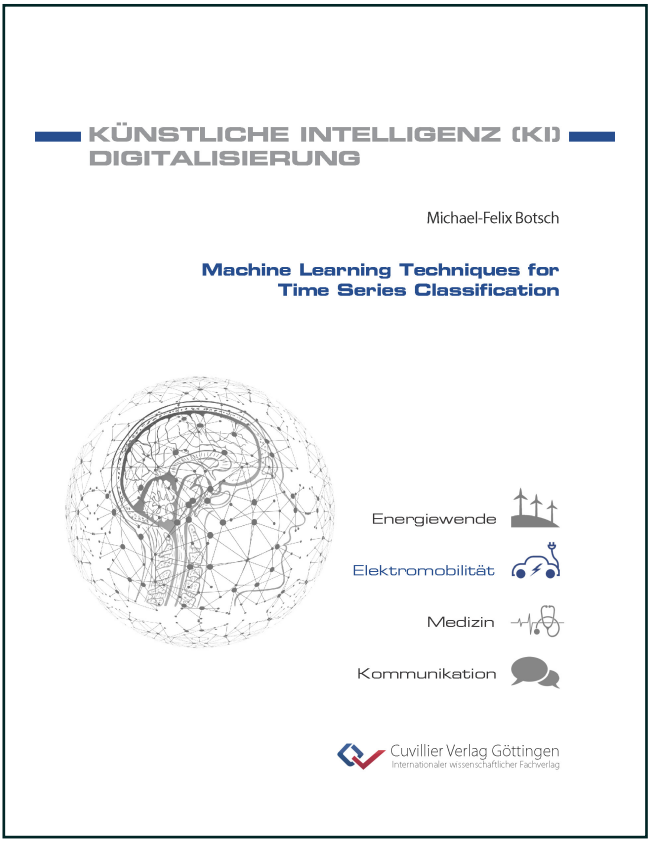
Michael Botsch (Autor)

Machine Learning Techniques for Time Series Classification



https://cuvillier.de/de/shop/publications/8834

1. Introduction

The usage of *machine learning* techniques for *on-line time series classification* is the main topic of this work. The motivation to deal with this task and its central challenges are described in Section 1.1. Major contributions of this thesis are summarized in Section 1.2.

1.1 Motivation

The classification of *temporal data* plays a central role in many fields, e. g., medicine, finance, speech recognition or monitoring of industrial processes. Approaches that are frequently applied in temporal classification tasks typically use statistical models like *Auto-Regressive Moving Average* (ARMA) or *Hidden Markov Models* (HMM). In cases where it is not possible to build accurate models that describe the considered time series the classification task is normally solved by using empirically designed *expert systems*. In recent years the interest in applying machine learning techniques to time series has grown, mainly driven by enhanced computational resources which are required for the large amount of data that has to be processed when dealing with time series. Thus, the difficult task of finding appropriate statistical models or the hand-crafted design of expert systems can be replaced by well-known machine learning methods or by new statistical learning algorithms that are emerging in order to deal with temporal data. Machine learning procedures automatically "learn" the mapping implementing the classifier from observations in such a way that the main task is the acquisition of representative observations rather than the design of analytical models. Hereby, the "learning" is accomplished by solving optimization problems in order to obtain the desired classification result for the available observations. Taking into account that most learning systems can be improved by integrating *domain knowledge* about the problem at hand, machine learning classifiers normally achieve a better performance than expert systems, although they can be designed with less effort. This makes machine learning an attractive approach to tackle classification problems.

Since time series are complex data structures the common framework used in machine learning for classification can not be applied straightforwardly and modifications must be included in order to take the temporal aspect into account. The common machine learning framework assumes that an object which has to be classified is described by *features*. Each feature represents a characteristic of the object. When dealing with temporal data, the object to be classified is described by trajectories which are represented in this work as time series. It is inappropriate to treat each sample in the series as a feature of the object to be classified for two reasons: firstly, this would lead to a huge number of features and secondly, it is rather the interdependence between samples that contains the characteristic information which is required for classification and should be integrated into features. In this work possibilities to represent time series in such a way that the common machine learning framework can be applied will be discussed.

The main difference of this thesis compared to other works that deal with the application of machine learning techniques for the classification of temporal data is the fact that the

time series considered here can change their class label over time so that the classification problem covers two main tasks: the detection of those time instances when the class label changes and the correct assignment of the class label. Works in the technical literature that deal with the application of machine learning to time series classification only consider the latter task and assume that the series are already divided into segments in such a way that each segment belongs to just one class. For practical applications including dynamic systems that generate time series which must be classified the former task is also highly important and therefore will be covered in this thesis. Whenever a new sample of a time series becomes available it has to be decided in real-time whether a class change has occurred and if yes, which one. Thus, the task will be called *on-line time series classification*.

Although many machine learning algorithms are able to solve a large number of classification problems that appear in practical applications with high accuracy they have the drawback of not being *interpretable*. For most classification tasks there is a tradeoff between interpretability and low error rates which stems from the inability of humans to imagine arbitrary hypersurfaces in high-dimensional spaces. Since interpretability is often desired as in case of safety critical applications the design of interpretable classifiers is an important issue that will be treated in this work.

