Geometric Clustering and its Applications in Binary Classification Problems

Michael Josef Öllinger

Vollständiger Abdruck der von der Fakultät für Informatik der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Gutachter:

1. Univ.-Prof. Dr. Andreas Brieden
2. Univ.-Prof. Dr. Stefan Pickl

Die Dissertation wurde am 10. Juli 2014 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 12. November 2014 angenommen. Die mündliche Prüfung fand am 24. November 2014 statt.

Universität der Bundeswehr München

Fakultät für Informatik

Geometric Clustering and its Applications in Binary Classification Problems

Michael Josef Öllinger

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

Für meine Mutter

Abstract

In this thesis, we show a classification procedure using methodologies of combinatorial optimization to partition the euclidean space into convex sets of prescribed number and size. After introducing the theoretical background, we present a clustering based classifier and compare it with established algorithms. We show that an iterative sequence based on a geometric clustering in each step leads to a segmentation of the data space especially suitable for our prediction task. Based on this procedure, we define corresponding binary classifiers and introduce a new probabilistic test procedure to evaluate the reliability of a clustering based prediction. Furthermore, we show the excellent performance of the new classification technique and demonstrate the clustering based test of hypotheses on real world data.

Zusammenfassung

Das in dieser Arbeit vorgestellte Klassifikationsverfahren basiert auf Methoden der kombinatorischen Optimierung und partitioniert einen geometrischen Raum in konvexe Mengen vorgeschriebener Anzahl und Größe. Im Anschluss an die theoretischen Grundlagen wird ein darauf aufbauender Vorhersagealgorithmus vorgestellt und mit etablierten Methoden verglichen. Die darin enthaltene iterative Sequenz besteht in jedem Schritt aus einem geometrischen Clustering und führt schließlich zu einer Aufteilung des Raumes. Basierend auf dieser Zerlegung in konvexe Zellen werden binäre Klassifikatoren definiert sowie ein neuer wahrscheinlichkeitstheoretischer Ansatz zur Beurteilung der Prognosegüte vorgestellt. Die hervorragende Prognosegenauigkeit des neuen Verfahrens wird im letzten Schritt anhand von bekannten Praxisdatensätzen analysiert. Dabei wird neben dem Vergleich mit anderen Algorithmen auch die vorgestellte stochastische Beurteilungsmethode einer clusterbasierten Vorhersage demonstriert.

Acknowledgement

During the years of working on my PhD thesis, many people provided me with their support. Some of them I want to thank particularly at the beginning of my thesis.

First of all, I want to thank my doctoral advisor Prof. Dr. Andreas Brieden. Without him this work would not have been possible. I'm especially grateful for the mathematical discussions and his ideas for new mathematical directions concerning my work. Furthermore, I want to thank him for creating a friendly and pleasant working atmosphere.

My thanks also go to Prof. Dr. Peter Gritzmann for his mathematical input. Additionally, I thank him for the possibility to do some of the research at the Department of Applied Geometry and Discrete Mathematics at the Technische Universität München.

Last but not least, I want to thank my friends and family.

My colleagues Marie, Bernhard and Falk became my friends during our time together at the university.

My greatest thank goes to my wife Martina. She supported me during the last years in every situation.

Contents

Part I.

Introduction

The International Data Corporation (IDC) study [42] predicts a 50−fold growth of the digital data from 2010 to the end of 2020 and a big gap between the gathered and the analyzed data. IDC presumes that at the end of the year 2020 only a tiny portion of the gigantic amount of data is actually analyzed but about one third of the digital information would be useful to be analyzed. Therefore, it becomes more and more important to have efficient methods at hand to handle big data (see Figure 0.1).

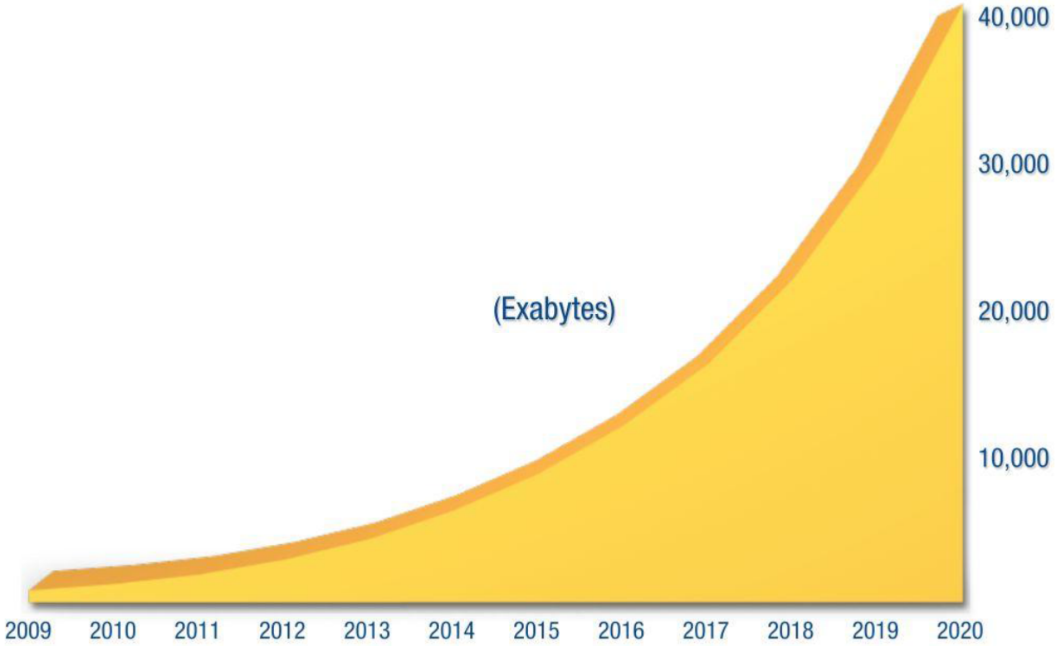

Figure 0.1.: Prediction of the growth of gathered data in the future (see [42]).

The technical advance in computer technology is not very useful if the corresponding software technology does not develop in the same way. Besides powerful hardware it is necessary to have suitable software and algorithms to handle big data.

The interdisciplinary field of data mining combines mathematics, statistics and computer sciences to retrieve information out of mostly large data sets. Data mining is widely used in a large variety of fields like economics (market research, credit scoring,...), science (physics, medicine,...) but also in national security or military affairs.

While data mining focuses on discovering new patterns, the similar field of machine learning concentrates on applying trained knowledge on new data. Both fields are very similar and often hard to distinguish. There are different definitions in the literature and it is often threatened equivalently with Knowledge Discovery in Databases (KDD). In [37], Fayyad et al. describe data mining as the following:

> Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns (or models) over the data.

Supervised and unsupervised learning are the two main categories in the field of machine learning. Unsupervised techniques discover structure in data without prior information.

In contrast, supervised learning trains a mapping based on a given set of input/output pairs (see [71]). If the output values are continuous, the problem is called regression. The problem is called classification in the case of categorical outputs.

Classification as a supervised learning technique assigns new data to a class based on a training set with known classes or assignments. For example, the training set could consist of one-hundred different people and the category is set to male or female. A classification algorithm, called classifier, assigns new data, in this case a new group of persons, without knowing their gender into the category using explanatory variables like weight, height or age.

There exist numerous classification techniques which differ from each other in many ways. Possible approaches are, for example, the estimation of the functional relationship, the estimation of conditional probabilities or the evaluation of the neighborhood of the data. Additionally, the combination of different techniques is an intuitive approach to achieve a good classification and prediction.

In this thesis we focus on classification problems with two possible outputs, i.e., binary classification. The new classification technique introduced in this work is based on another important field in data mining, which is the discovery of groups with similar attributes. These groups are called clusters. Even though a cluster is not precisely defined (see [34]), in the following we interpret a cluster as a group of points or objects. Assigning data to clusters is called cluster analysis or clustering. It is a core task in the field of data mining (see [46]).

The cluster analysis was first used by anthropologists in 1932 (see [31]) and soon became an important technique of statistical data analysis in many scientific fields (see [85] and [3]). Clustering can be done by numerous different ways. The techniques differ in the kind of data they can handle, for example, qualitative or quantitative input values. For an overview of different clustering techniques see [91]. In this work we focus on clusterings with similarity defined by geometrical proximity expressed in (Euclidean) distance measures.

This thesis is based on the work of Brieden and Gritzmann on geometric clusterings, their representation via polytopes and their application on real life problems like the consolidation of farmland (see [17], [19], [21], [22] [23] and [24] for details). The algorithmic approach is adopted as a classifier for binary classification problems in this thesis.

The joint work with Brieden and Gritzmann applies geometric clustering to binary classification problems. The introduced classifier combines supervised and unsupervised learning techniques. The introduced clustering based classifier is a supervised learning approach because of the training step followed by the classification in the next step. On the other hand, it is unsupervised as it discovers structure in the training data by means of cluster analysis.

The main part of this new classifier consists of an iterative sequence that solves a linear optimization program in every step. The result of this algorithm is a local optimum with respect to given parameter, a clustering particularly suitable for the classification task. The resulting clusters are pairwise linearly separable. Additionally, the clusters form a convex cell decomposition which is a key component for the prediction task.

In the first part of this thesis we present the theoretical background of binary classification problems. Therefore, we introduce the problem and describe the naive Bayes, the logistic regression and the k-nearest neighbor as basic classifiers. Additionally, we present evaluation techniques which will be useful later in this work to measure the performance of the clustering based classifier.

The next part outlines the clustering approach and its application to binary classification tasks. At first, we present theoretical mathematical aspects of the clustering approach like the identification of a feasible clustering with polytopes and the mathematical properties of the introduced clustering. After these theoretical aspects related to combinatorial optimization, the clustering technique is adopted as a classifier. Therefore, the classification or data mining content of the first part is integrated in the context of geometric clustering and vice versa. Additionally, a new statistical evaluation approach is introduced in Section 4.10. It relies on the clustering setting as it defines a pair of hypotheses for each individual cluster. Evaluating these hypotheses by computing a test statistic based on a classified data set leads to a measure of reliability expressed by a pair of statistical p-values.

In the following part, we present and discuss empirical results. They show the excellent performance of the new clustering based classifier. The results base on real world data sets like the German Credit and the Census Income data set. These broadly used data sets are provided by the UCI Machine Learning Repository, a well known database for machine learning. Besides the performance evaluation of the classifier, this part also demonstrates the new statistical evaluation approach on real world data.

In the conclusion as the last part of this thesis we summarize and discuss the main results and identify possible future work.

Part II.

Binary Classification

In general, the term classification covers any context of prediction or forecast based on currently available information (see [66]). In the context of machine learning or statistics, classification is regarded as an algorithmic procedure applied on data with unknown classes or labels but known features. The classification procedure is trained on data where both the labels and the features are known (see [66]).

Classification with more than two classes or labels is called multi-class classification. In this thesis we focus on the task of classifying binary labels.

At first, we give a formal problem description and the definition of a classifier and the corresponding components like the scoring function, the classification error and estimation principles. From a probabilistic point of view the introduction gives the reader the basic mathematical definitions for classification and prediction used in this work.

After the formal introduction of classification and binary classification, important basic concepts and their implementation are illustrated. Therefore, the naive Bayes, the logistic regression and the k-nearest neighbor procedure as examples for different types of binary classification are presented. They are examples for generative, discriminative and nearest neighbor techniques. Besides their use for comparison, the basic concepts of these approaches will be found in the new clustering approach introduced in the next part of this work. The end of this part refers to model evaluation techniques. The concepts of parameter adjustments and performance measures for classification techniques will be used later in this work to evaluate the new classification approach. For a broad introduction to machine learning see, for example, [13] and [47].

1. Problem Description

The task of classification in the context of machine learning or mathematical statistics is to predict the categorical label of a data point. For a broad view on the learning respectively classification problem see also [4], [65] and [71]. If the prediction technique uses the information of a training set with known input values and labels, it is called supervised learning. In statistics and in machine learning the levels of measurement are important criteria of distinction. If the output variables are quantitative, the naming convention for prediction is regression. Predicting qualitative outputs is called classification in the literature (see [47] for an overview). The methodology depends on the type of variable. In this work we focus on the prediction of binary labels leading to the binary classification problem. The following formal definition originates from the concept of statistical learning theory (see for example [70]). It interprets the input and output values as a realization of independent and identically distributed (i.i.d.) random variables.

Definition 1.0.1 (Binary Classification Problem)
Let $\{x_i\}_{i=1}^n = \{x_{i1}, ..., x_{id}\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$ with $x_i, y_i \in \mathbb{R}^d$ be n i.i.d. realizations of the (discrete) stochastic variables $X = (X_1, ..., X_d)$ and Y with an unknown joint probability distribution $\mathcal{D}_Z := \mathcal{D}_X \times \mathcal{D}_Y$ consisting of two probability distributions $\mathcal{D}_X$ and $\mathcal{D}_X$. The random variable Z with the underlying probability distribution $\mathcal{D}_Z$ is defined as $Z := X \times Y = \{(x, y | x \in X, y \in Y)\}$.

Then $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^d$ is called the (training) data set with input values X and labels Y.

A function $h \in \mathcal{H} := \{h : X \to Y\}$ is called classifier. h is called binary classifier if $|\{y_i : y_i \in Y\}| = 2$.

Finding a classifier $h : X \to Y$ by minimizing the so-called (generalization) error of h, $err(h, \mathcal{D}) : \mathcal{H} \to \mathbb{R}$ on the probability distribution $\mathcal{D}_Z$,

$$err_P(h, \mathcal{D}) = P(h(x) \neq y | (x, y) \sim \mathcal{D}_Z)$$

with

$$h(x) = \arg \min_{\{h \in \mathcal{H}\}} P(h(x) \neq y | (x, y) \sim \mathcal{D}_Z)$$

is called a classification problem and binary classification problem if h is binary.

Definition 1.0.1 states that the task of classification is to find a 'good' mapping from the input space X to the output space Y. This is equivalent to learn or train a classifier under the assumption that $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^d$ is a sample from a fixed but unknown joint distribution. The definition of the error in Definition 1.0.1 originates from the statistical learning theory and will also be useful when algorithms are compared. Input values X of Definition 1.0.1 are also called features, attributes or co-variates in the literature (see [71]) and therefore these terms will be used as synonyms in the following. Additionally, without loss of generality, the binary labels belonging to Y will be set to $\{0, 1\}$. In practice, as the probability distribution is estimated by the given training data, a more useful error definition is the so-called sample error or observed error instead of the theoretical concept of the generalization error.

Definition 1.0.2 (Sample Error)

Let $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0, 1\}$ be a data set and $h : X \to Y$ a binary *classifier. The sample or observed (classification) error of the set is defined as*

$$err_S(h(x); S) = \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \ .$$

As the error in Definition 1.0.1 is more of theoretical interests, we focus on the sample error err_S in the following. It provides an unbiased estimation of the real error err_P as it is the mean estimation and will be relevant later in this thesis. Often, the performance of the classification result is measured differently, depending on the classification task. In chapter 3 there is a more detailed overview of performance measures for binary classification.

Error estimation depends on the type of the data set S, relating to binary, discrete or continuous valued labels for Y. We focus on discrete valued input samples X and corresponding binary output values Y. For the input values X, this is only a small limitation as real valued labels need to be previously clustered in most cases and are therefore reduced to a discrete number of values or classes.

The basic idea behind supervised learning is to train an algorithm on a given training set and use this information on the data set that needs to be predicted, often called testing set. Training or learning are used synonymously from now on. The two stages of training and testing are equivalent to the inference and decision stage in decision theory (see [13]).

A classifier mostly classifies by assigning a real-valued score to a new element. This leads to the scoring function used for binary classification. This function needs to be computed by the training data and induces a classification rule. Training a classifier is therefore done by computing the scoring function.

Definition 1.0.3 (Scoring Function)

Let $h : X \to Y$ be a binary classifier, then a function $f : \mathbb{R}^d \to \mathbb{R}$ with

$$h(x) = \begin{cases} 1, & f(x) \geq \omega \\ 0, & otherwise \end{cases}$$

is called a scoring function with threshold value $\omega \in \mathbb{R}$.

The value ω defines the threshold for the class assignment. If not stated otherwise, the default threshold value is set to zero in the following. Commonly used scoring functions are, for example, linear scoring functions. Linear means that their general functional form is $f(x) = a^T x + b$ with $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$ (see Figure 1.1). In this case the scoring function is equivalent to a separating hyperplane. The separating hyperplane leads to the so called decision boundary $DB := \{x : a^T x + b = 0\}$. $DB^{\geq} := \{x : a^T x + b \geq 0\}$ is the (positive) half-space with every point assigned 1. Consequently in $DB^{<} := \{x : a^T x + b < 0\}$ every point is assigned 0. Of course, the data is mostly not linearly separable, leading trivially to an error in such cases. [1]

The separation of the data by a hyperplane into cells can be interpreted as a special case of our new approach in Part III. The two half-spaces $DB^{\geq}$ and $DB^{<}$ can be interpreted as two cells of a convex clustering. While linear binary classifiers have the same number of cells and classes, the later introduced clustering technique releases this connection. It partitions the multidimensional data space into a given number of convex, pairwise linearly separable cells. The cluster number in our new classification approach can be chosen freely. It is not restricted to two convex cells like the binary classification approaches introduced in Section 2.1 and 2.2 of Chapter 2. Definition 1.0.1 gives the underlying concept of a joint probability distribution for the joint random

[1] Finding an optimal separating hyperplane leads to support vector machines (for details see [71]).

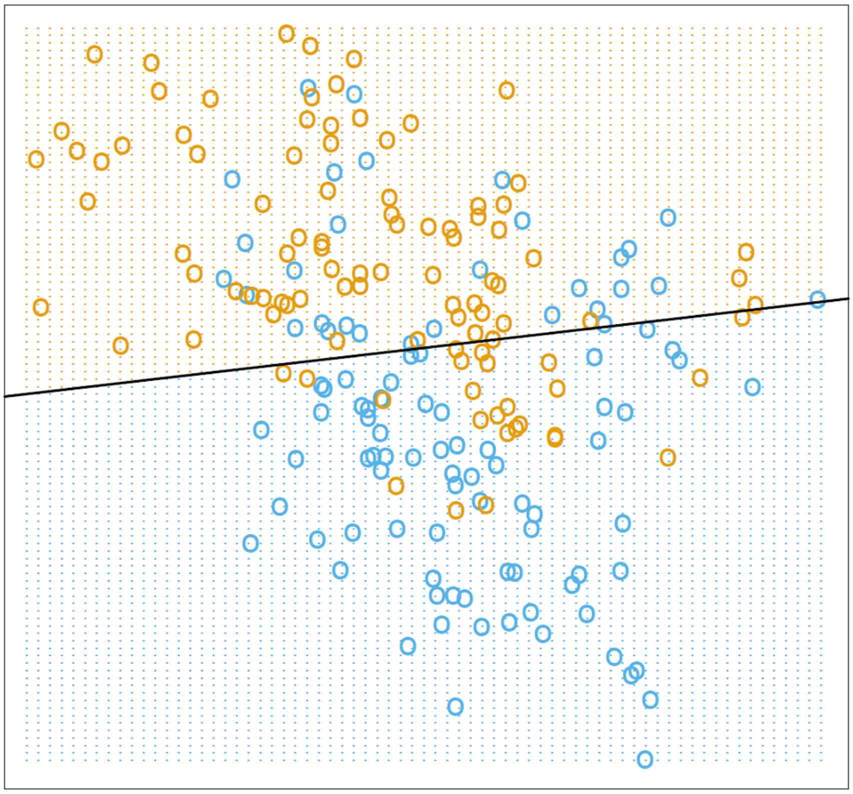

Figure 1.1.: The figure shows a two-dimensional classification example with a linear decision boundary. As a convex clustering, this corresponds to a clustering with two clusters. The orange region represents the part of the input space classified as $orange = 1$, while blue represents the second class $blue = 0$. The color of the points represents the true value (see [47]).

variable $Z := X \times Y$, $\mathcal{D}_Z$. From a probabilistic point of view, a classification procedure estimates the conditional probability $P(Y|X)$ based on the given training data. From a practical point of view, the underlying probability distributions are unknown and their parameters have to be estimated. Therefore, in the next step, underlying estimation concepts are introduced. They will be used later in this work in explicit classification approaches. For an overview of the following content see for example [68] or [71].
Because of the categorical output, the estimation by an ordinary least squares approach is not suitable. Therefore, the concept of the maximum likelihood estimation and the maximum a posteriori estimation will be introduced next. Both principles are used later in Sections 2.1 and 2.2 for the naive Bayes approach and the logistic regression.
Furthermore, these concepts will be applied to our clustering based classifier in Part III of this work. They lead to cluster values and scoring functions derived from the training data assigned to clusters.

A common approach to construct prediction models or estimate their parameters is the statistical concept of maximum likelihood (ML) parameter estimation. The adjustable parameters are chosen to maximize the probability to generate the underlying data set. As mentioned before, each pair of points (x_i, y_i), $i \in \{1, ..., n\}$ of the set $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0, 1\}$ is a representation of n independent and identically distributed (i.i.d.) random variables $Z_i = X_i \times Y_i$, $i \in \{1, ..., n\}$ with unknown joint probabilities $P(x_i, y_i|\theta)$ and unknown parameter θ. The assumption of independent and identically distributed variables leads to the joint probability

$$P(S|\theta) = \prod_{i=1}^{n} P(X = x_i, Y = y_i|\theta)$$

for the set S. The maximum likelihood estimation maximizes the likelihood for the data set S by maximizing the probability with respect to θ,

$$\theta_{ML} = \arg\max_{\theta} P(S|\theta) = \arg\max_{\theta} \prod_{i=1}^{n} P(X = x_i, Y = y_i|\theta) \ ,$$

under the assumption of a joint probability distribution $\mathcal{D}_Z = \mathcal{D}_X \times \mathcal{D}_Y$. Logarithmization leads to the equivalent maximization because of the monotonicity of the logarithm. This leads to the so-called log-likelihood

$$\theta_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \ln(P(X = x_i, Y = y_i|\theta)) \ .$$

The log-likelihood is easier to calculate and therefore commonly used.

A more general concept of prediction is based on the Bayes' rule

$$P(\theta|S) = \frac{P(S|\theta)P(\theta)}{P(S)} \ , \tag{1.1}$$

which is often interpreted as

$$posterior = \frac{likelihood \cdot prior}{evidence} \ .$$

It connects the concept of the maximum likelihood estimation with a more general concept of estimation. It originates from Bayesian concept learning and leads by the Bayes' rule (1.1) to another estimation concept explained in the next step (see [71]).

Besides maximizing the *likelihood* of a data set another possibility is to maximize the *posterior* under a pre-given *prior*.

This concept is called maximum a posteriori (MAP) estimation and is related to the maximum likelihood estimation mentioned before. It originates from

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|S).$$

and 'switches' the condition compared to the maximum likelihood estimation. The maximum a posteriori estimation uses the most probable class label and is equivalent to the mode of the posterior distribution. The posterior distribution with probabilities $P(\theta|S)$ and the assumption of a probabilistic parameter θ leads to the formulation

$$\theta_{MAP} = \arg\max_{\theta} P(S|\theta)P(\theta) = \arg\max_{\theta} \prod_{i=1}^{n} P(X = x_i, Y = y_i|\theta)P(\theta)$$

by applying the Bayes' rule (1.1). Like the maximum likelihood approach it has an equivalent logarithmized version:

$$\theta_{MAP} = \arg\max_{\theta} \sum_{i=1}^{n} ln(P(X = x_i, Y = y_i|\theta)) + ln(P(\theta)) \ .$$

The maximum likelihood estimation can be interpreted as a simplification of the maximum a posteriori estimation with θ uniformly distributed.
Contrary to the maximum likelihood estimation, the maximum a posteriori estimation allows to incorporate prior knowledge. The reason for this generalization of the maximum likelihood approach is mostly practical and will be useful in Chapter 2, as well as the maximum likelihood estimation itself.

2. Algorithmic Approaches

In this chapter, we present several algorithmic concepts for the binary classification task.

There are many different approaches to classify data. They can be split into two main groups differing in the estimation of the conditional probabilities $P(Y|X)$. The first popular way of determining the conditional probabilities is computing them indirectly via the Bayes' rule

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{\sum\limits_{i=1}^{l} P(X = x|Y = y_i)P(Y = y_i)}$$

for l classes, or

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x|Y = 0)P(Y = 0) + P(X = x|Y = 1)P(Y = 1)}$$

in the binary case. This approach leads to the so-called generative approaches like the naive Bayes classification method introduced in Section 2.1. The term generative in this case refers to the possibility of generating sample data. It originates from modeling both the distribution of the input data as well as the distribution of the output data (see [13]).

The second approach directly determines the conditional class probabilities $P(Y|X)$. The logistic regression introduced in Section 2.2 is an example for such a discriminative approach.

The following two Sections 2.1 and 2.2 introduce these basic examples for discriminative and generative methods. Both techniques divide the feature space into two groups by a linear decision boundary.
In the first step, the theoretical background will be introduced. After this basic introduction, the parameter estimation will be explained. Additionally, there is a broad discussion whether discriminative or generative models are better for classification (see e.g. [72] and [92]). Even though naive Bayes and logistic regression are rather old techniques for classification, they are still very popular and widely used (see [26]). Additionally, they are basic, individual classifiers and are often combined to more complex techniques (see for an overview [82]). The parameter estimation for both classifiers will be done by the already in Chapter 1 introduced maximum likelihood and maximum a posteriori estimation.
After these two examples for discriminative and generative classifiers, the k-nearest neighbor approach is introduced in Section 2.3. It is a different classification technique as it is a nonlinear classifier and belongs to the nearest neighbor approaches. Additionally, it has some similarities to our new approach introduced in Part III of this thesis.

2.1. Naive Bayes

As a first example for an often used classification method we present the naive Bayes approach. Classifiers like the generative naive Bayes are linked to Bayesian networks and the Bayes' rule for conditional probabilities. They are basic examples for generative classifiers. The underlying principle is the indirect estimation of the conditional probabilities $P(Y|X)$ by estimating the conditional probabilities $P(X|Y)$ (for further information of the naive Bayes approach see [68]). The fundamental equation is the Bayes' rule

$$P(Y=y|X_1=x_1,...,X_d=x_d)=\frac{P(X_1=x_1,...,X_d=x_d|Y=y)P(Y=y)}{\sum\limits_{i=1}^{l}P(X_1=x_1,...,X_d=x_d|Y=y_i)P(Y=y_i)}$$

already mentioned in the introduction of this section (see [77] for details). The above equation contains the conditional probabilities

$$P(X_1=x_1,...,X_d=x_d|Y=y)\ ,$$

which are impractical to estimate. The main assumption of the naive Bayes approach is the conditional independence of the stochastic input variables. Conditional independence is similar to the usual concept of stochastic independence and is defined in the following Definition 2.1.1.

Definition 2.1.1 (Conditional Independence)
Let X, Y and Z be random variables. Then X and Y are conditionally independent given Z if and only if the probability distribution of X is independent of the value of Y given Z. This is equivalent to

$$P(X=x_i|Y=y_j,Z=z_k)=P(X=x_i|Z=z_k),\forall i,j,k\ .$$

The above definition implies that conditional independence holds for two random variables X and Y given a third random variable Z if and only if they are independent in their conditional probability distribution given Z (see [29]). Definition 2.1.1 of the conditional independence leads to

$$P(X_1=x_1,...,X_d=x_d|Y=y)=\prod_{i=1}^{d}P(X_i=x_i|Y=y) \qquad (2.1)$$

for variables $X_1, ..., X_d$. This represents the factorization of a joint probability $P(X_1 = x_1, ..., X_d = x_d|Y = y)$ into its marginals $P(X_i = x_i|Y = y)$, $i = 1, ..., d$, and directly leads to the main assumption of the naive Bayes classifier.

The 'naive' independence between the input values $X_1, ..., X_d$ could be seen as oversimplifying but [93] gives some theoretical reasons why it is nevertheless effective in most cases. The conditional independence leads to a new formulation of the conditional probability.

$$P(Y = y|X_1 = x_1, ..., X_d = x_d) = \frac{P(Y = y) \prod\limits_{i=1}^{d} P(X_i = x_i|Y = y)}{\sum\limits_{j=1}^{l} P(Y = Y_j) \prod\limits_{i=1}^{d} P(X_i = x_i|Y = y_j)} \tag{2.2}$$

is the fundamental equation for the naive Bayes classifier. The result is an estimation of a complex multidimensional probability distribution via independent, one-dimensional distributions also called marginals. This is useful for high dimensional data as it helps to avoid problems arising from increasing dimensions or number of variables. With a growing number of input variables, the amount of data, necessary to estimate the combined probabilities, grows exponentially. The avoidance of this 'curse of dimensionality' (also known as combinatorial explosion) leads to the efficiency and effectiveness of the naive Bayes method despite the extreme simplification (see [41] and [93]).

Equation (2.2) gives the probability for a label under the condition of a new instance $x = (x_1, ..., x_d)$. The corresponding distributions are estimated from the training data.

The classification rule is

$$h(x) = \arg \max_{y \in \{y_1, ..., y_l\}} \frac{P(Y = y) \prod_{i=1}^{d} P(X_i = x_i|Y = y)}{\sum_{j=1}^{l} P(Y = y_j) \prod_{i=1}^{d} P(X_i = x_i|Y = y_j)} .$$

As the denominator is irrelevant for the maximization, it can be simplified to

$$h(x) = \arg \max_{y \in \{y_1,...,y_l\}} P(Y = y) \prod_{i=1}^{d} P(X_i = x_i | Y = y).$$

Like the logistic regression in the next Section 2.2, naive Bayes represents a linear classifier. This is obvious after the logarithmization which leads to the equivalent maximization task in the following equation:

$$h(x) = \arg \max_{y \in \{y_1,...,y_l\}} \ln(P(Y = y)) + \sum_{i=1}^{d} \ln P(X_i = x_i | Y = y)\ .$$

Remark 2.1.2

As shown in [75], for a multinomial distributed random variable $X = (X_1, ..., X_d)$ the naive Bayes is a linear classifier with the scoring function

$$f_{NB}(x) = \frac{P(Y = 1)}{P(Y = 0)} \prod_{i=1}^{d} \frac{P(X_i = x_i | Y = 1)}{P(X_i = x_i | Y = 0)} - 1 \tag{2.3}$$

in the binary case. This leads to the (linear) decision boundary

$$DB_{NB} := \left\{ x \in \mathbb{R}^d : \ln\left(\frac{P(Y = 1)}{P(Y = 0)}\right) + \sum_{i=1}^{d} \ln\left(\frac{P(X_i = x_i | Y = 1)}{P(X_i = x_i | Y = 0)}\right) x_i = 0 \right\}.$$

and the corresponding classifier

$$h_{NB}(x) = \begin{cases} 1, & f_{NB}(x) > 0 \\ 0, & otherwise \end{cases}.$$

In the next step, we show the parameter estimation based on the training data. As mentioned in the previous Chapter 1, there exist different estimators. At first, we introduce the maximum likelihood estimators $\theta_{ML}(Y)$ for $P(Y)$ and

$\theta_{ML}(X_i|Y)$ for $P(X_i|Y)$, $i = 1, ..., d$, respectively.

