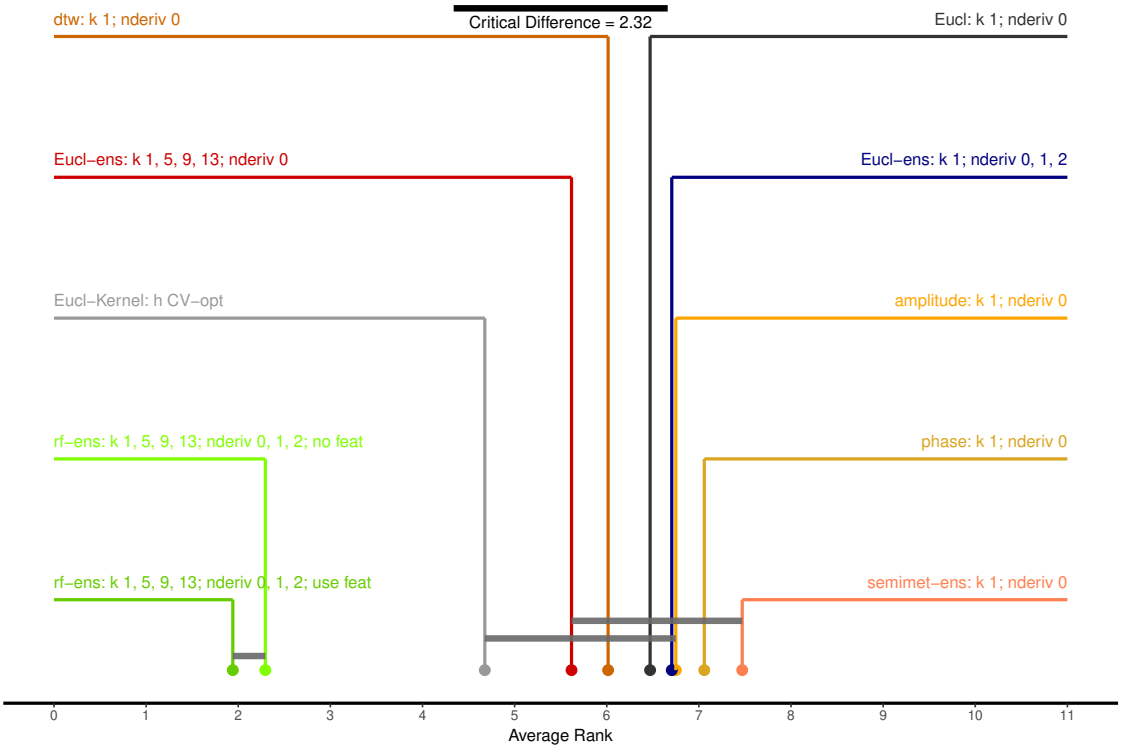

Figure 29: Critical difference diagram ($\alpha = 0.05$) for the benchmark analysis for the real-world data sets. Depicted are the full nearest neighbor ensemble, the random forest ensemble, their CV optimal counterpart, the reference models and the nearest neighbor estimators using warping distances.

6 Discussion

This thesis introduces and implements two functional data classification methods, the nearest neighbor ensemble of Fuchs et al. (2015) and the random forest ensemble. Both classification methods are based on ensembling nearest neighbor estimators for functional data. The random forest model creates more flexible nonlinear combinations of the estimations of the base models and can additionally include multivariate covariables into the model. Both methods are comprehensively implemented in the new **R**-package `classiFunc` (Maierhofer, 2017). This package can be used through the user friendly interface of the `mlr`-package (Bischl et al., 2016), which increases its versatility.