The main application that will be discussed is the design of classifiers for detection and categorization of *car crashes*. The aim hereby is the deployment of safety systems, e. g., belt tensioners or airbags, at time instances where the best-possible protection of passengers is assured. This is a safety critical application where a decision must be taken based on temporal data stemming from sensors, e. g., deceleration or pressure sensors which are incorporated in modern cars. State of the art algorithms for this task are expert systems which rely on empirical experience and require lots of hand-crafted "trial and error" steps for each car type. Therefore, applying machine learning procedures represents a means to reduce the development costs and to improve the flexibility and performance of the employed algorithms. Solving the car crash classification problem with machine learning techniques requires the usage of all topics that were mentioned above: a suitable representation of time series, the detection of those time instances when a class change occurs, i. e., when to deploy safety systems, a high accuracy, i. e., the correct decision about what safety systems to deploy, and interpretability.

1.2 Outline and Major Contributions of the Thesis

The thesis is divided into two main parts. The focus of the first part that is covered by Chapters 2 and 3 is the classification task based on machine learning techniques whereas the second part—Chapters 4 to 7—deals with temporal data, the possibilities to classify time series using machine learning procedures, and the car crash application.

Chapter 2 starts by introducing the basic concepts of machine learning. Hereby, the generalization ability of classification systems is treated in more detail, i. e., the property of classifiers to take the best decision not only for the data that has been used in the design phase but also for new, unseen data. In this context the bias-variance framework is very useful since it gives insight into the tradeoff that must be found when choosing the complexity of the model that is used to learn the underlying process generating the data. Since a main aim in the design of classifiers is a good generalization ability also methods that can be applied to measure the performance will be discussed in this context. Chapter 2 continues

by presenting important machine learning algorithms. The aim is to convey the basics of some learning algorithms that have proven their suitability on a large number of practical applications and to introduce the relatively new field of ensemble learning. In this part of the thesis an approach to explain the good generalization ability of ensemble techniques for classification tasks using the bias-variance framework is presented. Similar approaches exist for regression tasks but there the bias-variance framework is different. Having stressed the advantages of ensemble techniques a special representative, the *Random Forest* (RF) algorithm, is introduced since it is the basic algorithm that is used throughout the thesis to design classification systems and to develop new algorithms. The success of a classification system does not only depend on the classification algorithm but also on the way how the data is represented for the algorithm. Thus, the so-called feature extraction is highly important, i. e., the extraction of a small but representative set of attributes that facilitate a good classification performance. The last part of Chapter 2 covers this topic.

A major contribution of this thesis is described in Chapter 3 where a possibility to design interpretable *Generalized Radial Basis Functions* (GRBF) based on the RF kernel is introduced. After showing how the RF kernel can be deduced from the kernel of a single decision tree, the similarity measure described by the RF kernel is used during the construction of GRBF classifiers in order to assure a good generalization performance. GRBF classifiers have some advantages that other classifiers do not have: they allow interpretability which is important in safety critical applications and they offer the option to reject decisions. Both properties will be used in the second part of the thesis for the car crash application. Chapter 3 describes in detail how GRBF can be constructed using the RF kernel and shows how interpretability can be achieved by using a constrained optimization step. Moreover, it is presented how the number of generalized radial basis functions can be reduced in the GRBF classifier while assuring a good classification performance. The chapter ends with experimental results that affirm the advantages of the proposed classification algorithm.

Chapter 4 describes the temporal classification problem, elaborating on the possibilities to represent temporal data, the difficulties that arise when dealing with the topic and possible ways to handle it. Some common approaches for the classification of temporal data are reviewed. Since the usage of suitable similarity measures between time series is a possible access to the classification task the chapter presents some standard measures and shows how these measures can be applied to construct class-specific prototypes. In Chapter 4 a new similarity measure is introduced which will be called *Augmented Dynamic Time Warping* (ADTW) similarity. Due to its property to capture both the similarity in shape as well as the duration of time series the ADTW similarity represents an adequate measure for the car crash application. In the final part of the chapter some possibilities to generate global or local features from time series are reviewed and a new type is introduced that will be called *event-based features*. The essence of event-based features is the usage of classifiers in the generation step in such a way that application specific events can be detected and the time instances when these events occur are incorporated into features.

The main application that is used in the second part of the thesis to evaluate the techniques that are developed in this work is presented in Chapter 5. The classification of car crashes is a challenging task for a couple of reasons. Firstly, the task involves two classification steps: the detection of a suitable time instance when to deploy safety systems and the decision about what safety systems to activate. Secondly, causality must be taken into account, i. e., at time instances when a decision must be taken only sensor signals up to this time stamp

are available, which leads to time series of different length. Thirdly, it is a safety critical application which comes along with high demands on the classification performance while favoring interpretable classifiers. The chapter describes how the performance of car crash classification systems can be measured. Both the time instance when to deploy as well as the decision about what safety systems to activate depends on the crash severity. Thus, measuring the performance must evaluate both aspects.