Remark 2.1.3

Let $S = \{(x_j, y_j)\}_{j=1}^n \subset \mathbb{R}^d \times \{0,1\}$ with $x_j = (x_{j1}, ..., x_{jd})$ be a sample of d stochastic random variables X_i, $i = 1, ..., d$. Then the maximum likelihood estimates for the naive Bayes classifier h_{NB} in Remark 2.1.2 are

$$\theta_{ML}(Y = y) = \frac{\sum_{j=1}^n 1_{\{y_j\}}(y)}{n}$$

and

$$\theta_{ML}(X_i = x|Y = y) = \frac{\sum_{j=1}^n 1_{\{y_j\}}(y) \cdot 1_{\{x_{ji}\}}(x)}{\sum_{j=1}^n 1_{\{y_j\}}(y)}, \ i = 1, ..., d \ .$$

If realizations do not occur in the given data together with y, then the estimation for $P(X_i = x|Y = y)$ would result in a division by zero. A possible solution is the maximum a posteriori approach which was introduced after the maximum likelihood estimation in Chapter 1. As the input data is discrete, the usually assumed prior follows a Dirichlet distribution which is the multivariate expansion of the beta distribution (see [38] and [52] for details).

Remark 2.1.4

Let $S = \{(x_j, y_j)\}_{j=1}^n \subset \mathbb{R}^d \times \{0,1\}$ with $x_j = (x_{j1}, ..., x_{jd})$ be a sample of d stochastic random variables X_i, $i = 1, ..., d$, and n_{X_i} the number of possible (discrete) values for X_i. Then the maximum a posteriori estimation for the (binary) naive Bayes classifier h_{NB} is

$$\theta_{MAP}(Y = y) = \frac{\sum_{j=1}^n 1_{\{y_j\}}(y) + \alpha}{n + 2\alpha}$$

and

$$\theta_{MAP}(X_i = x|Y = y) = \frac{\sum_{j=1}^n 1_{\{y_j\}}(y) \cdot 1_{\{x_{ij}\}}(x) + \alpha}{\sum_{j=1}^n 1_{\{y_j\}}(y) + n_{X_i}\alpha}, i = 1, ..., d$$

with $\alpha > 0$ as a smoothing parameter.

The naive Bayes approach is widely used because of its fast but effective prediction technique. The conditional independence assumption leads to a rather small number of required training samples. Modeling the conditional probability $P(Y|X)$ as a product of marginals results in a parameter reduction (see [68], [72] and [86]).
In the next section, we present another widely used approach for binary classification, the logistic regression. While the naive Bayes approach uses a detour via the Bayes' rule, the logistic regression approach models the conditional probabilities directly.

2.2. Logistic Regression

Linear regression models or ordinary least squares are not suitable for binary classification problems. The outputs of the ordinary least squares naturally are not restricted to $[0, 1]$. Additionally, with binary labels, the assumption of normal distributed errors is violated. A solution is the logistic regression model (see [73]). Although it has the term 'regression' in its name it is rather used for classification than for regression.
There are a lot of different equivalent specifications of the logistic regression. It can be motivated via numerous ways, for example, as a generalized model or as a perceptron. As we focus on the logistic regression as a method for binary classification problems we will not discuss the theoretical background more detailed (see [1], [10] and [50] for further information). The logistic regression model is a discriminative classification approach in contrast to the generative naive Bayes classifier. While naive Bayes used the Bayes' rule to model the conditional probability $P(Y|X)$ by $P(X|Y)$, the logistic regression models $P(Y|X)$ directly. The concept of the logistic regression as a method

for binary classification is to map the feature vector x to a real number, such that large positive numbers are set to the binary label $y = 1$ and negative numbers are set to $y = 0$. This can be done by calculating the weighted sum of the parameter vector $\theta = (\theta_1, ..., \theta_d) \in \mathbb{R}^d$ with a feature vector $x \in \mathbb{R}^d$ in the functional model $f : \mathbb{R}^d \to \mathbb{R}$ with

$$f(x) = \theta_0 + \theta^T x,\ \ \theta_0 \in \mathbb{R}\ .$$

Obviously, this is a linear function with values in the range $]-\infty, \infty[$. The logistic function $g : \mathbb{R} \to [0, 1]$ with

$$g(x) = \frac{1}{1 + \exp(-x)}$$

maps the interval $]-\infty, \infty[$ to the interval $[0, 1]$ (see Figure 2.1). It is applied

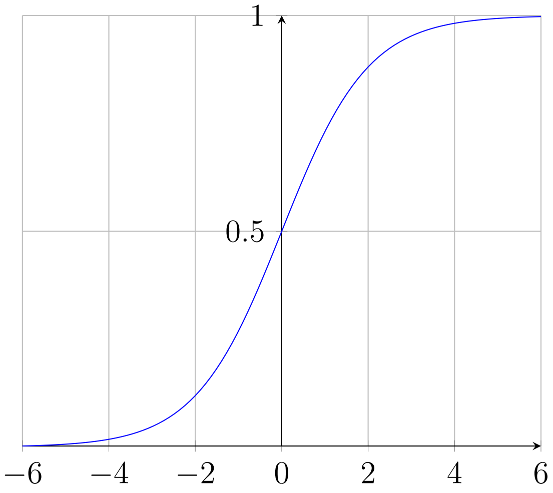

Figure 2.1.: The logistic function $g(x) = \frac{1}{1+\exp(-x)}$.

to model the required conditional class probabilities

$$P(Y=0|X_1=x_1,...,X_d=x_d)=\frac{1}{1+\exp(\theta_0+\sum_{i=1}^d\theta_i x_i)} \tag{2.4}$$

$$P(Y=1|X_1=x_1,...,X_d=x_d)=\frac{\exp(\theta_0+\sum_{i=1}^d\theta_i x_i)}{1+\exp(\theta_0+\sum_{i=1}^d\theta_i x_i)}\ . \tag{2.5}$$

Equations (2.4) and (2.5) lead with

$$\frac{P(Y=1|X_1=x_1,...,X_d=x_d)}{P(Y=0|X_1=x_1,...,X_d=x_d)}=\exp(\theta_0+\sum_{i=1}^d\theta_i x_i)$$

to the decisions

$$1,\text{ if }\frac{P(Y=1|X_1=x_1,...,X_d=x_d)}{P(Y=0|X_1=x_1,...,X_d=x_d)}\geq 1$$

and

$$0,\text{ if }\frac{P(Y=1|X_1=x_1,...,X_d=x_d)}{P(Y=0|X_1=x_1,...,X_d=x_d)}<1\ .$$

The corresponding classifier h_{LOG} with scoring function f_{LOG} is based on

$$\ln\left(\frac{P(Y=1|X_1=x_1,...,X_d=x_d)}{P(Y=0|X_1=x_1,...,X_d=x_d)}\right)=\theta_0+\sum_{i=1}^d\theta_i x_i$$

and is a linear classifier like the naive Bayes approach.

Remark 2.2.1

The logistic regression is a linear classifier with the scoring function

$$f_{LOG}(x)=\ln\left(\frac{P(Y=1|X_1=x_1,...,X_d=x_d)}{P(Y=0|X_1=x_1,...,X_d=x_d)}\right)=\theta_0+\sum_{i=1}^d\theta_i x_i,$$

the decision boundary $DB_{LOG}:=\{x\in\mathbb{R}^d:\theta_0+\sum_{i=1}^d\theta_i x_i=0\}$ *and the* *classifier*

$$h_{LOG}(x) = \begin{cases} 1, & f(x) \geq 0 \\ 0, & otherwise \end{cases} .$$

The logarithmized ratios $\ln\left(\frac{P(Y=1|X_1=x_1,\ldots,X_d=x_d)}{P(Y=0|X_1=x_1,\ldots,X_d=x_d)}\right)$ are called logits, log-odds or the logarithm of the odds and lead to an expression equivalent to the linear regression model (see [51] for details).

The estimation of the parameter vector θ, which is equivalent to the training step, can be done with a conditional log likelihood estimation

$$\theta_{ML} = \max_{\theta} \sum_{i=1}^{n} \ln P(Y = y_i | X = x_i, \theta).$$

In the binary case, this is equivalent to

$$\begin{aligned} \theta_{ML} &= \max_{\theta} \sum_{i=1}^{n} \left(Y_i \ln P(Y_i = 1|X = x_i, \theta) + (1 - Y_i) \ln P(Y_i = 0|X = x_i, \theta)\right) \\ &= \max_{\theta} \sum_{i=1}^{n} \left(Y_i \ln\left(\frac{P(Y_i = 1|X = x_i, \theta)}{P(Y_i = 0|X = x_i, \theta)}\right) + \ln P(Y_i = 0|X = x_i, \theta)\right) \\ &= \max_{\theta} \sum_{i=1}^{n} \left(Y_i(\theta_0 + \theta^T x_i) - \ln(1 + exp(\theta_0 + \theta^T x_i))\right) \ , \end{aligned}$$

which has no closed form. Therefore, it can be calculated with the gradient descent approach (see [68]).

The maximum a posteriori estimation (MAP) described in Chapter 1 is commonly used to avoid problems arising in real world cases. One possible problem is the infinitely high number of solutions in the case of linear separability of the data points. A solution to this problem is the addition of a penalty term that originates from the maximum a posteriori estimate. The MAP estimate incorporates a prior on the unknown and now probabilistic parameter θ assuming the zero-mean Gaussian distribution for $P(\theta)$. This leads to the search for the parameter vector θ_{MAP} with

$$\theta_{MAP} = \arg\max_{\theta} \sum_{i=1}^{n} \ln(P(Y_i = 1|X = x_i, \theta)) + \ln P(\theta)$$

or

$$\theta_{MAP} = \arg\max_{\theta} \sum_{i=1}^{n} \ln(P(Y_i = 1|X = x_i, \theta)) - \frac{\lambda}{2}\theta^T\theta \ .$$

The penalty term is related to the variance of θ if $P(\theta)$ is a zero-mean Gaussian distribution. Like in the maximum likelihood estimation, the underlying convex function has no closed form and is often solved with Newton-Raphson iterations, also called iterative re-weighted least squares (see [13]).

The logistic regression directly models the conditional probabilities and is therefore a broadly used classifier in the group of discriminative models. Besides the logistic regression, support vector machines and neural networks also belong to this group of classifiers. Discriminative models often yield to very good results by estimating the conditional probabilities $P(Y|X)$ directly. In contrast, generative models like the naive Bayes introduced in Section 2.1 estimate the conditional probabilities $P(Y|X)$ by applying the Bayes' rule. Despite the different categorization of the logistic regression and the naive Bayes, both classification techniques are linked. For example, the Gaussian naive Bayes classifier as modification for continuous input values is motivated by Equations (2.4) and (2.5). [1] Furthermore, the logistic regression can also be used for classification problems with more than two classes leading to the multinomial logistic regression (see [13] for more information).

Both linear classifiers, the naive Bayes approach in Section 2.1 and the logistic regression in Section 2.2 are widely used for binary and also multiclass classi-

[1] The Gaussian naive Bayes modification assumes continous valued input variables following a Gaussian distribution (see [68] for details).

fication problems.

In the following, we summarize the main results Ng and Jordan showed in [72]. They showed that discriminative approaches like the logistic regression are almost always to be preferred in terms of the so-called asymptotic error. It is smaller for generative techniques compared to discriminative methods like the naive Bayes.

With err_{S_n} we denote the sample error based on a training set of size n, drawn from the underlying distribution $\mathcal{D}_Z$ of $Z := X \times Y$. The so-called asymptotic error err_{S_∞} leads to the generalization error

$$err_P(h, \mathcal{D}) = P(h(x) \neq y | (x, y) \sim \mathcal{D}) = E(1_{\{h(x) \neq y | (x,y) \sim \mathcal{D}\}}) \ ,$$

introduced in Definition 1.0.1. The generalization error is the expected value of the misclassification rate when averaged over future data (see [71]). As $\mathcal{D}$ is unknown, it is estimated by samples drawn from a superset. Converging the number of samples to infinity ($n \to \infty$) leads to the asymptotic error err_{S_∞}.

In the following, $err_{S_\infty}(h_{LOG})$ is the asymptotic error of the logistic regression and $err_{S_\infty}(h_{NB})$ the asymptotic error of the (Gaussian) naive Bayes algorithm.

Ng and Jordan showed in [72] that the logistic regression has a smaller asymptotic error than the naive Bayes and both converge to their asymptotic errors at different rates. While the asymptotic error $err_{S_\infty}(h_{LG})$ of the logistic regression holds the inequality

$$err_{S_n}(h_{LOG}) \leq err_{S_\infty}(h_{LOG}) + \mathcal{O}\left(\sqrt{\frac{d}{n}}\right) ,$$

the corresponding inequality for the asymptotic error $\hat{err}_n(h_{NB})$ of (Gaussian) naive Bayes approach is

$$err_{S_n}(h_{NB}) \leq err_{S_\infty}(h_{NB}) + \mathcal{O}\left(\sqrt{\frac{\log(d)}{n}}\right).$$

Additionally, the (Gaussian) naive Bayes requires $\mathcal{O}(\log(d))$ to converge while the logistic regression takes $\mathcal{O}(d)$. Ng and Jordan also show that logistic regression outperforms the (Gaussian) naive Bayes approach most of the time. Only in cases when there are few samples in the training set, naive Bayes performs better than the logistic regression. In contrast to this results, Xue and Titterington show that for real world data it is not clear whether discriminative or generative approaches are more suitable (see [92]).

2.3. *k*-Nearest Neighbor

We introduced the naive Bayes and the logistic regression as linear classifiers. The k-nearest neighbor (KNN) approach is a density estimation technique. It is a nonparametric, nonlinear classification approach. Without assuming a functional form, it is similar to the later introduced new clustering approach as it classifies new instances by their proximity to training instances (see for example [13]). Besides the linear classifiers introduced in the sections before, the nearest neighbors approach is an intuitive but different classification technique. In the concept of the so-called k-nearest neighbor, an instance x is classified by the most common amongst its k-nearest neighbors. The predicted value for x is the the label in the k-neighborhood with the highest frequency. Figure 2.2 shows an example with a 15-nearest neighbor classifier. In the case of $k = 1$, a new instance is assigned with the label of its nearest neighbor (see Figure 2.3).

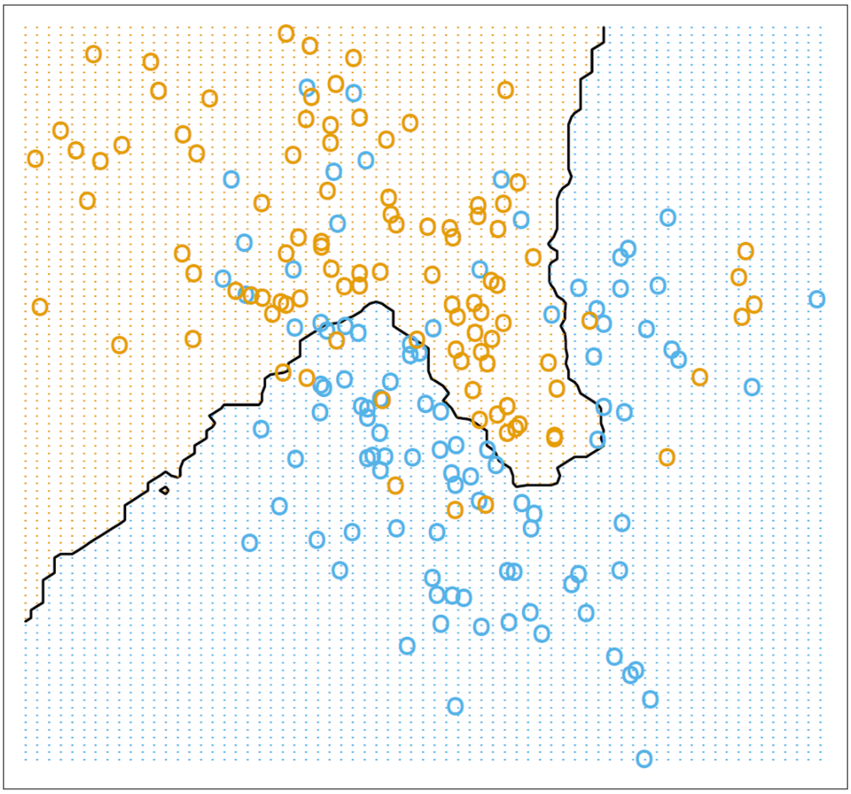

Figure 2.2.: The 15-nearest neighbor approach for the same data as in Figure 1.1 with the resulting piecewise linear decision boundary. The orange region represents the part of the input space classified as *orange* = 1, while the blue region represents the second class *blue* = 0. The color of the points represents the true value of their label (see [47]).

Definition 2.3.1

Let $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^d$ be a data set. Then the k-neighborhood $N_k(x)$ of a point $x \in \mathbb{R}^d$ in the data set S is given by

$$N_k(x) := \{X \subset S : |X| = k \wedge d(x, x_i) \leq d(x, x_j), \forall x_i \in X, \forall x_j \in S \backslash X\}$$

under the metric d.

The estimation for a new instance x and the corresponding scoring function is derived from

$$f(x) = \frac{\sum_{x_i \in N_k(x)} y_i}{\sum_{x_i \in N_k(x)} 1_{\{(x,y) \in S : x \in N_k(x)\}}} .$$

Remark 2.3.2

Let $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^d$ be a data set. The k-nearest neighbor is a classifier with the scoring function

$$f_{KNN}(x) = \frac{\sum_{x_i \in N_k(x)} y_i}{\sum_{x_i \in N_k(x)} 1_{\{(x,y) \in S : x \in N_k(x)\}}} - 0.5,$$

and the classifier

$$h_{KNN}(x) = \begin{cases} 1, & f_{KNN}(x) \geq 0 \\ 0, & otherwise \end{cases} .$$

$N_k(x)$ is the k-neighborhood of x.

A common proximity measure for the k-nearest neighbor approach depends on distance measures like the Euclidean distance. Therefore, even if it is not dependent on stringent assumptions on the data, it is dependent on the underlying distance measure. While by definition, the introduced linear classifiers naive Bayes and logistic regression have linear decision boundaries, the k-nearest neighbor approach results in a piecewise linear decision boundary. It

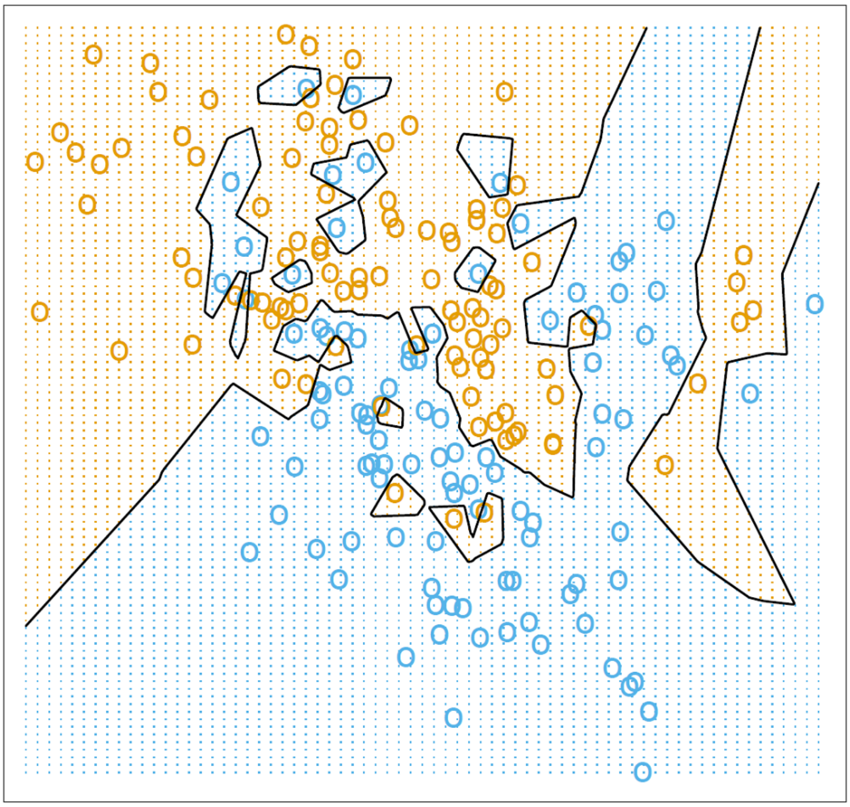

Figure 2.3.: The 1-nearest neighbor approach for the same data as in Figure 1.1 and 2.2 with the resulting piecewise linear decision boundary. The orange regions represent the parts of the input space classified as *orange* $= 1$, while the blue regions represent the second class *blue* $= 0$. The color of the points represents the actual value of their label (see [47]).

is composed of hyperplanes that form perpendicular bisectors of pairs of points from different classes ([13]). In case of $k = 1$ the 1-nearest neighbor approach produces a Voronoi tessellation (see [32]) like seen in Figure 2.3. This Voronoi tessellation is closely related to the segmentation of the data space into convex cells of the classification approach introduced in Part III of this thesis.

The basic concepts of the linear classifiers introduced in Section 2.1 and 2.2 as well as the k-nearest neighbor method will be found in our new clustering technique. The nearest neighbor techniques lead to a partition of the data space but does not allow a given number of distinct cells or areas. The similarity measure is a crucial point and the prediction depends on the chosen distance metric.

The k-nearest neighbor algorithm is very simple but effective and belongs, like the introduced naive Bayes and logistic regression, to the top ten of the data mining algorithms (see [90]). Nevertheless, it has some crucial disadvantages. At first, the computational complexity is $\mathcal{O}(d \cdot n)$ to find the exact nearest neighbor of a single test point. Therefore, the algorithm can become quite slow in high dimensional spaces. Secondly, the whole training data has to be stored in the memory and last but not least the distance measure has to be specified.

The clustering approach for classification introduced in Part III of this work allows a classification based on the values of the neighborhood similar to the k-nearest neighbor technique. One crucial difference of our approach is the possibility of a given number of cells instead of neighbors. Additionally, our technique has a complexity depending on this given number of cells and not on the number of points. Therefore, the k-nearest neighbor algorithm becomes slow with an increasing number of instances as its complexity is $\mathcal{O}(d{\cdot}n)$. There are some techniques to speed up KNN or the underlying nearest neighbor problem but the complexity is still dependent on the number of training points (see for example [27], [28] or [87]).

3. Model Evaluation

Chapter 1 introduced the theoretical basics and the definitions for the binary classification problems. After exemplary classification techniques in Chapter 2, we now present relevant evaluation techniques based on the error definition in Chapter1. This includes, for example, the often used statistical performance measures sensitivity and specificity. Also enhanced measures like the receiver operating characteristic and the related area under curve (AUC) are introduced. Additionally, statistical evaluation techniques like confidence estimation and the cross-validation techniques are highlighted to complete the basics for evaluating classification results and comparing classification algorithms. The introduced content will be applied to evaluate and compare our classification approach on real world data in Part IV.

3.1. Binary Accuracy Measures

The following topics are related to [36] and [64]. The goal of achieving a 'good' classification depends on the used evaluation criteria. What is regarded as a good classifier depends on the underlying performance measure. In medical studies, identifying someone falsely as sick is usually not as bad as not identifying a sick person at all. This leads to different measures of a good classification and different error definitions for optimization. The error definitions in Chapter 1 are now extended and embedded in the field of information retrieval.

Therefore, we will focus on the empirical error related to the evaluation of the sample sets instead of the theoretical probabilistic error.

Definition 3.1.1 (Classification Accuracy)
Let $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$ be a data set, $h(x)$ a binary classifier and $err_S(h(x); S)$ the corresponding sample error. The (sample) classification accuracy is defined as

$$acc_S(h(x); S) = 1 - err_S(h(x); S) \ .$$

As we focus on the sample error and not on theoretical errors, we will use the term error and sample error and the corresponding terms for accuracy equivalently in the following. For any binary classifier $h(x)$ there are two possible errors that are summed up in the classification error:

- misclassifying a sample with label 0 as 1 (so called false positive *fp*)
- misclassifying a sample with label 1 as 0 (so called false negative *fn*)

The resulting false positive and false negative rates are important, for example, in clinical studies. The above measures in this context represent healthy people incorrectly identified as sick (*fp*) or sick people incorrectly identified as healthy (*fn*).

Definition 3.1.2 (Binary Performance Measures)
Let $S = \{(x_i, y_i\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$ be a data set, $h(x)$ a binary classifier and $err_S(h(x); S)$ the corresponding sample error.
The true positives (tp) are defined as

$$tp(h(x); S) = \sum_{i=1}^{n} 1_{\{h(x_i)=1 \wedge y_i=1\}}$$

and the true negatives (tn) as

$$tn(h(x);S) = \sum_{i=1}^{n} 1_{\{h(x_i)=0 \wedge y_i=0\}} \ .$$

The corresponding false positives (fp) and false negatives (fn) are defined as

$$fp(h(x);S) = \sum_{i=1}^{n} 1_{\{h(x_i)=1 \wedge y_i=0\}}$$

and

$$fn(h(x);S) = \sum_{i=1}^{n} 1_{\{h(x_i)=0 \wedge y_i=1\}}$$

The sensitivity (sens) and the specificity (spec) are defined as

$$sens(h(x);S) = \frac{tp(h(x);S)}{tp(h(x);S) + fn(h(x);S)}$$

and

$$spec(h(x);S) = \frac{tn(h(x);S)}{fp(h(x);S) + tn(h(x);S)} \ .$$

The precision (prec) is defined as

$$prec(h(x);S) = \frac{tp(h(x);S)}{tp(h(x);S) + fp(h(x);S)}$$

and the recall (rec) is equivalent to the already defined sensitivity.

The negative prediction value (npv) of a classification is defined as

$$npv(h(x);S) = \frac{tn(h(x);S)}{tn(h(x);S) + fn(h(x);S)} \ .$$

The different terminologies mostly depend on their original scientific field. The specificity and the sensitivity are widely used in medical classification problems. In this context, the specificity represents the number of truly healthy people among all persons classified as healthy. The sensitivity corresponds to the 'hit rate' or the number of actually sick classified people compared to all persons classified as sick. In the context of pattern recognition and information retrieval, precision (also known as confidence) is interpreted as the probability that a positive label is actually true (see [74]). In the following, to simplify the further reading, we will write tp instead of $tp(h(x); S)$ and so on. The classification accuracy and the (sample) classification error can be expressed in the setting of Definition 3.1.2.

Remark 3.1.3

The sample error $err(h(x); S)$ can be expressed as

$$err_S = \frac{fp + fn}{tp + tn + fp + fn} = \frac{fp + fn}{n}$$

and the corresponding sample accuracy $acc_S(h(x); S)$ as

$$acc_S = \frac{tp + tn}{tp + tn + fp + fn} = \frac{tp + tn}{n}.$$

The following Table 3.1 gives an overview of the above introduced performance measures of Definition 3.1.2. The solitary consideration of the sensitivity and the specificity has little use from a theoretical point of view. The reason is always assigning 1 or 0 would lead to a value of 100% for *sens* and *spec*, respectively. Therefore, both values should be interpreted pairwise. The threshold parameter ω of the scoring function, 'tunes' the relation between the sensitivity

$h(x_i)\backslash y_i$	$y_i = 1$	$y_i = 0$	
$h(x_i) = 1$	tp	fp	$prec = \frac{tp}{tp+fp}$
$h(x_i) = 0$	fn	tn	$npv = \frac{tp}{tp+fp}$
	$sens = \frac{tp}{tp+fn}$	$spec = \frac{tn}{fp+tn}$	

Table 3.1.: Overview of binary classification measures.

and the specificity. The resulting pairs lead to the so-called receiver operating characteristic curve (ROC-curve). It is a two-dimensional graph with the *sens* on the vertical axis and 1-*spec* on the horizontal axis. A complete randomly assigned labeling would converge to a ROC-curve equivalent to the bisecting line (see Figure 3.1).

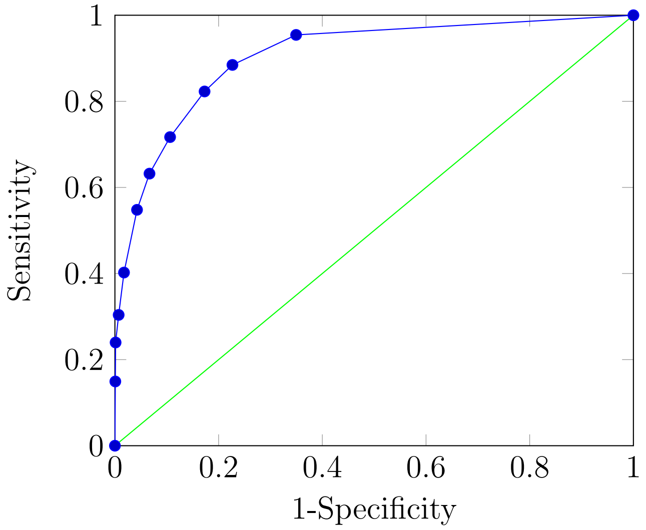

Figure 3.1.: ROC-curve in scenario T_{lin}^{500} for S^{train} for the later introduced Census Income data set in Section 6.1. The green bisecting line represents a classification achieved by random assignment.

ROC-curves allow to easily compare the performance of different classifiers. They visualize information of the sensitivity and the specificity and additionally lead to another often used measure, the area under curve or AUC . The AUC represents the area between the X-axis and the ROC-curve and gives a point estimation for the performance of a classifier. Therefore, the AUC value always lies in the interval $[0, 1]$. 1 would represent a classification accuracy of 100% and 0.5 would be the expected result of a random assignment.

Of course, the introduced measurements depend on the underlying data. In a worst case scenario, if all training labels are assigned with 1 and all testing labels are assigned with 0, no classifier can perform well. To compare the classifiers based on the introduced values, a good estimation of a representative combination of a training and testing set is required to get a good estimation of the measures above.[1]

3.2. Statistical Evaluation

The evaluation measures in the previous Section 3.1 have no direct probabilistic component. From the probabilistic point of view introduced in Section 1, the given data is a realization of the stochastic random variable $Z = X \times Y$. This motivates the following content based on Chapter 5 in [68].

Besides the deterministic measures introduced in Section 3.1, it is also possible to compute confidence intervals to evaluate the performance of classifiers. They are closely related to our new statistical evaluation approach introduced in Section 4.10. Additionally, they are used for the evaluation of the classification approach on real-world data later in this work.

In the next step, we link the introduced error definitions of Chapter 1, the theoretical generalization error

$$err_P(h, \mathcal{D}) = P(h(x) \neq y | (x, y) \sim \mathcal{D})$$

of a classifier h and the sample error

$$err_S(h(x); S) = \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \ .$$

The statistical estimation of confidence intervals allows the comparison of these two errors. Of course, the sample error is the most probable value for the

[1] Of course, there are plenty of functions to measure the performance like the popular Brier score. It uses the actual value of the scoring function $f(x)$ in comparison to its binary label and not the class label $h(x)$: $BS(h(x); S) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2$ (see [49]).

true error. It depends on a (randomly) chosen sample set. Like introduced in Chapter 1, a data set $S = \{(x_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^d \times \{0,1\}$ is interpreted as n i.i.d. realizations of random variables $Z = X \times Y$ from an unknown joint distribution $\mathcal{D}_Z = \mathcal{D}_X \times \mathcal{D}_Y$. For multiple samples $S_1, ..., S_k$ every sample error $err_{S_i}, i \in \{1, ..., k\}$ gives an estimation of the real, but unknown error err_P. It is therefore a realization of the random variable err_S. The random variable $r = n \cdot err_S$ follows a binomial distribution with the parameter $p = err_P$ and the sample size n. [2] Because of

$$E(err_S) = err_P = p,$$

err_S is also an unbiased estimator for err_P. Additionally, if n represents the cardinality of the sample set, the standard deviation of the sample error is given by

$$\sigma_{err_S} = \sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{err_P(1-err_P)}{n}},$$

leading with the substitution of err_P by err_S to the estimation

$$\hat{\sigma}_{err_S} = \sqrt{\frac{err_S(1-err_S)}{n}}$$

of the standard deviation σ_{err_S}.