This thesis thoroughly examines the nearest neighbor ensemble and the random forest ensemble in a set of benchmark experiments. This empirical comparison allows drawing conclusions about underlying structures in the modeling approaches. The random forest model outperforms the nearest neighbor ensemble by a large margin. The nearest neighbor ensemble has problems detecting useful base models from a large number of uninformative base models. When the majority of base models is uninformative, the nearest neighbor ensemble completely fails; its prediction accuracy drops below random guessing. The random forest ensemble is more robust to uninformative base models. It filters out useless base models and achieves a similar predictive performance. In contrast to the nearest neighbor ensemble, the random forest ensemble clearly outperforms the reference models used in the benchmark analysis. Ensembling over numbers of nearest neighbors k, derivative order and semimetrics in the nearest neighbor ensemble does not improve performance. Optimizing these parameters using cross validation, instead of ensembling over them, leads to better results and easier interpretation with only slightly inflated variance.

In conclusion, the random forest ensemble is a superior alternative to the nearest neighbor ensemble. It boasts a better predictive performance given the same base models and allows additional multivariate covariables.

Outlook

In future research, the unlimited possibilities for ensemble methods should be further investigated. For example, Fuchs et al. (2016) show that using a restricted multinomial logit model is one promising alternative to the weighted sum in the nearest neighbor ensemble. To improve predictive performance in the random forest ensemble, it could be built with an optimized set of semimetrics and explanatory variables created by dimension reduction techniques such as functional principal component analysis. It will be important for future research to directly compare new ensembles against existing classification methods. Bagnall et al. (2016) provide an inspiring example of direct model comparisons, which should be followed as the field evolves. To keep up with the explosion of available functional data in a principled way, continued research into methods for funtional data classification will be essential.

Acknowledgements

I would like to thank Professor Sonja Greven for her assistance and guidance throughout my thesis. Her help and time is much appreciated and enabled this project. Additionally, I would like to thank Karen Fuchs for her help with the implementation of her classification method in the `classiFunc` package, Almond Stöcker, who introduced me to the Square Root Velocity Framework, and Xudong Sun, my guide into the `mlr`-package. Thank you all for your patient help.

References

Bagnall, A., J. Lines, A. Bostrom, J. Large, and E. Keogh (2016). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 1–55.

Besse, P. C., H. Cardot, and F. Ferraty (1997). Simultaneous non-parametric regressions of unbalanced longitudinal data. *Computational Statistics & Data Analysis 24*(3), 255–270.

Bischl, B., M. Lang, J. Richter, J. Bossek, L. Judt, T. Kuehn, E. Studerus, L. Kotthoff, and S. Julia (2016). *mlr: Machine Learning in R*. R package version 2.8.

Bortolan, G. and J. Willems (1993). Diagnostic ecg classification based on neural networks. *Journal of Electrocardiology 26*, 75–79.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and Regression Trees*. CRC Press.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review 78*(1), 1–3.

Chen, Y., E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista (2015, July). The UCR Time Series Classification Archive. `www.cs.ucr.edu/~eamonn/time_series_data/`.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research 7*(Jan), 1–30.

Eilers, P. H. and B. D. Marx (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, 89–102.

Fahrmeir, L., A. Hamerle, and G. Tutz (1996). *Multivariate statistische Verfahren*. Walter de Gruyter GmbH & Co KG.

Fahrmeir, L., T. Kneib, S. Lang, and B. Marx (2013). *Regression: models, methods and applications*. Springer Science & Business Media.

Febrero-Bande, M. and M. Oviedo de la Fuente (2012). Statistical computing in functional data analysis: The R package fda.usc. *Journal of Statistical Software 51*(4), 1–28.

Ferraty, F. and P. Vieu (2003). Curves discrimination: a nonparametric functional approach. *Computational Statistics & Data Analysis 44*(1), 161–173.

Fourier, J. (1822). *Théorie analytique de la chaleur*. Chez Firmin Didot, père et fils.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association 32*(200), 675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics 11*(1), 86–92.

Fuchs, K., J. Gertheiss, and G. Tutz (2015). Nearest neighbor ensembles for functional data with interpretable feature selection. *Chemometrics and Intelligent Laboratory Systems 146*, 186–197.

Fuchs, K., W. Pößnecker, and G. Tutz (2016). Classification of functional data with k-nearest-neighbor ensembles by fitting constrained multinomial logit models. *arXiv preprint arXiv:1612.04710*.

Gini, C. (1912). Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi*.