In Chapter 6 a contribution of this thesis to the problem of on-line time series classification is presented. The approach is an expansion of the RF algorithm to temporal data which is why it will be denoted *Scenario-Based Random Forest* (SBRF) algorithm. SBRF comes along with all advantages of the RF algorithm making it possible to compute an honest estimate of the classification performance without putting aside a subset of observations from the available data set. A highly important property of the SBRF algorithm for practical applications is its ability to perform feature selection. Reducing the number of features that are used in the classification task not only decreases the computational load but it also facilitates a good generalization ability. On-line classification using the SBRF approach is performed by computing at each time stamp a feature vector and assigning this feature vector to a class. The chapter ends by applying the SBRF algorithm to the car crash classification task. The results for two data sets that stem from real car crashes are shown.

In Chapter 7 the temporal time series classification problem is explicitly divided into two subtasks: detecting time instances when class changes possibly occur and assigning class labels. The former subtask segments a time series into intervals that belong to the same class and the latter assigns class labels to the segments. In contrast to the SBRF approach, here the label is not computed at each time instance but only when the segmentation classifier signals a possible class change. The construction of linear segmentation classifiers is discussed and than adapted to the car crash classification task. In the final part of the chapter two labeling classification techniques are applied to the car crash datasets. The first is realized using GRBF classifiers that are constructed as described in Chapter 3 and the second using the ADTW similarity measure from Chapter 4.

The thesis ends with Chapter 8 where the presented methods for the car crash application are compared and aspects for future work in the field of on-line time series classification using machine learning techniques are discussed.

1.3 Notation

Throughout the thesis vectors and matrices are denoted by lower and upper case bold letters. Random variables are written using *sans serif* fonts. The matrix $\boldsymbol{I}_n$ is the $n \times n$ identity matrix, $\boldsymbol{e}_i$ its i-th column, and $\mathbf{0}_n$ the n-dimensional zero vector. The symbol "$*$" denotes the element-wise multiplication, $\mathrm{tr}\{\cdot\}$ the trace of a matrix, $\mathrm{E}_{\mathsf{x}}\{\cdot\}$ the expectation with respect to x, $(\cdot)^{\mathrm{T}}$ transpose, $\lfloor\cdot\rfloor$ floor, $\lceil\cdot\rceil$ ceil, $\mathrm{O}(\cdot)$ the Landau symbol, and $\|\cdot\|$ the norm of a vector. A list of the most important symbols that are used in the thesis can be found in Appendix B. Expressions are emphasized by writing them in italic type.

2. Machine Learning

The branch of science that deals with the automatic discovery of regularities in data through the use of computer algorithms is called *machine learning*. If the discovery of regularities in data is not necessarily coupled to the use of computers one talks about *statistical learning*. Machine learning plays an important role in the areas of data mining, artificial intelligence, statistics and in various engineering disciplines. The focus of this thesis lies on the latter, aiming to use machine learning for the design of technical systems that have to react to signals coming from the environment by tuning the parameters of an adaptive model in such a way that an application-specific behavior is realized.

The first section of this chapter introduces the basics underlying statistical learning. Section 2.2 presents state of the art algorithms for machine learning with a focus on linear basis expansion models, *Classification and Regression Trees* (CART), and the *Random Forest* (RF) algorithm since these methods are the basis for techniques that are developed later in the thesis for the task of temporal classification. Section 2.3 addresses the problem of finding the most compact and informative representation of data which is then used by a machine learning algorithm to realize the desired behavior.

2.1 Basics of Statistical Learning

Many relations that are found by statistical learning methods in data can be represented in the form of *classification* or *regression* functions. Classification and regression aim at estimating values of an attribute of a system based on previously measured attributes of this system. Given a set of measured observation attributes $\boldsymbol{v} = [v_1, \ldots, v_{N'}]^{\mathrm{T}} \in \mathbb{R}^{N'}$, statistical learning methods estimate the values of a different attribute y. If y takes on continuous numerical values, i. e., $y \in \mathbb{R}$ one talks about regression and if it takes on discrete values from a set of K categorical values, called classes, i. e., $y \in \{c_1, \ldots, c_K\}$ one talks about classification. Often a preprocessing of the observation vector $\boldsymbol{v}$ is performed in order to simplify the mapping

$$\tilde{f} : \mathbb{R}^{N'} \to \mathbb{R},\ \boldsymbol{v} \mapsto y \quad \text{for regression and} \tag{2.1}$$

$$\tilde{f} : \mathbb{R}^{N'} \to \{c_1, \ldots, c_K\},\ \boldsymbol{v} \mapsto y \quad \text{for classification.} \tag{2.2}$$

Preprocessing plays a very important role being a possibility to introduce *a priori* knowledge about the considered machine learning problem. This preprocessing transforms the observation vector $\boldsymbol{v}$ into the so-called feature vector $\boldsymbol{x} \in \mathbb{R}^N$. Defining feature vectors is the most common and convenient means of data representation for classification and regression problems. A pair $(\boldsymbol{x}, y)$ is called a pattern, $\boldsymbol{x}$ the "input" and y the "output" or "target". Because the measured attribute values are subject to variations which often cannot be described deterministically, a statistical framework must be adopted [Vid03]. In this framework, $\boldsymbol{x}$ is the realization of the random variable $\mathbf{x}$ and y of the random variable y. One can think of the

mapping from $\boldsymbol{v}$ to y or the mapping from $\boldsymbol{x}$ to y as a black box representing the process of interest.[1]

