# Statistical Methods to Enhance Clinical Prediction with High-Dimensional Data and Ordinal Response

Dissertation

zur Erlangung
des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

vorgelegt von
**Andreas Leha**
aus Forchheim

Göttingen, 2015

**Betreuungsausschuss**

Prof. Dr. Tim Beißbarth
Prof. Dr. Stephan Waack

**Mitglieder der Prüfungskommision**

Referent: Prof. Dr. Tim Beißbarth
Koreferent: Prof. Dr. Stephan Waack

*Weitere Mitglieder der Prüfungskommision*

Prof. Dr. Edgar Wingender
Prof. Dr. Burkhard Morgenstern
Prof. Dr. Winfried Kurth
Prof. Dr. Ramin Yahyapour

**Tag der mündlichen Prüfung:** 25.03.2015

# Abstract

Advancing technology has enabled us to study the molecular configuration of single cells or whole tissue samples. Molecular biology produces vast amounts of high-dimensional omics data at continually decreasing costs, so that molecular screens are increasingly often used in clinical applications.

Personalized diagnosis or prediction of clinical treatment outcome based on high-throughput omics data are modern applications of machine learning techniques to clinical problems. In practice, clinical parameters, such as patient health status or toxic reaction to therapy, are often measured on an ordinal scale (e.g. *good*, *fair*, *poor*).

The prediction of ordinal end-points is commonly treated as a simple multi-class classification problem, disregarding the ordering information contained in the response. But classifiers that do not consider the order in the response may loose prediction accuracy and may even produce unexpectedly disordered predictions.

Classical approaches to model ordinal response directly, including for instance the cumulative logit model, are typically not applicable to high-dimensional data.

We present *hierarchical twoing (hi2)*, an algorithm for classification of high-dimensional data into ordered categories. *hi2* combines the power of well-understood binary classification with ordinal response prediction. An open-source implementation of *hi2* is made available.

A comparison of several approaches for ordinal classification on real world data as well as simulated data shows that established classification algorithms especially designed to handle ordered categories fail to improve upon state-of-the-art non-ordinal classification algorithms. In general, the classification performance of an algorithm is dominated by its ability to deal with the high-dimensionality of the data. We demonstrate that our algorithm *hi2* shows consistently strong performance and outperforms its competitors in many cases.

# Zusammenfassung

Der technologische Fortschritt ermöglicht es heute, die moleculare Konfiguration einzelner Zellen oder ganzer Gewebeproben zu untersuchen. Solche in großen Mengen produzierten hochdimensionalen Omics-Daten aus der Molekularbiologie lassen sich zu immer niedrigeren Kosten erzeugen und werden so immer häufiger auch in klinischen Fragestellungen eingesetzt.

Personalisierte Diagnose oder auch die Vorhersage eines Behandlungserfolges auf der Basis solcher Hochdurchsatzdaten stellen eine moderne Anwendung von Techniken aus dem maschinellen Lernen dar. In der Praxis werden klinische Parameter, wie etwa der Gesundheitszustand oder die Nebenwirkungen einer Therapie, häufig auf einer ordinalen Skala erhoben (beispielsweise *gut*, *normal*, *schlecht*).

Es ist verbreitet, Klassifikationsproblme mit ordinal skaliertem Endpunkt wie generelle Mehrklassenproblme zu behandeln und somit die Information, die in der Ordnung zwischen den Klassen enthalten ist, zu ignorieren. Allerdings kann das Vernachlässigen dieser Information zu einer verminderten Klassifikationsgüte führen oder sogar eine ungünstige ungeordnete Klassifikation erzeugen.

Klassische Ansätze, einen ordinal skalierten Endpunkt direkt zu modellieren, wie beispielsweise mit einem kumulativen Linkmodell, lassen sich typischerweise nicht auf hochdimensionale Daten anwenden.

Wir präsentieren in dieser Arbeit *hierarchical twoing (hi2)* als einen Algorithmus für die Klassifikation hochdimensionler Daten in ordinal Skalierte Kategorien. *hi2* nutzt die Mächtigkeit der sehr gut verstandenen binären Klassifikation, um auch in ordinale Kategorien zu klassifizieren. Eine Opensource-Implementierung von *hi2* ist online verfügbar.

In einer Vergleichsstudie zur Klassifikation von echten wie von simulierten Daten mit ordinalem Endpunkt produzieren etablierte Methoden, die speziell für geordnete Kategorien entworfen wurden, nicht generell bessere Ergebnisse als state-of-the-art nicht-ordinale Klassifikatoren. Die Fähigkeit eines Algorithmus, mit hochdimensionalen Daten umzugehen, dominiert die Klassifikationsleisting. Wir zeigen, dass unser Algorithmus *hi2* konsistent gute Ergebnisse erzielt und in vielen Fällen besser abschneidet als die anderen Methoden.

# Acknowledgements

There are many people I would like to thank, for this thesis would not have been possible to be completed without them.

In the first place I want to thank my supervisor Prof. Tim Beissbarth for his guidance throughout the whole project, for the always open door, for his patience, for good coffee in the office, and above all for sharing of his knowledge. I also want to thank PD Klaus Jung for all his time and statistical advise and for being not only a boss but also a mentor and a friend. And I want to thank my second supervisor and referee Prof. Stephan Waack for his effort and time.

I owe many thanks to my friends and former colleagues at the institute, especially to Frank Kramer and Silvia von der Heyde for sharing an office and listening to my complaints about R. I want to thank Stephan Artmann, Manuel Nietert, Michaela Bayerlová, Alexander Wolff, and Astrid Wachter for many fruitful discussions and for creating such a nice environment to work in.

I owe many thanks to Annalen Bleckmann for the good collaboration and for answering all my questions on clinical data. Many thanks go to Katharina Kramer and Frank Konietschke for additional advise in statistical questions.

I want to thank my father for reading this thesis and for his many good comments.

And finally, I want to thank my family and especially my wife for their support and understanding when I took my time off to work on this thesis.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

Modern medicine has seen advances beyond the dreams of former times. Technological progress and scientific methods have led to a medicine capable to implant artificial hearts, regrow kidneys from skin cells, vaccinate, and defeat illnesses like Polio to near extinction. Life expectancy in Germany has increased one full generation in just 100 years (from 47.41 / 50.68 years (male / female) in 1910 to 76.64 / 82.58 years in 2009, table 2.2.9 in Willand, 2014). We live longer and more comfortable than ever. The advances are also revealed by the fact that obesity and back pain are listed among the most widespread diseases in western countries (Chou et al., 2007; Olshansky et al., 2005).

And yet, we have – despite all advances – not won the *war on cancer* (DeVita, 2002). We still are not able to cure numerous illnesses – especially forms of neurodegenerative diseases. And still, patients are treated individually only on the surface. Better safety control is the main reason for giving patients standard treatment: It is sometimes only possible to show the effectiveness and to assess side effects of a treatment on a large cohort of equally treated patients in randomized clinical trials. Therefore, the drug admission agencies require in their guidelines the conduct of such trials and limit the use of drugs to their primary target[(1)].

Individually tailored treatment or *personalized medicine* describes approaches where treatment is adapted to each patient individually based on patient specific characteristics. For such individual treatment the efficacy and the non-inferiority are hard to verify. But on the other hand, efficacy is possible on a much higher level with individualized treatment. Simple dosage increase, for instance, for a patient with weak treatment response could improve

[(1)] Although, *unapproved uses of approved products* are allowed within tight boundaries.

treatment success for this patient without increasing toxic reaction for other patients.

Individualized treatment on a subgroup level is known from *patient stratification* where groups of patients are distinguished (based on age or gender, e. g.). Patient stratification is commonly used already and is one key factor in treatment decision.

Today advances in technology make it possible to perform patient stratification at a much deeper level, as it is now possible to measure molecular features (such as gene expression) on individual patients. Considering molecular features in clinical decisions is also called *precision medicine*. Such molecular features have proven to be highly predictive components in diagnostic, prognostic, and predictive tests (Mehta et al., 2010). With classifiers trained on panels of molecular features treatment efficacy, toxic response or survival probability can be predicted individually for each patient (Paik, Shak, et al., 2004; Salazar et al., 2010; Bleckmann et al., 2015). With such predictions at hand the treatment decision can be guided to individual paths for individual patients.

Therapies for highly diverse illnesses that form many subtypes can benefit especially from this individualization. Cancer is an extreme example: Tumor cells show a disrupted molecular machinery, but mainly without a prominent master cause (Smith, 2013). In contrast, many levels of the molecular composition of the cell interact in complex ways and many malfunctions have been shown to relate to cancer-like behaviour of the cell.

But we are just starting to use molecular features to stratify patients (Nowsheen et al., 2012). Single predictive biomarkers have been discovered early and have been used in clinical practice for some time. Prominent examples are BRCA1 and the receptor gene HER2 in breast cancer. Mutations in BRCA1 – as discovered by Hall et al., 1990 – constitute a strong genetic predisposition for developing breast cancer and have additionally been shown to be predictive, e.g. with respect to chemotherapy (Quinn et al., 2007). An overexpression of the HER2 receptor leads to worse prognosis for these patients, but can be treated effectively by application of receptor blockers (Brügmann and Sorensen, 2011). Gene panels are the next step and examples such as MammaPrint ('t Veer et al., 2002) or Oncotype DX (Paik, Shak, et al., 2004; Paik, Tang, et al., 2006) demonstrate that looking at several genes at once can yield an

improved classification result. Gene expression or mRNA expression is arguably the most commonly used type of molecular feature (Zadran, Remacle, and Levine, 2013). But also miRNA expression (Nana-Sinkam and Croce, 2013), protein abundance (Hanash, Pitteri, and Faca, 2008), or epigenetic features such as DNA methylation (Ziech et al., 2010) are examples of molecular features that have proven to be predictive (Nowsheen et al., 2012).

To reflect the mechanisms of complex diseases such as cancer, current research projects aim at integrating more levels of molecular features into their models to further improve the classification performance.

So, while we see a lot of approaches that allow for more and more complex models on the features side, the outcome is often even simplified to binary two-class decision problems. Survival is grouped into *long term survivor* and *short term survivor*, treatment response is given as *responder* and *non-responder* and toxicity is measured as *affected* and *not affected*. Such binarization simplifies model construction and makes the results easily interpretable. On the other hand it ignores that the outcome is often given in more detail. Survival is usually measured on a continuous time scale, treatment response can be continuous (e.g. abundance of a protein in blood) or multi-class as in *partial response*, *major response*, *complete response*, and toxic reaction is often also multi-class as in *not affected*, *mildly affected*, *strongly affected*, *severely affected*.

## 1.1 Aim and Organization of This Thesis

The focus of this thesis is the classification into ordered classes. That means that the outcome of the classification problems we look at is *ordinal*, i.e. takes values from several classes where these classes can be ordered, as in the example above: *partial response* $<$ *major response* $<$ *complete response*.

Approaches that classify into an ordinal outcome can be compared to several alternative strategies.

- Binary classification. Compared to the commonly used binary classification, taking into account all available classes enables the classifier to produce more detailed results.

- Non-ordinal multi-class classification. One can classify into the available categories while ignoring the order among these. In comparison to such approaches we want to examine to what extent the information that is given in the order of the outcome classes can be used to improve the classification.

- Regression. Instead of ignoring the order between the outcome classes one can choose to emphasize it: One can transform the outcome classes to numbers on a continuous scale (e.g. *partial response* = 0; *major response* = 1; *complete response* = 2) and perform a regression. Using that approach one has to introduce a metric and impose a distance between the classes and one has to choose a model, e.g. a linear relation between predictors and this transformed outcome.

Additionally, we focus on approaches that are able to deal with high dimensional data as we are especially interested in molecular features, which will typically produce high dimensional data sets.

The aim of this thesis is threefold: We want to introduce our new method *hierarchical twoing (hi2)*, we want to evaluate the performance of existing methods in comparison to hi2, and we want to discuss the benefits from directly considering the ordinal response structure for the classification.

This thesis is organized as follows: We start with a motivating example in section 1.2 that presents a typical scenario that would benefit from improved classification based on the ordinal response structure. The following section deals with high dimensional data (section 1.3) concluding the introduction. Chapter 2, *Methods*, starts with an important section on how to correctly evaluate a classifier when the response is on an ordinal scale (section 4.2). In the sequel we present possible existing approaches to ordinal classification in section 2.3.

Our own method hi2 will be presented in section 4.1 in the chapter *Results*. We will present the method behind hierarchical twoing in section 4.1.1 and our implementation within the package hi2 for the statistical programming framework R (R Core Team, 2014) in section 4.1.3.

In between the chapters *Methods* and *Results* there will be a chapter *Material* (chapter 3) detailing the data used to evaluate the different methods in the

results (section 4.2). Both, the material and the evaluation, are split in two parts: First, we will present several real data sets (sections 3.1 and 4.2.2.1) which all contain cancer data, but for different types of cancer and different types of molecular features. We will begin with an evaluation of publicly available data sets from other groups and present the data from our research group after that.

The second part in both, the material and the evaluation, contains a simulation study (sections 4.2.2.2 and 3.2), in which we investigate specific properties of the input data. Different simulation settings are utilized to cover the different data properties.

The results are discussed in chapter 5 and the thesis is wrapped up in the concluding remarks in chapter 6.