The random variable $r = n \cdot err_S$ follows the binomial distribution converging to the normal distribution for $n \to \infty$. Therefore, err_S is approximated by a normal distribution with the same mean and the same standard deviation for sufficiently large n ($n \geq 30$, see [68]). This leads to an interval estimation introduced in Remark 3.2.1.

[2] This can be easily seen as $n \cdot error_S$ is the number of misclassified labels by the classifier h with the probability of $p = err_P$ for each label to be misclassified.

Remark 3.2.1

Let $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$ *be a data set and* h *a classifier. Then the* $1\text{-}\alpha$ *confidence interval for the sample error* err_S $(= err_S(h(x); S))$ *is given by*

$$\begin{aligned} I^{err}_{(1-\alpha)} &= \left[l^{err}_{1-\alpha}, u^{err}_{1-\alpha}\right] \\ &= \left[err_S - z_{(1-\frac{\alpha}{2})}\sqrt{\frac{err_S(1-err_S)}{n}}; err_S + z_{(1-\frac{\alpha}{2})}\sqrt{\frac{err_S(1-err_S)}{n}}\right] \end{aligned}$$

with $z_{(1-\frac{\alpha}{2})}$ *being the* $(1-\frac{\alpha}{2})$*-quantile of the standard normal distribution. For the classification accuracy* $acc_S = 1 - err_S$*, the corresponding interval is given by*

$$\begin{aligned} I^{acc}_{(1-\alpha)} &= \left[l^{acc}_{1-\alpha}, u^{acc}_{1-\alpha}\right] \\ &= \left[acc_S - z_{(1-\frac{\alpha}{2})}\sqrt{\frac{acc_S(1-acc_S)}{n}}; acc_S + z_{(1-\frac{\alpha}{2})}\sqrt{\frac{acc_S(1-acc_S)}{n}}\right] . \end{aligned}$$

Besides the two-sided confidence interval in Remark 3.2.1, one-sided intervals are calculated analogously.

Confidence interval estimates for the error err_P allow a better performance evaluation of a classifier. This holds especially in cases when there is no given partition into training and testing data like in the later introduced German Credit data set. Additionally, the concept of confidence intervals leads to hypothesis testing and will be helpful in our new statistical evaluation technique introduced in Section 4.10.

These estimation techniques are closely related to the k-fold cross-validation presented in the next section. It improves the estimation by a partition of the data into k sample sets.

3.3. Model Validation Techniques

To evaluate the performance of a binary classification technique it is important to have representative training and testing data sets. The k-fold cross-validation is one of the most common techniques to evaluate the performance of an algorithm. Additionally, the k-fold cross-validation is the basic cross-validation technique leading to similar approaches as, for example, the $N \times k$-fold cross-validation (see [30]). It is also applied to optimize the adjustable parameters according to the desired performance measure. The later introduced clustering based classifier will be evaluated and adjusted using a k-fold cross-validation technique. For further information see [83], [43], [33] and [60]. In the concept of the k-fold cross-validation, the data set S is uniformly, randomly partitioned into k folds $F = \{F_1, ..., F_k\}$ with $S = \bigcup_{i=1}^{k} F_i$, $F_i \neq F_j \ \forall i \neq j$, of approximately equal size. The classifier is trained and tested k times. In every step $j \in \{1, ..., k\}$ it is trained on $S_j = F \backslash F_j$ and tested on F_j (see [60] and [76]). The resulting average estimation is the cross-validation estimate of the error and the accuracy, respectively (see Remark 3.3.1).

Remark 3.3.1

Let $S = \{(x_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^d \times \{0, 1\}$ be a data set and h a classifier. Furthermore, let $F = \{F_1, ..., F_k\}$ be a partition of S into k sets F_j, $j = 1, ..., k$, with similar size and $S = \bigcup_{i=1}^{k} F_i$, $F_i \neq F_j$ for all $i \neq j$.

Then the k-fold cross-validation sample error err_S^{CV} of h and $F = \{F_1, ..., F_k\}$ is defined as

$$err_S^{CV}(h(x); F) = \frac{1}{n} \sum_{i=1}^{k} \sum_{(x,y) \in F_i} (h(x) - y)^2.$$

The k-fold cross-validation accuracy of h is

$$acc_S^{CV}(h(x); F) = 1 - err_S^{CV}(h(x); F) \ .$$

Of course, the concept of cross-validation can be evaluated via a unspecified scoring function f. The algorithmic concept is shown in Algorithm 1.

The error err_S^{CV} and the accuracy acc_S^{CV} of a k-fold cross-validation are interpreted as random variables as the training sets are chosen randomly from the original data set. Therefore, it is a realization of i.i.d. random variables (see Definition 1.0.1).

```
Algorithm 1: $k$-fold cross-validation
  Input: data set $S = \{(x_i, y_i)\}_{i=1}^n$; number of folds $k$;
  Output: $k$-fold scoring function $f$;
  Partition $S$ randomly into $k$ sets $\{F_1, ..., F_k\}$ of similar size.
  for $i \in \{1, ..., k\}$ do
      1. Classify the training set $S \backslash S_i$.
      2. Classify the testing set $S_i$.
  end
  Calculate the average $k$-fold scoring function $f$.
```

As an estimator for the real error or the real accuracy they are unbiased under the assumption that the classification technique is stable. Stable in this context means the classifier makes the same predictions for any of the k perturbations. In this case, the two estimators have approximately the variance

$$Var(err_S^{CV}) = err_S^{CV} \cdot (1 - \frac{err_S^{CV}}{n}) \tag{3.1}$$

and

$$Var(acc_S^{CV}) = acc_S^{CV} \cdot (1 - \frac{acc_S^{CV}}{n}) \tag{3.2}$$

(see [60] and [18]). This leads to better results as the bias of err_S^{CV} and acc_S^{CV} decreases when k is increased. The variance does not depend on the number of folds k in this case. In [60], Kohavi shows that there is almost no change in the variances (3.1) or (3.2) when the number of folds is varied.

An intuitive extension to the k-fold cross-validation is the so-called complete cross-validation which is the average of all $\binom{n}{n/k}$ (if $\frac{n}{k} \in \mathbb{N}$) possibilities for choosing sets of size $\frac{n}{k}$ out of the superset of size n. In practice, complete cross-validation is usually not suitable as the effort is way too high (see [60] for details).

The cross-validation will be relevant in Part IV of this work to evaluate the new classification approach.

Part III.

Clustering for Binary Classification Problems

Brieden shows the approximation of discrete convex norm maximization in [19] and together with Gritzmann numerous additional theoretical results concerning the relationship between feasible clusterings and polyhedrals (see [19], [20], [21], [22] and [23]). The practical application of the underlying theoretical concept was already shown for the consolidation of farmland in [19], [20] or with Borgwardt in [17]. Part II gave an introduction to the task of binary classification including the problem description, basic classification approaches and validation techniques. Now, based on the above mentioned theoretical and algorithmic work for geometric clustering, we introduce its application for binary classification problems and present theoretical results in joint work with Brieden and Gritzmann.

The task of clustering is to assign instances to groups (see [3]). These groups should be similar with respect to a chosen measure. A common criterion is geometric proximity like in the k-nearest neighbor approach of Section 2.3.

The main idea behind the new classification technique is the application of a

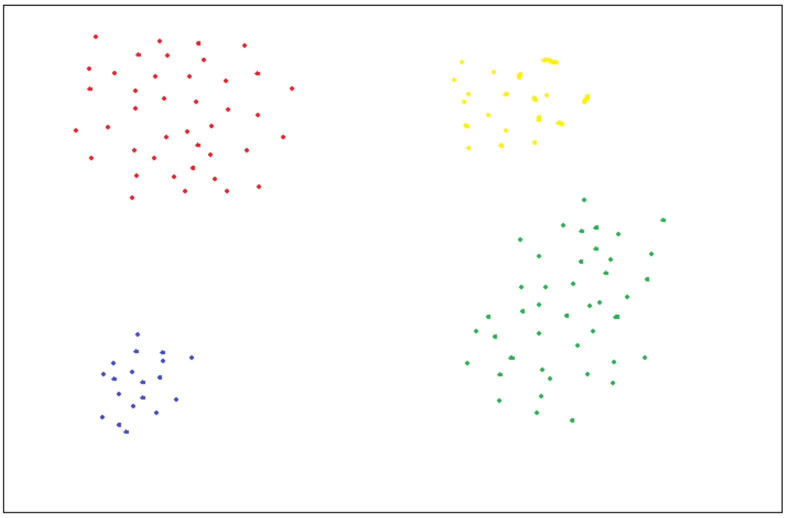

Figure 3.2.: Example instances in a two dimensional sample space.

geometric clustering and its special characteristic, the induced decomposition of the data space into convex cells.

In the first step, the new approach assigns the given data to a given number of convex clusters (see Figure 3.2). In the next step, a so called power diagram is composed which partitions the data-space into convex polyhedral cells, belonging to the clustering (see Figure 3.3). This partition is used in the next

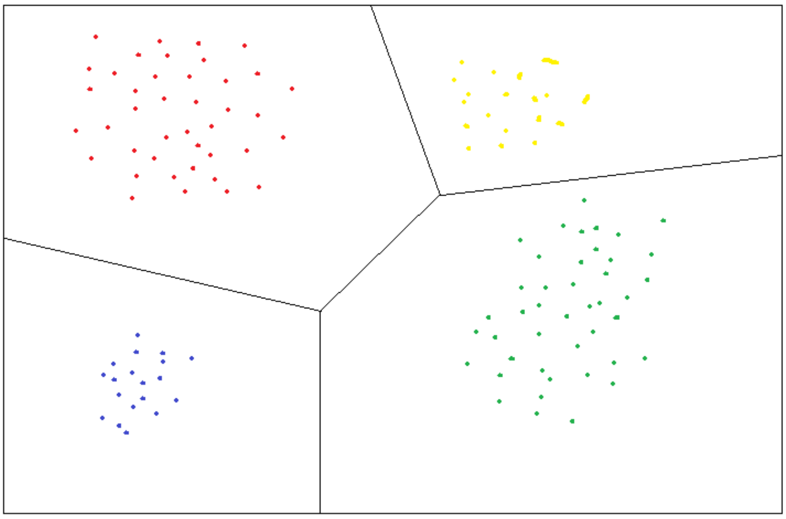

Figure 3.3.: Computation of a convex cell decomposition (power diagram) based on a feasible clustering.

step for the classification task of new data. To do so, the cells, i.e., clusters are assigned with a prediction value, based on the labels of the clustered training data (see Figure 3.4 and 3.5). New instances, for example the testing data, are assigned to the cluster in the last step. The predictive value depends on the cell the point lies in. It is labeled with the corresponding prediction value of the cluster (see Figure 3.6). In Section 4.1 of this work, we introduce the theoretical groundwork of geometric clustering. It is based on the work of Brieden and Gritzmann mentioned above. Furthermore, we present new results of the underlying structure. These results allow the application of a new

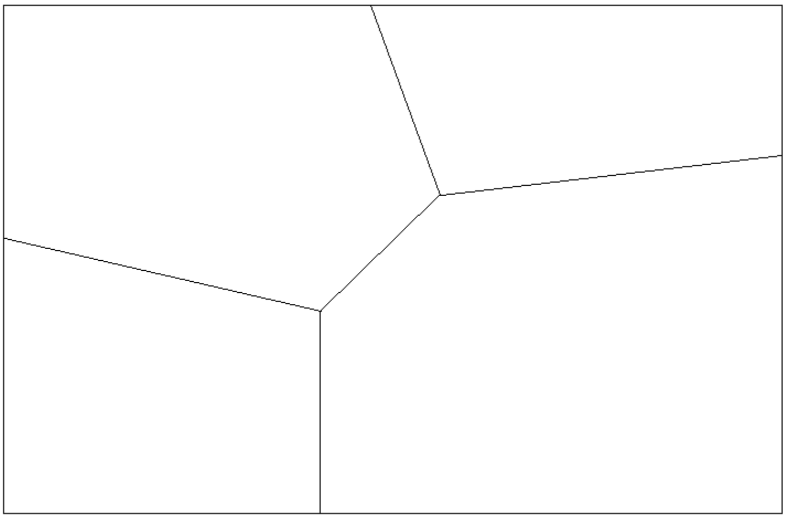

Figure 3.4.: Remaining convex cell decomposition as as segmentation of the data space.

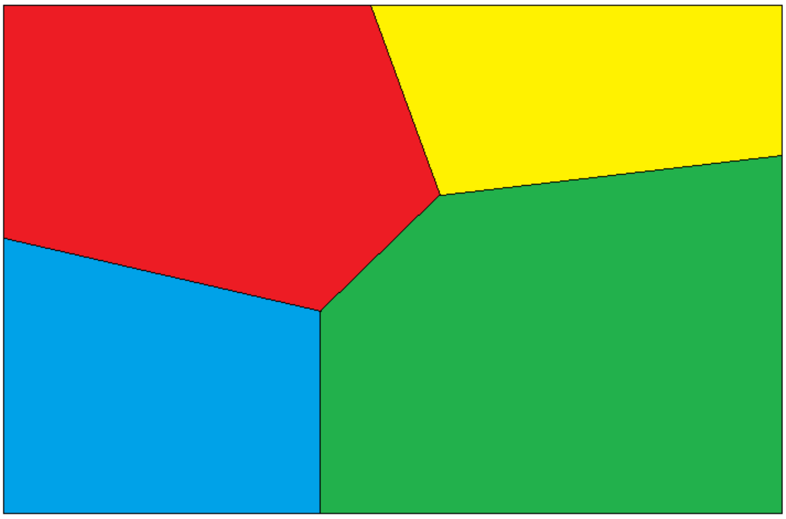

Figure 3.5.: Calculation of predictive values for each cluster.

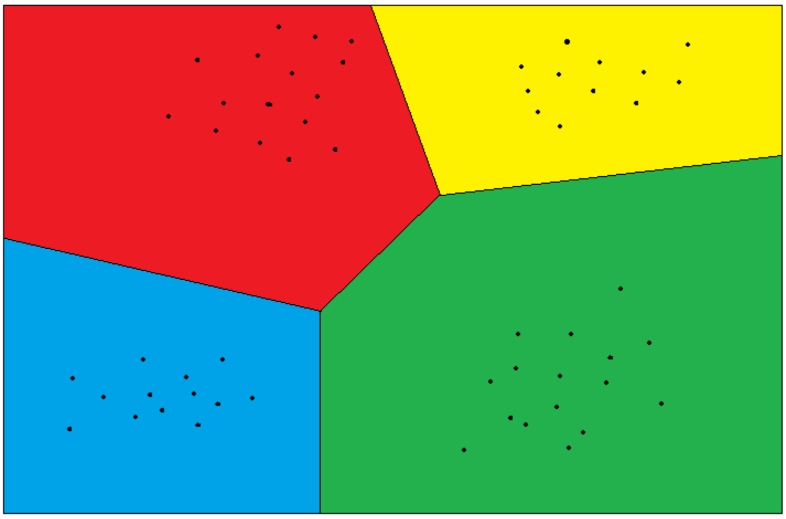

Figure 3.6.: Labeling new instances with the cluster value, depending on the underlying convex cell.

algorithmic approach to the already defined binary classification task. They lead to a new classification technique which is introduced as a result of the cell decomposition and the transformation in the later sections of this part. This new algorithmic sequence based on a feasible clustering is especially suitable for classification tasks.

Additionally, the later sections link the presented new technique to the topics in Part II of this work.

4. Geometric Clustering as a New Binary Classifier

In the first section of this chapter, we introduce the theoretical principles of the clustering approach. They represent the main ideas of the classification algorithm based on the work of Brieden and Gritzmann (see for example [20], [21] and [22]). Basic definitions are given on the mathematical interpretation of a clustering and how it is related to polytopes. We show the connection between a vertex of a polytope, a feasible clustering and a cell partition of the $\mathbb{R}^d$, called power diagrams. It leads to a partition of the Euclidean space into convex polyhedral cells. The possibility to compute a convex polyhedral cell decomposition, a so-called power diagram, is a fundamental principle of the new classification approach. The cell decompositions lead to the application of the geometric clustering approach for classification problems.

Additionally, the data transformation technique introduced in Section 4.2 allows the computation of a clustering for non-metric data by using the conditional probabilities. It is closely related to the value difference metric introduced in [81]. In Section 4.4, the new clustering approach is examined as a supervised learning approach for binary classification problems as introduced in Part II. We adopt the definitions and terms of classification techniques to the new clustering based classification approach. Additionally, we compare it to binary classification procedures. This leads to the definition of scoring functions for the new clustering approach in Section 4.5 based on the computation of instances of the training data assigned to the clusters. The weighting of data in Section 4.6 originates from the transformation technique of Section 4.2. It determines which input variables are selected and weighted due to their importance. The adjustment of the clustering parameters to optimize the prediction, like the number of clusters, boundaries, etc. is the topic of Section 4.7. It introduces relevant techniques to additionally improve the quality of prediction. In the end of this part, we present different scoring functions for the introduced classification technique. They are used on real-world data in Part IV of this work.

4.1. Clustering as a Norm Maximization Problem

Clustering or cluster analysis as a field of unsupervised learning offers a broad variety of different techniques and algorithmic approaches. We focus on geometric clustering. Therefore, we form cluster or group objects referring to their geometric information. It is expressed by the position of a point in the data space, mostly the $\mathbb{R}^d$, and the distance measured by an ellipsoidal norm, for example the Euclidean norm.

In this section, we interpret the first step of the classification task introduced in Part II as finding (geometrical) similarity. It is represented by the geometrical proximity in a given data set S. At first, it is a deterministic approach without the probabilistic point of view of Part II. The statistical component will be added afterwards. The clustering technique is unsupervised as it does not use the labels y_i but the input values x_i. Similarity of instances as the origin of a good estimation is measured by geometrical proximity leading to convex clusterings in the following. At first, basic definitions of a clustering are given and extended.

Definition 4.1.1 (Clustering)

A k-clustering $C := (C_1, ..., C_k)$ is a partition of a set $X \subset \mathbb{R}^d$ into k non-empty sets $C_1, ..., C_k$. C_i is called the i-th cluster of the Clustering C with $i \in \{1, ..., k\}$.

Let $C := (C_1, ..., C_k)$ be a k-clustering of a data set X. Then $\kappa_i = |C_i|$ is the size of the i-th cluster for $i \in \{1, ..., k\}$ and $|C| := (|C_1|, ..., |C_k|)$ is called the shape of a k-clustering.

With $X \subset S$ we denote the subset of input values, $X = \{x_i\}_{i=1}^n$. As we regard geometric clusterings, X is a subset of $\mathbb{R}^d$ in all the clusterings of this work. The reason we can constrain ourselves to $\mathbb{R}^d$ is the transformation technique explained later in Section 4.2. In addition, we focus on clusterings with given sizes and bounds for each cluster. The advantage is the possibility to 'control' the clusters in terms of a minimum number of points and therefore predetermined estimation reliability. Even if the new objective function introduced later in this work does not require strict bounds to 'fill' all clusters evenly, they could be used to adjust a clustering. The given sizes are possible because of the underlying bounded-shape partition polytope introduced later in this section. Therefore, we extend the general definition of a clustering to a (k, l, u)-clustering with given lower (l) and upper (u) integer bounds.

Definition 4.1.2 ((k, l, u)-Clustering)

A k-clustering $C := (C_1, ..., C_k)$ with $l = (l_1, ..., l_k), u = (u_1, ..., u_k) \in \mathbb{N}^k$ and $l_i \leq \kappa_i \leq u_i$ for $i \in \{1, ..., k\}$ is called (k, l, u)-clustering.

(k, l, u)-clusterings are equivalent to so-called integer balanced clusterings in [23] and represent clusterings that fulfill the given pair of lower and upper size restrictions (l, u).

Remark 4.1.3

For a given set X of size n and lower and upper bounds (l, u) the number of feasible clusterings is

$$\sum_{\substack{l_i \leq \kappa_i \leq u_i, i \in \{1,\dots,k\} \\ \sum_i \kappa_i = n}} \frac{n!}{\prod_{i=1}^{k} \kappa_i! \prod_{i=1}^{k} m_i!}$$

with $m_i := |\{\kappa_i : \kappa_i \in \{\kappa_1, ..., \kappa_k\}\}|$.

Remark 4.1.3 shows the enormous number of all feasible clusterings fulfilling lower and upper size restrictions.

The lower and upper bounds determine the feasibility of a clustering, i.e., whether a clustering exists that fulfills the restriction given by the bounds.

Definition 4.1.4 (Set of (Feasible) Clusterings)

$\mathcal{C}(X, k) := \{C : C$ is a k-clustering of $X\}$ is the set of k-clusterings of X and

$$\mathcal{C}(X, k, l, u) := \{C : C \text{ is a } (k, l, u)\text{-clustering of } X\}$$

is the set of (k, l, u)-clusterings of X.

Of course, there are trivial requirements for lower and upper bounds to allow feasible clusterings.

Remark 4.1.5

Let X be a set of size n and $(l, u) \in \mathbb{N}^2$ lower and upper bounds. A clustering $C := (C_1, ..., C_k)$ is feasible with respect to (l, u) if

$$\sum_{i=1}^{k} l_i \leq n \leq \sum_{i=1}^{k} u_i$$

holds.

In the following, we will always assume that the bounds will allow feasible clusterings. Additionally, with the number of cluster k and the bounds (l, u) clear from the context and fixed, we use the informal term bounded-shape clustering (BSC) for a clustering respecting these numbers. Also, we use the notation $BSC(X) = BSC(k, l, u) := \mathcal{C}(X, k, l, u)$.

Especially for classification a clustering with bounds is very helpful. A given minimal number of points in a cluster leads to better statistical estimations because of the law of great numbers.

In the next step, we focus on the relationship between vertices of special polytopes and the assignment of a clustering. Brieden and Gritzmann show in [22] and [23] that vertices of a polytope can be identified with a clustering. Therefore, the clustering is represented by a vector consisting of the sums of the points of each cluster. Each resulting clustering can be (of course not uniquely) identified with the center of gravity or the sum of all points (see [23]).

Definition 4.1.6 (Cluster Sum and Center of Gravity)

Let $C := (C_1, ..., C_k)$ be a clustering of a set X. Then the cluster sum of a cluster C_i is defined as $s_i := \sum\limits_{x \in C_i} x$ for $i \in \{1, ..., k\}$. The vector $v(C) := (s_1^T, ..., s_k^T)^T \in \mathbb{R}^{d \cdot k}$ is called the cluster sum vector. The center of gravity c_i of a Cluster C_i is defined as $c_i := \frac{s_i}{\kappa_i}$ for $i \in \{1, ..., k\}$.

While the center of gravity as the center of a cluster is more intuitive as representative in the first place, the cluster sum allows lower and upper bounds. In our case, the solution can be identified as a vertex of a special polytope, the already mentioned bounded-shape partition polytope[1]. The set of all cluster sum vectors lead to this polytope. It is the convex hull of the cluster sum

[1] The bounded-shape partition polytope is also related to a network flow problem (see [54]))

vertices and can be expressed by linear constraints or a linear program.

Definition 4.1.7 (Set of Cluster Sum Vectors)
Let X be a subset of $\mathbb{R}^d$. Then $V := V(X; k, l, u) := \{v(C) : C \in \mathcal{C}(X, k, l, u)\}$ is the set of all cluster sum vectors.

In the next step the bounded-shape partition polytope is defined as the convex hull of all feasible (k, l, u)-clusterings of a set S.

Definition 4.1.8 (Bounded-Shape Partition Polytope (BSPP))
Let X be a subset of $\mathbb{R}^d$. The bounded-shape partition polytope is defined as the convex hull of all cluster sum vectors

$$BSPP = BSPP(k, l, u) = BSPP(k, l_1, ..., l_k, u_1, ..., u_k) := conv V(X; k, l, u) \ .$$

Each vertex of the bounded-shape partition polytope is related to a clustering by its cluster sum vector representation.

Lemma 4.1.9
Let v^* be a vertex of a $BSPP$. Then there is exactly one (k, l, u)-clustering $C = (C_1, ..., C_k)$ with $v(C) = v^*$. We call this the clustering of v^*.
Proof:
The proof is given in [23] as the bounded-shape partition polytopes are contained in the subspace of the described gravity bodies. □

Lemma 4.1.9 allows the identification of clusterings with polytopes which was already shown by Barnes, Hoffman and Rothblum in [11]. This leads to linear constraints and to the solution of a linear program representing a feasible clustering. Additionally, the vertices, i.e., the corresponding clusterings have an additional useful characteristic for the classification task. These clusterings are pairwise separable and allow a cell decomposition.

As shown in [54], the computation of a vertex of the bounded-shape partition polytope can be done in the following linear system. It is the representation by hyperplanes and corresponds to the vertex characterization of the $BSPP$ shown in Definition 4.1.8.

Definition 4.1.10 (BSPP)

Let $k, l_i, u_i \in \mathbb{N}$ for all $i \in \{1, ..., k\}$ with $\sum_{i=1}^{k} l_i \leq n \leq \sum_{i=1}^{k} u_i$. We call the polytope defined by the constraints

$$\begin{array}{rcll} \sum_{j=1}^{n} \xi_{ij} & \leq & u_i & (i \leq k) \\ \sum_{j=1}^{n} \xi_{ij} & \geq & l_i & (i \leq k) \\ \sum_{i=1}^{k} \xi_{ij} & = & 1 & (j \leq n) \\ \xi_{ij} & \geq & 0 & (i \leq k, j \leq n) \end{array}$$

the bounded-shape partition polytope $BSPP(k, l, u)$.

Hwang, Onn and Rothblum show in [54] that a bounded-shape clustering can be computed as a solution $(\xi_{ij}^*) \in \{0,1\}^{k \times n}$ of the following integer system:

$$\max \ f(\xi)$$

$$\sum_{j=1}^{n} \xi_{ij} \leq u_i \quad (i \leq k) \tag{4.1}$$

$$\sum_{j=1}^{n} \xi_{ij} \geq l_i \quad (i \leq k) \tag{4.2}$$

$$\sum_{i=1}^{k} \xi_{ij} = 1 \quad (j \leq n) \tag{4.3}$$

$$\xi_{ij} \in \{0,1\} \quad (i \leq k, j \leq n) \; .$$

Hwang, Onn and Rothblum also showed that the bounded-shape partition problem can be solved in polynomial time (see [53], [55]). The reason is the total unimodularity of the underlying matrix derived from constraints (4.1), (4.2) and (4.3) of the problem formulation above.

Because the corresponding matrix is totally unimodular (see [54]), every solution of the relaxation is integral. Therefore, it suffices to solve the following relaxation of the program above:

$$\begin{array}{rcll} & \max \; f(\xi) & & \\ \sum_{j=1}^{n} \xi_{ij} & \leq & u_i & (i \leq k) \\ \sum_{j=1}^{n} \xi_{ij} & \geq & l_i & (i \leq k) \\ \sum_{i=1}^{k} \xi_{ij} & = & 1 & (j \leq n) \\ \xi_{ij} & \geq & 0 & (i \leq k, j \leq n) \end{array}$$

With the interpretation as vertices of the bounded-shape partition polytope, we can show that each clustering resulting from the relaxation of the maximization problem is a separable clustering.

Definition 4.1.11 (Linear Separability)

Let $A, B \subset \mathbb{R}^d$. A and B are weakly linearly separable if there is a hyperplane $H_{a,\mu} := \{x \in \mathbb{R}^d : a^T x = \mu\}$ with $a \in \mathbb{R}^d \backslash \{0\}$ and $\mu \in \mathbb{R}$ such that $A \subset H^{\geq}_{a,\mu} := \{x \in \mathbb{R}^d : a^T x \geq \mu\}$ and $B \subset H^{\leq}_{a,\mu} := \{x \in \mathbb{R}^d : a^T x \leq \mu\}$.

A and B are strictly linearly separable if there is a hyperplane $H_{a,\beta} \subset \mathbb{R}^d$ with $a \in \mathbb{R}^d \backslash \{0\}$ and $\beta \in \mathbb{R}$ such that $A \subset H^{>}_{a,\mu}$ and $B \subset H^{<}_{a,\mu}$.

Definition 4.1.12 (Separability of Clusterings)
Let $C := (C_1, ..., C_k)$ be a k-clustering of X. C allows (weak, strict) linear separation (or is weakly, strictly linearly separable) if C_i and C_j are (weakly,strictly) linearly separable for any $i \neq j$, $i, j \in \{1, ..., k\}$.

The separability of clusters as a similarity criterion in the terms of an intuitive 'good' characterization was mentioned before when linear classifiers were introduced in Part II of this work. The next theorem shows that every vertex of the polytope represents a linearly separable clustering. This result was first proven by Barnes, Hoffman and Rothblum in [11] for a similar polytope.

Theorem 4.1.13
Let v^* be a vertex of the $BSPP$. Then the BSC C^* associated with $v^* = v(C^*)$ allows strict linear separation.
Proof.
See [11]. □

Besides the separability of a clustering, a vertex of the bounded-shape partition polytope has a second characteristic. It leads to a special arrangement of a cell decomposition with every cluster lying in a polytopal cell.
In the next step, we show an additional attribute a clustering should have to be considered as a good clustering and that leads to a good classification. This attribute is the pairwise distance of the centers.
Every vertex of the bounded-shape partition polytope is a feasible clustering of the maximization problem described on page 67. A good clustering in our case should be separable firstly and the pairwise (Euclidean) distance between the centers of each cluster should be maximized secondly. The graphical examples in Figure 4.1 and 4.2 show the need of the separability and the maximized distance. Separability itself does not lead automatically to intuitive 'good'

clusterings. Figure 4.1 shows that without an additional property a clustering could be separable but it would not be the 'natural' choice. If the distance between the points should be maximized, the result is a clustering which would be intuitively valuated as good. The reason is that the points of each cluster are clearly separated (e.g. Figure 4.2). The maximization of the pairwise distance between the centers of a clustering would lead to the following type of maximization:

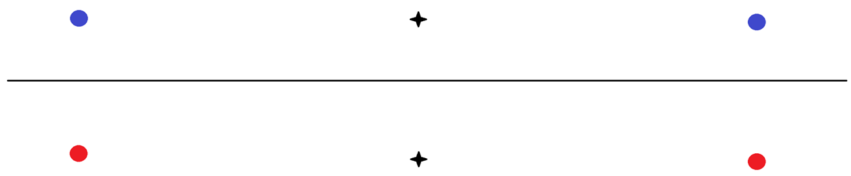

Figure 4.1.: The points are separated but the distance between the centers is minimal.

$$\max_{C=(C_1,\ldots,C_k)\in\mathcal{C}} \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} \omega_i\omega_j \left\| c_i - c_j \right\|^q , \qquad (4.4)$$

with weights $\omega_i \in \mathbb{R}, i \in \{1, ...k\}$. This is a norm maximizing approach including for example the Euclidean norm for $q = 2$. Unfortunately, this would lead to a nonlinear optimization problem which is known to be $\mathbb{NP}$-hard (see [15]). The illustration in Figure 4.3 shows that norm maximization can be interpreted as scaling up a unit ball to fit the feasible region. The nonlinear norm maximization problem can be piecewise linearly approximated and the

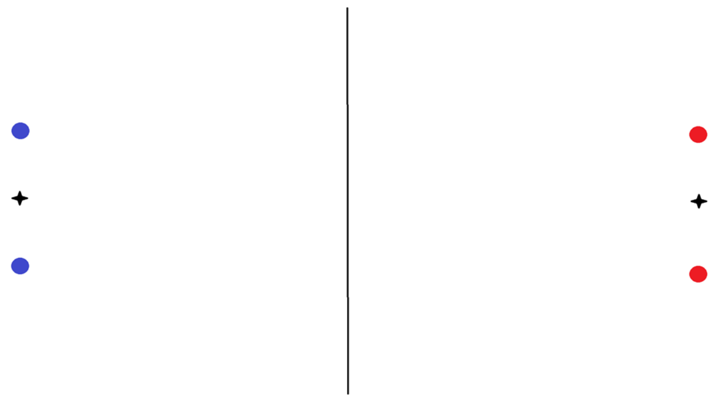

Figure 4.2.: The points are separated and the distance between the centers is maximal.

optimal norm maximal vertex could be therefore iterative calculated as shown in Figure 4.4 and 4.5. The algorithmic implementation of this procedure is shown later in Section 4.3.