In machine learning one is interested both in generating from the observation $\boldsymbol{v}$ a feature vector $\boldsymbol{x}$ that is suitable for the application at hand and in estimating the mapping from $\boldsymbol{x}$ to y using a set of M already known correspondences, the so-called training set[2]

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_M, y_M)\}. \tag{2.3}$$

Whereas most of the literature focuses on finding an estimate of the mapping from $\boldsymbol{x}$ to y, i. e., on computing a suitable mapping

$$f : \mathbb{R}^N \to \mathbb{R},\ \boldsymbol{x} \mapsto \hat{\mathsf{y}}, \quad \text{for regression and} \tag{2.4}$$

$$f : \mathbb{R}^N \to \{c_1, \ldots, c_K\},\ \boldsymbol{x} \mapsto \hat{\mathsf{y}}, \quad \text{for classification,} \tag{2.5}$$

it should be noted that for a good performance of the learning system, which enables to predict accurately the output y for a new unseen measurement vector $\boldsymbol{v}$, the construction of the feature vector $\boldsymbol{x}$ is extremely important. Fig. 2.1 shows that the computed output $\hat{\mathsf{y}}$ can only be a good estimate of the target y corresponding to $\boldsymbol{v}$ if both the feature extraction and the mapping f are chosen properly. The topic of generating suitable feature vectors

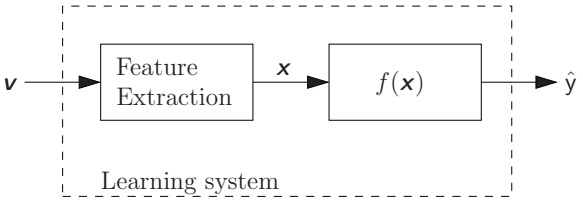


Figure 2.1: Learning system

is discussed in Section 2.3. The current and the following section focus on computing the mapping f.

In order to determine a suitable mapping f, a measure for the quality of the mapping is required. Based on statistical decision theory, not the quality but the lack of quality of f is measured. Firstly, a *loss* $\mathcal{L}(\mathsf{y}, \hat{\mathsf{y}})$ must be defined which assigns a cost to the prediction $\hat{y} = f(\boldsymbol{x})$, knowing that the true value is y. The measure for the lack of quality is the *prediction risk* $R(f)$—also called *generalization error*—which is defined as the expectation of $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x}))$ over the $\boldsymbol{x}, \mathsf{y}$-space. For regression one obtains

$$R(f) = \mathrm{E}_{\boldsymbol{x},\mathsf{y}}\left\{\mathcal{L}(\mathsf{y}, f(\boldsymbol{x}))\right\} = \int_{\mathbb{R}^N} \int_{\mathbb{R}} \mathcal{L}(y, f(\boldsymbol{x}))\, \mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = y)\, \mathrm{d}y \mathrm{d}\boldsymbol{x}, \tag{2.6}$$

[1] In regression the mapping performed by the black box can be modeled by $\mathsf{y} = f_{\mathrm{true}}(\boldsymbol{x}) + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is noise and f_{true} the noiseless mapping from $\boldsymbol{x}$ to y. In classification the mapping can be represented by $\mathsf{y} = f_{\mathrm{discr}}(f_{\mathrm{true}}(\boldsymbol{x}) + \boldsymbol{\varepsilon})$, where f_{discr} maps the real valued expression $f_{\mathrm{true}}(\boldsymbol{x}) + \boldsymbol{\varepsilon}$ to the classes c_k.

[2] This kind of learning problem is called *supervised learning* because in the training set to each input vector $\boldsymbol{x}_m$ the corresponding target y_m is known. In *unsupervised learning* only a set of feature vectors without their corresponding targets is available and instead of predicting the output for an unseen input the task changes to describing how the data is organized or clustered.

where $\mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = y)$ is the joint probability density function of $\boldsymbol{x}$ and y. For classification the risk is defined as

$$R(f) = \mathrm{E}_{\boldsymbol{x},\mathsf{y}}\left\{\mathcal{L}(\mathsf{y}, f(\boldsymbol{x}))\right\} = \int_{\mathbb{R}^N} \sum_{k=1}^{K} \mathcal{L}(c_k, f(\boldsymbol{x}))\, \mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = c_k)\, \mathrm{d}\boldsymbol{x}. \tag{2.7}$$

In this framework, the aim in machine learning is to find the function $f_{\mathrm{B}}(\boldsymbol{x})$ which minimizes the prediction risk $R(f)$

$$f_{\mathrm{B}} = \operatorname*{argmin}_{f} \{R(f)\}. \tag{2.8}$$

The function $f_{\mathrm{B}}(\boldsymbol{x})$ is called the *Bayes regression function* or *Bayes classifier*, respectively.