## 1.2 Introductory Example: Treatment Response

There are many applications where high-dimensional data should be used to classify into ordered categories. Several biological examples are presented in section 4.2.2.1. As our motivating example in this section we want to present a prediction problem, where therapy response should be predicted for patients with locally advanced rectal cancer. The patients are all treated within the CAO/ARO/AIO-04 study and receive *neoadjuvant* combined radio-chemo-therapy prior to the surgery (Sauer et al., 2012) . The prediction of the efficacy of this pre-operative treatment is one of the questions in the focus of the clinical research group 179 *Biological Basis of Individual Tumor Response in Patients with Rectal Cancer* (Ghadimi, 2014).

Such prediction would be highly valuable in terms of individualized treatment. There is a great variability in the observed response of different patients to that radio-chemo-therapy (Mihaylova et al., 2011). While some patients show complete response – the tumor cells are dead at the end of the treatment – other patients show no response – the tumor continues to grow. And all stages in between can be observed as well. Access to information on the efficacy of the neoadjuvant treatment *prior to the treatment* would be of great help in guiding patients along different treatment paths: Non-responding patients could be spared the pre-operative treatment (with all its toxic side effects) and could

be subjected to surgery right away. Weakly responding patients might benefit from intensified treatment in form of higher dose of radiation or a different set of chemicals. And complete responders might respond similarly well to lower, hence less toxic, dosages of radiation.

An accurate prediction of the level of response to the currently applied radio-chemo-therapy is the prerequisite to even studying such proposals of different treatment.

For the construction of a classifier one reasonable hypothesis is that gene activity in the tumor should be predictive with respect to the response of the tumor to radio-chemo-therapy. Gene expression data is, thus, measured from a biopsy taken prior to any therapy via Microarrays (Platform: Agilent-026652 Whole Human Genome Microarray 4x44K v2).

Figure 1.1 depicts the treatment line for all patients in these studies. Highlighted in red is the desired prediction, where gene expression data from microarrays is used to train a classifier for the treatment response. Treatment response in this case is assessed by the *tumor regression grade (TRG)* as defined by Dworak, Keilholz, and Hoffmann, 1997 (see Figure 1.2). The TRG is a score, that takes one of the 5 levels TRG0, TRG1, ..., TRG4. It is assigned by a pathologist who examines tissue sections from the resection specimen removed during the surgery. It is a semiquantitative measure based on the amount of tumor cells and the amount of fibrosis in the sample.

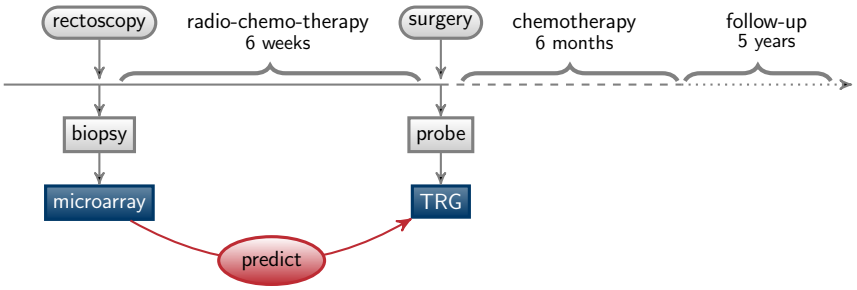



**FIGURE 1.1** *Line of treatment for rectal cancer patients. The therapy is centered around the surgery at which the tumor is removed. Following the surgery there is the* adjuvant *therapy which is a chemo-therapy with a duration of six months. The* neoadjuvant *therapy is a combined radio-chemo-therapy which lasts 6 weeks. There is a biopsy taken prior to the therapy, from which gene expression was measured using microarrays. The hypothesis in this application is, that the gene expression is predictive with respect to the treatment response (the tumor regression grade (TRG)) to the neoadjuvant radio-chemo-therapy.*

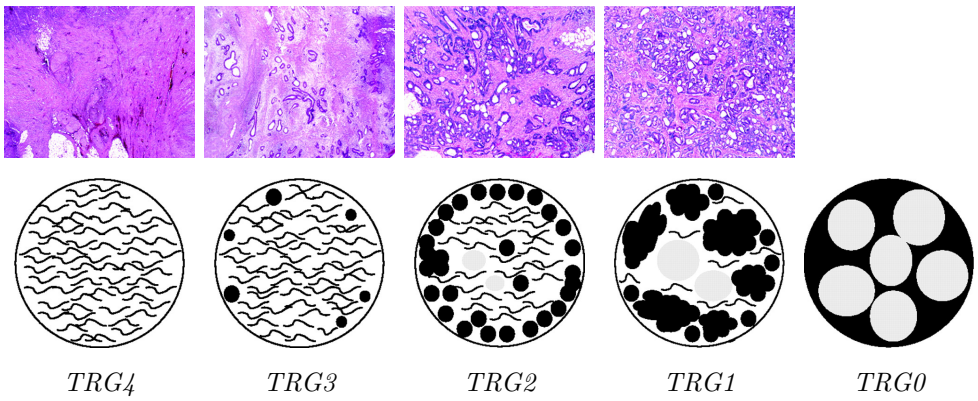



**FIGURE 1.2** *The tumor regression grade (TRG) as described by Dworak, Keilholz,* and *Hoffmann, 1997, is a semiquantitative measure based on the amount of tumor versus the amount of fibrosis visible in the tissue section samples. The top row in this figure gives example images for such tissue section samples (with kind permission from Rödel et al., 2005), the bottom row gives corresponding schematic pictures (with kind permission from Rubbia-Brandt et al., 2007) where areas coloured in black depict cancer cells, areas coloured in gray show necrotic tissue and areas filled with fibrils depict fibrosis. The TRG takes one of the five levels TRG0, TRG1, TRG2, TRG3, or TRG4. A sample is assigned TRG4 when no tumor cells are present and the tumor is replaced completely by fibrosis (complete regression). TRG3 corresponds to tissue with a small number of scattered tumor cells. More tumor cells but still predominating fibrosis is classified into TRG2. In TRG1 the tumor predominates. And in TRG0 there is no tumor regression, so that the sample mostly contains tumor cells.*

In this case the prediction problem translates to training a classifier that uses the gene expression values to predict the TRG. The TRG value assigned by a pathologist is available for all patients, such that this constitutes a supervised classification problem. The classes (TRG0, ..., TRG4) are ordinally scaled and the number of features (29 055 probes with unique probe ID) greatly exceeds the number of samples (231 patients with TRG annotation) which makes this a high-dimensional problem. So, the task to predict the TRG from gene expression exactly fits the scope of this thesis.

The data set will be available under the accession number *GSE40492* from the gene expression omnibus data base (Edgar, Domrachev, and Lash, 2002).

## 1.3 High Dimensionality

Most types of data on molecular features have in common that they are composed of many single features: The ensembl database (Flicek et al., 2012) in its current version 74 has annotation of 20517 protein coding genes, there are

currently 1872 potential miRNAs listed in miRBase (Kozomara and Griffiths-Jones, 2013), the estimated number of different proteins in human cells go up to over 1 000 000 (Nørregaard Jensen, 2004), and profiling of the DNA methylome will generate tens of millions of data points (Lister et al., 2009).

Therefore, molecular data sets will typically encompass many more features (such as genes) than samples (such as patients). Data with

$$p \gg N, \tag{1.1}$$

where $p$ denotes the number of features and the number of samples is represented with $N$, are called *high dimensional* and are challenging in principle as well as technically.

If the number of dimensions increases while the number of samples stays fixed, the samples will be distributed in a growing space, so that the larger space will be covered more and more sparsely.

In order to achieve levels of sample coverage in a high dimensional space comparable to the levels in a low dimensional space, the number of samples has to increase exponentially, which quickly becomes unfeasible even for moderate numbers of dimensions (chapter 2.5 in Hastie, Tibshirani, and Friedman, 2009). If 100 uniformly distributed samples provide a good coverage of the 1-dimensional space, $100^{10}$ uniformly distributed samples are needed to achieve the same coverage in the 10-dimensional space. Since molecular data typically has not 10 but 10 000 dimensions, any realistic sample size will lead to sparse coverage.

This is the sampling aspect of the phenomenon called *curse of dimensionality* (Bellman, 1961), a term which is used to describe all challenges that arise from high dimensionality in the data.

One consequence of the sparse sample coverage is, that local methods such as k nearest neighbours (kNN, Fix and Hodges, 1951; Cover and Hart, 1967) or kernel methods (e.g. the support vector machine (SVM), Bennett and Campbell, 2000) might not work well in high dimensions as with sparse sampling the neighbourhood might become too large.

On the other hand, in many applications, the high-dimensional space is not covered uniformly. Instead, the data often live on manifolds or sub-spaces

where many dimensions are meaningless towards the target and do not add information but are random noise. Such data are called to have a low *intrinsic dimension*. Thus, locally adaptive methods can be successfully applied in many high-dimensional settings (Braun, Buhmann, and Müller, 2008; Kpotufe, 2011). Examples are locally adaptive kNN (Domeniconi, Peng, and Gunopulos, 2002), tree based methods such as CART (Breiman et al., 1984), C4.5 (Quinlan, 1986; Quinlan, 1993), and C5.0 (Kuhn and Johnson, 2013) or kernel based methods (Bennett and Campbell, 2000).

Dimension reduction techniques can help to deal with high dimensional data as well. Again, there are locally adaptive methods that try to reflect the structure of a possible manifold embedded into the high dimensional space. Prominent examples here are Isomap, locally linear embedding, kernel PCA, or self organizing maps. In contrast to these locally adaptive methods there are also global projection methods that transform the data into spaces where the dimensions carry information on the structure of the data. Here, prominent examples are principal component analysis PCA that maximizes correlation, or independent component analysis ICA, that maximizes independence. Often projection methods are followed by a *feature selection* – where only the first and most informative features are retained – and are, hence, often inaccurately called feature selection methods in the literature.

Transforming the data – also called *feature extraction* – can lead to better discriminatory properties since the transformed data is, for example in PCA, ordered by variability. The downside is that the transformed features do not have a direct physical interpretation.

Therefore, the even more direct approach to handle high dimensional data by selecting a subset of informative features without transformation is often preferred. The physical meaning is retained in this approach and the information on the selected features itself is also highly valuable. In the introductory example, for instance, not only a prediction of the therapy response would be useful, but also a list of predictive genes which could be potential drug targets.

There are several directions to follow in order to select the informative features (Guyon and Elisseeff, 2003; Saeys, Inza, and Larrañaga, 2007). Depending on when the feature selection is performed in relation to the classification, three categories of feature selection methods can be distinguished:

1. *Filter* methods separate the feature selection from the classification and perform the feature selection prior to any classification. This independence from the classification makes it possible to compare different classifiers on the same data and keeps the computational costs relatively low. The main disadvantage is that filter methods ignore possible interactions between the feature set and the classifier. Examples are univariate tests such as t-test or ANOVA.

2. *Wrapper* methods integrate the feature selection with the classification. Several feature sets are defined and used in the classification. This has the advantage that possible interactions between the feature set selection and the classifier are captured. Of course, the price is an increased risk of overfitting as well as a heavier computational burden. Prominent examples are simulated annealing, or genetic algorithms.

3. Some classification algorithms have an *embedded* feature selection. These algorithms do not need to be combined with an external feature selection, but implicitly perform their own feature selection. Examples here include classification trees or SVMs.

Another categorization can be deployed based on the variable selection itself. *Ranking based* methods evaluate the predictive power of each feature individually, whereas *subset based* methods try to select subsets of features that have together predictive power.

As we will discuss in Section 4.1, hi2 can be able to handle high-dimensional data depending on the supplied binary base learner. If the provided binary base learner can handle high-dimensional data, then that transfers directly to hi2.

# 2 Methods

In this chapter we will present several established methods that can be used to classify samples into ordered categories in a high-dimensional feature space (section 2.3). We will use these methods as comparisons in the evaluation of the performance of our method hi2. To enable us to do such benchmarking and to increase our understanding of the particular demands on ordinal classifiers, (section 4.2) will discuss how to properly evaluate a classifier when the response is on ordinal scale.

## 2.1 Building and Evaluation of Classifiers

This section is based on chapter 7.2 of Hastie, Tibshirani, and Friedman, 2009 and starts with some notation: We want to train a classifier that uses data in $p$ features on $N$ samples. These would be $p = 29\,055$ expression values on $N = 231$ patients in the rectal cancer data from the introductory example. We will use $X$ to denote the feature variables and $G$ to denote the response variable. There are $L$ different class (or group) labels (the $L = 5$ TRG levels in the rectal cancer data) which are collected for each sample in a vector $\mathbf{g} = (g_1, \ldots, g_N)$. The values of the $p$ variables are recorded in the matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ where each component $\mathbf{x}_i$ is again a vector of length $p$ containing the values of all $p$ measured variables on sample $i$. We assume that there is an unknown relation $f$ from the features to the class:

$$f(X) = G \tag{2.1}$$

In this notation, a classifier is a function $\hat{f}$ that produces estimates $\hat{f}(x)$ where $x$ is one of the $\mathbf{x}_i,\ i = 1, \ldots, N$.

(We will apply the functions $f(.)$ and $\hat{f}(.)$ as well as the below defined functions $L(.,.)$ and $p_l(.)$ to both, matrices (such as $\mathbf{X}$, e.g.) and vectors (such as $x$, e.g.), but for the sake of simpler notation we will not distinguish these cases with different symbols.)