Brieden and Gritzmann show in [23]that in the case of $\sum_{\{x \in X\}} x = 0$ the norm maximization in (4.4) is equivalent to the maximization of the total linear inter cluster distance (4.5) with $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$ and $\omega_i \in \mathbb{R}$ for $i = 1, ..., k$:

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{x_i \in C_i} \sum_{x_j \in C_j} (a_i - a_j)^T (s_i - s_j), \tag{4.5}$$

with s_i being the cluster sum of cluster C_i, $i \in \{1, ..., k\}$. Therefore, we use a linear approach with a objective function $a^T v$ with $a \in \mathbb{R}^{d \cdot k}$ and $v = v(C) := (s_1^T, ..., s_k^T)^T \in \mathbb{R}^{d \cdot k}$ as a cluster sum approach

$$\max_{C=(C_1,...,C_k) \in \mathcal{C}} \sum_{i=1}^{k} a_i^T c_i \kappa_i = \max_{C=(C_1,...,C_k) \in \mathcal{C}} \sum_{i=1}^{k} a_i^T s_i \ . \tag{4.6}$$

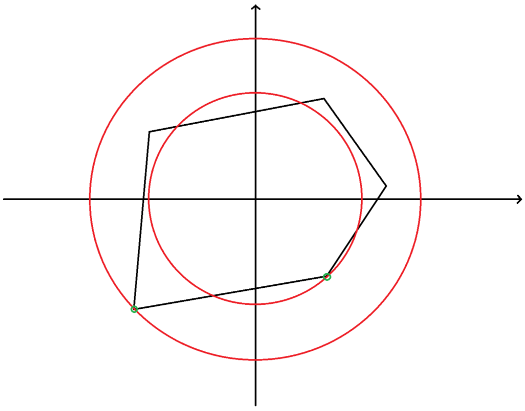

Figure 4.3.: Norm maximization over a polytope.

This is related to the so-called least-squares assignment (LSA). Least-squares assignments are connected to power diagrams introduced later in this section (see [6] and [23]).

Definition 4.1.14 (Least-Squares Assignment (LSA))
Let $C = (C_1, ..., C_k)$ be a (k, l, u)-clustering of $X \subset \mathbb{R}^d$ and $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$. C is called a least-squares assignment (LSA) of X to a if and only if it minimizes

$$\sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} \left|\left|x_j - a_i\right|\right|^2$$

over the $BSPP(k, l, u)$.

Finding a clustering with a minimal least-squares assignment is similar to finding the clustering with the minimal intra-cluster variance (if the site vector a is replaced with the centers of gravity c which is also a desirable feature of a good clustering).

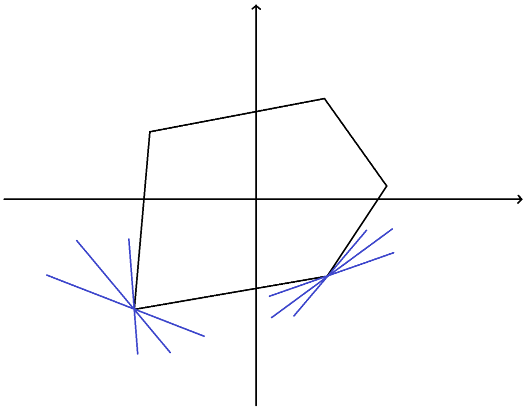

Figure 4.4.: The nonlinear problem is piecewise approximated by linear functions.

The cluster sum approach (4.6) is defined in the next step as an approximation of the least-squares assignment, the so-called cluster sum assignment. It is motivated by application of the clustering technique for classification tasks. Optimizing the objective functional of a least-squares assignment minimizes the variance of the points assigned to a cluster. Therefore, outliers could be identified and would often be assigned to a corresponding cluster with only a few points. This characteristic of a least-squares assignment is often desirable but not for our designated use as a classifier. For this task, the clusters should contain enough points to precisely estimate the values used for prediction. In clusters containing only outliers, new instances assigned to these clusters would receive a 'bad' estimation achieved by only a few points (that are even outliers). A possible solution is to force a minimum number of points into a cluster by setting lower bounds. Another possibility is the cluster sum assignment defined in the next step which leads to proper clusters for estimation without

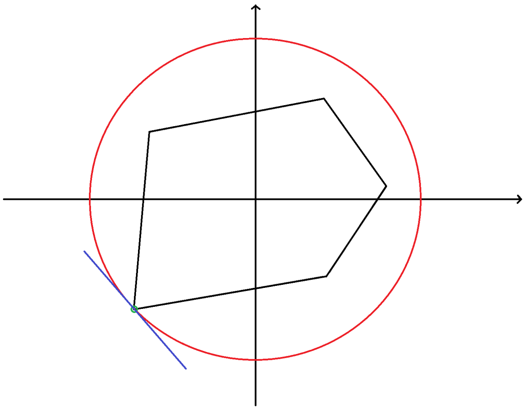

Figure 4.5.: A norm maximal vertex is an optimal clustering.

the setting of strict lower bounds.

Definition 4.1.15 (Cluster Sum Assignment (CSA))
Let $C = (C_1, ..., C_k)$ be a (k, l, u)-clustering of $X \subset \mathbb{R}^d$ and $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$. C is called a cluster sum assignment (CSA) of X to a if and only if it maximizes

$$\sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} a_i^T x_j$$

over the $BSPP(k, l, u)$.

Empirical results show that the cluster sum assignment consists of proper filled clusters. They are perfectly suitable for the classification task introduced in later sections. We show the link to the least-squares assignment in the next theorem.

Theorem 4.1.16

Let $||a_i|| = 1$ for all $i \in \{1, ..., k\}$. Then minimizing

$$\sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij} \, ||x_j - a_i||^2$$

is equivalent to maximizing

$$\sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij} a_i^T x_j \ .$$

Proof.

Minimizing

$$\sum_{i=1}^{k}\sum_{j=1}^{n} -2\xi_{ij} a_i^T x_j$$

is equivalent to minimizing

$$\sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij}(||a_i||^2 - 2a_i^T x_j)$$

if $||a_i|| = 1$ for all $i \in \{1, ..., k\}$. □

This theorem shows that with standardized sites a_i, the resulting clusterings of LSA and CSA are the same. Without the standardization in the proof of Theorem 4.1.16, the term

$$\sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij} \, ||a_i||^2$$

would be influenced by the chosen assignment, i.e., the clustering. The components, i.e., sites $a_i^T \in \mathbb{R}^d$ of the vector $a^T = (a_1^T, ..., a_k^T)^T$ with the highest value $||a_i||$ would 'draw' the points in its direction. Therefore, standardizing a_i leads to more balanced clusterings in terms of the size of the resulting clusters. Additionally, the following example shows that without standardization, a cluster sum assignment and a least-squares assignment could differ from each other.

Example 4.1.17

Let

$$X := \{x_1; x_2; x_3\} = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \begin{pmatrix} \text{-1} \\ \text{-1} \end{pmatrix}; \begin{pmatrix} 0.5 \\ \text{-0.5} \end{pmatrix} \right\}$$

and

$$a := \left(\begin{pmatrix} 1+\lambda \\ 0 \end{pmatrix}; \begin{pmatrix} 0 \\ \text{-1} \end{pmatrix} \right) .$$

At first, let x_1 and x_3 be assigned to cluster C_1 and x_2 to cluster C_2. This leads for $\lambda = 0$ to an indifferent clustering $\mathcal{C}^1 = \{C_1^1, C_2^1\} = \{\{x_1, x_3\}; \{x_2\}\}$ compared with $\mathcal{C}^2 = \{C_1^2, C_2^2\} = \{\{x_1\}; \{x_2, x_3\}\}$ with respect to the same cluster sum assignment value of 2.5 *for both clusterings and the same least-squares assignment value of* 2.5.

Theorem 4.1.16 shows that with the standardization of the vector a, the optimal LSA is always equivalent to the optimal CSA.

In general, without $||a_i|| = 1$ for all $i \in \{1, ..., k\}$, the least-squares assignment is not identical with the maximal cluster sum assignment.

For $\lambda = 2$ and $\lambda = -0.5$ the following two tables show the different LSA and CSA decisions with all possible clusterings $\mathcal{C}^0$, $\bar{\mathcal{C}^0}$, $\mathcal{C}^1$, $\bar{\mathcal{C}^1}$, $\mathcal{C}^2$, $\bar{\mathcal{C}^2}$, $\mathcal{C}^3$ and $\bar{\mathcal{C}^3}$. While the most important clusterings $\mathcal{C}^1$ and $\mathcal{C}^2$ have already been introduced, the notation $\bar{\mathcal{C}^i}$ represents the 'opposite' of $\bar{\mathcal{C}^i}$, i.e. $\bar{\mathcal{C}^i} = \{C_2^1, C_1^1\}$ instead of $\mathcal{C}^i = \{C_1^1, C_2^i\}$. Therefore, with $\mathcal{C}^0 = \{C_1^0, C_2^0\} = \{\{x_1, x_2, x_3\}; \{\}\}$, $\mathcal{C}^3 = \{C_1^3, C_2^3\} = \{\{x_1, x_2\}; \{x_3\}\}$ and the mentioned opposites, the following tables list the LSA and the CSA values for all possible clusterings.

$\lambda = 2$	$\mathcal{C}^0$	$\bar{\mathcal{C}^0}$	$\mathcal{C}^1$	$\bar{\mathcal{C}^1}$	$\mathcal{C}^2$	$\bar{\mathcal{C}^2}$	$\mathcal{C}^3$	$\bar{\mathcal{C}^3}$
CSA	*1.5*	*0.5*	*5.5*	*-3.5*	*4.5*	*-2.5*	*0.5*	*1.5*
LSA	*28.5*	*6.5*	*12.5*	*22.5*	*6.5*	*24*	*22.5*	*12.5*

While in the case of $\lambda = 2$, the best clustering in terms of a maximal CSA value is $\mathcal{C}^1$, it is $\mathcal{C}^2$ in terms of a minimal LSA value.

$\lambda =$ -0.5	$\mathcal{C}^0$	$\bar{\mathcal{C}^0}$	$\mathcal{C}^1$	$\bar{\mathcal{C}^1}$	$\mathcal{C}^2$	$\bar{\mathcal{C}^2}$	$\mathcal{C}^3$	$\bar{\mathcal{C}^3}$
CSA	*0.25*	*0.5*	*1.75*	*-1*	*2*	*-1.25*	*0.5*	*0.25*
LSA	*4.75*	*6.5*	*2.5*	*8.75*	*2.75*	*4*	*5*	*6.25*

In the second case, with $\lambda =$ -0.5, choosing by the CSA value would lead to $\mathcal{C}^2$ but choosing by the LSA value would lead to $\mathcal{C}^1$.

This example shows that maximizing the cluster sum assignment value could generally result in completely different clusterings compared to least-squares assignments.

If a_i is standardized and the set X is replaced with the centered set $X^c = X - \frac{\sum_{i=1}^n x_i}{n}$ the cluster sum assignment can be interpreted as a clustering with the centers of gravity c_i being pushed away from the unit ball with respect to directions a_i. In Section 4.3, we show a special iterative sequence which converges to

$$\sum_{i=1}^{k} \kappa_i \, ||c_i|| = \sum_{i=1}^{k} ||s_i|| \ .$$

It computes an optimal cluster sum assignment in each step.

In the next step we show that we can compute a linearly separable clustering by maximizing the cluster sum assignment over the bounded-shape partition polytope.

Theorem 4.1.18

Let $X \subset \mathbb{R}^d$ be a set, $k, l_i, u_i \in \mathbb{N}$ for all $i \in \{1, ..., k\}$ with $\sum_{i=1}^k l_i \leq n \leq \sum_{i=1}^k u_i$ the parameter set, $BSPP(k, l, u)$ the corresponding bounded-shape par-

tition polytope and $a := (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$.
Then we can find a vertex v^* *of* $BSPP$ *with* $a^T v^* \geq a^T v$ *for any* $v \in BSPP$ *by solving a linear program.*
Proof:
Hwang, Onn and Rothblum showed in [54] that the matrix of the bounded-shape partition polytope is total unimodular. Therefore, every solution (ξ_{ij}^*) *of the following relaxation is integral (see [78]):*

$$\max \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} a_i^T x_j$$

$$\begin{array}{cccc} \sum_{j=1}^{n} \xi_{ij} & \leq & u_i & (i \leq k) \\ \sum_{j=1}^{n} \xi_{ij} & \geq & l_i & (i \leq k) \\ \sum_{i=1}^{k} \xi_{ij} & = & 1 & (j \leq n) \\ \xi_{ij} & \geq & 0 & (i \leq k, j \leq n) \end{array}$$

Lemma 4.1.9 shows that every vertex represents a feasible bounded-shape clustering which implies the assumption. □

Theorem 4.1.18 shows the first core concept of our new classification approach. The clustering used for the classification procedure is optimal with respect to its cluster sum value and represents a vertex in the bounded-shape partition polytope. Because of its unimodularity, the calculation of an integer clustering is done in polynomial time (see [54]) and there even exist strongly polynomial-time algorithms (see [53]). Additionally, Theorem 4.1.13 shows the property of linear separability for each pair of clusters. This separability leads to a partition of $\mathbb{R}^d$ into convex polyhedral cells, also known as power diagrams. These decompositions can be interpreted as a generalization of a Voronoi tessellation and represent the next basic principle of our new classification approach. For a general survey of power diagrams see [5].

Partitioning the Euclidean space into cells with every cell containing a cluster leads to the new classification procedure shown in the later Section 4.3.

Definition 4.1.19 (Power Diagram)

Let $a := (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$ *with* $a_i \neq a_j$ *for all* $i, j \in \{1, ..., k\}$ *with* $i \neq j$ *and* $\Sigma = (\sigma_1, ..., \sigma_k) \in \mathbb{R}^k$*. Then the* i*-th power cell* $P_i^{a,\Sigma}$ *is defined as*

$$P_i^{a,\Sigma} = \{x \in \mathbb{R}^d : \|x - a_i\|_2^2 - \sigma_i \leq \|x - a_j\|_2^2 - \sigma_j, \forall j \in \{1, ..., k\} \backslash i\}$$

and $\mathcal{P}^{a,\Sigma} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ *is the* (a, Σ)*-power diagram.*

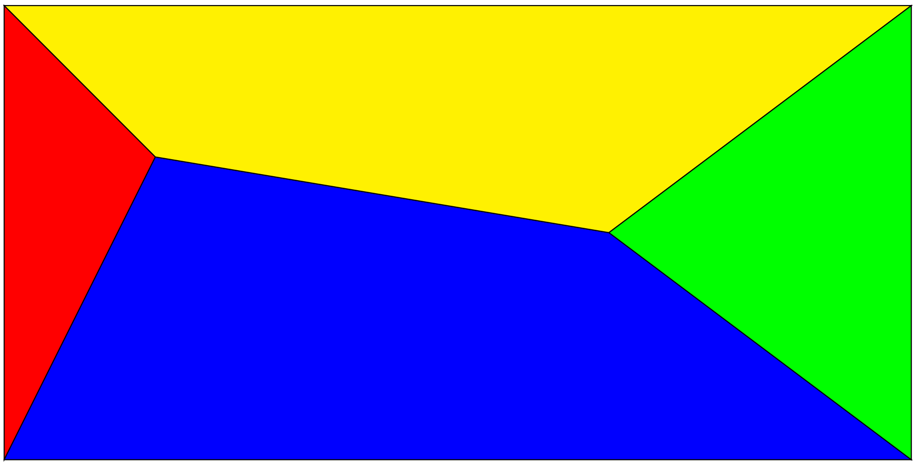

Figure 4.6.: A power diagram of $\mathbb{R}^2$, consisting of 4 cells.

These power diagrams can be interpreted as a generalization of Voronoi diagrams as every Voronoi diagram always represents a power diagram (see [6]) The next corollary shows that the clustering associated with a vertex of the bounded-shape partition polytope induces a power diagram and therefore partitions the metric space into convex cells.

Corollary 4.1.20

Let v^* be a vertex of the BSPP, $C := (C_1, ..., C_k)$ be the BSC associated with v^* and let $a := (a_1^T, ..., a_k^T) \in \mathbb{R}^{d \cdot k}$ with $a^T v^* > a^T v$ for any $v \in BSPP \backslash \{v^*\}$. Then there exists a (a, Σ)-power diagram $\mathcal{P}^{a,\Sigma} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ with $C_i \subset int(P_i^{a,\Sigma})$ for $i \in \{1, ..., k\}$.

Proof.

See [23] as the bounded-shape partition polytope is contained in the linear subspace in the described gravity bodies. □

Like the computation of the clustering shown in Theorem 4.1.18, the power diagram $\mathcal{P}^{a,\Sigma} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ can also be computed by solving a linear program.

Theorem 4.1.21

Let $X \subset \mathbb{R}^d$ of size n and $k, l_i, u_i \in \mathbb{N}$ for all $i \in \{1, ..., k\}$ with $\sum_{i=1}^k l_i \leq n \leq \sum_{i=1}^k u_i$. Let $C := (C_1, ..., C_k)$ be a (k, l, u)-clustering with $v^* := v^*(C)$ the vertex of the corresponding polytope $BSPP(k, l, u)$ and $a \in \mathbb{R}^{d \cdot k}$ with $a^T v^* > a^T v$ for any $v \in BSPP \backslash \{v^*\}$.

Then we can calculate a (a, Σ)-power diagram $\mathcal{P}^{a,\Sigma} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ with $C_i \subset int(P_i)$, for all $i \in \{1, ..., k\}$ by solving a linear program.

Proof:

Let $(\xi_{ij}^*) \in \{0, 1\}^{k \times n}$ be the optimum corresponding to the vertex $v^*(C)$ in the linear program of Theorem 4.1.18 and $A := \{(i, j) : \xi_{ij}^* \neq 0\}$. Referring to Section 4 in [23], the solution μ_i^* of the following linear program

$$\min \sum_{i=1}^{k} \kappa_i \mu_i + \sum_{j=1}^{n} \eta_j$$

$$\mu_i + \eta_j \quad \geq \quad \gamma_{ij} \quad (i, j \in A)$$

leads with

$$\Sigma = (\sigma_1, ..., \sigma_k), \ \sigma_i = \|a_i\|^2 - 2\mu_i^*, (1 \leq i \leq k)$$

to a (a, Σ)*-power diagram* $\mathcal{P}^{a,\Sigma} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ *with*

$$P_i^{a,\Sigma} = \bigcap_{i \neq j} \{x : a_{ij}^T x \leq \mu_j^* - \mu_i^*\} \ .$$

□

The introduced cell decomposition by power diagrams allows to easily assign a point to a cell by evaluating the separating hyperplanes. With the calculation of a clustering by solving a linear program the cell decompositions provide the mathematical background for the new classification approach.
While the underlying data was set to a subset $X \subset \mathbb{R}^d$ in this section, in practice common data has nominal, ordinal or mixed level of scale. In the next section we present a technique to apply the clustering approach on data that is not real valued by replacing the data with its conditional expectations. Additionally, this introduces a statistical component to the deterministic clustering procedure.

4.2. Transformation by Conditional Probabilities

The following section will introduce the general principles of the data transformation technique. Additionally, this links the new approach to stochastic topics introduced in Part II. It it closely related to the value difference metric introduced in [81]. The transformation of the input variables into metric scaled values allows the use of the geometric clustering approach for data of every level of scale. After the transformation, our clustering algorithm can perform an optimal clustering solution in the multidimensional Euclidean space.

The underlying probabilistic concept is similar to the naive Bayes approach introduced in Section 2.1. For a basic overview of the underlying stochastic principles and definitions see for example [62]. The 'naive' assumption of conditional independence is expressed by the following equation:

$$P(X_1 = x_1, ..., X_d = x_d | Y = y) = \prod_{i=1}^{d} P(X_i = x_i | Y = y) \ .$$

This equation was applied to model the desired probability $P(Y = y | X_1 = x_1, ..., X_d = x_d)$ by the Bayes' rule. The underlying assumption of conditional independence reduced the probability $P(Y = y | X_1 = x_1, ..., X_d = x_d)$ to the probabilities $P(X_i = x_i | Y = y)$, which are easier to determine.

A similar assumption is used in the following for the transformation technique. It models the conditional probabilities $P(Y = y | X_1 = x_1, ..., X_d = x_d)$ by the marginal conditional probabilities $P(Y = y | X_i = x_i)$, $i \in \{1, ..., d\}$, based on the conditional expected value. The conditional expected value

$$E(Y | X_1 = x_1, ..., X_d = x_d) = \sum_{y \in \Omega_y} y P(Y = y | X_1 = x_1, ..., X_d = x_d)$$

consists of the corresponding conditional probabilities. The following assumption states that the conditional expected value is a combination of the one dimensional conditional expected values $E(Y | X_i = x_i)$, $i = 1, ..., d$. In the next step, this leads to

$$P(Y = y | X_1 = x_1, ..., X_d = x_d) = \sum_{i=1}^{d} \beta_i P(Y = y | X_i = x_i)$$

with

$$\beta_i = \hat{\beta}_i \frac{P(Y = y|X_1 = x_1, ..., X_d = x_d)}{P(Y = y|X_i = x_i)}, \; \sum_{i=1}^{d} \hat{\beta}_i = 1 \; .$$

The result is a convex combination

$$\sum_{i=1}^{d} \beta_i E(Y|X_i = x_i) \tag{4.7}$$

with

$$\beta_i = \hat{\beta}_i \frac{E(Y|X_1 = x_1, ..., X_d = x_d)}{E(Y|X_i = x_i)}, \; \sum_{i=1}^{d} \hat{\beta}_i = 1 \; .$$

The conditional expected values $E(Y|X_i = x)$, i.e., their estimators are the new values replacing the original input values of the i-th feature X_i. With this transformation the data can be clustered in the next step. In the binary case, the conditional expected values relate to a Bernoulli-distributed random variable. Therefore, if x is fixed, they are equivalent with the conditional probabilities.

Remark 4.2.1

If Y is a binary random variable, $Y \sim Be(p)$, then

$$E(Y|X = x) = P(Y = 1|X = x) =: p_{|x}$$

holds.

Besides a transformation setting, this also leads to a linear convex weighting approach introduced in Section 4.6.

The conditional expected values in (4.7) allow to cluster non-metric data with the geometrical approach introduced in Section 4.1. In the binary case, all conditional expectations are equivalent to conditional probabilities. Therefore, the data is actually mapped to the d-dimensional unit cube $[0, 1]^d$. Obviously, the transformed data set consists only of real values. Additionally, the transformation calculates the one-dimensional estimators for the conditional expected values

$$E(Y|X_i = x),\ i \in \{1, ..., d\},\ x \in \Omega_i\ ,$$

with Ω_i as the sample space of the random variable X_i. These are equivalent to the conditional probabilities

$$P(Y = 1|X_i = x), i \in \{1, ..., d\},\ x \in \Omega_i\ ,$$

in the binary case. After the transformation, each instance $x = (x_1, ..., x_d)$ is represented by the vector of their conditional expected values

$$(x_1, ..., x_d) \to (E(Y|X_1 = x_1), ..., E(Y|X_d = x_d))$$

or the equivalent conditional probabilities

$$(P(Y = 1|X_1 = x_1), ..., P(Y = 1|X_d = x_d))$$

in the binary case. As the vector consists of expected values or probabilities, the geometric clustering approach of Section 4.1 is applicable.

In the next step, the conditional expected values are estimated by the corresponding conditional means of a data set S.

Remark 4.2.2

Let $S = \{(x_j, y_j)\}_{j=1}^{n} \subset \mathbb{R}^d \times \{0, 1\}$ with $x_j = (x_{j1}, ..., x_{jd})$ be a sample of d stochastic random variables X_i, $i = 1, ..., d$, and Y.

Then

$$\tilde{x}_{y|x} = \theta(Y = y|X_i = x) = \frac{\sum_{j=1}^{n} 1_{\{y_j\}}(y) \cdot 1_{\{x_{ji}\}}(x)}{\sum_{j=1}^{n} 1_{\{x_{ji}\}}(x)}, \textit{ for } i = 1, ..., d$$

is an unbiased estimator for the conditional expected value $E(Y|X_i = x)$.

Proof:

The definition of the conditional probability leads to

$$P(Y = y|X_i = x) = \frac{P(Y = y, X_i = x)}{P(X_i = x)}, \;\; i \in \{1, ..., k\} \; .$$

Therefore, the estimators

$$\theta(Y = y, X_i = x) = \frac{\sum_{j=1}^{n} 1_{\{y_j\}}(y) \cdot 1_{\{x_{ji}\}}(x)}{n}$$

and

$$\theta(X_i = x) = \frac{\sum_{j=1}^{n} 1_{\{x_{ji}\}}(x)}{n}$$

consist of the (conditional) frequencies and lead to a mean estimation of the conditional expected value $E(Y|X_i = x)$. □

The following remark shows that the mean of each transformed variable $\tilde{x}_{y|x}$ is equivalent to the mean of the labels y.

Remark 4.2.3

Let $S = \{(x_j, y_j)\}_{j=1}^{n} \subset \mathbb{R}^d \times \{0, 1\}$ and $\tilde{x}_{y|x}$ the estimator for the conditional expected value $E(Y|X_i = x)$ of Remark 4.2.2.

Then

$$\frac{1}{n} \sum_{j=1}^{n} \tilde{x}_{y|x_{ji}} = \frac{1}{n} \sum_{j=1}^{n} y_j$$

holds for $i \in \{1, ..., d\}$.

It follows from Remark 4.2.2 and corresponds to the property $E(E(Y|X)) = E(Y)$ for conditional expected values (see [62]).

The estimation technique uses the labels of Y and the features of the input variables $X_1, ..., X_d$ based on the given training data. All possible features of the random variables $X_1, ..., X_d$ are included in their sample spaces $\Omega_1, ..., \Omega_d$. Algorithm 2 shows the transformation of the data set by replacing these features with the estimations of the conditional expected values.

Algorithm 2: Computation of the transformed training data

```
Input: training data set $S^{train} = \{(x_j^{train}, y_j^{train})\}_{j=1}^n \subset \mathbb{R}^d \times \{0,1\}$;
       sample spaces $\Omega_1, ..., \Omega_d$ of $X_1, ..., X_d$;
Output: $\tilde{S}^{train} = \{(\tilde{x}_j^{train}, y_j^{train})\}_{j=1}^n$;
        $\{\{\tilde{x}_{y|x_{ji}} : x_{ji} \in \Omega_i\}\}_{i=1}^d \subset \mathbb{R}^{|\Omega_1| \times ... \times |\Omega_d|}$;
for $i = 1$ to $d$ do
    for $x \in \Omega_i$ do
        calculate $\tilde{x}_{y|x}$
    end
end
Generate the transformed data set:
for $i = 1$ to $d$ do
    for $j = 1$ to $n$ do
        $\tilde{x}_{ji}^{train} \leftarrow \tilde{x}_{y|x_{ji}}$
    end
end
Return $\tilde{S}^{train}$ and $\{\{\tilde{x}_{y|x_{ji}} : x \in \Omega_i\}\}_{i=1}^d$.
```

In a second step, the data which should be classified needs a transformation based on these estimations $\{\{\tilde{x}_{y|x_{ji}} : x \in \Omega_i\}\}_{i=1}^d \subset \mathbb{R}^{|\Omega_1| \times ... \times |\Omega_d|}$.

While the training data is transformed based on the labels of the training set, the testing data has to be adjusted in a different way. As the testing labels are unknown, the features of the testing data $\{x_j^{test}\}_{j=1}^m \subset \mathbb{R}^d$ have to be replaced with the estimations for the conditional expected values of the training data $\{\{\tilde{x}_{y|x_{ji}} : x \in \Omega_i\}\}_{i=1}^d \subset \mathbb{R}^{|\Omega_1| \times \ldots \times |\Omega_d|}$. The result is a 'new' testing data set determined by Algorithm 3.

Algorithm 3: Assignment of the testing data

```
**Input**: testing data set $\{x_j^{test} = (x_{j1}^{test}, ..., x_{jd}^{test})\}_{j=1}^m \subset \mathbb{R}^d$;
           $\{\{\tilde{x}_{y|x_{ji}} : x_{ji} \in \Omega_i\}\}_{i=1}^d \subset \mathbb{R}^{|\Omega_1| \times \ldots \times |\Omega_d|}$;
**Output**: $\{\tilde{x}_j^{test} = (\tilde{x}_{j1}^{test}, ..., \tilde{x}_{jd}^{test})\}_{j=1}^m$;
Assign the testing data:
for $i = 1$ to $d$ do
    for $j = 1$ to $m$ do
        $\tilde{x}_{ij}^{test} \leftarrow \tilde{x}_{y|x_{ji}}$
    end
end
Return $\{\tilde{x}_j^{test}\}_{j=1}^m$.
```

In the next step, we can compute a clustering based on the transformed training data. Additionally, we can assign the transformed testing data to a cluster by evaluating its position with respect to the separating hyperplanes of the power diagram induced by the clustering.

4.3. Geometric Clustering for Classification Problems

In this section, we complete the introduction of optimal geometric clusterings as a foundation for a new technique to classify binary data sets. So far, we presented the mathematical introduction of a geometric clustering in Section

4.1 and the transformation technique in Section 4.2.

Now we describe and analyze the algorithmic implementation of the geometric clustering. It is included in an iterative sequence in which we compute a clustering in each step. This sequence converges to a cluster sum assignment (CSA) introduced in Section 4.1. Furthermore, we prove termination and compare it to a least-squares assignment (LSA). The introduced iterative clustering based sequence is the foundation for the classifiers defined in the following sections and is applied to real-world data sets in Part IV of this work.

At first, we show the computation of a single clustering. Algorithm 4 generates an optimal clustering with respect to a given site vector $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$. It is equivalent to the search for an optimal vertex in the bounded-shape partition polytope introduced in Section 4.1 and a solution of the linear program in Theorem 4.1.18. The underlying data is the transformed feature set $\{\tilde{x}_j^{train}\}_{j=1}^n$ described in Section 4.2. Besides $\{\tilde{x}_j^{train}\}_{j=1}^n$ and the site vector $a = (a_1^T, ..., a_k^T)^T$, the cluster number k and bounds (l, u) are additional input parameters.