Commonly used loss functions in regression are the absolute error $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x})) = |\mathsf{y} - f(\boldsymbol{x})|$ or the squared error $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x})) = (\mathsf{y} - f(\boldsymbol{x}))^2$ whereas in classification problems all possible values of the loss $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x}))$ can be stored in a $K \times K$ matrix. Often the loss for classification is chosen to be the 0/1-loss

$$\mathcal{L}(\mathsf{y}, f(\boldsymbol{x})) = 1 - \delta(\mathsf{y}, f(\boldsymbol{x})) = \begin{cases} 0, & \text{if} \quad \mathsf{y} = f(\boldsymbol{x}) \\ 1, & \text{otherwise,} \end{cases} \tag{2.9}$$

where $\delta(\cdot, \cdot)$ is a function that generates the output 1 if its arguments are equal and 0 otherwise.

The probability density functions are normally not known in practical applications and therefore $f_{\mathrm{B}}(\boldsymbol{x})$ cannot be computed. Thus, the empirical risk is introduced which defines a measure for the lack of quality of $f(\boldsymbol{x})$ based on the training set $\mathcal{D}$

$$R_{\mathrm{emp}}(f, \mathcal{D}) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(y_m, f(\boldsymbol{x}_m)). \tag{2.10}$$

The empirical risk from (2.10) is an unbiased estimate of $R(f)$ obtained from the training set $\mathcal{D}$. Learning procedures can now be applied in order to find an estimate of the mapping from $\boldsymbol{x}$ to y by minimizing $R_{\mathrm{emp}}(f, \mathcal{D})$.

2.1.1 Parametric and Nonparametric Techniques for Classification

The joint probability density $\mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = c_k)$ offers a complete summary of the uncertainty associated to the random variables $\boldsymbol{x}$ and y, such that the knowledge of $\mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = c_k)$ allows the computation of the optimal classifier f_{B}. An arbitrary classification algorithm implementing the mapping f segments the input space $\mathbb{R}^N$ into regions $\mathcal{X}_\ell$, in such a way that all inputs $\boldsymbol{x} \in \mathcal{X}_\ell$ are assigned to the class c_ℓ. Therefore, the prediction risk from (2.7) can be rewritten as

$$R(f) = \mathrm{E}_{\boldsymbol{x},\mathsf{y}}\left\{\mathcal{L}(\mathsf{y}, f(\boldsymbol{x}))\right\} = \sum_{k=1}^{K} \sum_{\ell=1}^{K} \int_{\mathcal{X}_\ell} \mathcal{L}(c_k, c_\ell) \mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = c_k) \mathrm{d}\boldsymbol{x}, \tag{2.11}$$

with $\mathcal{L}(c_k, c_\ell)$ representing the costs of assigning an input $\boldsymbol{x}$ to the output c_ℓ when the true class is c_k. During the design of a classification algorithm one aims at choosing the regions

$\mathcal{X}_\ell$ in such a way that the prediction loss from (2.11) is minimized. As a consequence the regions $\mathcal{X}_\ell$ emerge by minimizing for each $\boldsymbol{x} \in \mathbb{R}^N$ the cost function $\sum_{k=1}^{K} \mathcal{L}(c_k, f(\boldsymbol{x}))\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}, \mathsf{y} = c_k)$ leading to the Bayes classifier

$$f_{\mathrm{B}}(\boldsymbol{x}) = \underset{f(\boldsymbol{x})}{\operatorname{argmin}} \left\{ \sum_{k=1}^{K} \mathcal{L}(c_k, f(\boldsymbol{x}))\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}, \mathsf{y} = c_k) \right\} \tag{2.12}$$

$$= \underset{c_\ell}{\operatorname{argmin}} \left\{ \sum_{k=1}^{K} \mathcal{L}(c_k, c_\ell)\mathrm{p}(\mathsf{y} = c_k | \boldsymbol{\mathsf{x}} = \boldsymbol{x}) \right\}, \tag{2.13}$$

since the common factor $\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x})$ can be eliminated.[3] Assuming the 0/1-loss from (2.9) the Bayes classifier can be simplified, resulting in the so-called *Maximum-A-Posteriori* (MAP) classifier[4]

$$f_{\mathrm{MAP}}(\boldsymbol{x}) = \underset{c_\ell}{\operatorname{argmax}} \{\mathrm{p}(\mathsf{y} = c_\ell | \boldsymbol{\mathsf{x}} = \boldsymbol{x})\}. \tag{2.14}$$

If additionally the priors $\mathrm{p}(\mathsf{y} = c_k)$ are equal for all classes the MAP decision rule can be further simplified to the so-called Maximum Likelihood classifier[5]

$$f_{\mathrm{ML}}(\boldsymbol{x}) = \underset{c_\ell}{\operatorname{argmax}} \{\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}) | \mathsf{y} = c_\ell\}. \tag{2.15}$$

In practical applications the probabilities that are required for the ML, MAP or Bayes decision rule are normally not known. Nevertheless, in some applications parameterized forms of the underlying densities $\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}, \mathsf{y} = c_k)$ or $\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x} | \mathsf{y} = c_k)$ are available, which makes it possible to use the training set in order to estimate the values of unknown parameters in the distributions. Then, based on the computed densities, one of the optimal decision rules from (2.13), (2.14) or (2.15) is applied. This approach is called a *parametric classification* method. On the other hand, if one uses methods for classification that do not require knowledge of the forms of the probability distributions, the approach is said to be *nonparametric*. A possible way to perform classification by applying nonparametric techniques is to estimate