Most classifiers will be imperfect, and the class label $\hat{f}(x)$ that is assigned by the classifier will be different from the true label $f(x)$ in some cases. The important question is, whether a given classifier is useful despite producing wrong results in some cases, and whether this classifier is doing better than some alternative classifier. Answering these questions is called *model assessment*. Now to assess the classifier, we use a test set $\mathcal{S} \subseteq \{1, \ldots, N\}$ of $|\mathcal{S}| = n$ samples. We will use $\mathbf{X}^{\mathcal{S}}$ as a shorthand for $(x_{ij})_{\substack{i \in \mathcal{S} \\ j=1,\ldots,p}}$ and $\mathbf{g}^{\mathcal{S}}$ as short hand for $(g_i)_{i \in \mathcal{S}}$. Model assessment is based on a comparison of the true classes $\mathbf{g}^{\mathcal{S}}$ to the predicted classes $\hat{f}(\mathbf{X}^{\mathcal{S}})$.

In classification problems the most commonly applied measure of classification performance is the *accuracy*:

$$\text{accuracy}(\mathbf{g}^{\mathcal{S}}, \hat{f}(\mathbf{X}^{\mathcal{S}})) = \frac{\sum_{i \in \mathcal{S}} I(g_i = \hat{f}(\mathbf{x}_i))}{|\mathcal{S}|}. \tag{2.2}$$

where

$$I(c) = \begin{cases} 1, & c \text{ true} \\ 0, & c \text{ false} \end{cases} \tag{2.3}$$

is the indicator function.

The accuracy is closely related to the *0-1 loss*:

$$L(\mathbf{g}^{\mathcal{S}}, \hat{f}(\mathbf{X}^{\mathcal{S}})) = \sum_{i \in \mathcal{S}} I(g_i \neq \hat{f}(\mathbf{x}_i)) \tag{2.4}$$

Both, accuracy and 0-1 loss are defined on the basis of the class predictions $\hat{f}(\mathbf{X}^{\mathcal{S}})$. Alternative measures can be defined on the basis of class probabilities $\hat{p}(\mathbf{X}^{\mathcal{S}})$ where for each of the $L$ class labels $\hat{p}_l(x) = Pr(G = l \mid x), l = 1, \ldots, L$.

On the basis of class probabilities the typical measure is the *log likelihood*:

$$loglik(\mathbf{g}^{\mathcal{S}}, \mathbf{X}^{\mathcal{S}}) = \sum_{i \in \mathcal{S}} \sum_{l=1}^{L} I(g_i = l) \log \hat{p}_l(\mathbf{x}_i) \tag{2.5}$$

where $L$ : number of classes.

The related loss function is -2 $\times$ the log likelihood.

The log-likelihood – in contrast to the accuracy – uses not only the class predictions but the class probabilities. Thus, the log-likelihood is able to differentiate classifiers that produce the same class prediction, but where one is more confident in the classification: High confidence in correctly classified samples (i.e. $Pr(G = l \mid x)$ close to 1) adds up to a higher log-likelihood than low confidence (i.e. $Pr(G = l \mid x)$ not much higher than $1/L$). Unfortunately, not all classifiers compute class probabilities. A prominent example is the 1-nearest-neighbour classifier.

The adequacy of both measures, the accuracy and the log-likelihood, for settings where the response $G$ is ordinal is discussed below.

All these measures assess the difference (or equality) between the predictions $\hat{f}(\mathbf{X}^{\mathcal{S}})$ and the true values $\mathbf{g}^{\mathcal{S}}$ and are, thus, a measure of the error the classifier commits. Depending on the data used to evaluate these measures, the error that is estimated might be the *training error* (also called *apparent error*) or the *test error* (also called *generalization error*). If the loss function is evaluated on the same data that was used to train the classifier ($\mathcal{S} = \mathcal{T}$, $\mathcal{T} \subseteq \{1, \ldots, N\}$ the training set), it measures the training error, which is defined as the average loss on the training samples:

$$\text{training error} = \frac{1}{N} \sum_{i=1}^{N} L(g_i, \hat{f}(x_i)) \tag{2.6}$$

Since the data used to evaluate the classifier was also used to train the classifier when considering the training error, the training error will generally overestimate the performance of the classifier when it comes to new and unseen data. The test error is evaluated on data that has not been used in the training phase ($\mathcal{S} \cap \mathcal{T} = \emptyset$) and is a much better indicator of the performance of a classifier. Figure 2.2 shows the difference of training and test error in a model tuning setting.

There are methods that try to mathematically adjust the training error in order to obtain an estimation of the test error. Most prominent examples are methods that adjust for model size like the Akaike Information Criterion (AIC)

or the Bayesian Information Criterion (BIC). These rely on model assumptions and need (an estimate of) the model degrees of freedom, which might not be obvious in all cases.


**FIGURE 2.1** *Data partitioning for predictive modelling. To perform the* model assessment *of the final model, an unseen* test set *is needed. And before that, for the* model selection *(e.g. parameter tuning) there is also need for an unseen data set, which is also called* validation set*. Different schemes are applied to arrive at such a data partition. Depicted here is the* hold-out *approach, where the model is trained with different parameters on the training set (coloured in red) and evaluated on the validation set (coloured in blue). The parameter resulting in the best performing model is chosen and used to train the classifier on the combined training and validation set. The performance of this final model is then evaluated on the test set (coloured in yellow).*

Alternatively, the test error can be estimated directly given an independent set of samples. For that, a set of samples needs to be hold out from the training of the classifier. These hold out samples form an independent data set that is used solely for the calculation of the error. Therefore, the full data set is typically split into a *training set* $\mathcal{T}$ and a *test set* $\mathcal{S}$. Often the training set is further split into the actual training set and a *validation set* $\mathcal{V}$ (see Figure 2.1). This second split is due to *model selection*: Many classification algorithms can be adapted to the data by tuning parameters. An often applied and general approach to set such parameters is a *grid search* in the parameter space. When a grid search for the best value of a parameter is conducted, values for this parameter are selected in advance and the classification is trained and evaluated for each of these values. The value giving the best classification performance (e.g. measured by the lowest loss) is then used.

Especially in the process of model selection it is of crucial importance to evaluate the performance of that classifier on unseen data, i.e. to estimate the generalization error. Without a careful evaluation of the classifiers performance on unseen data it is easy to tune too much so that the classifier is adapted too much to the training data (a phenomenon called *overfitting*).

The parameter $k$ in the $k$-nearest-neighbors classifier (knn) is an illustrative example. The nearest neighbour classifier segments the feature space into regions that are assigned to one class label by looking at the $k$ nearest data points from the training data. Usually the Euclidean distance is used as metric.

Thus, the parameter $k$ controls the size of the neighborhood in the feature space that is used to determine the class of any given point in that feature space. If in the extreme case $k = 1$ is chosen then the classifier will be obviously 100% accurate on the training data, but will most likely be overfitting and generalize poorly to other unseen data. Figure 2.2 demonstrates this effect when the gene expression data from the rectal cancer patients (see section 1.2) is used to predict the patients' pathological lymphnode status: While the accuracy on the training data rises from about 70% to 100% when the neighbourhood shrunk from 10 to 1, the accuracy on unseen test data decreases from 66% to 53%.

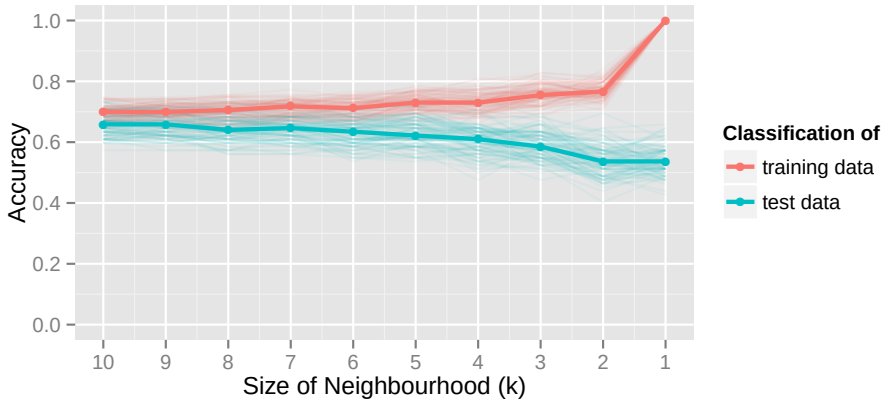



**FIGURE 2.2** *Model tuning. Many models have parameters that allow some tuning to adapt to the data at hand. Often there is a trade-off between a model that is tuned very much to the training data and a model that generalizes well. The plot shows on the y-axis the accuracy of a k-nearest-neighbours classifier for different values of k (displayed along the x-axis). These are results obtained from the rectal cancer gene expression data set when the response is the pathological lymphnode status after neoadjuvant treatment and surgery (0,1,2 - according to uicc tnm classification). Shown are results from 100 runs, where in each run the data has been split randomly into 2/3 training data and 1/3 test data. The red curves show the accuracy on the training itself (*re-classification*) and the blue curves show the accuracy on the test data. The thin and transparent lines give the individual curves from the individual runs and the solid thick curve is the median from all runs. When the neighbourhood is chosen to consist only of the closest data point (k=1) then the accuracy on the training data rises to 100 %. This is not reflected on the test data.*

There are several ways of how to arrive at independent training and test sets.

The *Hold Out* approach is the most straight forward way, where the data is simply split into training and test set (or – more complex – according to the scheme of Figure 2.1). The test set is set aside and used only at the very end when the performance of the classifier is evaluated. The *hold out* scheme is

often applied as it is simple to understand and easily deployed. The typical split uses 2/3 of the samples for training and the remaining 1/3 for the test (Dobbin and Simon, 2011). The classifier will be biased as it does not use all available data, but only a fraction (e.g. 2/3) of the data. And the split itself might introduce a bias, for instance in the case that all extreme samples go into the test set. Therefore, a non-random but careful selection of the test set might be appropriate to ensure that the class fractions are similar in the training and test set or to maximize the difference between the samples in the training set. See Martin et al., 2012 for a discussion on data splitting.

In many applications, data are scarce and the *hold-out* approach will perform poorly (Molinaro, Simon, and Pfeiffer, 2005): If the full data set is small already, holding out further samples from the training will result in a poorly fitted classifier. Additionally, the hold-out test set might be too small to get reliable estimates on the test error. To quote Hawkins, Basak, and Mills, 2003: "Hold-out samples of tolerable size, by contrast, do not match the cross-validation itself for reliability in assessing model fit and are hard to motivate."

Therefore, efforts have been made to use all the data for both, training and testing, through resampling techniques. Most prominent representatives of resampling techniques are *cross-validation (CV)* which is used in the analyses presented later in this work and *bootstrapping*.

Cross validation increases the data usage while still staying close to the hold out approach. In a $V$-fold cross validation, the data are split into $V$ (roughly) equally sized parts. Each of these $V$ parts is used as a test set for a classifier that is trained on the $V - 1$ other parts. Thus, $V$ classifiers are trained and evaluated. The reported performance is then the average performance across the $V$ folds.

There are two ways commonly applied to obtain confidence intervals for the error estimates resulting from the cross validation. Unfortunately, both are flawed. First, one can aggregate the predictions of the $V$ folds. Since the data are split into the $V$ parts, each sample appears as part of a test set once, so that the cross validation produces one prediction for each sample. These aggregated predictions can be used to calculate a confidence interval of the performance measure. In case of the accuracy, for example, the aggregated predictions can be treated as a sample drawn from a binomial distribution and

the confidence intervals can be based on that distribution (e.g. through the method by Clopper and Pearson, 1934). This distribution assumption does not hold, though, as the individual predictions are not independent, because the training sets overlap to a great extent. This leads to overly narrow confidence intervals (Jiang, Varma, and Simon, 2008).

A second widely used approach is to do repeated cross validation and to use the empirical distribution of the error estimates for these repetitions to obtain empirical confidence intervals. Vanwinckelen and Blockeel, 2012 however demonstrate, that these empirical confidence intervals are misleading. While repeating the cross validation does reduce the variance in the estimate (which is known to be quite high in cross validation, see Kohavi, 1995), the estimate is still biased and the confidence might even become too narrow to include the performance measure (test error, e.g.).

Since confidence intervals around the estimates produced by cross validation are, thus, still an unsolved problem, we follow Hastie, Tibshirani, and Friedman, 2009 and resort to show only standard errors without any claim on the significance.

As a final note on cross validation we want to discuss the parameter $V$ that controls the number of folds in a cross validation. $V$ balances bias and variance: The lower $V$ is chosen the smaller is the subset on which the classifier is trained, which increases the bias introduced by doing a cross validation (Vanwinckelen and Blockeel, 2012): The performance of the trained classifier must be expected to be worse, when less samples are given to the training set. On the other hand, the larger $V$ is chosen, the smaller are the individual test sets, which increases the variance in the estimate. "There is no free lunch" and we are not aware of any formal analysis detailing the choice of $V$ in relation to particular aspects in the data. But it is quite generally agreed, that 10-fold cross validation is a good compromise and yields good results (Hastie, Tibshirani, and Friedman, 2009).