Algorithm 4: Calculation of the clustering C

Input: $\{\tilde{x}_j^{train}\}_{j=1}^{n} \subset \mathbb{R}^d$, k, l_i, $u_i \in \mathbb{N}$ with $\sum_{i=1}^{k} l_i \leq n \leq \sum_{i=1}^{k} u_i$;
initial site vector $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$;

Output: (k, l, u)-clustering $C = (C_1, ..., C_k)$;

Solve the linear program

$$
\begin{array}{cccc}
 & \max \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} a_i^T x_j & & \\
\sum_{j=1}^{n} \xi_{ij} & \leq & u_i & (i \leq k) \\
\sum_{j=1}^{n} \xi_{ij} & \geq & l_i & (i \leq k) \\
\sum_{i=1}^{k} \xi_{ij} & = & 1 & (j \leq n) \\
\xi_{ij} & \geq & 0 & (i \leq k, j \leq n)
\end{array}
$$

and return a feasible solution $(\xi_{ij}^*) \in \{0, 1\}^{k \times n}$ as the assignment.

Lemma 4.3.1

Algorithm 4 computes a (k, l, u)-clustering by linear programming with $k \cdot n$ variables and $(k + 1) \cdot n + 2 \cdot k$ constraints.

We use the solution of Algorithm 4 as input for another linear program. Therefore, the standardized sums of the clustering solution are applied as new sites

$$(a_1, ..., a_k) \leftarrow (\frac{c_1}{||c_1||}, ..., \frac{c_k}{||c_k||}) = (\frac{s_1}{||s_1||}, ..., \frac{s_k}{||s_k||})$$

of a linear program with the unchanged restrictions, i.e., the same underlying *BSPP*. Because of $c_i = \frac{s_i}{\kappa_i}$, the standardized sum $\frac{s_i}{||s_i||}$ of a cluster C_i is equivalent to the standardized center $\frac{c_i}{||c_i||}$. This iterative step is repeated until the solution does not change anymore.

```
Algorithm 5: The iterative cluster sum approach
  Input: $\{\tilde{x}_j^{train}\}_{j=1}^n \subset \mathbb{R}^d$, $k$, $l_i$, $u_i \in \mathbb{N}$ with $\sum_{i=1}^k l_i \leq n \leq \sum_{i=1}^k u_i$;
         initial site vector $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$;
  Output: $(k, l, u)$-clustering $C = (C_1, ..., C_k)$;
    1. Apply Algorithm 4 for the site vector $a = (a_1^T, ..., a_k^T)^T$ to obtain
       clustering $C$ and the related assignment $(\xi_{ij}) \in \{0,1\}^{k \times n}$.
    2. Update each site $a_i$ as the standardized cluster sum $\frac{s_i}{||s_i||}$ with
       $s_i := \sum_{j=1}^n \xi_{ij} x_j$.
       If the objective function value increases during the last iteration,
       go to 1., else return the current assignment and sites.
```

Theorem 4.3.2 (Iterative Clustering for a Cluster Sum Assignment) *Algorithm 5 terminates with a feasible (k, l, u)-clustering that is a cluster sum assignment. The clustering allows a (a, Σ)-power diagram.*

Proof:

We first prove that standardized cluster sums $\frac{s_1}{||s_1||}, ..., \frac{s_k}{||s_k||}$ with $s_i := \sum_{j=1}^{n} \xi_{ij} x_j$ for all $i \in \{1, ..., k\}$ of a clustering $C = (C_1, ..., C_k)$ consists of optimal sites $\frac{s_i}{||s_i||}$ with respect to a cluster sum assignment.

If the sequence in Algorithm 5 terminates, the result is a cluster sum assignment with a feasible power diagram.

Let $C := (C_1, .., C_k)$ be a fixed clustering with cluster sums $s_1, ..., s_k \in \mathbb{R}^d$ and

$$\Theta(C, A) := \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} \frac{x_j^T a_i}{\|a_i\|}$$

is the objective value for the clustering C with the sites $A = \{a_1, ..., a_k\}$. In the first step, we show that with a fixed clustering $C := (C_1, .., C_k)$, $\Theta(C, A)$ is maximal for $a_i = s_i$.

Let further be $C^{(l)}$ the resulting clustering of the l-th iteration computed with sites $A^{(l)} := \{a_1^{(l)}, ..., a_k^{(l)}\}$. Therefore, $A^{(l|1)} = \{a_1^{(l|1)}, ..., a_k^{(l|1)}\} = \{s_1^{(l)}, ..., s_k^{(l)}\}$ consists of the cluster sums of the clustering $C^{(l)}$.

Because of

$$\sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} \frac{x_j^T a_i}{\|a_i\|} = \sum_{i=1}^{k} \frac{s_i^T a_i}{\|a_i\|} = \sum_{i=1}^{k} \|s_i\| \left(\frac{s_i}{\|s_i\|} \right)^T \left(\frac{a_i}{\|a_i\|} \right)$$

and the Cauchy Schwarz inequality $-1 \leq \left(\frac{s_i}{\|s_i\|}\right)^T \left(\frac{a_i}{\|a_i\|}\right) \leq 1$*, the inequality*

$$\begin{aligned}
\Theta(C^{(l)}, A^{(l|1)}) &:= \sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij}^{(l)} \frac{x_j^T a_i^{(l|1)}}{\left\|a_i^{(l|1)}\right\|} \\
&= \sum_{i=1}^{k} \frac{{s_i^{(l)}}^T s_i^{(l)}}{\left\|s_i^{(l)}\right\|} \\
&\geq \sum_{i=1}^{k} \frac{{s_i^{(l)}}^T a_i^{(l)}}{\left\|a_i^{(l)}\right\|} \\
&= \sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij}^{(l)} \frac{x_j^T a_i^{(l)}}{\left\|a_i^{(l)}\right\|} \\
&=: \Theta(C^{(l)}, A^{(l)})
\end{aligned}$$

holds. Because of

$$\begin{aligned}
\Theta(C^{(l)}, A^{(l|1)}) &= \sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij}^{(l)} \frac{x_j^T a_i^{(l|1)}}{\left\|a_i^{(l|1)}\right\|} \\
&= \sum_{i=1}^{k} \frac{{s_i^{(l)}}^T s_i^{(l)}}{\left\|s_i^{(l)}\right\|} \\
&\leq \max_{(\xi_{ij})\in\{0,1\}^{k\times n}} \sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij} \frac{x_j^T s_i^{(l)}}{\left\|s_i^{(l)}\right\|} \\
&= \sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij}^{(l|1)} \frac{x_j^T s_i^{(l)}}{\left\|s_i^{(l)}\right\|} \\
&= \sum_{i=1}^{k}\sum_{j=1}^{n} \xi_{ij}^{(l|1)} \frac{x_j^T a_i^{(l|1)}}{\left\|a_i^{(l|1)}\right\|} \\
&=: \Theta(C^{(l|1)}, A^{(l|1)}) \ ,
\end{aligned}$$

the following sequence is increasing monotonously:

$$\Theta(C^{(l)}, A^{(l)}) \leq \Theta(C^{(l)}, A^{(l|1)}) \leq \Theta(C^{(l|1)}, A^{(l|1)}) \ . \tag{4.8}$$

Algorithm 5 terminates because of its Step 2 and the monotonously increasing sequence (4.8). In each iteration we compute a cluster sum assignment as a vertex of the corresponding bounded-shape partition polytope. Therefore, the last iteration leads to a (a, Σ)-power diagram due to Theorem 4.1.21. □

Applying a standardized cluster sum vector as a new direction leads to better clusterings with respect to the objective value of the cluster sum assignment in the next step. The new approach applies the standardized cluster sums $\frac{s_1}{\|s_1\|}, ..., \frac{s_k}{\|s_k\|}$ of the resulting clustering C of Algorithm 4 as new sites in the next step. Theorem 4.3.2 proves that the iterative sequence (4.8) is increasing monotonously. Therefore, Algorithm 5 terminates with a feasible clustering inducing a power diagram described in Section 4.1. Additionally, the iterative approach leads to a clustering with

$$\Theta(C^{(l)}, A^{(l|1)}) \xrightarrow{x \to \infty} \max \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} \frac{x_j^T s_i^*}{\|s_i\|} = \sum_{i=1}^{k} \frac{s_i^{*T} s_i^*}{\|s_i\|} = \sum_{i=1}^{k} \|s_i^*\| \ .$$

Theorem 4.1.16 shows that computing a least-squares assignment is equivalent to a cluster sum assignment if the sites a_i are standardized. Therefore, the standardization leads to clusterings that are least-squares assignments and cluster sum assignments.

Corollary 4.3.3

The result of Algorithm 5 is a cluster sum assignment and a least squares assignment, simultaneously.

Proof:

Theorem 4.1.16 shows that for standardized sites, a least-squares assignment is equivalent to a cluster sum assignment. □

Borgwardt, Brieden and Gritzmann show similar results for least-squares assignments in [16]. Their iterative sequence based on least-squares assignments uses non-standardized centers of gravity as new sites in every step. The next corollary shows that this sequence based on least-squares assignments also terminates in the case of standardized sites.

Corollary 4.3.4

Algorithm 3 in [16] also terminates if each site is updated with the standardized center of gravity $\frac{c_i}{\|c_i\|}$ instead of the non-standardized center of gravity c_i. The resulting least-squares assignment is also a cluster sum assignment.

Proof:

Theorem 4.1.16 shows that for standardized sites, a least-squares assignment is equivalent to a cluster sum assignment. Therefore, Algorithm 3 in [16] terminates with standardized centers of gravity in each step because of Theorem 4.3.2. □

In general, least-squares assignments and cluster sum assignments are not identical as we show in Example 4.1.17 in Section 4.1. For cluster sum assignments, the standardization is necessary. The following example shows that without the standardization, the monotonously increasing sequence (4.8) cannot be guaranteed.

Example 4.3.5

Let

$$X := \{x_1; x_2; x_3\} = \left\{ \begin{pmatrix} -4 \\ 1 \end{pmatrix}; \begin{pmatrix} 1 \\ -3 \end{pmatrix}; \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\}$$

be the point set for a clustering with 2 clusters,

$$a^0 = \left(\begin{pmatrix} 10 \\ 0 \end{pmatrix}; \begin{pmatrix} 0 \\ \text{-60} \end{pmatrix} \right)$$

the initial sites and

$$\Theta(C^{(l)}, A^{(l)}) := \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij}^{(l)} x_j^T a_i^{(l)} = \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij}^{(l)} x_j^T a_i^{(l)}$$

the objective function of a cluster sum assignment given in Definition 4.1.15. Then the initial assignment achieved by the maximization of

$$\Theta(C^{(0)}, A^{(0)}) = \sum_{i=1}^{k} \sum_{j=1}^{n} \xi_{ij} x_j^T a_i$$

has a value of $\Theta(C^{(0)}, A^{(0)}) = 200$ *and the optimal clustering*

$$C^{(0)} = \{\{x_1\}; \{x_2, x_3\}\}$$

with the corresponding (non-standardized) centers

$$c^0 = \left(\begin{pmatrix} \text{-4} \\ 1 \end{pmatrix}; \begin{pmatrix} 0 \\ \text{-2} \end{pmatrix} \right) = a^1$$

as the next sites.

This leads to the value

$$\Theta(C^{(0)}, A^{(1)}) = \begin{pmatrix} \text{-4} \\ 1 \end{pmatrix}^T \begin{pmatrix} \text{-4} \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ \text{-4} \end{pmatrix}^T \begin{pmatrix} 0 \\ \text{-2} \end{pmatrix} = 25$$

and in the next step by maximizing $\Theta(C, A^{(1)})$ with the centers of $C^{(0)}$ as new sites $A^{(1)}$ to a maximal value of

$$\Theta(C^{(1)}, A^{(1)}) = \begin{pmatrix} \text{-5} \\ 0 \end{pmatrix}^T \begin{pmatrix} \text{-4} \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ \text{-3} \end{pmatrix}^T \begin{pmatrix} 0 \\ \text{-2} \end{pmatrix} = 26.$$

The corresponding clustering is $C^{(1)} = \{\{x_1, x_3\}; \{x_2\}\}$ with its new centers

$$c^1 = \left(\begin{pmatrix} \text{-2.5} \\ 0 \end{pmatrix}; \begin{pmatrix} 1 \\ \text{-3} \end{pmatrix} \right) =: a^2 \ .$$

Without standardization, this leads in the next step to the maximal objective value of

$$\Theta(C^{(1)}, A^{(2)}) = \begin{pmatrix} \text{-5} \\ 0 \end{pmatrix}^T \begin{pmatrix} \text{-2.5} \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ \text{-3} \end{pmatrix}^T \begin{pmatrix} 1 \\ \text{-3} \end{pmatrix} = 22.5,$$

showing that without standardized sites, the iterative sequence is usually not increasing.

The use of a cluster sum assignment is motivated by the use for classification as described in Section 4.1. A cluster sum assignment is more suitable than a least-squares assignment because the resulting clusters tend to be equally filled with points. In contrast, least-squares assignments often identify outliers which can lead to poorly filled clusters.

Theorem 4.3.6 (Runtime Upper Bound)
The number of iterations of Algorithm 5 is bounded by

$$(40ek^2n)^{(d+1)k-1} \ .$$

Proof:

Since Algorithm 5 computes a clustering that allows a strongly feasible power diagram, the proof follows from Theorem 3 in [16].

The upper bound shows that the following algorithm has a polynomial running time for fixed d and k. Algorithm 5 is similar to the well known k-means algorithm which iteratively calculates a clustering leading to a local optimum. The calculation of a norm maximal global optimum of the k-means algorithm is $\mathbb{NP}$-hard as shown in [2] and [63].

Besides the computation of the clustering a second component of the new classification approach is the induced cell-decomposition introduced in Section 4.1. In the next step, Algorithm 6 is presented that computes an (a, Σ)-power diagram based on the results of Algorithm 5.

With the computations of Algorithm 6, the position of the separating hyperplanes $H_{a,\mu}$ is determined with Corollary 4.1.20 and Theorem 4.1.21.

Algorithm 6: Calculation of the (a, Σ)-power diagram

```
Input: clustering $C = (C_1, ..., C_k)$ with vector $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$;
       assignment $(\xi_{ij}^*) \in \{0,1\}^{k \times n}$;
       cluster sizes $\kappa_i = |C_i|$, $i \in \{1, ..., k\}$;
Output: $(a, \Sigma)$-power diagram $\mathcal{P} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ with
        $\{\mu_{ij} := \mu_j^* - \mu_i^*\}, i, j \in \{1,...,k\}$;
Compute the solution $(\mu_i^*, \eta_j^*)$ of
    $\min \sum_{i=1}^{k} \kappa_i \mu_i + \sum_{j=1}^{n} \eta_j$
    $\mu_i + \eta_j \geq \gamma_{ij} \quad (i, j \in A)$
with $A := \{(i,j) : \xi_{ij}^* \neq 0\}$ which leads with
    $\Sigma = (\sigma_1, ..., \sigma_k), \sigma_i = \|a_i\|^2 - 2\mu_i^*, (1 \leq i \leq k)$
to a $(a, \Sigma)$-power diagram $\mathcal{P}^{a,\Sigma}$ with $P_i^{a,\Sigma} = \bigcap_{i \neq j} \{x : a_{ij}^T x \leq \mu_j^* - \mu_i^*\}$ .
```

After the computation of the clustering and the cell decomposition, the (transformed) testing data set can be classified by the use of the cell decomposition. The following algorithm computes the assignment of the testing data according to the geometrical information with respect to the separating hyperplanes. The value $C_a(\tilde{x}_i^{test}) \in \{1, ..., k\}$ is therefore the cluster assignment index of an instance, i.e., point $\tilde{x}_i^{test}$.

Algorithm 7: Assignment with a (a, Σ)-power diagram

```
Input: $\{\tilde{x}_i^{test}\}_{i=1}^m \subset \mathbb{R}^d$, $(a, \Sigma)$-power diagram $\mathcal{P} = (P_1^{a,\Sigma}, ..., P_k^{a,\Sigma})$ and
       $\{\mu_{ij} := \mu_j^* - \mu_i^*\}, i, j \in \{1,...,k\}$;
Output: labeled testing set $\{(\tilde{x}_i^{test}, C_a(\tilde{x}_i^{test}))\}_{i=1}^m$;
for $i = 1$ to $m$ do
    $l := 1$;
    for $j = 2$ to $k$ do
        if $a_{lj}^T \tilde{x}_i^{test} > \mu_{lj}$ then
            $l = j$;
        end
        $j = j + 1$;
    end
    $C_a(\tilde{x}_i^{test}) = j$
end
```

Lemma 4.3.7

Algorithm 7 has a running time of $O(m \cdot d \cdot (k-1))$.

Proof:

The labeling of each point takes $O(d \cdot (k-1))$. □

After the testing data is assigned to the clusters of the underlying clustering, the instances need to be labeled which completes the classification process. The computation of typical cluster values, closely related to the scoring function, can be done by different techniques. It is not related to the clustering process itself but to the evaluation of the labels of the training data assigned to the clusters. Therefore, the next section links the clustering approach and its use as a classifier to the terminology introduced in Part II.

4.4. Clustering as a Classifier

In this section the new clustering approach is examined as a supervised learning approach for binary classification problems.

Most of the techniques for binary classification perform badly when the binary labels are not equally balanced over the data. The more unequal the distribution of the class labels is, the more the accuracy of the majority class differs from the minority class. Especially in classification tasks arising in medicine, this leads to severe problems. Woods et al. give an example of identifying breast cancer in the 'Mammography Data Set' containing 10921 'negatives' and 260 'positives' (see [89]). In data sets like the mentioned 'Mammography Data Set' with the ratio of negatives and positives differ naturally significant from each other, the conditional accuracies are often extremely different. While in examples like this, the majority class has an accuracy of nearly 100%, the minority class is often single-digit. A lot of techniques are used to handle this important problem. Sampling techniques are commonly used which try to adjust the size of the minority or the majority so that the ratio between the cardinality of the labels tend to 1 (see [48],[56]and [79]).

The new classification approach introduced in Section 4.3 is based on geometric clusterings and transformation technique using the conditional frequencies introduced in Section 4.2. The concept of similarity by the conditional probabilities in Section 4.2 is related to the probabilistic view already introduced in Part II of this work. The underlying data is interpreted as an i.i.d. sample, i.e., realizations of random variables $X = (X_1, ..., X_d)$ and Y, originating from a sample space with an unknown, joint probability distribution $\mathcal{D}_Z = \mathcal{D}_X \times \mathcal{D}_Y$ of $Z = X \times Y$. The same sample space is a simplifying but necessary assumption in most cases. In contrast, the clustering approach assumes that each cluster describes an own sample space. Therefore, there are k sample spaces

Ω_i, $i \in \{1, .., k\}$, in this interpretation with the n realizations originating from these sample spaces. The clustering naturally identifies the points with the same underlying distribution. This leads to excellent results in data that contains heterogeneous groups, violating the assumption of the same underlying sample space. Each identified cluster C_i refers to its own underlying sample space and to a cluster dependent set of random variables $X_{C_i} = (X^1_{C_i}, ..., X^d_{C_i})$ and Y_{C_i} with the joint distribution $\mathcal{D}_Z(C_i) = \mathcal{D}_X(C_i) \times \mathcal{D}_Y(C_i)$.
Therefore, in the binary case, the clustering identifies groups with a high conditional probability for 1 and 0, respectively.
In the next step, we apply the technical definitions of Part II to introduce the clustering based classifier. So far, the data set S consists of the feature set $\{x_i\}_{i=1}^n$ and the label set $\{y_i\}_{i=1}^n$. In Section 4.2, the feature set was replaced by the estimations for the conditional expected values. From now on, this will be used as feature set $\{x_i\}_{i=1}^n$.
Firstly, the assignment of points to a cluster is formalized.

Definition 4.4.1 (Cluster Assignment Vector)
Let $C := (C_1, ..., C_k)$ be a clustering of the set $\{x_i\}_{i=1}^n$ and $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $\mu_{ij} \in \mathbb{R}$, $i, j \in \{1, ..., k\}$, the corresponding (a, Σ)-power diagram. Then

$$\vec{C}_a(x) = (C_{a_1}(x), ..., C_{a_k}(x)) \in \{0,1\}^k$$

with

$$C_{a_i}(x); = \prod_{j=1}^{k} 1_{\{a_{ij}^T x \leq \mu_{ij}\}},\ 1 \leq i \leq d$$

is called the cluster assignment vector of x.
The set $C_a(x) := \{i \in \{1, ..., k\} : C_{a_i}(x) = 1, C_{a_i}(x) \in \vec{C}_a(x)\}$ is called the cluster assignment of x.

This definition represents the assignment of points to clusters encoded in the cluster assignment vector. For future work, it could also be set from discrete $\{0,1\}$ to continuous $[0,1]$ which would correspond to fuzzy clustering (see [69] for an overview).
The assignment of a point or instance to a cluster in Definition 4.4.1 is not necessarily unique. In this case, which of course happens quite rare in practice, one possible solution is an assignment in lexicographical order.

Definition 4.4.2
Let $\vec{C}_a(x) := (C_{a_1}(x), ..., C_{a_k}(x)) \in \{0,1\}^k$ be a cluster assignment vector. Then the lexicographical cluster assignment vector is defined as

$$\vec{C}_a^{lex}(x) = (C_{a_1}^{lex}(x), ..., C_{a_k}^{lex}(x)) \in \{0,1\}^k$$

with

$$C_{a_i}^{lex}(x) := \arg \min_{j \in \{1,...,k\}} \{C_{a_j}(x) \in \vec{C}_a(x) : C_{a_j}(x) = 1\},\ 1 \leq i \leq d\ .$$

The index $C_a^{lex}(x) := \{i \in \{1,...,k\} : C_{a_i}^{lex}(x) = 1, C_{a_i}^{lex}(x) \in \vec{C}_a^{lex}(x)\}$ is called the lexicographical cluster assignment of x.

With the above definition of a lexicographical cluster assignment vector, every assignment of an instance to a cluster is unique. This feature is required for a distinct classification procedure. Unless stated otherwise, we assume the lexicographical cluster assignment of x when we speak of the cluster or the cluster number of x. The following definition of the cluster value corresponds to the scoring function. The scoring function defined in Definition 1.0.3 assigns a score to an instance x. The same concept holds for this new approach by addressing a cluster number to the instance x and the corresponding value of the cluster as score for x.

Definition 4.4.3 (Cluster Value)
Let $C := (C_1, ..., C_k)$ be a clustering of the set $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$. The function

$$f(x|C) = \sum_{i=1}^{k} f_i(C_i) \cdot C_{a_i}^{lex}(x)$$

with the lexicographical cluster assignment vector $\vec{C}_a^{lex}(x) = (C_{a_1}^{lex}(x), ..., C_{a_k}^{lex}(x))$ is called the cluster value of the clustering C for x.
Hereby, $f_i : \mathcal{C} \rightarrow \mathbb{R};\ C_i \rightarrow f_i(C_i)$ is called cluster value of the cluster C_i.

The function of the cluster value in Definition 4.4.3 is not specified explicitly. It is only defined as a real value assigned to a cluster depending, for example, on the labels of the training instances assigned to the cluster. Intuitive computations of cluster values will be given in the next section. Additionally, scoring functions of different clusterings could be combined and lead to the maximum and mean evaluation in Section 4.8 and 4.9. While the naive Bayes and the logistic regression are linear classifiers, the k-nearest neighbor technique has a piecewise linear decision boundary. This also holds for the geometric clustering approach as the convex polyhedral cell decompositions trivially leads to a piecewise linear decision boundary.
Based on the scoring function defined above, the following definition of the classifier completes the formal introduction of the new clustering approach as a binary classifier.

Definition 4.4.4 (Clustering Based Classifier)
Let $C := (C_1, ..., C_k)$ be a clustering of the set $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $a = (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $\mu_{ij} \in \mathbb{R},\ i, j \in \{1, ..., k\}$, the corresponding

(a, Σ)-power diagram. Let further be $f(x|C)$ the corresponding cluster value. Then

$$f_{clust}(x|C) = \sum_{i=1}^{k} f_i(C_i) \cdot C_{a_i}^{lex}(x)$$

is the scoring function, and

$$h_{clust}(x|C) := \begin{cases} 1, & f_{clust}(x|C) \geq \omega \\ 0, & otherwise \end{cases}$$

is the clustering based classifier of C with threshold value $\omega \in \mathbb{R}$.

This definition presents the clustering based classifier as a new individual concept of a supervised learning technique. Actually, it is a combination of supervised and unsupervised learning. On the one hand, it uses the unsupervised learning technique of a clustering approach. On the other hand, it has prior knowledge of the labels as it evaluates them for the transformation technique of Section 4.2 to estimate the conditional expected values. Additionally, the labels are used to train the scoring function, i.e., to compute the cluster values. Algorithm 8 shows the concept of the new clustering based classifier and gives an additional overview of the previous algorithmic results in this part.

The clustering based classifier creates a partition of the multidimensional metric space $\mathbb{R}^d$ into k convex polyhedral cells by computing a (a, Σ)-power diagram. The linear classifiers introduced in Chapter 2 also induce two linear cells via the decision boundary. Therefore, the new approach can be seen as a generalization in terms of dividing the data space into polyhedral decision regions or cells. In the special case of two clusters, the iterative clustering approach is trivially a linear classifier. While this shows a connection to the linear classifiers, there is also a similarity to the k-nearest neighbor as they both have piecewise linear decision boundaries.

Algorithm 8: Clustering based classification

```
Input: data set $S^{train} := \{(x_i^{train}, y_i^{train})\}_{i=1}^n$ as training set;
       data set $\{x_i^{test}\}_{i=1}^m$ as testing data to be classified;
Output: data set $S^{test} := \{(x_i^{test}, y_i^{test} =: h_{clust}(\tilde{x}_i^{test}))\}_{i=1}^m$;
    1. Apply Algorithm 2 on $S^{train} := \{(x_i, y_i)\}_{i=1}^n$ to update the training
       data set with the estimation of its conditional probabilities
       to get $\tilde{S}^{train} := \{(\tilde{x}_i^{train}, y_i^{train})\}_{i=1}^n$.
    2. Apply Algorithm 3 on $\{x_i^{test}\}_{i=1}^m$ to update the testing data set with
       the estimation of its conditional probabilities (based on $S^{train}$)
       to get $\{\tilde{x}_i^{test}\}_{i=1}^m$.
    3. Apply Algorithm 5 on $\{\tilde{x}_i^{train}\}_{i=1}^n$ to get clustering $C$.
    4. Apply Algorithm 6 on $C$ to get a $(a, \Sigma)$-power diagram $\mathcal{P}^{a,\Sigma}$.
    5. Apply Algorithm 7 to get the cluster assignments $\{C_a(\tilde{x}_i^{test})\}_{i=1}^m$
       based on $C$.
    6. Evaluate $h_{clust}$ and return $S^{test} := \{(\tilde{x}_i^{test}, h_{clust}(\tilde{x}_i^{test}|C))\}_{i=1}^m$.
```

Depending on the scoring function, there are many ways to define specific clustering based classifiers. In the next section, we introduce some intuitive scoring functions for the clustering based classification.

4.5. Scoring Functions for Clustering Based Classification

In the last sections, the geometric clustering was introduced as a binary classifier. The underlying scoring function, i.e., the underlying cluster value function

$f(x|C) = \sum_{i=1}^{k} f_i(C_i) \cdot C_{a_i}^{lex}(x)$ was not specified exactly. The intuitive concept is that the value of the scoring function for an instance x depends on the cluster the point is assigned to. This is similar to binary linear classifiers (like naive Bayes or logistic regression) which decide via two cells which of the two labels $\{0, 1\}$ is assigned. The clustering approach depends on the transformed data set of Section 4.2 and the resulting similarity expressed by geometrical proximity. Additionally, estimating the probabilities

$$p := P(Y = 1) = 1 - P(Y = 0)$$

of the Bernoulli distributed random variable is trivially equivalent to estimating its mean. An underlying functional relationship between the output labels Y and the input values $X = (X_1, ..., X_d)$ is therefore not needed explicitly. As mentioned in Section 4.4, the clustering based classification assumes not one underlying sample space Ω but k underlying sample spaces $\Omega_1, ...\Omega_k$. Each identified cluster C_i refers to its own underlying sample space. Furthermore, every sample space Ω refers to another set of random variables $X_{C_i} = (X_{C_i}^1, ..., X_{C_i}^d)$ and Y_{C_i}. Therefore, the computation of a cluster value $f_i(C_i)$ is equivalent to the estimation of

$$p(C_i) := P(Y_{C_i} = 1) = 1 - P_{C_i}(Y_{C_i} = 0)$$

for each cluster C_i.

Remark 4.5.1 shows an intuitive estimator for the cluster values calculated by the average number of positive labels. As the output random variable Y_{C_i} is Bernoulli distributed, the intuitive scoring function is also called Bernoulli scoring function. It is equivalent to a maximum likelihood estimation of the expected value.

Remark 4.5.1 (MLE for the Bernoulli Scoring Function)
Let Y_{C_i} be a Bernoulli distributed random of the labels of cluster C_i. Then the cluster values of the clustering based Bernoulli scoring function $f^{be}(x|C)$ are given by

$$f_i^{be}(C_i) = \frac{\sum_{i=1}^{n} 1_{\{C^{lex}(x_i)=i \wedge y_i=1\}}(x_i, y_i)}{\sum_{i=1}^{n} 1_{\{C^{lex}(x_i)=i\}}(x_i)},\ i \in \{1, ..., k\}\ .$$

As $f_i^{be}(C_i)$ is a mean estimation, it is the maximum likelihood estimator for the expected value of Y_{C_i}.

This remark represents the intuitive scoring approach as every cluster gets its score depending on the relative frequency of the labels of the training set. It also shows that this intuitive estimator is the best linear unbiased estimator (BLUE) as it is the arithmetical mean as estimation for the expected value of a random variable.

As introduced in Chapter 1, an often used alternative to the maximum likelihood estimation is the maximum a posteriori estimation. With prior information, the MAP estimation is often preferred. With the underlying parameter belonging to a Binomial or Bernoulli distributed random variable, a common distribution for the parameter itself is the beta distribution $Beta(\alpha, \beta)$ with the two positive shape parameters $\alpha, \beta > 0$ (see for example [12]).

Remark 4.5.2 (MAP for the Bernoulli Scoring Function)
Let Y_{C_i} be a Bernoulli distributed random variable of the labels of cluster C_i. Then the maximum a posteriori estimator for the mean of Y_{C_i} and the cluster value $f_i^{be}(C_i)$ of the Bernoulli scoring function $f^{be}(x|C; (\alpha; \beta))$ are given by

$$f_i^{be}(C_i; (\alpha_i; \beta_i)) = \frac{\sum_{i=1}^{n} 1_{\{C^{lex}(x_i)=i \wedge y_i=1\}}(x_i, y_i) + \alpha_i - 1}{\sum_{i=1}^{n} 1_{\{C^{lex}(x_i)=i\}}(x_i) + \beta_i + \alpha_i - 2},\ i \in \{1, ..., k\},$$

with $\alpha_i, \beta_i > 0$.
α_i *and* β_i *are the parameters of the corresponding Beta distribution with the parameter* $p(C_i) \sim Beta(\alpha_i, \beta_i)$ *of the underlying Bernoulli distribution of* Y_{C_i}.