[3] According to Bayes rule the joint probability density can be decomposed into

$$\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}, \mathsf{y} = c_k) = \mathrm{p}(\mathsf{y} = c_k | \boldsymbol{\mathsf{x}} = \boldsymbol{x})\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}).$$

[4] The MAP classifier results from the simplification

$$f_{\mathrm{B}}(\boldsymbol{x}) = \underset{c_\ell}{\operatorname{argmin}} \left\{ \sum_{k=1}^{K} (1 - \delta(c_k, c_\ell))\mathrm{p}(\mathsf{y} = c_k | \boldsymbol{\mathsf{x}} = \boldsymbol{x}) \right\}$$

$$= \underset{c_\ell}{\operatorname{argmax}} \left\{ \sum_{k=1}^{K} \mathrm{p}(\mathsf{y} = c_k | \boldsymbol{\mathsf{x}} = \boldsymbol{x}) - \sum_{k=1}^{K} (1 - \delta(c_k, c_\ell))\mathrm{p}(\mathsf{y} = c_k | \boldsymbol{\mathsf{x}} = \boldsymbol{x}) \right\} = \underset{c_\ell}{\operatorname{argmax}} \{\mathrm{p}(\mathsf{y} = c_\ell | \boldsymbol{\mathsf{x}} = \boldsymbol{x})\}.$$

[5] Due to Bayes rule the MAP classifier can be simplified for equal priors:

$$f_{\mathrm{ML}}(\boldsymbol{x}) = \underset{c_\ell}{\operatorname{argmax}} \{\mathrm{p}(\mathsf{y} = c_\ell | \boldsymbol{\mathsf{x}} = \boldsymbol{x})\} = \underset{c_\ell}{\operatorname{argmax}} \left\{ \frac{\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x} | \mathsf{y} = c_\ell)\mathrm{p}(\mathsf{y} = c_\ell)}{\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x})} \right\} = \underset{c_\ell}{\operatorname{argmax}} \{\mathrm{p}(\boldsymbol{\mathsf{x}} = \boldsymbol{x}) | \mathsf{y} = c_\ell\}.$$

$\mathrm{p}(\boldsymbol{x} = \boldsymbol{x}, \mathsf{y} = c_k)$ or $\mathrm{p}(\boldsymbol{x} = \boldsymbol{x}|\mathsf{y} = c_k)$ without using any model of the densities, e. g., with kernel methods, and then to take a decision based on the estimated densities and one of the Eqs. (2.13), (2.14) or (2.15).

One of the fundamental problems in statistical learning is the so-called *curse of dimensionality* which states that with increasing dimensionality of the input space it becomes harder to construct accurate statistical models. A manifestation of the curse of dimensionality that can be easily remembered is the fact that the sampling density is proportional to $M^{1/N}$, where M is the size of the training set and N the dimensionality of the input space. If one divides a region of the input space into regular cells, then the number of cells grows exponentially with the dimensionality of the input space and as a consequence one needs an exponentially large number of examples in the training set to ensure that the cells are not empty. Therefore, the methods that are applied in statistical learning use assumptions that lie beyond the observed data to perform well in high dimensional spaces. More details on the curse of dimensionality can be found in [Bel61] or [HTF01].

Almost all machine learning procedures that are used in practice are nonparametric techniques and some of them are presented in Section 2.2. When considering a classification problem the structure in the observed data is a result of the fact that the data is produced by a source which has certain statistical properties in such a way that the generated observations $\boldsymbol{v}$ do not fill out the whole input space $\mathbb{R}^{N'}$. Moreover, observations $\boldsymbol{v}$ which lie close to each other with respect to a certain metric that is determined by the data source also belong to the same class. Nonparametric procedures exploit these facts by specifying metrics for measuring the neighborhood in the input space. Kernel methods or CART explicitly specify the metric, whereas methods based on linear combinations of basis functions implicitly define a metric. It is very important to know that all methods that aim to escape the curse of dimensionality have explicitly or implicitly defined a metric for measuring the neighborhoods. This metric does not allow the neighborhood to be simultaneously small in all directions [HTF01]. Thus, every nonparametric technique makes intrinsic statistical assumptions about the data but in general it can not be predicted in advance if these assumptions are valid, i. e., if the algorithm is suitable for the considered classification task. A frequently adopted approach in machine learning is to check which of the available classifiers performs well, in such a way that the most adequate algorithm from the cost and performance point of view can be chosen.

2.1.2 Learning and Generalization in Classification Tasks

The aim of a machine learning algorithm is to find a "good" approximation of the mapping from $\boldsymbol{x}$ to y. The approximation is "good" if it generalizes well, i. e., if it represents the underlying systematic properties of the data such that the prediction loss is small. In this context the question arises whether a classification method is superior or inferior to other methods, if no prior assumptions about the nature of the classification task are made. The answer to this question is given by the *no free lunch theorem* which states that there are no context-independent or usage-independent reasons to favor one classifier over another [Sch94]. A superior classifier can only exist if the probability over the class of problems is not uniform. If an algorithm is superior to another it is due to the fact that its intrinsic assumptions fit the particular problem better. The no free lunch theorem is an essential theoretical result

since it states that for a new classification task one should focus on the prior knowledge that one has about the process generating the data.