Throughout this work the performance estimation is based on 10-fold cross-validation. But still we want to mention here that *bootstrapping* is another possible re-sampling method which can be used for the purpose of estimating classifier performance. The bootstrap was introduced by Efron, 1979 and its

application to model assessment is discussed especially in Efron and Tibshirani, 1986.

In a simulation (as in our simulation results presented in section 4.2.2.2) there is no need for re-sampling. Here one can simply use the simulation to draw more samples: If we are interested in estimating some measure $\alpha$ (the accuracy, e.g.), we can perform $R = 1000$ simulation runs – where in each run a sample is drawn from the chosen distribution – to produce an estimate for $\alpha$ which we call $\hat{\alpha}_r,\ r = 1, \ldots, R$. The estimate from the simulation is then calculated as the average over all estimates from the individual simulation runs

$$\bar{\alpha} = \frac{1}{R} \sum_{r=1}^{R} \hat{\alpha}_r \tag{2.7}$$

With real data, however, we do have a data set of fixed size and we can not simply draw more data samples from the underlying (unknown) distribution. Therefore, the bootstrap resorts to draw the needed samples from the original data set itself. Each bootstrap sample is of the same size as the full data and is drawn from the full data set with replacement. The estimation of the performance measure is done on the samples that are not drawn (the *out-of-bag* samples). The final estimate is then calculated according to 2.7.

Similar to the CV, the training of the classifier is not done on the full data set but on the bootstrap sample omitting the out-of-bag samples. Thus, again there is a bias of the estimate to be pessimistic. Some extensions of the bootstrap have been proposed to minimize this bias. On average, 63.2 % of the samples in the full data set will be part of the bootstrap sample. To compensate for the smaller training set and the resulting weaker performance of the classifier, the 632-bootstrap (Efron, 1983) uses 63.2% of the (pessimistic) bootstrap estimate and the remaining 36.8% of the (optimistic) estimate from reclassification on the training set. Further improvements in the 632+ bootstrap are discussed in Efron and Tibshirani, 1997.

## 2.2 Suitable Error Measures for Ordinal Response Settings

In the previous section we have seen how to get an estimate of a performance measure even in the case of a limited number of available samples. In this section we want to discuss which measures are suitable in the case of ordinal response.

### *2.2.1 Accuracy*

The accuracy is the common way of classifier evaluation. It is simple, interpretable and established. But it does not take into account the ordinal structure of the response.

The accuracy is also called *percentage agreement* or *correct classification rate* in the literature. Equivalent measures are the 0-1 loss or the *misclassification error rate*.

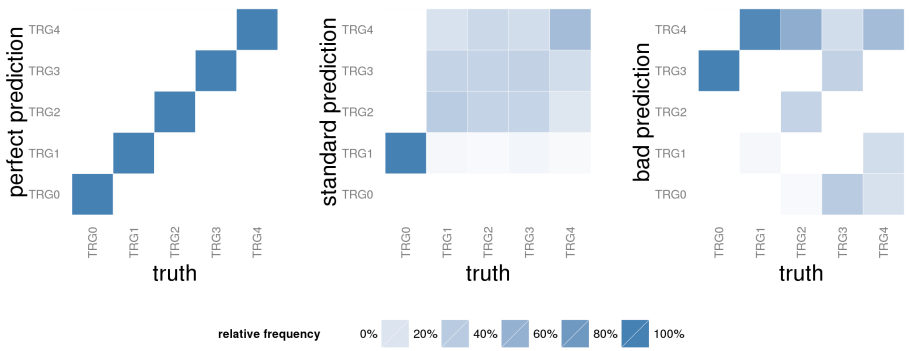



**FIGURE 2.3** *Assessing different predictions. The rectal cancer data contains samples from 5 groups. Here we display three different predictions of that grouping: the left panel shows a perfect prediction, the center panel shows the actual prediction from an SVM, the right panel shows the same prediction but manipulated to be worse.*
*For all three predictions the Figure presents a heatmap of the contingency table where each cell contains the fraction of all samples in that column that fall into that cell. Thereby, dark colours represent higher numbers.*
*The entries in the right panel are constructed from the real prediction in the center panel by 'worsening' the prediction by two levels (the prediction of TRG1 for the sample with true class TRG0 is worsened to TRG3, e.g.).*
*The accuracy for the clearly worse prediction in the right panel is the same as for the better prediction in the center panel, while Kendall's $\tau$ is able to catch that difference.*

Equation 2.2 has the definition of the accuracy. It only depends on the number $|\mathcal{S}|$ of samples in the test set and the number of samples where the classification is correct ($\sum_{i \in \mathcal{S}} I(g_i = \hat{f}(\mathbf{x}_i))$). What is lacking for the ordinal case is some distinction among the wrongly classified samples.

As a demonstration we use a re-classification, resulting from an SVM trained on the rectal cancer data (see Section 1.2) and applied to the same (training) data again. The results are shown in Figure 2.3 in the center panel where we show a contingency table of the true classes (columns) vs. the classification result (rows). Additionally, the left panel shows a perfect classification and the right panel shows a manipulated version of the predictions of the SVM. For that manipulation every classification that was incorrect was worsened two additional levels (if possible), such that a classification result 'TRG1' for a true value 'TRG0' was shifted to 'TRG3' in the manipulated version. Obviously, the manipulated classification is worse than the 'original' SVM result. But since the manipulation did not change the numbers of correctly classified samples (the diagonal elements in the contingency table are the same), the accuracy is the same for both, the SVM classification as well as the manipulated version.

As a consequence, the accuracy should not be used solely to evaluate classifiers for ordinal response, but at most in conjunction with a measure which is capable of capturing the 'level of incorrectness' (Baccianella, Esuli, and Sebastiani, 2009).

In the literature, several measures have been proposed. Natarajan et al., 2007 propose to use not a single measure but to take all possible binary splits of the response and calculate Cohen's $\kappa$ for each of the resulting 2x2 tables. The advantage is, that the two vectors that are compared (in our case the truth and the classification result) need not be on the same scale, but one could have a higher number of levels. And additionally, this method reveals which splits are easier to classify and which are harder by giving a separate measure for each split. But since this approach does not produce a single number, it is not very well suited as a measure to benchmark different classifiers.

For the comparison of the true class levels with the classification result we do not need the flexibility that is offered by Natarajan et al., 2007. Instead it is possible to directly view the true class levels $\mathbf{g}^{\mathcal{S}}$ and the classification result $\hat{f}(\mathbf{X}^{\mathcal{S}})$ as two raters and calculate a measure for inter-rater-agreement which

is apt for ordinal data with ties. A common choice is the weighted Cohen's $\kappa$ (Cohen, 1960, Cohen, 1968) which is also used internally in Natarajan et al., 2007. Weighted Cohen's $\kappa$ is basically the ratio of the observed agreements to the expected agreements where the disagreements are weighted increasingly with higher distances. Thus, these weights punish wrong classification. Typical weights increase linearly or quadraticly with the difference between the classification results and the truth. The weights of Cohen's $\kappa$ are new (hyper-)parameters which do not have the obvious best default.

This is why, as an alternative, Kendall's $\tau$ is proposed as a measure for the performance of ordinal classifiers (Baccianella, Esuli, and Sebastiani, 2009, Sánchez-Monedero et al., 2013).

### 2.2.2 Kendall's $\tau$

Kendall's $\tau$ (Kendall, 1938) is a non-parametric measure of correlation between two paired vectors. A short introduction to Kendall's $\tau$ is in Noether, 1981; the main reference is the book by Kendall, 1975.

Formally, we want to measure the correlation between the prediction of a classifier $\hat{f}(\mathbf{X}^{\mathcal{S}}) = (\hat{g}_i)_{i \in \mathcal{S}}$ and the true class label $\mathbf{g}^{\mathcal{S}} = (g_i)_{i \in \mathcal{S}}$ where $\mathcal{S}$ is the set of samples in the test set under inspection.

The idea of Kendall's $\tau$ is to compare pairs of samples, and to calculate a measure of correlation based on the concepts of *concordant* and *discordant* pairs: All $\binom{n}{2} = \frac{1}{2}n(n-1)$ possible pairs $(g_i, g_j), i \neq j \in \mathcal{S}$ are compared to the corresponding pair $(\hat{g}_i, \hat{g}_j)$. If $g_i < g_j$ and $\hat{g}_i < \hat{g}_j$ these pairs are called *concordant*, if $g_i < g_j$ and $\hat{g}_i > \hat{g}_j$ they are called *discordant* – analogously for the opposite direction.

The basic definition of Kendall's $\tau_{\mathrm{a}}$ is:

$$\tau_a \coloneqq \frac{n_c - n_d}{\frac{1}{2}n(n-1)} \tag{2.8}$$

where $n$ is the number of ranked samples,

$n_c$ is the number of concordant pairs, and

$n_d$ is the number of discordant pairs.

It is easily visible from 2.8 that $\tau_{\mathrm{a}}$ takes values between -1 and 1, where $\tau_a = 1$

if all pairs are concordant, i.e. perfect correlation and $\tau_a = -1$ if all pairs are discordant, i.e. perfect negative correlation. If there are as many concordant as discordant pairs – which is *un-correlation* in terms of Kendall's $\tau$ – $\tau_a = 0$.

Any pair $(g_i, g_j)$ with $g_i = g_j$ is called a *tie* in $\mathbf{g}^{\mathcal{S}}$. Similarly, a pair $(\hat{g}_i, \hat{g}_j)$ with $\hat{g}_i = \hat{g}_j$ is a tie in $\hat{f}(\mathbf{X}^{\mathcal{S}})$. In the classification setting where there are only a few (e.g. 5) classes, there are always many ties.

The measure in equation 2.8 is called $\tau_\mathrm{a}$ because it does not account for ties.

If there is a tie in either $(g_i, g_j)$ or $(\hat{g}_i, \hat{g}_j)$, this pair is neither concordant nor discordant and gets a 'score' of 0, i.e. contributes neither to $n_c$ nor to $n_d$. Thus, the nominator in the definition of $\tau_a$ gets smaller.

It is possible to still use $\tau_\mathrm{a}$ also in the situation of existing ties. But the (probably not desired) consequence is that perfectly correlated vectors with ties would be given a $\tau$ smaller than 1 (see table 2.1).

**TABLE 2.1** *Perfect classification result. In this example the classification result $\hat{f}(\mathbf{X})$ perfectly matches the true class labels $G$. Still, Kendall's $\tau_a$ comparing $\hat{f}(\mathbf{X})$ and $G$ is 2/3 (as there is one tie and only 2 out of three possible pairs are counted). Kendall's $\tau_b$ was developed to give 1 in situations like this.*

| | | | |
|---|---|---|---|
| $G$ | 0 | 0 | 1 |
| $\hat{f}(\mathbf{X})$ | 0 | 0 | 1 |

The alternative measure $\tau_b$ subtracts the indistinguishable (because tied) pairs from the total number of possible pairs. This is done for both vectors $(g_i)_{i\in\mathcal{S}}$ and $(\hat{g}_i)_{i\in\mathcal{S}}$. If we call the number of ties in the true class labels $T$ and in classification result $U$ Kendall's $\tau_b$ is defined as

$$\tau_b := \frac{n_c - n_d}{\sqrt{(\frac{1}{2}n(n-1) - T)(\frac{1}{2}n(n-1) - U)}} \tag{2.9}$$

where pairs tied in at least one ranking do not count to $n_c$ nor to $n_d$. In the case of no ties $\tau_a = \tau_b$ as it should be.

Note that in case one of the vectors is fully tied – in our case if the classifier puts all samples into the same class – the denominator of $\tau_b$ becomes 0 and, thus, $\tau_b$ is not computable.

Other measures that are applicable in the ordinal case are the *mean absolute error (MAE)* or the *average mean absolute error (AMAE)* (Baccianella, Esuli, and Sebastiani, 2009). Both count the number of levels each prediction $\hat{g}_i$ is away from the true class level $g_i$ and take the mean. While the MAE simply takes the mean over all samples, the AMAE calculates the MAE within each group separately and averages over all groups in the end. The AMAE was developed to get a more robust measure in case of unbalanced groups. As alternative to the average MAE, also maximum MAE and minimum MAE have been proposed (Cruz-Ramirez et al., 2011).

Cardoso and Sousa, 2011 propose the *ordinal classification index* as a new measure. The ordinal classification index tries to include a measure of consistent direction (as Kendall's $\tau$) and a measure of distance between the vectors. The main argument in their paper is that Kendall's $\tau$ only measures the consistent ordering but is missing any notion on absolute difference. Their measure is defined as a minimal path through a graph representing the contingency table between $\hat{f}(\mathbf{X}^{\mathcal{S}})$ and $\mathbf{g}^{\mathcal{S}}$. It is not a single formula but needs some implementation and is computationally involved. The benefits on real data are hard to estimate. Thus, the proposed ordinal classification index has not seen much adoption.

## 2.3 Related Work

This section introduces some methods that can be applied for classification with ordinal response. Some of these are not applicable in high-dimensional settings and some are not targeted to ordinal responses specifically. Table 2.2 on page 49 gives a systematic overview. This overview is by far not exhaustive, as there are many methods for classification and regression; listed are only those methods that are prominent and relate to ordinal classification. In that table the part listing the available ordinal methods usable for high-dimensional problems is the largest, which suggests that there are many methods available. This impression is misleading, though. The table is just the most detailed in that section. To date ordinal classification has not received much attention compared to nominal classification, although there has been increasing attention given to ordinal classification in recent years (Cardoso and Sousa, 2011; Lin

and Li, 2012; Archer and Williams, 2012; Galimberti, Soffritti, and Maso, 2012; Sánchez-Monedero et al., 2013; Leha, Jung, and Beißbarth, 2013).