In the case of $\alpha = \beta = 1$ this leads to the case of 'no prior information', i.e., every value $p(C_i) \in [0, 1]$ has the same probability. α and β can be interpreted as pseudo counts of 1 and 0, respectively. As further possible scoring functions the cluster value could consist of the cluster centers c_i or for example their closest neighbor in the training data. Additionally, adopting approaches like the logistic regression or the naive Bayes to a cluster could also be the fundamental of a scoring function.
Besides the calculation of the scoring function, another possibility of enhancing the classification results is the combination of different scoring functions. This is possible because of the initial target direction that could lead to different iterative clustering results and therefore slightly different predictions. It will be explained and applied in the Sections 4.8 and 4.9.

4.6. Weighting the Data

In addition to the scoring function, variable and feature selection is an important part of data mining (see [14] and [45] for an overview). It is also important in the clustering based classification.
While the calculation of the cluster values and the scoring function is similar to classic classification techniques, the now introduced weighting of the data refers to the geometrical interpretation of the clustering. The reason is the replacement of the original data by their conditional expected values.
The cluster should represent typical groups of similar instances. In the next step, the information is used to predict the corresponding binary labels. Without any further adjustment, the d input values are treated equally. Assuming

an unknown functional relationship between the input variables $X_1, .., X_d$ and Y leads intuitively to a weighting procedure of the input variables. Although this seems to conflict with the assumption in previous sections (e.g. every cluster originates from its own sample space), practical results show improved accuracy for weighted data compared to unweighted data.

Especially the transformation technique introduced in Section 4.2 allows linear weighting based on the conditional expected values to estimate the conditional expected value $E(Y|X_1, ..., X_d)$ by

$$y_i = \sum_{i=1}^{d} \beta_i x_i + \epsilon = \sum_{i=1}^{d} \beta_i E(Y|X_i) + \epsilon \ ,$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \ .$$

The computation of the weighting can be done, for example, by an ordinary least squares regression. Algorithm 8 of the clustering based classifier is complemented with a weighting procedure after the transformation step. The enhanced version is presented as Algorithm 9.

The results tend to be better on a weighted data set compared to an unweighted data set. Therefore, Part IV includes results for the weighted and unweighted case to compare the performance. In the context of this chapter, the logistic regression introduced as a classifier in Section 2.2 could also be used to model or estimate the functional relationship between the input values and the binary labels to measure the influence of each input variable (see for example [40] or [58]).

In addition to the functional weighting models like the linear weighting model introduced before or the mentioned logistic regression, there are numerous other techniques to weight a data set or to select or rank variables based on their importance (see [45] for an overview). This could be an interesting aspect of future work.

```
Algorithm 9: Clustering based classification with weighting procedure
Input: data set $S^{train} := \{(x_i^{train}, y_i^{train})\}_{i=1}^n$ as training set;
       data set $\{x_i^{test}\}_{i=1}^m$ as testing data to be classified;
Output: data set $S^{test} := \{(x_i^{test}, y_i^{test} =: h_{clust}(\tilde{x}_i^{test}))\}_{i=1}^m$;
    1. Apply Algorithm 2 on $S^{train} := \{(x_i, y_i)\}_{i=1}^n$ to update the training
       data set with the estimation of its conditional probabilities
       to get $\tilde{S}^{train} := \{(\tilde{x}_i^{train}, y_i^{train})\}_{i=1}^n$.
    2. Apply a weighting procedure on $\{(\tilde{x}_i^{train}, y_i^{train})\}_{i=1}^n$.
    3. Apply Algorithm 3 on $\{x_i^{test}\}_{i=1}^m$ to update the testing data set with
       the estimation of its conditional probabilities (based on $S^{train}$)
       to get $\{\tilde{x}_i^{test}\}_{i=1}^m$.
    4. Apply the weighting of step 2 on set $\{\tilde{x}_i^{test}\}_{i=1}^m$.
    5. Apply Algorithm 5 on $\{\tilde{x}_i^{train}\}_{i=1}^n$ to get clustering $C$.
    6. Apply Algorithm 6 on $C$ to get a $(a, \Sigma)$-power diagram $\mathcal{P}^{a,\Sigma}$.
    7. Apply Algorithm 7 to get the cluster assignments $\{C_a(\tilde{x}_i^{test})\}_{i=1}^m$
       based on $C$.
    8. Evaluate $h_{clust}$ and return $S^{test} := \{(\tilde{x}_i^{test}, h_{clust}(\tilde{x}_i^{test}|C))\}_{i=1}^m$.
```

4.7. Adjusting the Clustering Parameters

Due to the two stage nature of the iterative clustering approach there are two main classes of adjustable parameters. In addition to classification related parameters like the scoring function or the weighting of the training data, the introduced classifier has its specific clustering parameters introduced in Section 4.1.

The first important clustering parameter is the given number of clusters. There have been a lot of different approaches to determine the optimal cluster number (see [80], [84], [67] and [44] for details). A common method is to evaluate parameter settings for the training set and then use the best result for classification.

The evaluation of different cluster numbers on the training set is done by the k-fold cross-validation technique already introduced in Section 3.3. For a number l of possible cluster numbers $k \in \{k_1, ..., k_l\}$ the k-fold cross-validation is performed on the training set and the cluster number with the highest chosen accuracy measure is used (see Algorithm 1).

Besides the number of the clusters, the clustering based classifier also allows to set lower and upper bounds for each cluster. Similar to the maximum a posteriori estimation, where knowledge of a a posterior distribution could be taken into account, lower and upper bounds allow the use of prior information of a minimum cluster or group cardinality. In the estimation of a probability, i.e., mean of a Bernoulli distributed random variable, the variance of the estimator f_i^{be} introduced in Remark 4.5.1 is decreasing with an increasing underlying sample size. Therefore, it could be useful to secure a minimum number of instances in each cluster to achieve a good estimation. Of course, if the bounds are chosen too tight, it leads to an infeasible clustering problem, as there exists no separable clustering.

Practical results in the next part of this work show that strict bounds are not needed to achieve a good classification accuracy. The reason is the underlying cluster sum assignment which tend to 'fill' the clusters equally without strict lower bounds.

Another adjustable clustering parameter is the approximation degree of the Euclidean unit ball by the number of initial linear target directions. This parameter, the number of directions, is primarily performance-driven which means the more directions the better. It is highly related to the enhanced scoring functions introduced in the next Sections 4.8 and 4.9.

4.8. Maximum Clustering Based Classifier

As described in the previous sections, the clustering based classifier creates with Algorithm 5 a cell decomposition of the data space with respect to a given initial vector $a = (a_1^T, ..., a_k^T)^T \in R^{d \cdot k}$. The clustering is a result of the iterative, converging sequence described in Section 4.3.

In the following, we extend this approach to the case of more than one initial site vector, i.e., a set $\mathcal{A} = \{a_1, ..., a_l\} \subset \mathbb{R}^{d \cdot k}$ of initial vectors for a number of clusters $k \in \mathbb{N}$ with $a_i = (a_{i1}^T, ..., a_{ik}^T)^T \in \mathbb{R}^{d \cdot k}$.

In this setting, there exist different resulting cell decompositions of the data space $\mathbb{R}^d$. The performance of each cell decomposition has to be linked to a representative value of the clustering. This value is then evaluated to select the best clustering with respect to the performance measure, mostly the classification accuracy.

The random target direction is chosen with respect to the value

$$\sum_{i=1}^{k} \|s_i^*\| \,, \tag{4.9}$$

which is the resulting objective value of the sequence in Section 4.3. The value (4.9) is highly correlated with the quality of the prediction, measured by the classification accuracy. Therefore, it represents a good performance measure for the quality of prediction.

With the information of more than one clustering, the high correlation between expression (4.9) and the classification accuracy seen on empirical data leads to the following classifier. It selects the best clustering with respect to (4.9).

Definition 4.8.1 (Maximum Clustering Based Classifier)
Let $\mathcal{C} = \{C_i\}_{i=1}^{m}$ with $C_i = (C_{i1}, ..., C_{ik})$ be a set of clusterings of the set $X \subset \mathbb{R}^d$ and $f_{clust}(x|C_i)$ the corresponding scoring function of the clustering based classifier $h_{clust}(x|C_i)$. Then the maximum scoring function is defined by

$$f_{clust}^{\max}(x) = \text{argmax}_{\{f_{clust}(x|C):C\in\mathcal{C}\}} \sum_{i=1}^{k} \|s_i^*\| \, .$$

The scoring function $f_{clust}^{\max}$ implies the classifier

$$h_{clust}^{\max}(x|\mathcal{C}) = \begin{cases} 1, & f_{clust}^{\max}(x) \geq \omega \\ 0, & otherwise \end{cases}$$

as the maximum clustering based classifier of $\mathcal{C}$ with threshold value $\omega \in \mathbb{R}$.

As the prediction result depends on the initial target vector, another approach can be used that does not rely on a single clustering but a set of different cell decompositions. While the definition above picks one single scoring function, the approach introduced in the next section composes a scoring function by the mean of all 'single' scoring functions. The assumed disadvantage of using only one clustering for classification leads to an advantage on the other side. The selection of a specific clustering allows the statistical testing approach introduced in Section 4.10 to be used with $h_{clust}^{\max}$.

4.9. Mean Clustering Based Classifier

In Section 4.1, the clustering based classification approach has been interpreted as an approximation of a norm maximization problem. The underlying norm is piecewise approximated by linear functions. These functions are, for example, chosen randomly or determined by a given set. After the initial target direction, the clustering based classifier iterates with the standardized cluster sums as new sites and converges to a clustering. Like the similar k-means approach for the clustering problem (see [35] for details), the results of the new clustering based classifier depend on initial values, for example, the initial target direction and the number of clusters. Combining the results achieved with several initial directions leads to an extended scoring function, the mean clustering based classifier.

Like the maximum evaluation technique described in Section 4.8, the mean evaluation technique uses the information of different clusterings. Every clustering is optimal with respect to the given sites and leads to a cell decomposition, i.e., power diagram.

Motivated by our empirical results, the now introduced mean evaluation technique shows a good performance especially with an increasing number of clusters. As the number of clusters is increased, the number of points 'remaining' for each cluster decreases. Therefore, the variance of the estimation of the cluster values tends to grow, too. Using the information of different clusterings compensates this effect, which is what we see in empirical results.

Therefore, we define the mean clustering based classifier, which uses the information of more then one clustering for each data point.

Definition 4.9.1 (Mean Clustering Based Classifier)

Let $\mathcal{C} = \{C_i\}_{i=1}^{m}$ *with* $C_i = (C_{i1}, ..., C_{ik})$ *be a set of clusterings of the set* $X \subset \mathbb{R}^d$

and $f_{clust}(x|C_i)$ *the scoring function of the clustering based classifier. Then the mean scoring function is defined by*

$$f_{clust}^{mean}(x) = \frac{1}{m} \sum_{i=1}^{m} f_{clust}(x|C_i) \ .$$

The scoring function f_{clust}^{mean} *implies the classifier*

$$h_{clust}^{mean}(x|\mathcal{C}) = \begin{cases} 1, & f_{clust}^{mean}(x) \geq \omega \\ 0, & otherwise \end{cases}$$

as the mean clustering based classifier of $\mathcal{C}$ *with threshold value* $\omega \in \mathbb{R}$*.*

The mean clustering based classifier can be interpreted as a new classification approach, consisting of iterative clustering classifiers with m different initial target directions. This leads to stable and good prediction results shown in Part IV on empirical data sets.

In comparison to the maximum based classification, the mean evaluation technique uses all the information a set of clusterings $\mathcal{C}$ provides. This leads to a reliable estimation with low variance.

In reference to Algorithm 8 or 9, the transformation and weighting procedures do not have to be executed for a new initial target direction. An algorithm with the described mean or maximum evaluation would repeat the steps 3 to 6 and 5 to 8 in Algorithm 8 and 9, respectively.

The introduced techniques enhance the clustering based classification. In Part IV of this thesis, we show the excellent performance of these classifiers on real-world data.

4.10. Clustering Based Test of Hypotheses

In Section 3.2, we introduced confidence intervals as a statistical approach to estimate the accuracy of a classifier. Based on confidence intervals, hypothesis tests can be used to compare classifiers (see [68]). In this section, we introduce a hypothesis test procedure relying on the partition of the data generated by the clustering. It provides a generalization of the usual evaluation of hypotheses as it consists of tests for each cluster C_i of a clustering $C = (C_1, ..., C_k)$, based on the predictive values generated by the scoring function.

Additionally, it leads to a reliability measure for a prediction and is joint work with Brieden and Hinnenthal (see working paper [25]).

The new clustering approach allows a decomposition of the data set into several underlying sample spaces. Each cluster represents a sample space generating its own prediction for the instances assigned to it.

The underlying probabilistic concept introduced in Section 4.4 assumes that the realizations originate from k sample spaces Ω_i, $i \in \{1, .., k\}$. Therefore, every cluster C_i refers to its own joint probability distribution $\mathcal{D}_Z(C_i) = \mathcal{D}_X(C_i) \times \mathcal{D}_Y(C_i)$ that allows a cluster based statistical test.

In general, a statistical hypothesis test is structured as illustrated in Figure 4.7 and consists of several stages. At first, the hypothesis must be set which corresponds to an underlying assumption, in our case the designated cluster value. This leads to the so-called null hypothesis $\mathcal{H}_0$ and the opposite, the alternative hypothesis $\mathcal{H}_1$. In the second step, the level of significance α should be set. It determines the probability that $\mathcal{H}_0$ is rejected, even though it is true. Furthermore, the level of significance sets the region for rejection of the calculated realization t of a test statistic T based on a test sample.

With a decreasing α, the rejection of $\mathcal{H}_0$ is getting harder. Therefore, if $\mathcal{H}_0$ is rejected with a low level of significance, the decision is *most probably* true

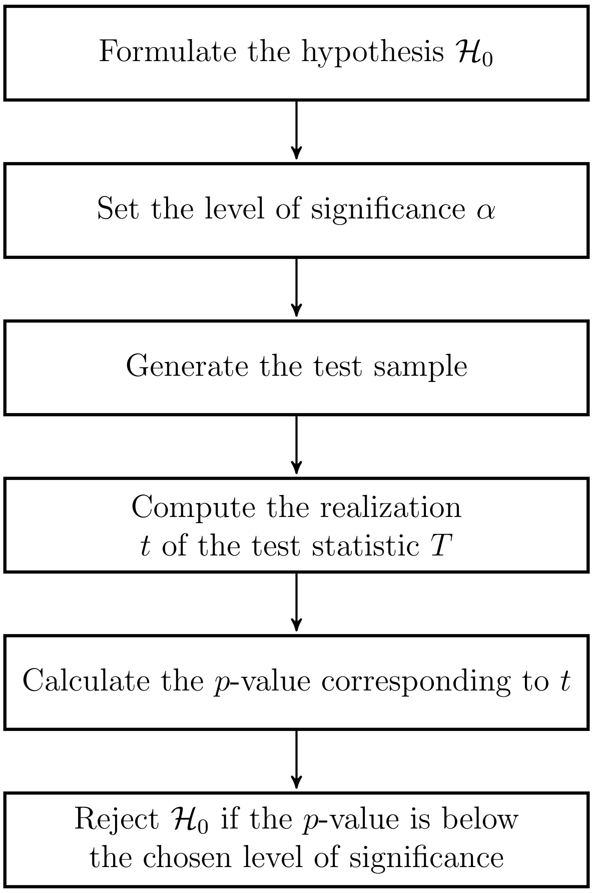
Formulate the hypothesis $\mathcal{H}_0$
Set the level of significance α
Generate the test sample
Compute the realization t of the test statistic T
Calculate the p-value corresponding to t
Reject $\mathcal{H}_0$ if the p-value is below the chosen level of significance

Figure 4.7.: Hypothesis testing procedure.

as the probability for a wrong decision (under the condition that $\mathcal{H}_0$ is true) is only α.

As only the described *type*1 error (rejecting $\mathcal{H}_0$ although $\mathcal{H}_0$ is actually correct) is controlled, accepting $\mathcal{H}_0$ has less explanatory power. For more details of statistical hypothesis tests see for example [39] or [88].

Every statistical test is based on certain statistical assumptions for the test sample. Common examples are the statistical independence or that the data originate from the same distribution.

As described in previous sections, a probabilistic assumption of the clustering based classification is that the sample set originates from k sample spaces Ω_i and every cluster C_i corresponds to a joint probability distribution $\mathcal{D}_Z(C_i) = \mathcal{D}_X(C_i) \times \mathcal{D}_Y(C_i)$.

In this new concept we now formulate hypotheses based on each cluster. As we are interested in the quality of a prediction, the hypotheses for each cluster C_i are related to their cluster value $f^{train}(C_i)$.

The main assumption in this approach states that the true (overall) prediction value

$$p_i := P(Y_{C_i} = 1) = 1 - P(Y_{C_i} = 0)$$

of a cluster C_i (referring to a Bernoulli distribution as introduced in Section 4.4 and 4.5) holds the following inequalities:

$$f_i^{train}(C_i) \cdot \delta_i^l \leq p_i \leq f_i^{train}(C_i) \cdot \delta_i^u \ . \tag{4.10}$$

These cluster values $f_i^{train}(C_i)$ are computed from the training data S^{train} and adjusted by a lower and an upper parameter δ_i^l and δ_i^u, respectively. The evaluation on the cluster values $f_i^{train}(C_i)$ of the training set S^{train} is calculated by the Bernoulli scoring function introduced in Remark 4.5.1:

$$f_i^{train}(C_i) = \frac{\sum_{i=1}^{n} 1_{\{C(x_i^{train})=i \wedge y_i^{train}=l\}}}{\sum_{i=1}^{n} 1_{\{C(x_i^{train})=i\}}}, i \in \{1, ..., k\} \ .$$

Like mentioned before, not accepting but rejecting $\mathcal{H}_0$ is the goal of a testing procedure. Therefore, for each cluster C_i two null hypotheses $\mathcal{H}_0^l(C_i)$ and $\mathcal{H}_0^u(C_i)$ are formulated. This leads to the following setting as the basic testing scheme for a cluster C_i:

$$\mathcal{H}_0^l(C_i) : p_i \leq f_i^{train}(C_i) \cdot \delta_i^l =: \hat{p}_i^l \tag{4.11}$$

$$\mathcal{H}_1^l(C_i) : p_i > f_i^{train}(C_i) \cdot \delta_i^l =: \hat{p}_i^l \tag{4.12}$$

$$\mathcal{H}_0^u(C_i) : p_i \geq f_i^{train}(C_i) \cdot \delta_i^u =: \hat{p}_i^u \tag{4.13}$$

$$\mathcal{H}_1^u(C_i) : p_i < f_i^{train}(C_i) \cdot \delta_i^u =: \hat{p}_i^u \ . \tag{4.14}$$

The null hypothesis (4.11) represents the assumption that the true value p_i is at most $\hat{p}_i^l$ while (4.13) corresponds to the assumption that the true cluster value p_i of the cluster C_i is at least $\hat{p}_i^u$.

Definition 4.10.1 (Clustering Based Set of Hypotheses)
Let $C := (C_1, ..., C_k)$ be a clustering. The set of lower and upper (null) hypotheses $\{(\mathcal{H}_0^l(C_i), \mathcal{H}_0^u(C_i))\}_{i=1}^k$ derived from (4.11) to (4.14) is called clustering based set of (null) hypotheses and $\{(\mathcal{H}_1^l(C_i), \mathcal{H}_1^u(C_i))\}_{i=1}^k$ is the corresponding set of alternative hypotheses.

The adjusting parameters (δ_i^l, δ_i^u), $i \in \{1, ..., k\}$, of the underlying hypotheses (4.11) to (4.14) can be modeled in several ways. A possibility is the dependence on the 'neighbor' values if the cluster values are sorted. This intuitive setting is introduced in the next definition.

Definition 4.10.2 (Comparing Distance)
Let $C := (C_1, ..., C_k)$ be a clustering with cluster values

$$0 =: f_0^{train}(C_0) \leq f_1^{train}(C_1) \leq f_2^{train}(C_2) \leq ... \leq f_k^{train}(C_k) \leq f_{k+1}^{train}(C_{k+1}) := 1$$

and $\{(\mathcal{H}_0^l(C_i), \mathcal{H}_0^u(C_i)\}_{i=1}^k$ the set of (null) hypotheses for the clustering.

Then the set $\{(\delta_i^l, \delta_i^u)\}_{i=1}^k \subset \mathbb{R} \times \mathbb{R}$ *with*

$$\delta_i^l = \lambda_i - (1-\lambda_i)\left(\frac{f_i^{train}(C_i)}{f_i^{train}(C_{i-1})}\right),\ i = 1, ..., k$$

and

$$\delta_i^u = \lambda_i + (1-\lambda_i)\left(\frac{f_i^{train}(C_{i+1})}{f_i^{train}(C_i)}\right),\ i = 1, ..., k$$

with $\lambda_i \in [0,1]$ *is called comparing distance parameter set. The corresponding set of null hypotheses* $\{(\mathcal{H}_0^l(C_i), \mathcal{H}_0^u(C_i))\}_{i=1}^k$ *is then called comparing distance hypotheses set.*

The comparing distance parameter set corresponds to convex combinations of the sorted cluster values. Of course, the sorting is just a permutation of the indeces and therefore no restriction.

The underlying cluster value is computed in the training step of the classification procedure, for example, as shown in Remark 4.5.1. The reliability of this estimation is examined in the next step by the evaluation of the labeled testing data S^{test}. This corresponds to step 4 in the usual statistical test procedure described in Figure 4.7. It is the computation of the realization t of a test statistic T derived from the value $f_i^{test}(C_i)$ for the cluster C_i. $f_i^{test}(C_i)$ is the cluster value of the testing data for C_i and it is computed by the classified testing set.

Definition 4.10.3 (Cluster Value for the Testing Data)
Let $C := (C_1, ..., C_k)$ *be a clustering and* $S^{test} := \{(x_i^{test}, y_i^{test})\}_{i=1}^m \subset \mathbb{R}^d \times \{0,1\}$ *a testing data set with* $y_i^{test} = h_{clust}(x_i^{test}|C), \forall i \in \{1, ..., m\}$, *and* $\{C^{lex}(x_i^{test})\}_{i=1}^m$ *the set of corresponding (lexicographical) cluster assignments. Then*

$$f_i^{test}(C_i) = \frac{\sum_{i=1}^m 1_{\{C^{lex}(x_i^{test})=i \wedge y_i^{test}=1\}}}{\sum_{i=1}^m 1_{\{C^{lex}(x_i^{test})=i\}}},\ i \in \{1, ..., k\},$$

is called the (Bernoulli) cluster value of C_i for the testing data.

The test statistic $T^l(C_i)$ and $T^u(C_i)$ are the standardized random variables resulting from the estimation of p_i. Their realizations $t(C_i)^l$ and $t(C_i)^u$ are computed with the estimations f^{test} for each cluster.

Definition 4.10.4 (Cluster Based Test Statistic)

Let $C := (C_1, ..., C_k)$ be a clustering, $\{(\mathcal{H}_0^l(C_i), \mathcal{H}_0^u(C_i))\}_{i=1}^k$ the set of hypotheses and $S^{test} := \{(x_i^{test}, y_i^{test})\}_{i=1}^m \subset R^d \times \{0,1\}$ a classified testing data set and $\{C^{lex}(x_i^{test})\}_{i=1}^m$ the set of corresponding (lexicographical) cluster assignments. Then

$$t^l(C_i) = \frac{f^{test}(C_i) - \hat{p}_i^l}{\sqrt{\frac{\hat{p}_i^l(1-\hat{p}_i^l)}{n^{test}(C_i)}}}$$

and

$$t^u(C_i) = \frac{f^{test}(C_i) - \hat{p}_i^u}{\sqrt{\frac{\hat{p}_i^u(1-\hat{p}_i^u)}{n^{test}(C_i)}}}$$

with

$$n^{test}(C_i) := \sum_{i=1}^{m} 1_{\{C^{lex}(x_i^{test})=i\}}$$

are the realizations for the lower and upper cluster based (standardized) test statistic $T^l(C_i)$ and $T^u(C_i)$ for cluster C_i. $\{(T^l(C_i), T^u(C_i))\}_{i=1}^k$ is the corresponding clustering based set of test statistics.

Remark 4.10.5

Let $C := (C_1, ..., C_k)$ be a clustering, $\{(\mathcal{H}_0^l(C_i), \mathcal{H}_0^u(C_i))\}_{i=1}^k$ the clustering based set of hypotheses and $\{(T^l(C_i), T^u(C_i))\}_{i=1}^k$ the corresponding set of test statistics. Then the test statistics are approximately normal distributed with

$$T^l(C_i) \sim \mathcal{N}\left(\hat{p}_i^l, \sqrt{\frac{\hat{p}_i^l(1-\hat{p}_i^l)}{n^{test}(C_i)}}\right)$$

and

$$T^u(C_i) \sim \mathcal{N}\left(\hat{p}_i^u, \sqrt{\frac{\hat{p}_i^u(1-\hat{p}_i^u)}{n^{test}(C_i)}}\right)$$

because of the central limit theorem.

Because of the relationship to the normal distribution we can derive the p-values (pv) as their quantiles. In the next step, these probabilities can be applied as a measure of reliability for the predicted cluster values.

Definition 4.10.6 (p-Values for the Clustering Based Classifier)
Let $C := (C_1, ..., C_k)$ be a clustering, $\{(\mathcal{H}_0^l(C_i), \mathcal{H}_0^u(C_i))\}_{i=1}^k$ the clustering based set of hypotheses and $\{(T^l(C_i), T^u(C_i))\}_{i=1}^k$ the corresponding set of test statistics.
Then $\{(pv_i^l, pv_i^u)\}_{i=1}^k$ is the set of corresponding p-values with

$$pv_i^l := P(T^l(C_i) > t^l(C_i)|\mathcal{H}_0^l(C_i))$$

and

$$pv_i^u := P(T^u(C_i) < t^u(C_i)|\mathcal{H}_0^u(C_i))$$

for the clustering based set of hypotheses.

The p-values can be interpreted by comparing them with given lower and upper levels of significance (α_i^l, α_i^u) for a cluster C_i. They represent the probability of obtaining a value for the test statistics of a cluster C_i that is at least as extreme as the one observed under the assumption that the hypotheses are correct. The smaller the p-value, the stronger the presumption against $\mathcal{H}_0$. So the pair of the lower and the upper p-values of a cluster indicates how 'trustable' a predicted cluster value $f_i^{train}(C_i)$ is.

Algorithm 10: Clustering Based Test of Hypotheses

```
Input: data set $S^{train} := \{(x_i^{train}, y_i^{train})\}_{i=1}^{n}$ as training set;
       classified data set $S^{test} := \{(x_i^{test}, y_i^{test})\}_{i=1}^{m}$ as testing set;
       corresponding clustering $C = (C_1, ..., C_k)$;
Output: set of $p$-values $\{(pv_i^l, pv_i^u)\}_{i=1}^{k}$;
   1. Calculate the cluster value $f^{train}(C_i)$ for each cluster $C_i$ in
      $C = (C_1, ..., C_k)$ based on the training data.
   2. Calculate the cluster value $f^{test}(C_i)$ for each cluster $C_i$ in
      $C = (C_1, ..., C_k)$ based on the testing data.
   3. Formulate the hypotheses $\mathcal{H}_0^{l_i}$ and $\mathcal{H}_0^{u_i}$ by computing $\hat{p}_i^l$ and $\hat{p}_i^u$.
   4. Compute the test statistics $t^{l_i}(C_i)$ and $t^{u_i}(C_i)$ for each cluster $C_i$.
   5. Compute the $p$-values $p_i^l$ and $p_i^u$ for each cluster $C_i$.
```

This new clustering based test of hypotheses in Algorithm 10 allows us to compute a probability for trusting the predicted values for an instance x. It enhances the fundamental concept of predicting a label by an additional measure of trust. Especially when results of different cluster numbers are compared, a tradeoff between accuracy and reliability is the upcoming effect. This can be seen in the next Part of this work when the new approach is demonstrated on real-world data.

Part IV.

Empirical Results

In the last part, the clustering approach for classification was introduced and motivated while this part focuses on practical results. The techniques introduced before are demonstrated and evaluated on real-world data.

The underlying technical implementation of the clustering technique, i.e., the corresponding Algorithms 8 and 9 was done in *JAVA* with the optimization problems of Chapter 4 computed by the optimization tool *Xpress by FICO*. All calculations have been performed on a personal computer with an Intel CORE i7 Q740 @ 1.73 GHz and 8GB RAM.

The first set is related to credit scoring. The German Credit data set consists of 1000 instances and the task is to classify whether a person has a good or a bad credit rating based on 20 input variables. It has no given separation into training and testing sets. Therefore a 10-fold cross-validation was performed to evaluate the clustering based classifier.

The second data set is the Census Income data set also known as the Adult data set. The task is to classify whether a person has an income above or below 50.000 \$. In contrast to the German Credit data set, the Census Income data set consists of more data points and less variables.

The results based on the German Credit data set show a first application of the clustering based classifier and its good performance. After these first results, the application of the clustering based classification on the Census Income data set underlines its good performance. Additionally, the set allows the demonstration and the application of the new clustering based test of hypotheses.

In each chapter, we describe the underlying data and explain the classification task in the first step. In the second step, we present the pre-processing and the experimental setup including the different parameter scenarios. After that, we show the corresponding excellent results and compare the performance of the new introduced approach with common classifiers, if possible.

5. The German Credit Data Set

The German Credit data set is frequently used for evaluating predictive algorithms for binary classification problems. It consists of 1000 instances with 20 input variable. The goal of the German Credit data set is to predict whether a risk has a good or a bad credit rating. The data was donated to the *UCI machine learning repository* by *Professor Dr. Hans Hofmann* from the *Institut für Statistik und Ökonometrie* at the *University of Hamburg, Germany.*

It was also used by the StatLog project to compare different classifiers (see [59]) and can be retrieved from the UCI machine learning repository (see [7]). An important difference to the later evaluated Census Income data set is the missing segmentation into training and testing instances. Therefore, it is necessary to generate a set of segmentations and to evaluate the method on each of these training/testing samples. As shown in Chapter 3, there are common evaluation techniques like the k-fold cross-validation. It is used in this case to compare the results with other techniques like those introduced in the Sections 2.1 to 2.3.

Even if the results can not be compared completely because of the random selection, the application of the 10-fold cross-validation technique and of confidence intervals indicates the excellent performance based on evaluation measures like the classification accuracy.

The characteristics of the German Credit data set are a high number of input variables and a comparatively small number of 1000 instances. There are 20 different variables excluding the target variable. Seven of these input variables have a numerical level of scale while the other thirteen have qualitative values. An explicit overview of the different variables is given in Table 5.1. The target variable indicates whether the person is actually a good or bad credit risk and if the credit was a default or not. In the given data set, exactly 30 % of the persons have a bad credit rating (0/negative) and 70 % have a good credit rating (1/positive).

variable	scale	values
status of existing account	qualitative	4
duration in months	numerical	33
credit history	qualitative	5
purpose	qualitative	11
credit amount	numerical	921
savings account/bonds	qualitative	5
present employment since	qualitative	5
installment rate in % of disposable income	numerical	4
personal status and sex	qualitative	5
other debtors/guarantors	qualitative	3
present residence since	numerical	4
property	qualitative	4
age in years	numerical	53
other installment plans	qualitative	3
housing	qualitative	3
# of existing credits at this bank	numerical	4
job	qualitative	4
# of people being liable to provide maintenance for	numerical	2
telephone	qualitative	2
foreign worker	qualitative	2
good or bad credit rating	qualitative	2

Table 5.1.: Variables of the German Credit data set.