2.1.2.1 Bias-Variance Decomposition

In the following the so-called *bias-variance* framework will be introduced to measure the suitability of a learning algorithm for a specific classification problem. Colloquially, the bias measures the suitability of the chosen learning model for the problem at hand and the variance the specificity of a realization of the learning model. Given a certain type of classifier, e. g., neural networks or CART, both bias and variance can be influenced when designing the algorithm, but the two terms are not independent, a decreasing bias leading to an increasing variance and vice versa. This tradeoff is discussed in detail below because it represents a key to understand the generalization ability of statistical learning algorithms.

By deciding to use a certain type of machine learning algorithm, e. g., neural networks or CART, one chooses the family of functions—also called *model space*—to which f belongs. If the model space is too large, f can be determined based on the training set $\mathcal{D}$ in such a way that the empirical risk is very small but the total risk $R(f)$ is large. This is called *overfitting* and the chosen model f has a high variance.[6] Intuitively, a model has a high variance if two different training sets $\mathcal{D}_1$ and $\mathcal{D}_2$ which are produced by the same source lead to significant different mappings f. On the other hand, if the mapping f differs significantly on average over the training sets from the optimal model f_B, then the learning model f has a high bias. This may occur for example if the model space is too small and the function f does not have the flexibility to estimate the optimal mapping from $\boldsymbol{x}$ to y.

Although regression is not the focus of this work, the bias-variance decomposition of the prediction risk for regression tasks will be introduced since it is required at various places in the thesis. In regression the most widely used loss function is $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x})) = (\mathsf{y} - f(\boldsymbol{x}))^2$, leading to the minimization of the mean squared error. With this loss function and taking into account that $f(\boldsymbol{x})$ is constructed based on a random training set D, the prediction risk can be decomposed as[7]

$$\begin{aligned}\mathrm{E}_{\boldsymbol{x},\mathsf{y},\mathsf{D}}\left\{(\mathsf{y} - f(\boldsymbol{x},\mathsf{D}))^2\right\} = &\underbrace{\mathrm{E}_{\boldsymbol{x}}\left\{\mathrm{E}_{\mathsf{y}|\boldsymbol{x}}\left\{\mathsf{y}^2\right\} - f_\mathrm{B}(\boldsymbol{x})^2\right\}}_{e_\mathrm{i}=\mathrm{E}_{\boldsymbol{x}}\{\mathrm{ile}(\boldsymbol{x})\}} + \underbrace{\mathrm{E}_{\boldsymbol{x}}\left\{(f_\mathrm{B}(\boldsymbol{x}) - \mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{f(\boldsymbol{x},\mathsf{D})\right\})^2\right\}}_{e_\mathrm{b}=\mathrm{E}_{\boldsymbol{x}}\left\{(\mathrm{bias}_{\hat{\mathsf{y}}|\boldsymbol{x}}\{\hat{\mathsf{y}}\})^2\right\}} \\ &+ \underbrace{\mathrm{E}_{\boldsymbol{x}}\left\{\mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{(f(\boldsymbol{x},\mathsf{D}) - \mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{f(\boldsymbol{x},\mathsf{D})\right\})^2\right\}\right\}}_{e_\mathrm{v}=\mathrm{E}_{\boldsymbol{x}}\left\{\mathrm{var}_{\hat{\mathsf{y}}|\boldsymbol{x}}\{\hat{\mathsf{y}}\}\right\}},\end{aligned} \tag{2.16}$$

where the optimal solution $f_\mathrm{B}(\boldsymbol{x})$ is the conditional mean estimator $\mathrm{E}_{\mathsf{y}|\boldsymbol{x}}\{\mathsf{y}\}$. This decomposition can be found in Appendix A.1. Whereas the first term on the right hand side, the expectation over the so-called *irreducible local error* $\mathrm{ile}(\boldsymbol{x})$, cannot be influenced and is due to the irreducible noise in the data, the bias and the variance can be changed by the choice of f. The bias measures the difference between the Bayes model $f_\mathrm{B}(\boldsymbol{x})$ and the average model $\mathrm{E}_{\mathsf{D}}\{f(\boldsymbol{x},\mathsf{D})\}$. The variance measures the dependency of $f(\boldsymbol{x},\mathsf{D})$ on the particular realization of D. From this additive decomposition in bias and variance it is clear that one should design the function $f(\boldsymbol{x},\mathsf{D})$ in such a way that both bias and variance are low.

[6] Since $f(\boldsymbol{x})$ depends on the random training set D, it will be denoted as $f(\boldsymbol{x},\mathsf{D})$ in the following.

[7] To cancel out the dependency of $f(\boldsymbol{x},\mathsf{D})$ on the training set, (2.6) and (2.7) have to be extended by taking also the expectation over the D-space.