To the knowledge of the author, in the field of ordinal classification within a high-dimensional feature space a systematic review and comparison of the available classification methods is still missing and beyond the scope of this work. But we will provide comparisons involving 8 methods from the list in table 2.2. Also, Sánchez-Monedero et al., 2013 take a good step into the direction of providing a thorough comparison.

As we have seen in the introduction already, ordinal classification is a problem located between classification and regression, where classification neglects the ordering in the response, while regression forces the response into a specific metric. Both ways are possible in principle, so that both ways appear in the list of applicable classification methods: linear discriminant analysis (LDA Fisher, 1936) and numeric regression (linear regression, e.g. Hastie, Tibshirani, and Friedman, 2009) are direct examples. Both, LDA and numeric regression, are not directly suited for high-dimensional data, though. Through penalization techniques such as regularization (Friedman, Hastie, and Tibshirani, 2010; Hastie, Tibshirani, and Friedman, 2009) or boosting (Bühlmann and Hothorn, 2007a), it is possible to apply many low dimensional models on high-dimensional data.

There are methods that are known to work well in high dimensions and do not need to be applied within a penalization framework. Most prominent examples are classification trees (Breiman et al., 1984; Quinlan, 1986) and the support vector machine (SVM Bennett and Campbell, 2000). Both of them are capable of performing both, classification and regression.

Some methods have been translated to the ordinal case: Classification trees can be built using ordinal split functions (Archer and Mas, 2009; Archer, 2010; Galimberti, Soffritti, and Maso, 2012), L1-penalization is available due to Archer and Williams, 2012. Support vector machines have been adapted to the ordinal setting (Chu and Keerthi, 2007), the k nearest neighbours are extended to the 'k k nearest neighbours' (kknn Hechenbichler and Schliep, 2006).

Additionally, there are some dedicated ordinal methods. The most common ordinal models (the cumulative link model (clm Agresti, 2010) and the continuation ratio model (crm Cox, 1988; Agresti, 2010)) are again not applicable

to high-dimensional data directly. The pairwise class distances for ordinal classification (PCDOC Sánchez-Monedero et al., 2013) have recently been developed solely for ordinal response data and are capable to classify using high-dimensional data.

Finally, the method by Frank and Hall, 2001 and our method (Leha, Jung, and Beißbarth, 2013) fall out from this scheme, as they are classification schemes that transform a binary base learner into an ordinal classifier. If that base learner is able to classify high-dimensional data, then the whole classifier is as well.

### *2.3.1 Non High Dimensional Methods*

This section describes classification methods that can not be directly applied to high-dimensional data due to reasons already sketched in Section 1.3.

Such methods are still worth to consider even if the feature space is high-dimensional. But in order to apply these methods it is necessary to wrap the classifier into a rigorous feature selection (see again Section 1.3).

#### 2.3.1.1 Non-Ordinal Methods

Due to the high number of available methods to perform general classification or regression, it is impossible to list and compare all of them. Instead we focus on three methods here, all of which are linked to the ordinal case: multinomial regression is the non-ordinal counterpart to the cumulative link model, linear discriminant analysis will be used as part of the base learner in hi2, and linear regression is an example of how regression can be used to do ordinal classification and is, therefore, part of the comparison later in this work – in a penalized (boosted) version.

**Linear Regression** Linear regression is a quite simple and yet highly powerful statistical technique to model given data in order to infer characteristics of either the given data or new and unseen data. Due to its simplicity it is widely applicable and well understood and forms the basis of many more advanced statistical methods.

There is extensive literature on linear regression, see for instance chapter 3 in Hastie, Tibshirani, and Friedman, 2009 with the references therein.

Linear regression is based on the assumption that there is a linear relationship between the predictors and the response. In that case, the model assumption is similar to Equation 2.1 and can be written like this:

$$Y = f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \epsilon, \text{ or more concrete} \tag{2.10}$$
$$y_i = f(\mathbf{x}_i) = \beta_0 + \sum_{j=1}^{p} x_{ij} \beta_j + \epsilon, \; i \in \mathcal{T}$$

where $Y$ : real-valued response,
$X = (X_1, \ldots, X_p)$ : feature vector,
$\mathbf{y} = (y_i)_{i \in \mathcal{T}}$ : real-valued response vector,
$\mathbf{X}^{\mathcal{T}} = (x_{ij})_{\substack{i \in \mathcal{T} \\ j=1,\ldots,p}}$ : matrix of feature vectors,
$p$ : number of features,
$\mathcal{T}$ : training set,
$\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ : unknown model parameters, and
$\epsilon$ : mean-zero random error
(assumed to be independent of $\mathbf{X}$).

Thus, the response $Y$ is assumed to be the sum of an intercept $\beta_0$ and a linear combination of the features (modulo the model error $\epsilon$).

The task in a linear regression is now to derive estimates $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \ldots, \hat{\beta}_p)$ for the true parameters $\boldsymbol{\beta}$.

The common approach here is *least squares* which minimizes the model error measured by the residual sum of squares:

$$RSS = \sum_{i \in \mathcal{T}} (y_i - \hat{y}_i)^2, \tag{2.11}$$

where $\hat{Y}^{\mathcal{T}} = (\hat{y}_i)_{i \in \mathcal{T}} = (\hat{\beta}_0 + \sum_{j=1}^{p} x_{ij} \hat{\beta}_j + \epsilon)_{i \in \mathcal{T}}$. The minimization is possible and results in the estimates

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \tag{2.12}$$

where $\mathbf{X} = (\mathbf{1}, \mathbf{X})$ is the matrix of feature vectors with the first column 1 (for the intercept).

As an example of a linear model fit, Figure 2.4 shows the results of a linear model (y axis) to model the TRG (true values on the x axis) using the 6 genes, where a univariate linear model for the TRG resulted in a q value (Storey and Tibshirani, 2003) $< 0.1$. As one can see from Equation 2.10, linear regression is not a classification method, but instead models continuous responses. To apply a linear regression model to our classification problem, we map the TRG levels to the numerical values 0, 1, 2, 3, and 4. The model will result in real values as predictions which is shown in the left panel of Figure 2.4. These real numbers are then rounded to the next full number in the set of allowed answers 0, 1, 2, 3, or 4 which is shown in the right panel of the same Figure.

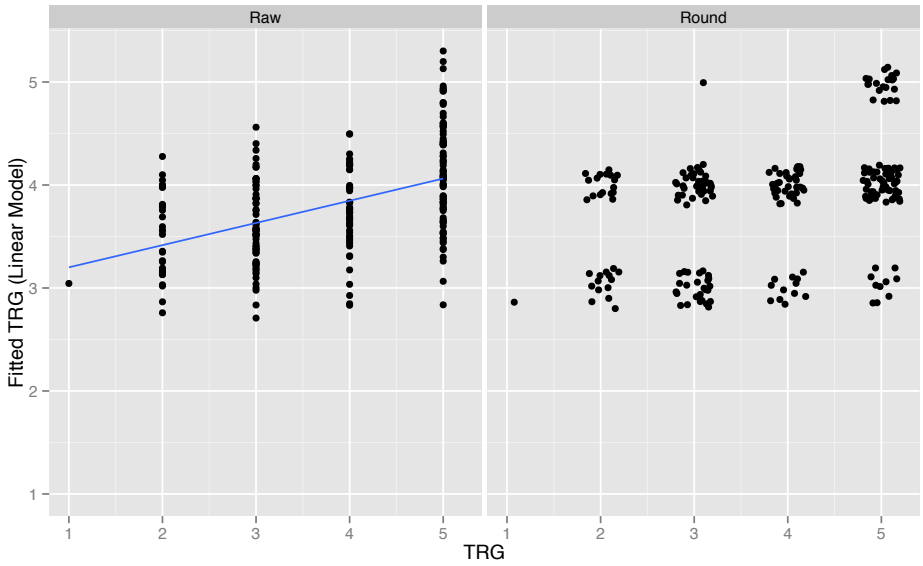



**FIGURE 2.4** *This plot shows the fitted values from a linear model (y axis) plotted against the original responses (x axis). The response is the TRG of the rectum cancer dataset – mapped to the numerical values 1,2,3,4,5. The terms in the model are the 6 genes, of which the expression values showed q values $< 0.1$ when used as the single predictor in a univariate linear model for TRG. The left panel shows the raw fitted values, which will generally result in real numbers and do not fall onto integers.*
*If linear regression is to be used to predict ordinal responses, these raw fitted values are rounded to the next number used in the representation of the original response (in this case 1,2,3,4,5). These rounded values are shown in the right panel.*

In high-dimensional problems with $p \gg n$ the features are necessarily highly correlated and some regularization is needed. In this work we choose

component-wise gradient boosting as our regularized version of linear models. For a detailed introduction, see Bühlmann and Hothorn, 2007b and Hofner et al., 2014, both of which also introduce the software package `mboost` that we used in our experiments.

In short, component-wise gradient boosting is a very flexible framework that can accommodate many modeling situations, among them linear models, which we use here. Component-wise boosting only considers (small, maybe of size 1) subsets of features at a time.

Roughly, the algorithm proceeds stepwise and

1. calculates the gradient of a chosen loss function at the estimate of the previous step,
2. fits a base learner model in each component,
3. chooses the component that best fits the gradient vector, and
4. updates the current estimate in the direction of the fit by a chosen step size

until a given maximum number of steps is reached.

The implicit feature selection happens in step 3. As the final model only contains features that have been chosen in at least one step.

The main parameters here are the base learner and the loss function. In our experiments we fit a regularized linear model with the `glmboost()` method using a linear base learner and the default $L_2$-loss.

**Linear Discriminant Analysis** There are two independent ways to arrive at the Linear Discriminant Analysis. One way goes back to Fisher, 1936 and presents LDA as technique for dimension reduction.

We present another way (due to Welch, 1939) which focuses on classification instead of dimension reduction.

We can write a classification problem in terms of the conditional probability of a sample belonging to the $l$th of $L$ classes given that the feature vector $X$ takes some observed values $x$. Thus, we are interested in

$$\underset{l=1,\ldots,L}{\arg\max}\, Pr(G = l \mid X = x) \tag{2.13}$$

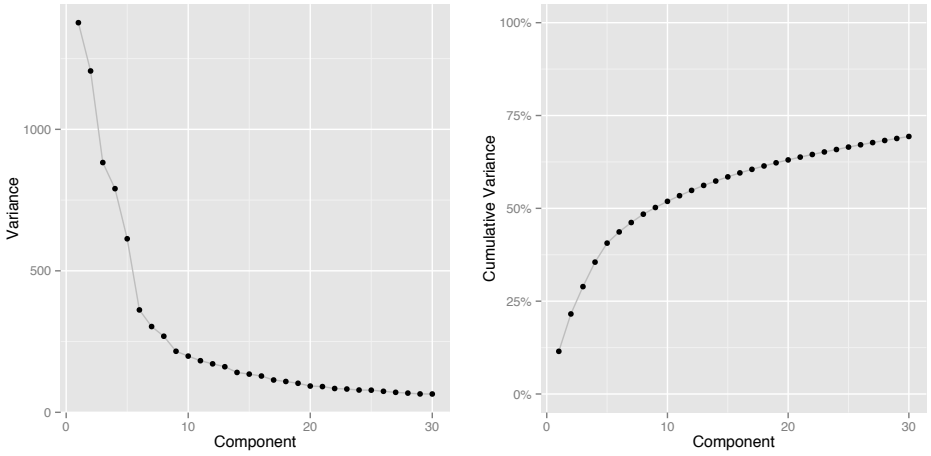



**FIGURE 2.5** *PCA component selection for the rectal cancer dataset. The left panel shows the* screeplot *where the variance in each principal component is plotted in descending order. The right panel shows a variation where the cumulative sum of variances are plotted – normalized by the total sum of the variances. Both plots are visual tools to help to decide on the number of principal components to retain.*

where $G = l$ is a shorthand for the response variable $G$ taking a value of the $l$th class.

Now, Bayes' theorem is applied to compute the individual class probabilities:

$$Pr(G = l|X = x) = \frac{\pi_l Pr(X = x \mid G = l)}{\sum_{m=1}^{L} \pi_m Pr(X = x \mid G = m)} \tag{2.14}$$

where $\pi_l$ denotes the prior probability of a sample belonging to the $l$th class.

In order to compute $Pr(G = l \mid X = x)$ via Equation 2.14 we need estimates for the prior class probabilities $\pi_l$ and for the class densities $Pr(X = x \mid G = l)$. The estimate for the prior class probabilities $\hat{\pi}_l$ is simply the prevalence of that class in the sample population.