Giorgino, T. et al. (2009). Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of Statistical Software 31*(7), 1–24.

Goldsmith, J. and F. Scheipl (2014). Estimator selection and combination in scalar-on-function regression. *Computational Statistics & Data Analysis 70*, 362–372.

Keogh, E. and C. A. Ratanamahatana (2005). Exact indexing of dynamic time warping. *Knowledge and Information Systems 7*(3), 358–386.

Maierhofer, T. (2017). *classiFunc: Classification of Functional Data*. R package version 0.1.0.

Meyer, D. and C. Buchta (2017). *proxy: Distance and Similarity Measures*. R package version 0.4-17.

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, Volume 18, pp. 263. International Biometric Soc 1441 I St, NW, Suite 700, Washington, DC 20005-2210.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Ramsay, J. O. and B. W. Silverman (2006). *Functional Data Analysis*. Wiley Online Library.

Ramsay, J. O., H. Wickham, S. Graves, and G. Hooker (2014). *fda: Functional Data Analysis*. R package version 2.4.4.

Rodríguez, J. J., C. J. Alonso, and J. A. Maestro (2005). Support vector machines of interval-based features for time series classification. *Knowledge-Based Systems 18*(4), 171–178.

Smith, J. W., J. Everhart, W. Dickson, W. Knowler, and R. Johannes (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261. American Medical Informatics Association.

Soetaert, K., K. Van den Meersche, and D. van Oevelen (2009). *limSolve: Solving Linear Inverse Models*. R package 1.5.1.

Srivastava, A. and E. P. Klassen (2016). *Functional and Shape Data Analysis*. Springer.

Suykens, J. A. and J. Vandewalle (1999). Least squares support vector machine classifiers. *Neural Processing Letters 9*(3), 293–300.

Thomas, D. H. (1981). How to classify the projectile points from monitor valley, nevada. *Journal of California and Great Basin Anthropology 3*(1), 7–43.

Tuddenham, R. D. and M. M. Snyder (1954). Physical growth of California boys and girls from birth to eighteen years. *Publications in Child Development. University of California, Berkeley 1*(2), 183.

Wang, J.-L., J.-M. Chiou, and H.-G. Müller (2016). Functional data analysis. *Annual Review of Statistics and Its Application 3*, 257–295.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Ye, L. and E. Keogh (2011). Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data mining and knowledge discovery 22*(1), 149–182.

Appendix

A Digital Supplement

Part of this thesis is a digital supplement. It consists of the folder `Master_Thesis`, which contains the code to reproduce all analysis and figures reported in this thesis as additional figures for the benchmark analysis. Further details are in the contained `README.txt` file.

B Vignette for `classiFunc`-package

classiFunc: Classification of Functional Data

Thomas Maierhofer

2017-05-29

This vignette gives a quick introduction to the key features and functions included in the `classiFunc` package. Please use the Project Page to suggest new features or report bugs. This package offers an extensible and efficient implementation of `k` nearest neighbor classification for functional data.

The following chunk gives a quick introduction to the usage of the `classiFunc` package.

```
library("classiFunc")

# classification of the ArrowHead data set
data("ArrowHead", package = "classiFunc")
classes = ArrowHead[,"target"]

set.seed(123)
# use 80% of data as training set and 20% as test set
train_inds = sample(1:nrow(ArrowHead), size = 0.8 * nrow(ArrowHead), replace = FALSE)
test_inds = (1:nrow(ArrowHead))[!(1:nrow(ArrowHead)) %in% train_inds]

# create functional data as matrix with observations as rows
fdata = ArrowHead[,!colnames(ArrowHead) == "target"]

# create a k = 3 nearest neighbor classifier with Euclidean distance (default) of the
# first order derivative of the data
mod = classiKnn(classes = classes[train_inds], fdata = fdata[train_inds,],
nderiv = 1L, knn = 3L)
# or create a kernel estimator
mod2 = classiKernel(classes = classes[train_inds], fdata = fdata[train_inds,])

# predict the model for the test set
# matrix with the prediction probabilities for the three classes
pred = predict(mod, newdata =  fdata[test_inds,], predict.type = "prob")
```