The bias-variance decomposition for classification is not equivalent to the bias-variance decomposition for regression since the loss function and thus the average error rate are defined in different ways. Whereas in regression one uses $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x})) = (\mathsf{y} - f(\boldsymbol{x}))^2$, in classification the commonly used loss function is the 0/1-loss $\mathcal{L}(\mathsf{y}, f(\boldsymbol{x})) = 1 - \delta(\mathsf{y}, f(\boldsymbol{x}))$. With the 0/1-loss the prediction risk for classification is

$$\begin{aligned} \mathrm{E}_{\boldsymbol{x},\mathsf{y},\mathsf{D}}\left\{\mathcal{L}(\mathsf{y}, f(\boldsymbol{x}, \mathsf{D}))\right\} &= \mathrm{E}_{\boldsymbol{x},\mathsf{y}}\left\{\mathrm{E}_{\mathsf{D}|\boldsymbol{x},\mathsf{y}}\left\{1 - \delta(\mathsf{y}, f(\boldsymbol{x}, \mathsf{D}))\right\}\right\} \\ &= \mathrm{E}_{\boldsymbol{x}}\left\{\mathrm{E}_{\mathsf{y}|\boldsymbol{x}}\left\{\mathrm{E}_{\mathsf{D}|\boldsymbol{x},\mathsf{y}}\left\{1 - \delta(\mathsf{y}, f(\boldsymbol{x}, \mathsf{D}))\right\}\right\}\right\} \\ &= \mathrm{E}_{\boldsymbol{x}}\left\{1 - \sum_{k=1}^{K} \mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{\delta(c_k, f(\boldsymbol{x}, \mathsf{D}))\right\} \mathrm{p}(\mathsf{y} = c_k|\boldsymbol{x})\right\}. \end{aligned} \tag{2.17}$$

There are a variety of bias-variance decompositions for classification in the literature which try to decompose the prediction risk when using the zero-one loss. An overview on these decompositions can be found in [Geu02] and three of them will be described in the following.

An intuitive way to decompose the prediction risk in a bias and a variance term will be discussed first. The equivalent to the irreducible local error from (2.16) can be set to

$$\mathrm{ile}(\boldsymbol{x}) = 1 - \mathrm{p}(\mathsf{y} = f_{\mathrm{B}}(\boldsymbol{x})|\boldsymbol{x}), \tag{2.18}$$

with the Bayes classifier being [see (2.8) and (2.17)]

$$f_{\mathrm{B}}(\boldsymbol{x}) = \underset{c_k}{\mathrm{argmax}}\left\{\mathrm{p}(\mathsf{y} = c_k|\boldsymbol{x})\right\}. \tag{2.19}$$

The equivalent to the bias error from (2.16) can be defined as

$$\mathrm{bias}_{\hat{\mathsf{y}}|\boldsymbol{x}}\{\hat{\mathsf{y}}\} = 1 - \delta(f_{\mathrm{B}}(\boldsymbol{x}), f_{\mathrm{maj}}(\boldsymbol{x})), \tag{2.20}$$

where

$$f_{\mathrm{maj}}(\boldsymbol{x}) = \underset{c_k}{\mathrm{argmax}}\left\{\mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{\delta(c_k, f(\boldsymbol{x}, \mathsf{D}))\right\}\right\}, \tag{2.21}$$

with other words the classifier choosing that class for which the majority among the distribution of classifiers vote which were built based on the distribution of the training set D. The expectation $\mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{\delta(c_k, f(\boldsymbol{x}, \mathsf{D}))\right\}$ is equal to the probability that a random realization of D leads to $f(\boldsymbol{x}, \mathsf{D}) = c_k$, i. e.,

$$\mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{\delta(c_k, f(\boldsymbol{x}, \mathsf{D}))\right\} = \int\limits_{\mathcal{D}:\, f(\boldsymbol{x},\mathcal{D})=c_k} \mathrm{p}(\mathsf{D} = \mathcal{D}|\boldsymbol{x})\mathrm{d}\mathcal{D} \;=\; \mathrm{p}_{\mathsf{D}}(f(\boldsymbol{x}, \mathsf{D}) = c_k). \tag{2.22}$$

In order to find an expression for $\mathrm{var}_{\hat{\mathsf{y}}|\boldsymbol{x}}\{\hat{\mathsf{y}}\}$ which measures the variation of $f(\boldsymbol{x}, \mathsf{D})$ with respect to the training set D the following intuitive definition can be used

$$\mathrm{var}_{\hat{\mathsf{y}}|\boldsymbol{x}}\{\hat{\mathsf{y}}\} = 1 - \mathrm{E}_{\mathsf{D}|\boldsymbol{x}}\left\{\delta\left(f_{\mathrm{maj}}(\boldsymbol{x}), f(\boldsymbol{x}, \mathsf{D})\right)\right\} = 1 - \mathrm{p}_{\mathsf{D}}(f(\boldsymbol{x}, \mathsf{D}) = f_{\mathrm{maj}}(\boldsymbol{x})). \tag{2.23}$$

Although the variance from (2.23) is a measure of the variability of the prediction, the decomposition of the local average error $\mathrm{E}_{\mathsf{y}|\boldsymbol{x}}\left\{\mathrm{E}_{\mathsf{D}}\left\{1 - \delta(\mathsf{y}, f(\boldsymbol{x}, \mathsf{D}))\right\}\right\}$, into the terms from