For the estimation of the class densities $Pr(X = x \mid G = l)$ LDA assumes the observations $X$ to be multivariate normal with group specific mean $\mu_l$ but the same covariance matrix $\mathbf{\Sigma}$, i.e. $X^{(l)} \sim \mathcal{N}(\mu_l, \mathbf{\Sigma})$, where $X^{(l)}$ is the subset of samples that belong to the $l$th class. Using this assumption of normally distributed data, Equation 2.14 can be transformed to the *linear discriminant functions*:

$$\delta_l(x) = x^T \mathbf{\Sigma}^{-1} \mu_l - \frac{1}{2} \mu_l^T \mathbf{\Sigma}^{-1} \mu_l + \log \pi_l \tag{2.15}$$

To estimate $\delta_l(x)$ we plug in the usual estimates for $\mu_l$ and $\mathbf{\Sigma}$:

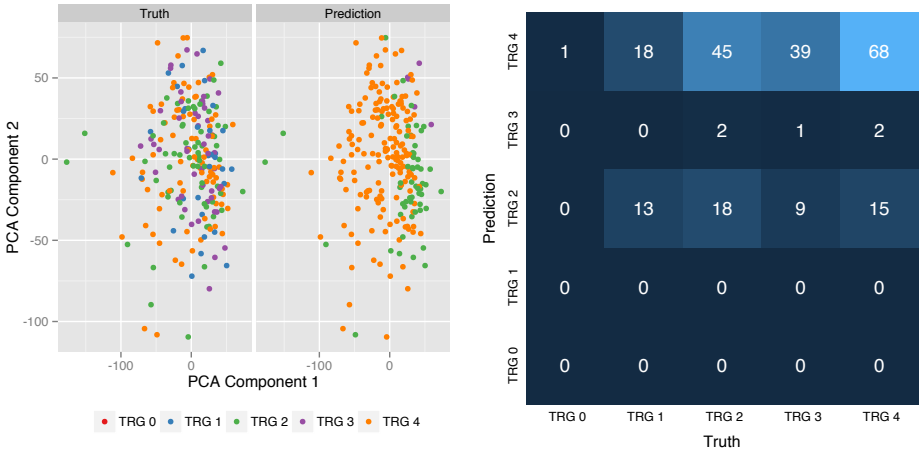



**FIGURE 2.6** *Results of an LDA on the first five principal components of the rectal cancer dataset. A PCA was performed on the rectal cancer dataset to reduce the number of dimensions. Based on visual inspection (see Figure 2.5) the first five components were retained.*
*The plots show the results of a re-classification using an LDA on these five components. The left panel shows both the true TRG as well as the predicted (re-classified) TRG for all patients in the data projected to the first two principal components.*
*It is clearly visible that the LDA is guided by the principal components as there is a separation of colours in the right part. The colours in the left part are well mixed, which visualizes the fact, that the true values are not separated by either of the first two components.*
*The right panel shows a heatmap of the contingency table between the true and the predicted TRG values (accuracy 38 %, kendalls $\tau$ 0.15).*

- $\hat{\mu}_l = \frac{1}{N_l} \sum_{i,\, g_i=l} x_i$, where $N_l$ is the number of class-$l$ observations
- $\hat{\mathbf{\Sigma}} = \frac{1}{N-L} \sum_{l=1}^{L} \sum_{i,\, g_i=l} (x_i - \hat{\mu}_l)(x_i - \hat{\mu}_l)^T$

As we see from 2.15, the LDA relies on $\mathbf{\Sigma}^{-1}$, the inversion of the covariance matrix. As a consequence, LDA can be applied only when there are more samples than features, i.e. when the problem is not high-dimensional. In order to use LDA on high-dimensional data, there is need for dimension reduction. There are two commonly used combinations:

- *Principle Component Analysis (PCA)*
  Here, the feature space is first transformed so that the axes align with the directions of highest variation in the data. To reduce the dimensionality, the last dimensions are omitted. If there is not much variation in one dimension, this dimension can hardly contribute to distinguishing the classes. On such way reduced data LDA is performed afterwards. Figure 2.6 shows some results of an LDA performed on the first five components

of a PCA of the rectal cancer dataset. Here the number of the components to retain was determined visually (see Figure 2.5). In a classification setting as this one, a more appropriate way to choose this number would be through cross validation, though. But the main concern of combining PCA and LDA is, that PCA does not take the response into account and can, thus, be misled by high but non-systematic variance. In essence, if the variability in the feature space is related to variability in the response, then PCA is a good choice. On the other hand, if variability in the feature space is not related to variability in the response, the first PCA components will be non-informative with respect to the response (Kuhn and Johnson, 2013).

- *Partial least squares discriminant analysis (PLS-DA)*
  The partial least squares have been developed in the context of regression (Wold, 1966; Wold, Martens, and Wold, 1983). Due to its superior performance compared to PCA in discrimination settings, it becomes increasingly popular in molecular biology as well as many other fields (Pérez-Enciso and Tenenhaus, 2003; Andersen et al., 2012; Serrano-Cinca and Gutiérrez-Nieto, 2013, , e.g.). For a good overview over the theoretical background and examples of application see Boulesteix and Strimmer, 2007. Similar to PCA the components returned by PLS are uncorrelated. But in contrast to PCA, where each component is in direction of the highest variance, the components of PLS are in the direction of highest co-variance with the outcome. PLS has also been called *supervised dimension reduction* as the dimension reduction is guided by the response to ensure maximal co-variance with the response. PCA is *unsupervised* in that sense. Several algorithms have been proposed to derive the PLS components, see Alin, 2009, for a comparison of algorithms for the case $N > p$. More interesting in this context are algorithms for $p > N$; see Rännar et al., 1994 for an example. In Figure 2.7 an LDA was conducted on the first five PLS components of the rectal cancer data. The results can be directly compared to the setting with PCA (Figure 2.6) and demonstrate the superiority of PLS in classification settings.

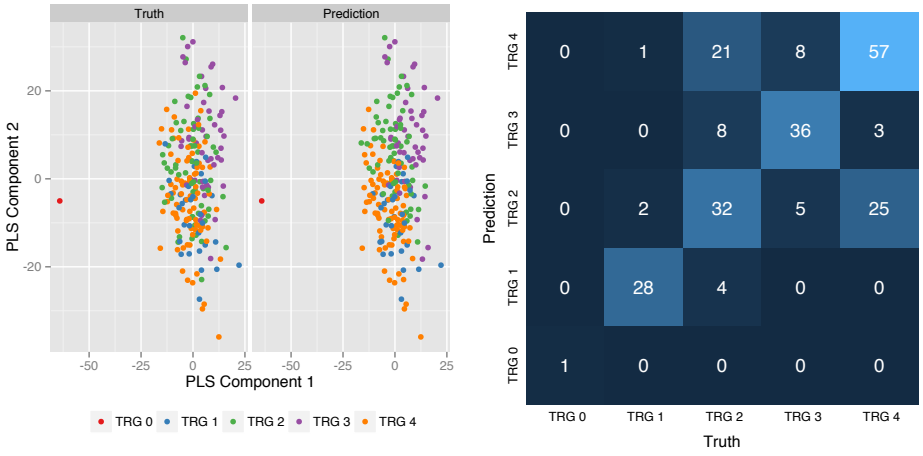



**FIGURE 2.7** *Results of an LDA on the first five components of a partial least squares fit on the rectal cancer dataset.*
*A PLS was performed on the rectal cancer dataset to reduce the number of dimensions. To match the settings of the PCA (Figure 2.5), the first five components were retained. The plots show the results of a re-classification using an LDA on these five components.*
*The left panel shows both the true TRG as well as the predicted (re-classified) TRG for all patients in the data projected to the first two components. The first component separates the first level (only one sample) from the others. The second component tries to separate the other levels but has problems to put TRG2 and TRG4 (the green and orange dots) in separation.*
*Compared to the PCA-LDA (Figure 2.6) the colours in the truth and prediction panels do not differ as much. This is also reflected in the right panel, which shows a heatmap of the contingency table between the true and the predicted TRG values (accuracy 66 %, kendalls $\tau$ 0.5).*

**Multinomial Logistic Regression** The logistic regression (also known as *logit regression*) as a model for 2-class problems is mostly credited to Berkson, 1944. It had been developed as an alternative to the *probit regression* and has – after a long period of hard discussion – by now become the more popular model (Cramer, 2003). The logistic regression is a method to facilitate linear regression to model a binary (i.e. 2 class) response. In order to apply linear regression, the binary response has to be transformed to a continuous value via a *link function*. In the logistic regression the logarithm of the odds ratio between the *success* class and the *failure* class, the *log-odds* also known as the *logit*, provides this link.

Let $\pi_{i1}$ denote the probability of sample $i$ to belong to the *success* class and $\pi_{i0}$ the probability of sample $i$ to belong to the *failure* class, so that $\pi_{ik} = Pr(G_i = k \mid \mathbf{x}_i = x)$, $k = 1, 2$. In a classification problem we assume that each sample $i$ must take exactly one of the 2 classes, i.e. $\pi_{i0} = 1 - \pi_{i1}$.

Using this notation, we can plug the log-odds into the left hand side of Equation 2.10 leading to the model equation

$$\ln \frac{\pi_{i1}}{1-\pi_{i1}} = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon,\ i \in \mathcal{T} \tag{2.16}$$

where
$\mathbf{X} = (x_{ij})_{\substack{i=1,\ldots,N \\ j=1,\ldots,p}}$ : the $N \times p$ matrix of feature vectors,
$p$ : number of features,
$\mathcal{T}$ : training set,
$\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ : the unknown model parameters, and
$\epsilon$ : random error.

Equation 2.16 can be solved for $\pi_{i1}$ giving the logistic function

$$\pi_{i1} = \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon)}},\ i \in \mathcal{T} \tag{2.17}$$

which takes any real value as input and yields a value in $]0;1[$. This is the motivation for modeling the log-odds because the values of the logistic function – being between 0 and 1 – can readily be interpreted as probabilities.

The model parameters $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ can be estimated by maximizing the likelihood. As the solution can not be given in closed form, usually an iterative procedure (mostly Newton-Raphson Ortega and Rheinboldt, 2000) is applied.

To arrive at class predictions a threshold can be used to discretize the modeled log-odds.

The logistic regression has been generalized in several ways to be applicable also to the case of $L > 2$ response classes. All different methods are based on the idea to replace the direct comparison of all classes with a set of binary comparisons. These binary comparisons are modeled as with logistic regression.

The multinomial logistic regression (McFadden, 1973; Agresti, 2002) is a generalization approach, that does not assume the $L$ response classes to be ordered.

Here, the idea is to take one of the $L$ classes as the reference class. Then, the log-odds of all other classes to that reference class are calculated. These log-odds are then modelled using logistic regression.

Let again $\pi_{il}$ denote the probability of sample $i$ to fall into class $l$ for each class $l = 1, \ldots, L$, so that

$$\pi_{il} = Pr(G_i = l) \tag{2.18}$$

where we assume again that each sample $i$ must take exactly one of the $L$ classes, i.e. $\sum_{l=1}^{L} \pi_{il} = 1$.

By convention, class $L$ is used as reference class. As there is no order among the classes, this choice is random and, indeed, the choice is irrelevant for the model result.

For each non-reference class $l = 1, \ldots, L-1$ the log-odds $\eta_{il}$ to the reference class are modeled via logistic regression as in:

$$\eta_{il} = \ln \frac{\pi_{il}}{\pi_{iL}} = \beta_0^{(l)} + \sum_{j=1}^{p} x_{ij} \beta_j^{(l)}. \tag{2.19}$$

These modeled log-odds can again be used to describe the original probabilities $\pi_{il}$:

$$\pi_{il} = \pi_{iL} e^{\eta_{il}},\ l = 1, \ldots, L-1 \tag{2.20}$$

With $\sum_{l=1}^{L} \pi_{il} = 1,\ i = 1, \ldots, N$ we can derive from Equations 2.20 that

$$\pi_{iL} = \frac{1}{1 + \sum_{l=1}^{L-1} e^{\eta_{il}}}, \tag{2.21}$$

which we can plug back into Equations 2.20 arriving at

$$\pi_{il} = \frac{e^{\eta_{il}}}{1 + \sum_{l=1}^{L-1} e^{\eta_{il}}},\ l = 1, \ldots, L-1. \tag{2.22}$$

Equation 2.21 together with Equations 2.22 form the multinomial regression model.

The regression coefficients $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ are numerically found by maximizing the likelihood using a numerical procedure. Typically Newton-Raphson (Ortega and Rheinboldt, 2000) or a quasi-Newton variant (Section 10.9 Press et al., 2007) is applied.

#### 2.3.1.2 Low Dimensional Ordinal Methods

Other possibilities to generalize the logistic regression to settings with more than two groups are ordinal methods that rely on a given order among the classes. Liu and Agresti, 2005 as well as Ananth and Kleinbaum, 1997 have an overview over these methods – please refer there for more details.

There exist other, less popular methods, such as an ordinal variant of LDA (Sun et al., 2010), which we will omit here.

For reference, we will introduce here shortly the two most commonly applied ordinal extensions to the logistic regression.

**The *Cumulative Link Model (CLM)*** As in the multinomial logistic regression the CLM (Walker and Duncan, 1967; McCullagh, 1980) dissects the problem into a set of binary problems which are subjected to logistic regression. There are two major differences to the multinomial logistic regression, though: (1) Instead of performing all pairwise comparisons to a chosen reference class, the CLM pools the classes on both sides of the decision boundary (see below) and (2) the regression coefficients (except for the intercept) are assumed to be the same for each of these binary comparisons.