5.1. Experimental Pre-Processing and Setup

In this section, we give a detailed description how the data set was transformed, how variables have been selected and what other pre-processing steps have been made. Additionally, the underlying parameter settings for the used clustering based classification are introduced.

Pre-Processing

The high number of possible combinations of the existing values of the 20 input variables (~ 47520000 for the qualitative variables) lead to an extremely small ratio between the actual and the possible combinations.

Before the clustering procedure, the data set has been revised in different terms.

In the first step, 9 of the 20 features or variables have been eliminated due to their small influence on the prediction value. This was determined by Kim and Sohn in [57] and used in our analysis as the first pre-processing step. Table 5.2 gives an overview of the variables used, their level of scale and the number of their occurring values. While the Census Income data set has a given separation into training and testing sets, it is not the case for this data set. The k-fold cross-validation was introduced as validation technique in Section 3.3 and will be applied in this case. We set the number of folds of the cross-validation to 10. This leads to training sets S_i^{train} and testing sets S_i^{test} for $i \in \{1, ..., 10\}$. The 10 folds where chosen randomly with size 100 for every fold, leading to a size of $|S_i^{train}| = 900$ for each training set and $|S_i^{test}| = 100$ for each testing set.

For every combination $(S_i^{train}, S_i^{test})$ of training and testing data, we have to apply the clustering based classification.

variable	scale	values
status of existing account	qualitative	4
duration in months	numerical	5
credit history	qualitative	5
purpose	qualitative	11
credit amount	numerical	5
savings account/bonds	qualitative	5
present employment since	qualitative	5
installment rate in % of disposable income	numerical	4
personal status and sex	qualitative	5
other debtors/guarantors	qualitative	3
present residence since	numerical	4
property	qualitative	4
age in years	numerical	53
other installment plans	qualitative	3
housing	qualitative	3
# of existing credits at this bank	numerical	4
job	qualitative	4
# of people being liable to provide maintenance for	numerical	2
telephone	qualitative	2
foreign worker	qualitative	2
good or bad credit rating	qualitative	2

Table 5.2.: Variables in the adjusted data set with 1000 instances after preprocessing. Nine variables have been eliminated (red) and two discretized by 20%-quantiles (green).

Experimental Setup

The 10 training/testing combinations generated by the described 10-fold cross-validation represent the input data for the clustering based classification. The chosen classifier was the mean clustering based classifier h_{clust}^{mean} as presented in Section 4.9 for each combination $(S_i^{train}, S_i^{test})$, $i \in \{1, ...10\}$. The number of evaluated clusterings, i.e., the size of the underlying clustering set $\mathcal{C}$ was $m = 50$ for each couple of training and testing data $(S_i^{train}, S_i^{test})$.

The parameter setup for the clustering step consisted of a number of 30 clusters $(k = 30)$, lower bounds $l_i = 20$, $i = \{1, ..., k\}$, no upper bounds and no weighting procedure. The chosen cluster number was a result of an internal cross-validation of cluster sizes $\{10, 20, 30\}$ on the training data. This was also done for the chosen lower bounds.

5.2. Experimental Results

In the following, we present the results of the 10-fold cross-validation and the performance comparison based on the detailed results in [9] and [82]. As [9] and [82] do not include results for the training data, we cannot compare our classifier for the training sets but only for the prediction results of the testing data.

In the following tables, the values for the performance measures of Section 3.3 are listed depending on the underlying set (columns labeled with 1 to 10). While the mean values (ø) in Table 5.3 and 5.4 are self-explanatory, the last column refers to the standard deviation of the cross-validation with the underlying data set size n as explained in Section 3.3. The positive values of the target variable refer to a good loan and the opposite therefore represents a bad loan. In the first step, the results based on the training set in Table 5.3 are analyzed.

On the training data we see an average classification accuracy of 77.89 % in a range from 76.44 % to 79.00 % with an average standard deviation of 1.38 % and an overall cross-validation standard deviation of 0.44 %. The sensitivity ($SENS$) on the training data is the percentage of good loans correctly labeled. It has an average value of 88.95 % with a minimum value of 86.56 % in S_8^{train} and a maximum value of 90.69 % in S_2^{train}. Specificity ($SPEC$) is a measure of how well the model is classifying bad loans correctly. Its average value is 52.06 % in a range between 43.98 % and 53.93 %. The AUC value as aggregation measure has an average value of 82.25 %.

S_i^{train}	1	2	3	4	5	6	7	8	9	10	ø	overall / cv
TP	558	575	568	552	553	558	564	541	572	563	560.4	-
FP	125	149	132	132	116	131	130	128	128	123	129.4	-
FN	73	59	64	71	78	71	65	84	61	70	$69,6$	-
TN	144	117	136	145	153	140	141	147	139	144	140.6	-
ACC (%)	78.00	76.89	78.22	77.44	78.44	77.56	78.33	76.44	79.00	78.56	77.89	-
σ_{ACC} (%)	1.38	1.41	1.38	1.39	1.37	1.39	1.37	1.41	1.36	1.37	1.38	0.44
$l_{0.95}^{ACC}$ (%)	75.29	74.13	75.53	74.71	75.76	74.83	75.64	73.67	76.34	75.87	75.18	77.03
$u_{0.95}^{ACC}$ (%)	80.71	79.64	80.92	80.17	81.13	80.28	81.02	79.22	81.66	81.24	80.60	78.75
ERR (%)	22.00	23.11	21.78	22.56	21.56	22.44	21.67	23.56	21.00	21.44	22.11	-
$SENS$ (%)	88.43	90.69	89.87	88.60	87.64	88.71	89.67	86.56	90.36	88.94	88.95	-
$SPEC$ (%)	53.53	43.98	50.75	52.35	56.88	51.66	52.03	53.45	52.06	53.93	52.06	-
AUC (%)	81.88	82.02	81.91	82.58	82.66	82.84	82.41	81.36	82.85	82.00	82.25	-

Table 5.3.: Credit scoring results for the scenario $Credit_{un_i}^{10cv}$ (S^{train}) with $k = 30$ and $l_i = 20$.

The next results refer to the testing data. Therefore, they represent the predictive classification performance of the clustering approach. They are compared to the mentioned results found in [9] and [82].

We focus on the comparison with individual classifiers like the introduced techniques in Sections 2.1 to 2.3, not on combined approaches also listed in [82]. The reason is that our clustering based classifier should also be seen as a new individual classifier which, of course, can be combined with other individual classifiers. Table 5.5 shows the performance for 17 classifiers plus the clustering approach as first classifier (see [9]). Our clustering technique ($CLUST$) takes the first place out of the listed 18 techniques if we compare our classification accuracy of the testing data of 75.48 % with the values listed in Table 5.5. The listed accuracies excluding our clustering technique range from 59.0 % to 75.1 %. The naive Bayes (NB) and the logistic regression (LOG) introduced in Sections 2.1 and 2.2 achieve an accuracy of 72.2 % and 74.6 %, respectively. The also listed k-nearest neighbor classifier achieves only an accuracy of 70.7 % with $k = 10$ and 68.9 % with $k = 100$.

Compared with the values listed in the second Table 5.6, consisting of the classification accuracy and AUC values of [82] for individual classifiers, the clustering technique also takes the first place with its average classification accuracy of 75.48 % on the testing data out of 17 techniques including the clustering approach. The average sensitivity of our approach on the testing data is 88.11 %. The comparative results in Table 5.5 cover a range from 79.5 % to 100 %. High values like the best value of 100 % often correspond with bad values for the specificity which is, for example, 0.00 % in this case for the 100-nearest neighbor technique. The average specificity of the clustering technique for the analyzed German Credit data set is 46.37 %. Compared with the values listed in Table 5.5, ranging from 0.00 % to 52.4 %, our approach takes the fourth place. The AUC values range from 62.0 % to 78.7 % in Table

S_i^{test}	1	2	3	4	5	6	7	8	9	10	ø	overall / cv
TP	58	58	60	69	59	61	60	70	61	61	61.7	-
FP	18	20	19	12	13	15	17	9	24	15	16.2	-
FN	11	8	8	8	10	10	11	5	6	6	8.3	-
TN	13	14	13	11	18	14	12	16	9	17	13.7	-
ACC (%)	71.00	72.00	73.00	80.00	77.00	75.00	72.00	86.00	70.00	78.79	75.48	-
σ_{ACC} (%)	4.54	4.49	4.44	4.00	4.21	4.33	4.49	3.47	4.58	4.11	4.27	1.36
$l_{0.95}^{ACC}$ (%)	62.11	63.20	64.30	72.16	68.75	66.51	63.20	79.20	61.02	70.73	67.12	72.81
$u_{0.95}^{ACC}$ (%)	79.89	80.80	81.70	87.84	85.25	83.49	80.80	92.80	78.98	86.84	83.84	78.15
ERR (%)	29.00	28.00	27.00	20.00	23.00	25.00	28.00	14.00	30.00	21.21	24.52	-
$SENS$ (%)	84.06	87.88	88.24	89.61	85.51	85.92	84.51	93.33	91.04	91.04	88.11	-
$SPEC$ (%)	41.94	41.18	40.63	47.83	58.06	48.28	41.38	64.00	27.27	53.13	46.37	-
AUC (%)	75.32	75.47	77.27	75.64	78.33	70.98	71.30	86.29	67.16	79.69	75.74	-

Table 5.4.: Credit scoring results for the scenario $Credit_{un_i}^{10cv}$ (S^{test}) with $k = 30$ and $l_i = 20$.

technique	*ACC* (%)	*SENS* (%)	*SPEC* (%)	*SENS*+*SPEC* (%)	*AUC* (%)
CLUST	75.48	88.11	46.37	134.5	75.74
LDA	74.6	79.5	52.4	131.9	78.4
QDA	71.0	79.5	52.4	131.9	71.8
LOG	74.6	89.5	41.9	131.4	77.7
LP	71.9	92.6	26.7	119.3	76.3
RBF LS-SVM	74.3	96.5	25.7	122.2	77.4
Lin LS-SVM	73.7	91.3	35.2	126.5	78.4
RBF SVM	74.0	92.6	33.3	125.9	77.2
Lin SVM	71.0	95.6	17.1	112.7	76.6
NN	73.7	85.2	48.6	133.8	78.7
NB	72.2	87.8	38.1	125.9	77.2
TAN	72.5	87.3	40.0	127.3	78.3
C4.5	72.2	88.2	37.1	125.3	74.7
C4.5 rules	71.0	91.3	26.7	118.0	62.0
C4.5 dis	74.6	87.3	46.7	134.0	74.6
C4.5 rules dis	74.3	89.5	41.0	130.5	64.4
KNN10	70.7	94.8	18.1	112.9	70.2
KNN100	68.6	100	0.00	100.0	76.1

Table 5.5.: Comparison of different classification algorithms in [9].
List of abbreviations: CLUST - clustering; LDA - linear discriminant analysis; QDA - quadratic discriminant analysis; LOG - logistic regression; LP - linear programming; RBF LS-SVM - radial basis function with least-squares support vector machines; Lin LS-SVM - linear kernels with standard least-squares support vector machines; RBF SVM - radial basis function with standard support vector machines; Lin SVM - linear kernels with support vector machines; NN - neural networks; NB - naive bayes; TNB - tree augmented naive bayes; C4.5 - a decision tree based algorithm with two modifications: conversion of the unpruned tree into a rule set (rule) and the discretization of the continuous data (dis); KNN10 & KNN100 - k-nearest neighbor with k= 10 & 100

technique	*ACC* (%)	*AUC* (%)
CLUST	75.48	75.74
ANN	74.90	79.10
B-Net	73.10	76.40
CART	69.30	70.60
ELM	73.50	77.80
ELM-K	74.70	79.40
J4.8	71.70	73.40
KNN?	73.50	77.20
LDA	74.80	78.40
LOR	74.70	78.40
LOR-R	74.20	77.80
NB	74.00	77.70
NN RBF	72.70	76.20
QDA	58.60	67.40
Lin SVM	74.20	78.20
RBF SVM	75.30	79.90
VP	67.50	68.00

Table 5.6.: Comparison of individual classification algorithms in [82].
List of abbreviations: CLUST - clustering; ANN - artificial neural network; B-Net - bayesian network classifier; CART - classification and regression trees; ELM - extreme learning Machines; ELM - extreme learning machines with kernels; J4.8 - open source implementation of the C4.5 algorithm; KNN - k-nearest neighbor; LDA - linear discriminant analysis; LOG - logistic regression; LOG-R - logistic regression with a L_1 regularizer; NB - naive bayes; NN RBF - neural networks with radial basis function; QDA - quadratic discriminant analysis; RBF SVM - radial basis function with support vector machines; Lin SVM - linear kernels with support vector machines; VP - voted perceptron

5.5 and 67.4 % to 79.9 % in Table 5.6 with the value for the clustering based classifier of 75.74 % positioned in the middle-field.

These first results for our clustering based classification of the German Credit data set consist of a solid specificity paired with a good sensitivity and an excellent overall classification accuracy compared with common individual classifiers listed in the Tables 5.5 and 5.6.

Of course, because of the cross validation setting, the true values can differ slightly but without the exact training/testing data sets there is always a small bias and the results here should be interpreted together with the bounds of the confidence intervals for the classification accuracy $[l_{0.95}^{ACC}, u_{0.95}^{ACC}]$ (see for example Table 5.4).

Additionally, Baesens assumes in [9] that most credit scoring data sets are only weekly nonlinear which results in relatively good performance of, for example, the logistic regression introduced in Section 2.2 with a classification accuracy of 74.6 %. Therefore, in a more complex data structure like in the next data set, our clustering based classifier performs even better.

6. The Census Income Data Set

The Census Income data set is a well known data set to test the quality of a prediction generated by data mining techniques for binary classification. It is also known as Adult data set which will be used synonymously in the following. Originally extracted by Barry Becker from the Census database of the year 1994 it is cited in over 50 publications and is therefore one of the most used data sets for the comparison of predictive algorithms.

It was first cited 1996 in [61] by Ron Kohavi and consists of 48842 instances, 45222 if instances with unknown values are removed. The number of input variables is originally 14, 7 being discrete and 7 continuous. A detailed overview is given in Table 6.1. The data set can be retrieved from the UCI machine learning repository (see [8]). Looking for duplicated entries, only 50 of these 45222 instances are duplicated. Even if we look just at the discrete (or qualitative) variables, there are 752640 possible combinations. The goal of prediction is whether a person earns more or less than 50000 $ and corresponds therefore to a binary classification problem.

A very important feature of this data set is the given separation into $\frac{2}{3}$ training and $\frac{1}{3}$ testing set with (train has 30162 instances and test 15060 if unknown values are removed). therefore, we can perfectly compare different data mining techniques as everyone uses the same samples for training and testing. It also leads to a broad comparative table as shown in Table 6.8.

In the first section, we examine the experimental setup while in Section 6.2,

variable	scale	values
age	continuous	74
workclass	discrete	8
fnlwgt	continuous	26741
education	discrete	16
education-num	continuous	16
marital-status	discrete	7
occupation	discrete	14
relationship	discrete	6
race	discrete	5
sex	discrete	2
capital-gain	continuous	122
capital-loss	continuous	97
hours-per-week	continuous	96
native-country	continuous	41

Table 6.1.: Variables in the Census Income data set with 45222 instances.

we show the experimental results for different parameter settings, i.e., scenarios and different scoring functions. Both the maximum evaluation and mean evaluation classifiers of Section 4.8 and 4.9 are demonstrated on this data set. Besides the important classification accuracy, like in the previous section, the sensitivity, the specificity and the *AUC* value are used for performance evaluation. Additionally, we use the given segmentation into training and testing data in the next chapter to demonstrate the new statistical hypothesis testing approach introduced in Section 4.10.

6.1. Experimental Pre-Processing and Setup

In this first section, we present the pre-processing procedures to adjust the data set for the classification task and introduce the different parameter settings.

Pre-Processing

Like the German Credit data set, the data has been revised in different terms before the application of the clustering based classifier. At first, incomplete instances have been removed so that the original data set shrunk from 32561 instances for the training data and 16281 for the testing data to 30162 and 15060 instances, respectively.

In a second step, the variable 'fnlwgt' has been eliminated as it is a check-sum variable and therefore not relevant for our clustering approach. In addition, the variable 'education' has been removed as it is doubled with the variable 'education-num'. This leads to a total of 12 input variables. The values of the metric scaled variables 'capital-gain' and 'capital-loss' have been discretized by 20 %-quantiles which reduces the number of originally 122 and 97 different values. Table 6.2 shows an overview of the variables after the described pre-processing steps. In the following, the resulting training and testing data will be set to S^{train} and S^{test}, respectively.

Experimental Setup

As described before, the whole data set consists of 14 input variables, originally. If a variable has no information for a persons income it should not be used for prediction. This is equivalent to be weighted with zero or being removed like the mentioned variables 'fnlwgt' and 'education'. The remaining variables should be weighted relative to their influence on the target value.

As described in Section 4.6, we use linearly weighted data and compare the results with the unweighted data in the first step. At first, a cluster number validated via a 3-fold cross-validation on a small range from 5 to 30 clusters on the training set is presented for the unweighted and the linearly weighted case. Secondly, we present the results for larger cluster numbers from 50 to 500

variable	scale	values
age	continuous	74
workclass	discrete	8
fnlwgt	continuous	26741
education	discrete	16
education-num	continuous	16
marital-status	discrete	7
occupation	discrete	14
relationship	discrete	6
race	discrete	5
sex	discrete	2
capital-gain	continuous	5
capital-loss	continuous	5
hours-per-week	continuous	96
native-country	continuous	41

Table 6.2.: Variables in the Census Income data set with 45222 instances after pre-processing. Two variables have been eliminated (red) and two discretized by 20 %-quantiles (green).

which will be motivated by results gathered through the previous scenarios.

As described in Section 4.1, because of the cluster sum assignments in each iteration, the clustering based classifier leads to non empty clusters of similar size even without strict lower bounds. Therefore, in all following scenarios, the lower bounds are set to 1 and no upper bounds were set to let the algorithm assign the training instances unrestricted.

Both enhanced clustering based classifiers introduced in Sections 4.8 and 4.9, h_{clust}^{max} and h_{clust}^{mean}, were evaluated. If not stated otherwise, the default threshold value ω of the underlying scoring function is set to 0.5. In this case, a cluster is assigned with 1, if at least 50 % of the labels of the training data in the cluster are labeled positive.

The corresponding initial sites $a := (a_1^T, ..., a_k^T)^T \in \mathbb{R}^{d \cdot k}$ were chosen randomly equally distributed out of the unit sphere and standardized as described in Section 4.3. Figure 6.1 gives an overview of the different scenarios, i.e., pa-

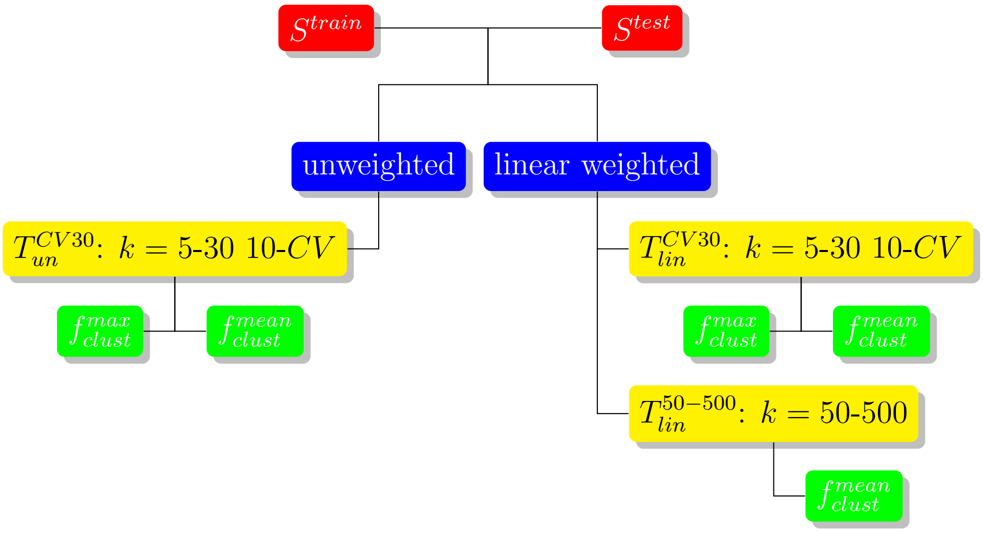

Figure 6.1.: Experimental overview of the scenarios for the evaluation on the Census Income data set.

rameter settings the results in Section 6.2 originate from. We used Algorithm 8 for the unweighted data and Algorithm 9 with a linear weighting procedure described in Section 4.6.

6.2. Experimental Results

In this section, we present the results for the different parameter settings listed in Table 6.1. The first results in Table 6.3 show that the unweighted data set leads to a worse prediction performance expressed in a lower classification accuracy if we compare scenario T_{un}^{CV30} with scenario T_{lin}^{CV30}. T_{un}^{CV30} represents the results of a performed cross-validation for cluster numbers between 5 to 30. The result is $k = 29$ as the best cluster number, evaluated on the 3-fold cross validation with the classification accuracy as the performance criterion. Both cross-validated classifications for the weighted and the unweighted case T_{un}^{CV30} and T_{un}^{CV30} result in a best cluster number of $k = 29$ (see Table 6.3 and 6.4). In the unweighted case an classification accuracy of 85.02 % is achieved on the training data and 84.89 % for the testing data with the classifier h^{max}_{clust}. The classifier h^{mean}_{clust} achieves corresponding accuracy values that are both higher with 85.12 % and 85.08 %, respectively. The same holds for the weighted case with h^{max}_{clust} achieving 85.08 % on the training and the testing data, while h^{mean}_{clust} performs better, with 85.15 % for both the training and the testing data. Additionally, Table 6.3 and 6.4 show that the important classification accuracy and also the sensitivities are better in the weighted case. This result holds for the maximum clustering based classifier h^{max}_{clust} as well as for h^{max}_{clust} and motivates the focus on the linearly weighted data set in the next analyses.

classifier	h^{max}_{clust}		h^{mean}_{clust}	
data set	S^{train}	S^{test}	S^{train}	S^{test}
TP	3979	1938	4198	2061
FP	988	513	1177	608
FN	3529	1761	3310	1638
TN	21666	10839	21477	10744
ACC (%)	85.02	84.89	85.12	85.08
σ_{ACC} (%)	0.21	0.29	0.20	0.29
$l^{ACC}_{0.95}$ (%)	84.62	84.32	84.72	84.51
$u^{ACC}_{0.95}$ (%)	85.43	85.46	85.53	85.65
ERR (%)	14.98	15.11	14.88	14.92
$SENS$ (%)	53.00	52.39	55.91	55.72
$SPEC$ (%)	95.64	95.48	94.80	94.64

Table 6.3.: Results for scenario T^{CV30}_{un} with cluster number $k = 29$.

Additionally, the h^{mean}_{clust} classifier performed better than the h^{max}_{clust} classifier on both the unweighted and the linearly weighted data if measured by the classification accuracy. Therefore, besides the focus on the weighted data, we also focus on the mean evaluation approach with h^{mean}_{clust} as classifier.

Scenario T^{50-500}_{lin} consists of 10 classifications with a cluster number from 50 to 500 in a distance of 50 clusters between each single classification. As mentioned in Section 4.9, the correlation between the underlying objective value of the clustering and the classification accuracy decreases with increasing number of

classifier	h_{clust}^{max}		h_{clust}^{mean}	
data set	S^{train}	S^{test}	S^{train}	S^{test}
TP	4251	2085	4250	2085
FP	1244	631	1220	621
FN	3257	1614	3258	1614
TN	21410	10721	21434	10731
ACC (%)	85.08	85.08	85.15	85.15
σ_{ACC} (%)	0.21	0.29	0.20	0.29
$l_{0.95}^{ACC}$ (%)	84.68	84.51	84.75	84.58
$u_{0.95}^{ACC}$ (%)	85.68	85.65	85.55	85.72
ERR (%)	14.92	14.92	14.85	14.85
$SENS$ (%)	56.62	56.37	56.61	56.37
$SPEC$ (%)	94.51	94.44	94.61	94.63

Table 6.4.: Results for scenario T_{lin}^{CV30} with cluster number $k = 29$.

clusters. Therefore, and additionally indicated by the first presented empirical results in the scenarios T_{un}^{CV30} and T_{lin}^{CV30}, the scenario T_{lin}^{50-500} is computed with h_{clust}^{mean} as classifier only. Table 6.5 as well as Figure 6.2 and 6.3 show that increasing the cluster number tend to better results in terms of the classification accuracy and sensitivity. Even though the specificity for the highest cluster

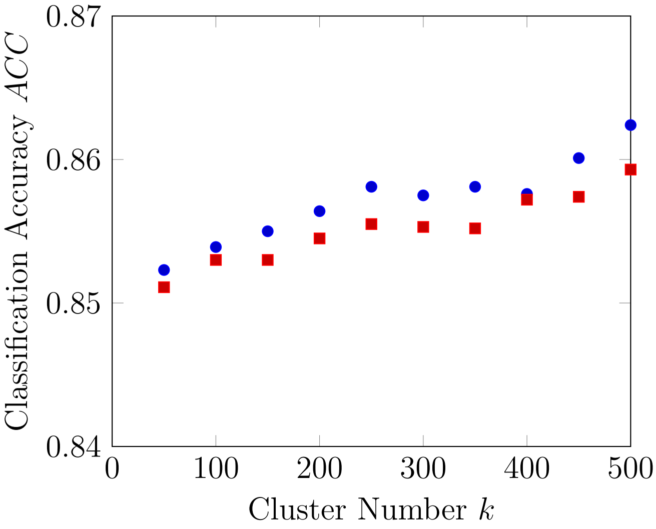

Figure 6.2.: Classification Accuracy for S^{train} (blue) and S^{test} (red) in scenario T_{lin}^{50-500}.

data set	S^{train}			S^{test}		
cluster number	ACC (%)	$SENS$ (%)	$SPEC$ (%)	ACC (%)	$SENS$ (%)	$SPEC$ (%)
50	85.23	57.87	94.30	85.11	57.59	94.07
100	85.39	58.90	94.17	85.30	58.77	93.94
150	85.50	56.51	95.11	85.30	55.86	94.89
200	85.64	59.31	94.37	85.45	58.63	94.19
250	85.81	61.04	94.01	85.55	62.05	93.79
300	85.75	59.52	94.45	85.53	58.80	94.24
350	85.81	58.54	94.85	85.52	57.46	94.66
400	85.76	61.11	93.93	85.72	60.85	93.82
450	86.01	60.91	94.32	85.74	60.11	94.09
500	86.24	65.13	93.23	85.98	64.50	92.98

Table 6.5.: Results in scenario T_{lin}^{50-500} for h_{clust}^{mean}.

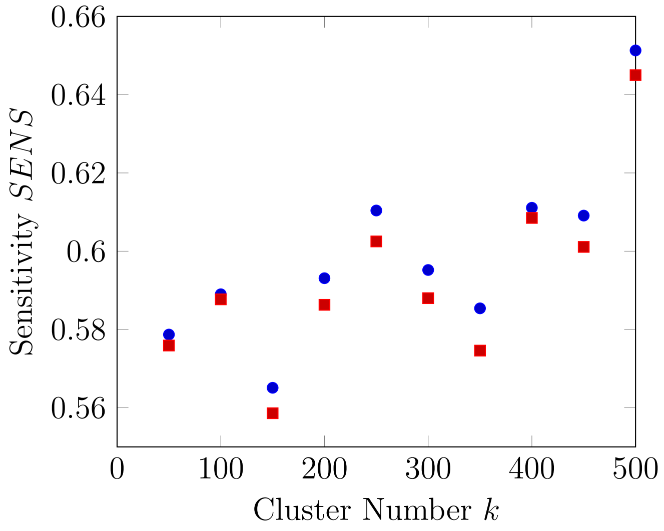

Figure 6.3.: Sensitivity for S^{train} (blue) and S^{test} (red) in scenario T_{lin}^{50-500}.

number $k = 500$ is slightly lower than for the other cluster sizes, the accuracy and the sensitivity are maximal for $k = 500$. Therefore, we focus on the highest cluster number and look into these results in detail in the following.

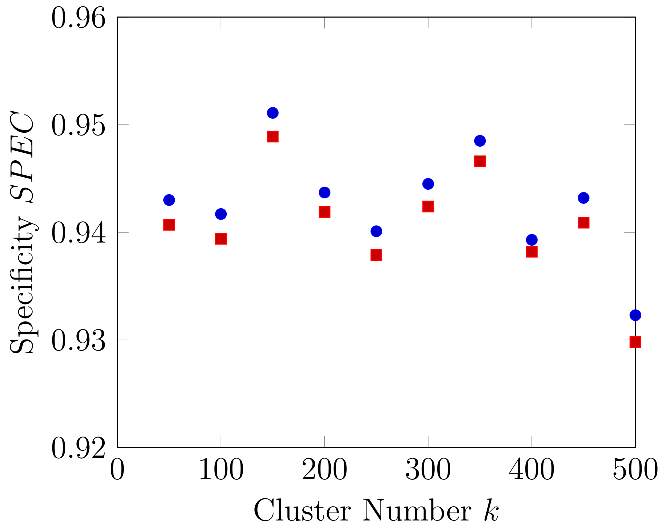

Figure 6.4.: Specificity for S^{train} (blue) and S^{test} (red) in scenario T_{lin}^{50-500}.

Table 6.5 shows the highest value of 86.24 % of the classification accuracy for the training set and 85.98 % for the testing set. Additionally, the values of the sensitivity are at their peak with the cluster number $k = 500$, being 65.13 % for the training and 64.50 % for the testing data.

As mentioned before, the default value for the threshold parameter of the used classifiers was $\omega = 0.5$. Tables 6.6 and 6.7 show the classification results for threshold parameters ω ranging from 0.1 to 0.9. Furthermore, *AUC* values of 91.21 % (train) and 90.49 % (test) result from the corresponding receiver operating characteristics shown in Figure 6.5.

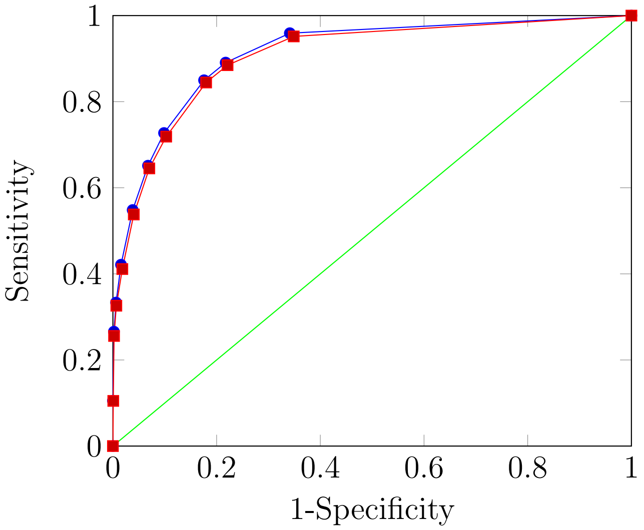

Figure 6.5.: ROC curve in scenario T_{lin}^{500} for S^{train} (blue) and S^{test} (red) with an AUC-value of 91.21 % (S^{train}) and 90.49 % (S^{test}).

The *ROC* curve in Figure 6.5 as well as Table 6.7 show that the intuitive threshold value $\omega = 0.5$ leads to the best performance, both for the training and the testing data. In the last step, these results are compared with other classification techniques like naive Bayes or the k-nearest neighbor introduced in Part II of this work.