All functionality of this package can also be accessed through the `mlr` package Project Page. For an introduction on how to use `mlr` check out the Online Tutorial. Currently, the learners are not merged into the Master branch of `mlr`. If you want to use the development version, please download the package from the Project Page. The following chunck gives a quick introduction on how to use the `classiFunc` learners in `mlr`.

```
# download and install the mlr branch containing the classiFunc learners
# devtools::install_github("maierhofert/mlr",
#                          ref = "fda_pull1_task")
```

```
library("mlr")

# classification of the ArrowHead data set
data("ArrowHead", package = "classiFunc")

# create the classiKnn learner for classification of functional data
lrn = makeLearner("fdaclassif.classiKnn")
# create a task from the training data set
task = makeFDAClassifTask(data = ArrowHead[train_inds,], target = "target")
# train the model on the training data task
m.mlr = train(lrn, task)

# predict the test data set
pred = predict(m.mlr, newdata = ArrowHead[test_inds,])
```

By using the `mlr` interface for this package, a multitude of new possibilities are available. One of the key features to be added by the `mlr` package is automatic hyperparameter tuning. In the following chunk a kernel estimator is created that automatically chooses its band width by cross validation.

```
# create the classiKernel learner for classification of functional data
lrn.kernel = makeLearner("fdaclassif.classiKernel", predict.type = "prob")

# create parameter set
parSet.bandwidth = makeParamSet(
  makeNumericParam(id = "h", lower = 0, upper = 10)
)

# control for tuning hyper parameters
# Use higher resolution in application
ctrl = makeTuneControlGrid(resolution = 11L)

# tuned learners
lrn.bandwidth.tuned = makeTuneWrapper(learner = lrn.kernel,
                                      resampling = makeResampleDesc("CV", iters = 5),
                                      measures = mmce,
                                      par.set = parSet.bandwidth,
                                      control = ctrl)

# train the model on the training data task
m = train(lrn.bandwidth.tuned, task)

# predict the test data set
pred = predict(m, newdata = ArrowHead[test_inds,])
```

The Brier score optimal ensemble proposed in Fuchs et al. (2015), Nearest neighbor ensembles for functional data with interpretable feature selection, can also be reproduced using the implementation in `mlr`. A newly implemented stacked learner aggregates the individual base learners to an ensemble learner by creating a weighted mean of their individual predictions. Other ensemble learners can easily be created.

```
# create the base learners
b.lrn1 = makeLearner("fdaclassif.classiKnn",
                     id = "Manhattan.lrn",
                     par.vals = list(metric = "Manhattan"),
                     predict.type = "prob")
b.lrn2 = makeLearner("fdaclassif.classiKnn",
                     id = "mean.lrn",
                     par.vals = list(metric = "mean"),
                     predict.type = "prob")
b.lrn3 = makeLearner("fdaclassif.classiKnn",
                     id = "globMax.lrn",
                     par.vals = list(metric = "globMax"),
                     predict.type = "prob")

# create an ensemble learner as porposed in Fuchs et al. (2015)
# the default uses leave-one-out CV to estimate the weights of the base learners as
proposed in the original paper
# set resampling to CV for faster run time
ensemble.lrn = makeStackedLearner(base.learners = list(b.lrn1, b.lrn2, b.lrn3),
                                  predict.type = "prob",
                                  resampling = makeResampleDesc("CV", iters = 5L),
                                  method = "classif.bs.optimal")

# create another ensemble learner
ensemble.rf.lrn = makeStackedLearner(base.learners = list(b.lrn1, b.lrn2, b.lrn3),
                                     super.learner = "classif.randomForest",
                                     predict.type = "prob",
                                     method = "stack.cv")

# train the model on the training data task
ensemble.m = train(ensemble.lrn, task)

# predict the test data set
pred = predict(ensemble.m, newdata = ArrowHead[test_inds,])
```

Statutory Declaration

I declare that I have developed and written the enclosed Master's Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. This thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

(Location, Date)	(Signature)