The pooling of classes means that for each class $l \in 1, \ldots, L-1$ the CLM uses the logistic regression equation to model $\gamma_{il} = Pr(G_i \leq l \mid x_i = x)$, which is the probability that the class of sample $i$ is at most $l$ given the observation $x$.

Logistic regression is used to model these probabilities, which means that the logit of these $\gamma_{il}$ is modeled with a linear expression for the explanatory variables $x$:

$$\ln \frac{\gamma_{il}}{1-\gamma_{il}} = \beta_0^{(l)} + \sum_{j=1}^{p} \beta_j x_{ij},\ l = 1, \ldots, L-1 \tag{2.23}$$

where $\beta_0^{(l)}$ is the class-specific intercept and $\beta_j,\ j = 1, \ldots, p$ are the regression coefficients for the feature variables. Since the values of $\beta_j,\ j = 1, \ldots, p$ are independent from the class, the CLM is also called *Proportional Odds Model*.

The reformulation and the parameter estimation (typically Newton-Raphson again) proceed very similar to ordinary logistic regression or multinomial regression.

**The Continuation Ratio Model** The continuation ratio model (Fienberg, 1980; McCullagh and Nelder, 1989) is a variation of the CLM which models the ratio of the probability of being in class $l$ to the probability of being in a class greater than $l$:

$$\ln \frac{Pr(G_i = l \mid x_i = x)}{Pr(G_i > l \mid x_i = x)} = \beta_0^{(l)} + \sum_{j=1}^{p} \beta_j x_{ij},\ l = 1, \ldots, L-1 \tag{2.24}$$

As with the CLM, the regression coefficients $\beta_1, \ldots, \beta_p$ are usually modeled to be independent of the class $l$ and only the intercept $\beta_0^{(l)}$ differs between the classes.

The continuation ratio model is also called *stopping ratio model* (Yee, 2010) or *sequential model* (Tutz, 1991).

### *2.3.2 High-Dimensional Methods*

In this section we briefly introduce some classification methods which are applicable to high-dimensional data. Since most methods are extensions of existing multi-class methods, we do not partition the section as a whole into an ordinal and a non-ordinal subsection, but – where applicable – present the methods in two parts: a general part and a description of the ordinal variant.

We will start with the widely used k-nearest-neighbours, support vector machines and classification trees. The subsection following that briefly describes other and less popular methods. We end this section with Frank and Hall's method *simple twoing* which is the basis for our method *hierarchichal twoing* which will be presented in chapter 4.

#### 2.3.2.1 Nearest Neighbour Classification: kNN and kkNN

We have introduced kNN (kNN, Fix and Hodges, 1951; Cover and Hart, 1967) already shortly in Section 2.1. For each point in the feature space kNN retrieves the class label of the $k$ nearest neighbours and assigns the majority vote as the class label for that point. As we have seen, the parameter $k$ is crucial and balances adaptivity to the local structure with generalization: the smaller $k$ is chosen the better will the classifier perform on the training data. The larger

$k$ is chosen the less partitions are generated. In the extreme case of $k$ equal to the number of samples, the feature space is not partitioned at all and kNN simply gives the class with the highest prevalence.

The idea of Hechenbichler and Schliep, 2006, is to also take into account the distance of the point to its $k$ nearest neighbours. It is a natural step to assign a higher weight to samples close to the point in question and less weight to points that are still in the neighbourhood but further away. This technique requires some work to standardize the dimensions in the input space in order to give similar weight to each of them, which is especially necessary for non continuous variables. See Hechenbichler and Schliep, 2006, for the details.

The distances are transformed into weights via a kernel (hence $k$kNN) and the majority vote is then replaced by the weighted majority vote.

For ordinal classifications Hechenbichler and Schliep, 2006 propose to use the weighted median of the neighbouring samples instead of the weighted majority vote.

Formally, the weighted majority vote of all samples in the neighbourhood $\mathcal{K}(x)$ of an unseen sample $x$ can be expressed as

$$\arg\max_{l \in L} \frac{\sum_{i \in \mathcal{K}(x)} w_i^{(x)} I(g_i = l)}{\sum_{i \in \mathcal{K}(x)} w_i^{(x)}}, \tag{2.25}$$

where $w_i^{(x)}, i \in \mathcal{K}(x)$ is the weight of training sample $i$ with respect to the unseen sample $x$. In the proposed ordinal version arg max is replaced by median.

We ran kkNN with a triangular kernel and $k = 7$ on the rectal cancer data and compared its performance to kNN. As the training error of kkNN with theses settings is 0 – this is expected, as the closest observation to each data point is the data point itself and this is given the highest weight in the triangular kernel – we performed a 10-fold cross validation here. Averaged over all 10 folds, kNN achieves an accuracy of 21% and a Kendall's $\tau$ of 0.17. Using kkNN the accuracy increases to 29% while Kendall's $\tau$ drops to 0.097.

Noteworthy are also the system requirements of kkNN. While the cross validation using kNN could be performed on a personal laptop in 40 seconds using less than 700MB of memory, kkNN required 21GB of memory on a large server and completed after 1 hour and 40 minutes.

#### 2.3.2.2 Support Vector Machines

The support vector machine is a comparatively young classification procedure which – although having roots further back in the past – go back to Boser, Guyon, and Vapnik, 1992 and Vapnik, 1995.

For some time there was a competition between SVMs and artificial neural networks (ANN, e.g. Dal Moro et al., 2006; Tonello, Vescini, and Caudarella, 2007). While the SVMs were favored for several years (Bennett and Campbell, 2000), ANNs gained popularity again more recently owing to the *deep learning* trend (see for example Schmidhuber, 2014 for an overview and many references or Deng, 2014 for a more thorough coverage of deep learning).

In its initial form the SVM is a binary classifier that finds a hyperplane in the feature space separating the samples from both classes. The hyperplane is constructed so that the distance to the samples closest to the hyperplane, the so called *margin*, is maximised. The closest samples themselves are called the *support vectors* as they determine the margin and, thus, the hyperplane.

The idea to find this separating hyperplane starts with two parallel separating hyperplanes the distance of which is maximised. The resulting hyperplane lies then half way between these two marginal hyperplanes. The procedure is a maximisation problem with constraints. If we denote the two initial parallel separating hyperplanes with

$$\begin{aligned} w \cdot x + b &= 1, \text{ and} \\ w \cdot x + b &= -1 \end{aligned} \tag{2.26}$$

than the distance is $\frac{2}{\|w\|}$, which is maximised when $\|w\|$ is minimised. To make sure that during the optimisation the plane remains separating between the two classes, we introduce constraints

$$g_i(w \cdot x_i - b) \geq 1, i = 1, \ldots, N \tag{2.27}$$

where we assume $g_i = 1$ if sample $i$ is in one class and $g_i = -1$ if sample $i$ is in the second class.

This is a convex quadratic programming problem which is robustly solvable by standard techniques.

Many extensions have been developed. One of the first extensions introduced *soft margins*. If the two classes are not linearly separable in the feature space, the initial definition of the SVM using the separating hyperplane is void. To deal with such situations soft margins allow some mis-labelling of samples. This is achieved by putting a penalty on mis-labeled samples. The mis-labelling is captured in so called *slack variables* $\xi_i$ for each data point $x_i$, $i = 1, \ldots, N$ which are included in the constraints

$$g_i(w \cdot x_i - b) \geq 1 - \xi_i, i = 1, \ldots, N \tag{2.28}$$

and in the minimisation problem. If we choose a linear penalty function that results in the minimisation

$$\arg\min_{w,b,\xi} \{\|w\| + C \sum_{i=1}^{N} \xi_i\} \tag{2.29}$$

where we introduced $C$ as a hyperparameter of the SVM.

We can further extend the SVM to be better suited for problems where the separation is highly non-linear. For such problems the 'kernel trick' is used to project the data into a feature space of (even) higher dimensions where the separation between the two classes is again linear in shape. In that space the SVM is trained and the resulting hyperplane is projected back into the lower dimensional original feature space which generally produces non-linear decision boundaries.

The kernel trick can best be shown on the dual minimisation problem to (2.29) which is given by (see for instance Cortes and Vapnik, 1995, for the derivation):

$$\begin{aligned} &\arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} g_i g_j \alpha_i \alpha_j x_i \cdot x_j - \sum_{i=1}^{N} \alpha_i \\ &\text{s.t.} \quad \sum_{i=1}^{N} g_i \alpha_i = 0 \text{ and} \\ &\qquad C \geq \alpha_i \geq 0, \\ &i = 1, \ldots, N. \end{aligned} \tag{2.30}$$

Now we want to project the samples $x_i$ into a higher-dimensional space using a projection $\theta$. In our optimisation problem 2.30 we will only need to replace $x_i \cdot x_j$ with $\theta(x_i) \cdot \theta(x_j)$. Now we apply Mercer's Theorem (Mercer, 1909) and are for some mappings $\theta$ allowed to replace the inner product of the mapped values with the application of a kernel function $K(x_i, x_j)$.

Widely used kernel functions include the polynomial kernel of degree $d$

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{2.31}$$

or the radial kernel with parameter $\gamma > 0$

$$K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2). \tag{2.32}$$

Besides swapping the inner product in 2.30 with the kernel function, the SVM proceeds exactly as before, but will now work in a highly non-linear fashion.

There are more extensions of the SVM, for instance the extension for regression problems. These are not interesting in this context and the reader is referred to Smola and Schölkopf, 2004, for details.

Highly relevant in the context of ordinal classification are the strategies to apply SVMs to problems with more than two classes.

Although some methods exist to apply SVMs to multi-class problems directly (e.g. Crammer and Singer, 2002; Lee, Lin, and Wahba, 2001), the most widely used strategy (Duan and Keerthi, 2005) is to perform a set of pairwise comparisons and combine the results. Prominent example is the *one-versus-one* approach where all pairwise comparisons are performed. For each sample the 'winner' is then chosen through majority voting. This is also the approach taken by the widely used implementation in `libsvm` (Chang and Lin, 2011).

An idea to generalize the SVM to ordinal problems assumes that the separating hyperplanes for all the decision boundaries should be parallel. In that way the ordinal SVM consists of $L - 1$ binary SVMs that are linked in that the hyperplanes share the direction $w$ and differ only in their *thresholds* $b_l$, $l = 1, \ldots, L - 1$.

This idea was proposed by Shashua and Levin, 2002. In order to retain the notion of the soft margin for each of the decision boundaries, Shashua and

Levin, 2002, proposed to restrict the slack variables for each boundary to the adjacent categories. That way, the error term in 2.29 becomes

$$C \sum_{l=1}^{L-1} \left( \sum_{i=1}^{N_l} \xi_i^l + \sum_{i=1}^{N_{l+1}} \xi_i^l \right) \tag{2.33}$$

where $N_l$ : number of samples in class $l$

$\xi_i^l$ : slack variable of sample $i$ in class $l$.

With accordingly adapted constraints 2.28 the ordinal SVM is complete.

Chu and Keerthi, 2007, realized that this formulation of an ordinal SVM might result in a solution where the thresholds $b_l$, $l = 1, \ldots, L-1$ are disordered with respect to the ordering of the classes $G_l$, $l = 1, \ldots, L$. They propose two ways to amend that. First, by adding an explicit constraint enforcing the desired order $b_1 \leq b_2 \leq \cdots \leq b_{L-1}$. And second, they propose to consider all samples in the error term – as opposed to considering only the samples from the adjacent classes. That way, the error term in 2.29 becomes

$$C \sum_{l=1}^{L-1} \left( \sum_{j=1}^{l} \sum_{i=1}^{N_j} \xi_i^j + \sum_{j=l+1}^{L} \sum_{i=1}^{N_j} \xi_i^j \right) \tag{2.34}$$

where $N_l$ : number of samples in class $l$

$\xi_i^l$ : slack variable of sample $i$ in class $l$.

Chu and Keerthi, 2007, show that this formulation automatically results in the correct order of the thresholds.

#### 2.3.2.3 Classification Trees

Tree based methods constitute another highly popular class of machine learning methods (Rokach and Maimon, 2008) that are suited well for high-dimensional problems (Breiman et al., 1984).

Tree based methods recursively partition the feature space into sub-spaces that are – according to a suitably chosen criterion (see below) – more homogeneous with respect to the class of the contained samples than the bigger space. Such recursive partitioning can be represented in a tree where the root

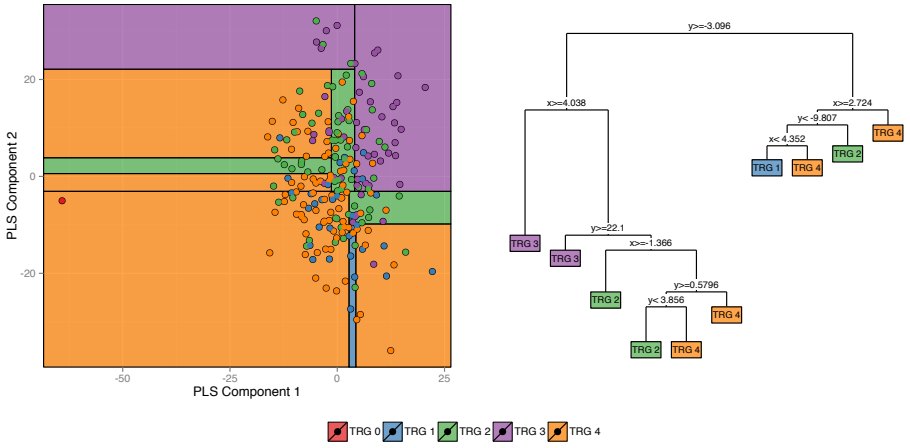



**FIGURE 2.8** *This is an illustration of the partitioning that is the outcome of classification trees. In order to be able to visualise the feature space we reduced the number of dimensions in the rectal cancer data to the first two components of a PLS regression (see Figure 2.7). The classification tree as implemented in the R package* `rpart` *was trained with the default parameters using the first two PLS components as features and the tumor regression grade as response.*
*The left panel shows the partitioning of the feature space as suggested by the classification tree. The right panel shows the tree representation. Each partition of the feature space and, thus, each leaf of the tree is coloured according to the TRG of the majority of data points falling into that partition. Note that the pruning removes many splits of the full and taller tree – especially the split that separates the only sample with TRG 0 from the rest.*

node corresponds to the full feature space which contains all samples from the dataset and where each inner node corresponds to some partition of the feature space which contains only the samples the features of which fall into that partition.