Table 6.8 is contained in the data set description of the Census Income data set and is a representative overview of the common binary classification techniques and their performance on the Census Income data set measured by the error rate. It is retrievable at the UCI machine learning repository (see [8]).

threshold ω	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TP	7201	6688	6380	5460	4890	4117	3163	2503	1995
FP	7730	4926	3982	2233	1533	863	354	137	44
FN	307	820	1128	2048	2618	3391	4345	5005	5513
TN	14924	17728	18672	20421	21121	21791	22300	22517	22610
ACC (%)	73.35	80.95	83.06	85.81	86.24	85.90	84.42	82.95	81.58
σ_{ACC} (%)	0.25	0.23	0.22	0.20	0.20	0.20	0.21	0.22	0.22
$l_{0.95}^{ACC}$ (%)	72.85	80.51	82.63	85.41	85.85	85.50	84.01	82.53	81.14
$u_{0.95}^{ACC}$ (%)	73.85	81.39	83.48	86.20	86.63	86.29	84.83	83.38	82.01
ERR (%)	26.65	19.05	16.94	14.19	13.76	14.10	15.58	17.05	18.42
$SENS$ (%)	95.91	89.08	84.98	72.72	65.13	54.83	42.13	33.34	26.57
$SPEC$ (%)	65.88	78.26	82.42	90.14	93.23	96.19	98.44	99.40	99.81
AUC (%)	91.21								

Table 6.6.: Results in scenario T_{lin}^{500} with different threshold values ω (S^{train}).

threshold ω	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TP	3520	3273	3124	2659	2386	1990	1521	1206	946
FP	3959	2510	2041	1167	797	457	206	75	25
FN	179	426	575	1040	1313	1709	2178	2493	2753
TN	7393	8842	9311	10185	10555	10895	11146	11277	11327
ACC (%)	72.51	80.49	82.62	85.34	85.98	85.61	84.16	82.94	81.54
σ_{ACC} (%)	0.36	0.32	0.31	0.29	0.28	0.29	0.30	0.31	0.32
$l_{0.95}^{ACC}$ (%)	71.79	79.86	82.01	84.77	85.43	85.05	83.58	82.34	80.92
$u_{0.95}^{ACC}$ (%)	73.22	81.13	83.22	85.90	86.54	86.17	84.74	83.54	82.16
ERR (%)	27.49	19.51	17.38	14.66	14.02	14.39	15.84	17.06	18.46
$SENS$ (%)	95.16	88.48	84.46	71.88	64.50	53.80	41.12	32.60	25.57
$SPEC$ (%)	65.13	77.89	82.02	89.72	92.98	95.97	98.19	99.34	99.78
AUC (%)	90.49								

Table 6.7.: Results in scenario T_{lin}^{500} with different threshold values ω (S^{test}).

The excellent performance can be seen as the clustering approach takes the first place because of the highest achieved classification accuracy of all listed classifiers. The listed techniques contain also state-of-the-art classifiers and not only basic individual classifiers like the already in Section 2.1 introduced naive Bayes which is taking rank 12 with a classification accuracy of 83.88 %. In combination with the empirical performance results for the German Credit data set presented in Chapter 5, these results show the excellent performance of the new clustering approach and its huge potential for classification tasks.

technique	*ERR* (%)	*ACC* (%)
CLUST	14.02	85.98
FSS NB	14.05	85.95
NBTree	14.10	85.90
C4.5-auto	14.46	85.54
IDTM (Decision Table)	14.46	85.54
HOODG	14.82	85.18
C4.5 rules	15.54	85.06
OC1	16.64	84.96
Voted ID3(0.6)	15.64	84.36
CN2	16.00	84.00
NB	16.12	83.88
Voted ID3(0.8)	16.47	83.53
T2	16.84	83.16
1R	19.54	80.46
KNN3	20.35	79.65
KNN1	21.24	78.58

Table 6.8.: Comparison table for S^{test} of common classification techniques (see [8]).

7. Statistical Hypothesis Evaluation on the Census Income Data Set

The following results are motivated by clustering based tests of hypotheses introduced in Section 4.10. This new approach allows to evaluate the reliability of a classification by assigning a pair of p-values (pv_i^l, pv_i^u) to each cluster C_i, $i \in \{1, ..., k\}$. These p-values correspond to a system of hypotheses

$$\mathcal{H}_0^l(C_i) : p_i \leq f^{train}(C_i) \cdot \delta_i^l =: \hat{p}_i^l$$

$$\mathcal{H}_1^l(C_i) : p_i > f^{train}(C_i) \cdot \delta_i^l =: \hat{p}_i^l$$

$$\mathcal{H}_0^u(C_i) : p_i \geq f^{train}(C_i) \cdot \delta_i^u =: \hat{p}_i^u$$

$$\mathcal{H}_1^u(C_i) : p_i < f^{train}(C_i) \cdot \delta_i^u =: \hat{p}_i^u$$

derived from cluster values $f^{train}(C_i)$ for each cluster based on the training data. The labeled testing data S^{test} assigned to the clusters C_i leads to the lower and upper realization $(t^l(C_i),\ t^u(C_i))$ of the corresponding test statistics $(T^l(C_i),\ T^u(C_i))$. In the next step, they allow the computation of special quantiles, the p-values

$$pv_i^l := P(T^l(C_i) > t^l(C_i)|\mathcal{H}_0^l(C_i))$$

and

$$pv_i^u := P(T^u(C_i) < t^u(C_i)|\mathcal{H}_0^u(C_i)) \ .$$

The single steps are described in detail in Section 4.10 including an algorithmic or procedural description in Algorithm 10.
The presented evaluation approach relies on the assignment of the training and the testing data to clusters. Therefore, the mean evaluation approach, i.e., the classifier h_{clust}^{mean} is not suitable as there is no specific clustering as underlying assignment. In contrast to the mean evaluation technique, the h_{clust}^{max} classifier allows the evaluation based on statistical hypothesis tests.

7.1. Experimental Pre-Processing and Setup

The clustering based test of hypotheses provides a reliability measure for the prediction as additional information. To demonstrate this new technique we use the Census Income data set.

Pre-Processing

The German Credit data set analyzed in Chapter 5 is not as suitable as the Census Income data set. The reason is the given segmentation in training and testing data which makes the use of a technique like the cross-validation unnecessary. We use the linear weighted Census Income data set introduced in Chapter 6 with the same pre-processing steps. That leads to a training data of size 30162 and a testing data set with 15060 instances and 12 input variables summarized in Table 6.2.

Experimental Setup

To illustrate the clustering based test of hypotheses, we choose a parameter setup with only few clusters. A smaller number of clusters allows reliable estimations of the predictive values and makes it easier to compare the single cluster values. Additionally, the cluster values differ from each other more clearly. The cluster numbers are 5 and 10 for the scenario T^5_{lin} and T^{10}_{lin}, respectively. Additionally, we do not apply lower and upper bounds as the results show proper filled clusters without bounds for both scenarios.

7.2. Experimental Results

At first, we evaluate the h^{max}_{clust} classifier on the training data to generate the clustering based set of lower and upper null hypotheses $\{\mathcal{H}^l_0(C_i), \mathcal{H}^u_0(C_i)\}^k_{i=1}$. Additionally, we use the comparing distance parameter set $\{(\delta^l_i, \delta^u_i)\}^k_{i=1}$, adjusting the underlying hypotheses (4.11) to (4.14). It consists of a constant convex combination as introduced in Definition 4.10.2 with the parameter $\lambda_i = 0.75$ for $i \in \{1, ..., k\}$, leading to

$$\delta^l_i = 0.75 - 0.25(\frac{f^{train}(C_i)}{f^{train}(C_{i-1})}), i = 1, ..., k$$

and

$$\delta^u_i = 0.75 + 0.25(\frac{f^{train}(C_{i+1})}{f^{train}(C_i)}), i = 1, ..., k$$

with

$$f^{train}(C_0) = 0 \text{ and } f^{train}(C_{k+1}) = 1.$$

The cluster indeces are permuted, so $f^{train}(C_i) \leq f^{train}(C_{i+1})$ holds for $i \in \{1, ..., k\}$.

The first underlying clustering scenario T^5_{lin} consists of five clusters ($k = 5$) and leads to the results listed in Table 7.1 for the training and testing data. The hypotheses $\mathcal{H}^l_0(C_i)$ and $\mathcal{H}^u_0(C_i)$ for $i \in \{1, ..., k\}$ are specified with the comparing distance parameter set $\{(\delta^l_i, \delta^u_i)\}^k_{i=1}$. In the next step, the classification of the testing data S^{train} is used to assign the test instances to the cluster and to compute the realizations of $T^l(C_i)$ and $T^u(C_i)$. These values then lead to the pair of p-values (pv^l_i, pv^u_i) for each cluster C_i (see Table 7.2). All but one predicted values are significant to the level 5 % in a range of ±0.25 % to the next smaller/larger predicted value or cluster. These results are compared with a second clustering scenario T^{10}_{lin} with the double number of clusters ($k = 10$) in the next step. The other parameters are unchanged and the resulting performance values are listed in Table 7.3.

data set	S^{train}	S^{test}
TP	4278	2100
FP	1328	676
FN	3230	1599
TN	21326	10676
ACC (%)	84.89	84.88
σ_{ACC} (%)	0.21	0.29
$l^{ACC}_{0.95}$ (%)	84.48	84.31
$u^{ACC}_{0.95}$ (%)	85.29	85.46
ERR (%)	15.11	15.12
$SENS$ (%)	56.98	56.77
$SPEC$ (%)	94.14	94.05

Table 7.1.: Results in scenario T^5_{lin} for clustering based test of hypotheses.

While all lower and upper p-values for each cluster in the scenario T^5_{lin} were below 5.2314 % and 7 out of 10 even below 0.0001 %, that statement holds only for 2 out of 20 p-values in the second scenario with 10 clusters.

Of course, the higher number of clusters lead to the smaller differences between the cluster values for the training and the testing data. This makes it harder

C_i	$n^{train}(C_i)$	$f^{train}(C_i)$	$n^{test}(C_i)$	$f^{test}(C_i)$	$\hat{p}_i^l$	$\hat{p}_i^u$	$t^l(C_i)$	$t^u(C_i)$	pv_i^l	pv_i^u
1	12147	2.17 %	6039	2.04 %	1.62 %	5.12 %	2.539	−10.867	0.5563 %	0.0000 %
2	3484	13.98 %	1774	12.68 %	11.02 %	17.43 %	2.230	−5.271	1.2873 %	0.0000 %
3	8925	27.79 %	4462	28.04 %	24.33 %	37.92 %	5.763	−13.611	0.0000 %	0.0000 %
4	3947	68.33 %	1968	67.53 %	58.19 %	75.07 %	8.397	−7.734	0.0000 %	0.0000 %
5	1659	95.30 %	808	95.42 %	88.56 %	96.47 %	6.129	−1.623	0.0000 %	5.2314 %

Table 7.2.: Results for the hypothesis testing approach for the clustering with a number of 5 clusters (scenario T_{lin}^5).

data set	S^{train}	S^{test}
TP	4280	2103
FP	1377	702
FN	3228	1596
TN	21277	10650
ACC (%)	84.73	84.73
σ_{ACC} (%)	0.21	0.29
$l_{0.95}^{ACC}$ (%)	84.33	84.16
$u_{0.95}^{ACC}$ (%)	85.14	85.31
ERR (%)	15.21	15.27
$SENS$ (%)	57.01	56.85
$SPEC$ (%)	93.92	93.82

Table 7.3.: Results in scenario T_{lin}^{10} for clustering based test of hypotheses.

to reject the null hypotheses as $\hat{p}_i^l$ and $\hat{p}_i^u$ are closer to $f^{test}(C_i)$. Additionally, there are fewer instances in a cluster at the average. Nevertheless, still 10 out of 20 p-values are lower than 1 % as listed in Table 7.4. These exemplary results demonstrate the new statistical evaluation approach for the clustering classifier as introduced in Section 4.10. They show the tradeoff, measured by statistical means, between the number of clusters and the reliability of the predicted value. The higher the number of clusters, the lower the difference between the (sorted) cluster values and the fewer instances per cluster used for training the classifier.

C_i	$n^{train}(C_i)$	$f^{train}(C_i)$	$n^{test}(C_i)$	$f^{test}(C_i)$	$\hat{p}_i^l$	$\hat{p}_i^u$	$t^l(C_i)$	$t^u(C_i)$	pv_i^l	pv_i^u
1	746	0.13 %	376	0.27 %	0.10 %	0.34 %	1.0121	−0.2453	15.5735 %	40.3130 %
2	8996	0.96 %	4429	0.88 %	0.75 %	2.44 %	1.0029	−6.7199	15.7950 %	0.0000 %
3	3080	6.88 %	1566	5.94 %	5.40 %	8.80 %	0.9407	−3.9909	17.3418 %	0.0033 %
4	3090	14.53 %	1574	13.60 %	12.62 %	16.58 %	1.1674	−3.1877	12.1524 %	0.0717 %
5	6498	22.75 %	3250	23.14 %	20.69 %	29.02 %	3.4432	−7.3833	0.0287 %	0.0000 %
6	2095	47.83 %	1051	47.29 %	41.56 %	52.13 %	3.7699	−3.1400	0.0082 %	0.0845 %
7	3102	65.02 %	1565	63.96 %	60.72 %	67.80 %	2.6227	−3.2463	0.4362 %	0.0585 %
8	896	76.12 %	432	76.62 %	73.34 %	79.90 %	1.5407	−1.6989	6.1694 %	4.4665 %
9	776	91.24 %	375	91.47 %	87.46 %	93.14 %	2.3444	−1.2860	0.9528 %	9.9229 %
10	883	98.87 %	433	98.85 %	96.96 %	99.15 %	2.2851	−0.6924	1.1154 %	24.4344 %

Table 7.4.: Results for the hypothesis testing approach for the clustering with a number of 10 clusters (scenario T_{lin}^{10}).

Part V.

Conclusion

Summary of the Results

In this thesis, we introduced a new approach for classification problems. The geometric clustering and its use for prediction offer a new way to solve classification problems. The underlying clustering uses geometrical proximity after transforming the data into its one-dimensional conditional expected values.

Besides the development of the new clustering based classifier, we show several new theoretical results based on the work of Brieden and Gritzmann for example in [19], [20], [21], [22] and [23].

These theoretical results like the termination of the iterative sequence on the basis of the introduced cluster sum assignments and its use for classification show necessary characteristics for the algorithmic implementation. Additionally, in Section 4.1 we show the connection and the difference of the introduced cluster sum assignment and the similar least-squares assignment (see for example [16]). Based on the theoretical framework in Section 4.1 and the data transformation technique explained in Section 4.2, in Section 4.3 we combine the combinatorial optimization with the field of supervised learning applying the conceptual framework of classifiers.

The theoretical concept of scoring functions as a part of a classifier is adopted in Section 4.5 to the clustering approach. We define the clustering based classification and two intuitive scoring functions leading to the maximum clustering based classifier and the mean clustering based classifier.

The transformation technique introduced in Section 4.2 allows the use of clustering techniques for non-metric input variables and in addition an useful weighting technique for practical applications. This is explained and motivated by statistical means in Section 4.6 and provides additional possibilities to enhance and adjust the quality of prediction in the following sections. Additionally, in the last section of Chapter 4 we present a new approach to evaluate the quality of a prediction from of a probabilistic point of view. The clustering based set of hypotheses allows corresponding tests of hypotheses and provides p-values for each cluster as quantitative reliability measures.

Besides the theoretical introduction and motivation of the clustering approach for classification, the empirical results of Part IV show the excellent performance of the new clustering based classifier applied to two real-world data sets. The classification of the Census Income and the German Credit data set show the excellent performance compared to established classification techniques. Depending on the performance measure, the new classification approach beats the existing techniques for example in terms of the classification accuracy. Additionally, the clustering based test of hypotheses introduced in Section 4.10 was demonstrated on the Census Income data set. These results allow the comparison of the prediction quality when different clustering based classifications are performed.

The new iterative clustering approach is different in various ways compared to the classification methods introduced in Part II of this thesis. In contrast to the clustering based classifier, the naive Bayes, the logistic regression and the k-nearest neighbor approach are obviously less complex but less flexible as they allow less adjustments. The new iterative classification approach combines supervised (training/testing classification) and unsupervised learning techniques (clustering) and achieves great performance by constructing a feasible convex cell decomposition of the data space. The partition is induced by a clustering with a given cluster number and the possible setting of bounds for a cluster. While the naive Bayes and the logistic regression divide the data space in two half-spaces, the new clustering approach allows a freely chosen number of cells. Therefore, it can be fitted appropriately to match the data structure. Additionally, the segmentation of the data space into a given number of clusters allows the statistical testing scheme introduced in Section 4.10 as a tiered reliability measure for the quality of prediction.

Outlook

The new clustering based classifier allows adjustments in many ways. The approach can be divided into three core components: the data transformation, the clustering including the cell decomposition and the calculation of the scoring function, i.e., the evaluation of the cluster cells. In each of these steps there are parameters and procedures that can be modified.

In the first part, the data transformation, the one-dimensional conditional expected values are estimated. Further work could concentrate on the extension to multidimensional conditional expectations. Besides the transformation, another weighting procedure for the input variables could be examined and analyzed.

In the second part of the new approach, the computation of the clustering, there a numerous adjustable parameters to be investigated like lower and upper bounds, the approximation of the ellipsoidal norm and last but not least the number of clusters. Additionally, the calculation of good initial sites or improvements of the iterative procedure, like steepest descent, could be analyzed.

The last core part of the new approach, the actual classification, could also be further adjusted. A variance based threshold on the classification values or even an extension of the estimation by using other classifiers like the naive Bayes or logistic regression on the cluster cells could improve the performance that is already excellent as the results in Part IV show.

The statistical hypothesis testing approach is another main component of this thesis and could be further analyzed. The setting of the underlying hypotheses and the detailed analysis of the mentioned tradeoff between classification accuracy and the reliability of the prediction is another possible aspect of future work. We introduced the comparing distance as a first parameter scheme for the statistical hypotheses setting. Additional analysis of different parameter settings in combination with different cluster numbers could also be an interesting extension of this part.

In practical applications different classifiers often are combined to benefit from their individual strength. A combination of established techniques with the new clustering based classifier could further improve the prediction quality for the binary classification problem.

List of Tables

List of Figures

References

[1] Agresti, A. *Categorical data analysis*, 2nd ed. Wiley Series in Probability and Statistics. Wiley-Interscience, 2002.

[2] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. NP-hardness of euclidean sum-of-squares clustering. *Machine Learning 75*, 2 (2009), 245–248.

[3] Anderberg, M. *Cluster analysis for applications.* Probability and mathematical statistics. Academic Press, 1973.

[4] Anthony, M., and Bartlett, P. L. *Neural network learning: theoretical foundations*, 1st ed. Cambridge University Press, New York, NY, USA, 2009.

[5] Aurenhammer, F. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing 16*, 1 (1987), 78–96.

[6] Aurenhammer, F., Hoffmann, F., and Aronov, B. Minkowski-type theorems and least-squares clustering. *Algorithmica 20*, 1 (1998), 61–76.

[7] Bache, K., and Lichman, M. UCI machine learning repository. `https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/`, 2014. [Online; accessed 09-July-2014].

[8] Bache, K., and Lichman, M. UCI machine learning repository. `http://archive.ics.uci.edu/ml/datasets/Adult`, 2014. [Online; accessed 09-July-2014].

[9] Baesens, B., Gestel, T. V., Viaene, S., Stepanova, M., Suykens, J., and Vanthienen, J. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society 54*, 6 (2003), 627–635.

[10] Balakrishnan, N. *Handbook of logistic distribution.* Mechanical Engineering Series. CRC Press LLC, 2010.

[11] Barnes, E., Hoffman, A., and Rothblum, U. Optimal partitions having disjoint convex and conic hulls. *Mathematical Programming 54*, 1-3 (1992), 69–86.

[12] Berger, J. *Statistical decision theory and Bayesian analysis.* Springer Series in Statistics. Springer, 1985.

[13] Bishop, C. *Pattern recognition and machine learning.* Information Science and Statistics. Springer, 2006.

[14] Blum, A., and Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence 97*, 1-2 (1997), 245–271.

[15] Bodlaender, H., Gritzmann, P., Klee, V., and Leeuwen, J. Computational complexity of norm-maximization. *Combinatorica 10*, 2 (1990), 203–225.

[16] Borgwardt, S., Brieden, A., and Gritzmann, P. A balanced k-means algorithm for weighted point sets. *CoRR abs/1308.4004* (2013).

[17] Borgwardt, S., Brieden, A., and Gritzmann, P. Geometric clustering for the consolidation of farmland and woodland. *The Mathematical Intelligencer* (2014), 1–8.

[18] Breiman, L. Heuristics of instability and stabilization in model selection. *The Annals of Statistics 24*, 6 (12 1996), 2350–2383.

[19] Brieden, A. *On the approximability of (discrete) convex maximization and its contribution to the consolidation of farmland.* Habilitation, 2003.

[20] Brieden, A., and Gritzmann, P. On the inapproximability of polynomial-programming, the geometry of stable sets, and the power of relaxation. In *Discrete and Computational Geometry*, B. Aronov, S. Basu, J. Pach, and M. Sharir, Eds., vol. 25 of *Algorithms and Combinatorics.* Springer Berlin Heidelberg, 2003, pp. 301–311.

[21] Brieden, A., and Gritzmann, P. A quadratic optimization model for the consolidation of farmland by means of lend-lease agreements. In *Operations Research Proceedings 2003*, D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt, Eds., vol. 2003 of *Operations Research Proceedings.* Springer Berlin Heidelberg, 2004, pp. 324–331.

[22] Brieden, A., and Gritzmann, P. On clustering bodies: geometry and polyhedral approximation. *Discrete & Computational Geometry 44*, 3 (2010), 508–534.

[23] Brieden, A., and Gritzmann, P. On optimal weighted balanced clusterings: gravity bodies and power diagrams. *SIAM J. Discrete Math. 26*, 2 (2012), 415–434.

[24] Brieden, A., Gritzmann, P., and Öllinger, M. The algorithmic implementation of geometric clusterings, manuscript.

[25] Brieden, A., Hinnenthal, M., and Öllinger, M. Probabilistic hypotheses in geometric clusterings, manuscript.

[26] Caruana, R., and Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA, 2006), ICML '06, ACM, pp. 161–168.

[27] Cayton, L. Fast nearest neighbor retrieval for bregman divergences. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)* (July 2008), A. McCallum and S. Roweis, Eds., Omnipress, Omnipress, pp. 112–119.

[28] Clarkson, K. Fast algorithms for the all nearest neighbors problem. In *Foundations of Computer Science, 1983., 24th Annual Symposium on* (Nov 1983), pp. 226–232.

[29] Dawid, A. P. Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B (Methodological) 41*, 1 (1979), 1–31.

[30] Dietterich, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation 10* (1998), 1895–1923.

[31] Driver, H., and Kroeber, A. *Quantitative expression of cultural relationship*, vol. Quantitative Expression of Cultural Relationships of *Publications in American Archeology and Ethnology.* University of California Press, Berkeley, 1932.

[32] Duda, R., Hart, P., and Stork, D. *Pattern classification.* Pattern Classification and Scene Analysis: Pattern Classification. Wiley, 2001.

[33] Efron, B. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association 78*, 382 (1983), pp. 316–331.

[34] Estivill-Castro, V. Why so many clustering algorithms: a position paper. *SIGKDD Explor. Newsl. 4*, 1 (June 2002), 65–75.

[35] Everitt, B., Landau, S., Leese, M., and Stahl, D. *Cluster analysis.* Wiley series in probability and statistics. Wiley, 2011.

[36] Fawcett, T. An introduction to ROC analysis. *Pattern Recogn. Lett. 27*, 8 (June 2006), 861–874.

[37] Fayyad, U., Piatetsky-shapiro, G., and Smyth, P. From data mining to knowledge discovery in databases. *AI Magazine 17* (1996), 37–54.

[38] Ferguson, T. S. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics 1*, 2 (1973), 209–230.

[39] Fisher, R. *Statistical methods for research workers.* Edinburgh Oliver & Boyd, 1925.

[40] Freedman, D., and Berk, R. Weighting regressions by propensity scores. *Evaluation Review 32*, 4 (2008), 392.

[41] Friedman, J. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery 1*, 1 (1997), 55–77.

[42] Gantz, J., and Reinsel, D. The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east. Tech. rep., International Data Company, December 2012.

[43] Geisser, S. The predictive sample reuse method with applications. *Journal of the American Statistical Association 70*, 350 (1975), pp. 320–328.

[44] Gordon, A. *Classification, 2nd Edition.* Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1999.

[45] Guyon, I., and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (2003), 1157–1182.

[46] Han, J. *Data mining: concepts and techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[47] Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference and prediction*, 2 ed. Springer, 2009.

[48] He, H., and Garcia, E. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on 21*, 9 (Sept 2009), 1263–1284.

[49] Hernández-Orallo, J., Flach, P. A., and Ramirez, C. F. Brier curves: a new cost-based visualisation of classifier performance. In *ICML* (2011), L. Getoor and T. Scheffer, Eds., Omnipress, pp. 585–592.

[50] Hilbe, J. M. *Logistic regression models.* Chapman & Hall/ Crc: Texts in Statistical Science Series. Chapman & Hall/CRC, 2009.

[51] Hosmer, D. W., and Lemeshow, S. *Applied logistic regression*, 2 ed. Wiley-Interscience Publication, 2000.

[52] Hsu, C. N., Huang, H. J., and Wong, T. T. Why discretization works for naive Bayesian classifiers. In *Proc. 17th International Conf.*

on Machine Learning (2000), Morgan Kaufmann, San Francisco, CA, pp. 399–406.

[53] Hwang, F., Onn, S., and Rothblum, U. A polynomial time algorithm for shaped partition problems. *SIAM Journal on Optimization 10*, 1 (1999), 70–81.

[54] Hwang, F. K., Onn, S., and Rothblum, U. G. Representations and characterizations of vertices of bounded-shape partition polytopes. *Linear Algebra and its Applications 278*, 1-3 (1998), 263–284.

[55] Hwang, F. K., Onn, S., and Rothblum, U. G. Linear-shaped partition problems. *Operations Research Letters 26*, 4 (2000), 159 – 163.

[56] Japkowicz, N., and Stephen, S. The class imbalance problem: a systematic study. *Intell. Data Anal. 6*, 5 (Oct. 2002), 429–449.

[57] Kim, Y. S., and Sohn, S. Y. Managing loan customers using misclassification patterns of credit scoring model. *Expert Systems with Applications 26*, 4 (2004), 567 – 573.

[58] King, G., and Zeng, L. Logistic regression in rare events data. *Political Analysis 9* (2001), 137–163.

[59] King, R. D., Feng, C., and Sutherland, A. STATLOG: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence 9*, 3 (1995), 289–333.

[60] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* (San Francisco, CA, USA, 1995), IJCAI'95, Morgan Kaufmann Publishers Inc., pp. 1137–1143.

[61] Kohavi, R. Scaling up the accuracy of naive Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), p. to appear.

[62] Kolmogorov, A. N. *Foundations of the theory of probability*, 2 ed. Chelsea Pub Co, June 1960.

[63] Mahajan, M., Nimbhorkar, P., and Varadarajan, K. The planar k-means problem is NP-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation* (Berlin, Heidelberg, 2009), WALCOM '09, Springer-Verlag, pp. 274–285.

[64] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to information retrieval.* Cambridge University Press, New York, 2008.

[65] McLachlan, G. J. *Discriminant analysis and statistical pattern recognition.* Wiley series in probability and mathematical statistics. J. Wiley and sons, New York, Chichester, Brisbane, 1992.

[66] Michie, D., Spiegelhalter, D. J., and Taylor, C. C., Eds. *Machine learning, neural and statistical classification.* Ellis Horwood, New York, NY, 1994.

[67] Milligan, G., and Cooper, M. An examination of procedures for determining the number of clusters in a data set. *Psychometrika 50*, 2 (1985), 159–179.

[68] Mitchell, T. M. *Machine learning.* McGraw-Hill, New York, 1997.

[69] Miyamoto, S., Ichihashi, H., and Honda, K. *Algorithms for fuzzy clustering: methods in c-means clustering with applications.* Studies in Fuzziness and Soft Computing. U.S. Government Printing Office, 2008.

[70] Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning.* The MIT Press, 2012.

[71] Murphy, K. P. *Machine learning: a probabilistic perspective.* Adaptive Computation and Machine Learning series. The MIT Press, Aug. 2012.

[72] Ng, A., and Jordan, M. On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)* (2001), vol. 14.

[73] Pohlmann, J. T., and Leitner, D. W. A comparison of ordinary least squares and logistic regression. *The Ohio Journal of Science 103*, 5 (Dec 2003), 118–125.

[74] Powers, D. M. W. Evaluation: from precision, recall and f-factor to ROC, informedness, markedness & correlation. Tech. Rep. SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2007.

[75] Rennie, J. D. M., Shih, L., Teevan, J., and Karger, D. R. Tackling the poor assumptions of naive Bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning* (2003), pp. 616–623.

[76] Rodriguez, J. D., Perez, A., and Lozano, J. A. Sensitivity analysis of k-fold cross-validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 3 (2010), 569–575.

[77] Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., and Tirri, H. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning 59*, 3 (2005), 267–296.

[78] Schrijver, A. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[79] Seiffert, C., Khoshgoftaar, T., Van Hulse, J., and Napolitano, A. RUSBoost: a hybrid approach to alleviating class imbalance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 40*, 1 (Jan 2010), 185–197.

[80] Smyth, P. Clustering using monte carlo cross-validation. In *KDD* (1996), pp. 126–133.

[81] Stanfill, C., and Waltz, D. Toward memory-based reasoning. *Commun. ACM 29*, 12 (Dec. 1986), 1213–1228.

[82] Stefan Lessmann, Hsin-Vonn Seow, B. B., and Thomas, L. C. Benchmarking state-of-the-art classification algorithms for credit scoring: a ten year update.

[83] Stone, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological) 36*, 2 (1974), 111–147.

[84] Tibshirani, R., and Walther, G. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics 14*, 3 (sep 2005), 511–528.

[85] Timm, N. *Applied multivariate analysis.* Springer texts in statistics. Springer, 2002.

[86] Wasserman, L. *All of statistics: a concise course in statistical inference.* Springer Publishing Company, Incorporated, 2010.

[87] Weber, R., Schek, H.-J., and Blott, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1998), VLDB '98, Morgan Kaufmann Publishers Inc., pp. 194–205.

[88] Weiss, N. *Introductory statistics.* Pearson Education, Limited, 2010.

[89] Woods, K. S., Solka, J. L., Priebe, C. E., Doss, C. C., Bowyer, K. W., and Clarke, L. P. Comparative evaluation of pattern recognition techniques for detection of microcalcifications. R. S. Acharya and D. B. Goldgof, Eds., vol. 1905, SPIE, pp. 841–852.

[90] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., and Steinberg, D. Top 10 algorithms in data mining. *Knowl. Inf. Syst. 14*, 1 (Dec. 2007), 1–37.

[91] Xu, R., and Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on 16*, 3 (May 2005), 645–678.

[92] Xue, J.-H., and Titterington, D. M. Comment on "on discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes". *Neural Process. Lett. 28*, 3 (Dec. 2008), 169–187.

[93] Zhang, H. The optimality of naive Bayes. In *FLAIRS Conference* (2004), V. Barr and Z. Markov, Eds., AAAI Press.