Trees that allow *multivariate* splits, i.e. splits involving more than one variable, typically choose linear combinations of the variables (leading to so called *oblique trees*) and have been studied for example by Loh and Vanichsetakul, 1988. An optimal linear split partitioning the feature space has been shown to be NP-hard to find (Heath, 1992), where a split is called *optimal* if it minimizes the number of misclassified samples. This holds even for the case of only two classes. Most common trees, however, use *univariate* splits, where each split is only in a single dimension of the feature space so that the splitting hyperplane is parallel to the remaining dimensions.

But even for trees with univariate splits the search for an optimal tree is NP-hard for various concepts of optimality (Naumov, 1991), e.g. the minimum

expected number of tests necessary for an unseen sample (Tu and Chung, 1992). Comparing trees giving the same classification result, the tree with the least nodes is considered to be preferable. This tree is of course less costly to evaluate and it will generalize better from the dataset (Murphy and Pazzani, 1994).

A widely applied heuristic to avoid the NP-hardness is the *greedy* approach, which at each split selects the variable and the split point in that variable leading to the highest ad-hoc improvement. There are some attempts to involve more splits at once which lead to non-binary trees (see Keprta, 1996, e.g.). These are exceptions, though, and most trees are built using one split in each node.

As the sample in Figure 2.8 shows, the partitioning of the feature space is adaptive to the training data and will be finer in regions with data points from diverse classes and more coarse in regions with data points all of the same class. This adaptivity can lead to highly overfitted trees that follow the training data too closely.

Two main approaches have been proposed to lessen the risk of overfitting: pruning and ensemble methods.

Figure 2.8 shows a pruned tree. The procedure here is to first grow the full tree until each leave (i.e. each subspace of the partitioning) contains only data points that belong to the same class. In a second *pruning* step, neighbouring leaves of the tree are merged into a bigger node. Depending on how often such a merge is done the resulting tree will have more or less nodes. One possible way to determine when to stop the pruning is to add a penalty term proportional to the number of nodes in that tree to the minimization criterion when choosing the tree (see the longintro of Therneau, Atkinson, and Ripley, 2014, for a deeper introduction to pruning).

Ensemble methods commonly used with classification trees as base learners are *bootstrap aggregation (bagging)* and *random forests*.

Briefly, the idea of bagging (Breiman, 1996) is to draw many samples from the training data with replacement and then to train the same classifier on each of these bootstrap samples. The overall classification is done via voting of all trained classifiers.

Random forests (Breiman, 2001) combine bagging with random decision trees (Amit and Geman, 1997; Ho, 1995). When the locally optimal split is

determined in each node during the growth of each of the trees in the bagging set, a tree in a random forest considers only a random subset of the available features. In other words the locally optimal split is found in a randomly chosen subspace of the full feature space.

Random forests remain highly popular as they are considered to be among the best available classification algorithms (Fernández-Delgado et al., 2014).

Two prominent procedures of fitting trees are *CART (classification and regression trees)* and the *C4.5; C5.0* procedure (Quinlan, 1993; Kuhn and Johnson, 2013), where C5.0 is the improved version of C4.5 which in turn is the improved version of the earlier ID3 algorithm. A comparison is beyond the scope of this work (see Tsoi and Pearson, 1991; Cerňak, 2012, e.g.).

Note also that – as indicated by the name *CART* – trees are not restricted to classification but can similarly be applied to regression problems.

The last missing component making up classification trees is the split criterion. The general strategy is to define a measure of *impurity* $I(A)$ on each node $A$ and to choose a split that maximizes

$$\Delta I = p(A)I(A) - (p(A_l)I(A_l) + p(A_r)I(A_r)) \tag{2.35}$$

where

$$p(A) = \sum_{l=1}^{L} \pi_l P(x \in A \mid \text{class}(x) = l), \tag{2.36}$$

$$\pi_l = \text{prior probability of class } l, and \tag{2.37}$$

$$A_l, A_r = \text{the child nodes of node } A. \tag{2.38}$$

The impurity $I(A)$ of a node is defined in terms of an impurity function $f$ either as a sum

$$I(A) = \sum_{l=1}^{L} f(p_{lA}) \tag{2.39}$$

or as the minimum of all pair-wise comparisons between the two classes of (disjoint) partitions of the available classes $\{1, \ldots, C\} = C_1 \uplus C_2$

$$I(A) = \min_{C_1, C_2} \left[ f(p_{C_1 A}) + f(p_{C_2 A}) \right] \tag{2.40}$$

where $p_{lA}$ is the proportion of members of class $l$ in the members of node $A$.

Two common choices for the impurity function $f$ are the *information index* (used in C4.5 / C5.0)

$$f(p_{lA}) = -p_{lA} \log(p_{lA}) \tag{2.41}$$

and the *Gini index* (typically default in CART implementations)

$$f(p_{lA}) = p_{lA}(1 - p_{lA}) = \sum_{m \neq l} p_{lA} p_{mA}. \tag{2.42}$$

**Ordinal Classification Trees** Classification trees as described above can quite easily be adapted to the ordinal setting.

The first possibility to do this is to restrict the *twoing* method to consider only partitions $C_1, C_2$ of $\{1, \ldots, C\}$ that preserve the ordering of the classes. In that case a partition $C_1 = \{1, 3\}, C_2 = \{2\}$ would not be allowed, e.g. The resulting method is referred to as *ordered twoing* (Breiman et al., 1984) and an implementation is given in `rpartOrdinal` (Archer and Mas, 2009; Archer, 2010) or `rpartScore` (Galimberti, Soffritti, and Maso, 2012).

Also the impurity function can be extended to incorporate information from the ordering of the classes. The *generalized Gini index* has been extended by Breiman et al., 1984, to include a loss function $\mathcal{L}(l, m)$:

$$f(p_{lA}) = \sum_{m \neq l} \mathcal{L}(l, m) p_{lA} p_{mA}. \tag{2.43}$$

where an ordinal loss function increases with the distance between the classes $l$ and $m$, so that

$$\forall l, m, n \text{ with } l < m < n : \mathcal{L}(l, m) \leq \mathcal{L}(l, n). \tag{2.44}$$

Typical loss functions are the *absolute loss* $\mathcal{L}(l, m) = |l - m|$ and the *quadratic loss* $\mathcal{L}(l, m) = (l - m)^2$. Regardless of the choice of the loss function, the need to assign a loss to each pair $(l, m)$ is similar to assigning distances between the classes and, thus, makes this approach somewhat similar to regression based classifiers.

Other ordinal splitting criteria were introduced in Piccarreta, 2004, and Piccarreta, 2008, including the following

$$f(p_{lA}) = \sum_{l=1}^{L} F_A(l)\,(1 - F_A(l)) \tag{2.45}$$

where $F_A(l)$ denotes the proportion of samples in node $A$ with class $\leq l$.

Our comparisons in chapter 4 include results from the ordered twoing method, which at the cost of being the computationally most expensive method provided the best results.

#### 2.3.2.4 Other Methods

Other methods which have been published but have not (yet) reached similar levels in popularity as the ones described above will be briefly described in this section.

We start with Archer and Williams, 2012, who present a LASSO penalized version of the continuation ratio models (see section 2.3.1.2). Using the $L_1$-penalization which shrinks most coefficients to 0 and, thus, performs a feature selection is the key idea to make CRMs applicable to high-dimensional problems. The authors provide two software packages `glmnetcr` and `glmpathcr` that use the LASSO implementation of `glmnet` and `glmpath` respectively.

A slightly different approach is presented in Archer, Hou, et al., 2014, which introduces an extension of the generalized monotone incremental forward stagewise method (GMIFS) – in contrast to the LASSO penalization used in Archer and Williams, 2012 – to also accomodate ordinal response settings. Using GMIFS the authors succeed in integrating not only CRMs but also CLMs using different link functions in unpenalized and penalized versions.

Another very different approach is presented in Sánchez-Monedero et al., 2013. While most specifically ordinal methods treat ordinal classification as a classification problem, their approach treats it as a regression problem. The interesting question then is, how to map the ordinal classes to numerical values. The proposed *Pairwise Class Distances (PCD)* are based on a latent variable model, which assumes that the ordinal measurements are generated from a latent continuous variable. The PCD tries to reverse the generative process

and derives an explicit projection of the features into $\mathbf{R}$. After that projection a simple regression can be used to train the classifier.

Ordinal logic regression as described in Wolf, Slate, and Hill, 2015, is a novel method that extends logic regression (Ruczinski, Kooperberg, and LeBlanc, 2003) to ordinal responses. The basis on logic regression, however, limits its applicability to situations with binary features only.

#### 2.3.2.5 Simple Twoing (si2)

This last section of the related work introduces the idea of Frank and Hall, 2001. We will call this idea *simple twoing (si2)* adapting terminology from Breiman et al., 1984. This section will lead into the presentation of hierarchical twoing which is an extension of this idea.

Their key idea was to translate an ordinal $L$-level classification problem into $L-1$ binary classification problems. The binarization is achieved by partitioning the $L$ levels into two groups preserving the ordering. As an example the 5 groups from the rectal cancer example *TRG0*, *TRG1*, *TRG2*, *TRG3*, and *TRG4* would be partitioned into the four partitions

- $\{TRG0\}$, $\{TRG1, TRG2, TRG3, TRG4\}$,
- $\{TRG0, TRG1\}$, $\{TRG2, TRG3, TRG4\}$,
- $\{TRG0, TRG1, TRG2\}$, $\{TRG3, TRG4\}$, and
- $\{TRG0, TRG1, TRG2, TRG3\}$, $\{TRG4\}$.

To predict the class of an unseen sample $x$, these $L-1$ binary classifiers are used to assign class probabilities to each of the $L$ classes. The class with the maximum class probability is then chosen as the (predicted) class of the unseen sample $x$. The assignment of class probabilities proceeds via the following scheme:

- $P(l = 1) = 1 - P(\text{class}(x) > 1)$
- $P(l = i) = P(\text{class}(x) > i - 1) - P(\text{class}(x) > i), \quad 1 < i < L$
- $P(l = L) = P(\text{class}(x) > L - 1)$

That means that the class probability of the classes 1 and $L$ each depend on one binary classifier and the class probabilities of the levels in between depend on the two 'neighbouring' classifiers.

The presented idea does not depend on a specific binary base learner, but any binary classifier can be trained and used as base learner as long as it is able to assign class probabilities. In their paper Frank and Hall use classification trees (the C4.5 implementation) as their base learner.

Depending on the chosen base learner simple twoing can scale well to high-dimensional problems.

| Method | Reference |
|---|---|
| *ordinal* | |
| *wrapper (high-dimensional or non high-dimensional)* | |
| si2 | Frank and Hall, 2001 |
| hi2 | Leha, Jung, and Beißbarth, 2013 |
| *high-dimensional* | |
| rpartOrdinal | Archer and Mas, 2009; Archer, 2010 |
| rpartScore | Galimberti, Soffritti, and Maso, 2012 |
| glmnetcr | Archer and Williams, 2012 |
| SVOREX | Chu and Keerthi, 2007 |
| ordinal KDA | Sun et al., 2010 |
| kknn | Hechenbichler and Schliep, 2006 |
| PCDOC | Sánchez-Monedero et al., 2013 |
| *non high-dimensional* | |
| cumulative link model | Agresti, 2010 |
| continuation ratio model | Cox, 1988; Agresti, 2010 |
| *non ordinal* | |
| *high-dimensional* | |
| SVM classification | Bennett and Campbell, 2000 |
| SVM regression | Vapnik, 1995 |
| classification trees | Breiman et al., 1984; Quinlan, 1986 |
| penalization methods | |
| - regularization | Friedman, Hastie, and Tibshirani, 2010 |
| - boosting | Bühlmann and Hothorn, 2007a |
| k nearest neighbours (kNN) | Fix and Hodges, 1951; Cover and Hart, 1967 |
| *non high-dimensional* | |
| LDA, QDA | Fisher, 1936<br>Hastie, Tibshirani, and Friedman, 2009 |
| multinomial logistic regression | McFadden, 1973; Agresti, 2002 |
| (linear) regression | Hastie, Tibshirani, and Friedman, 2009 |

**TABLE 2.2** *Systematic view on classification algorithms and their relation to ordinal response and high dimensional feature space. In this table, we distinguish at the top level specifically ordinal methods from more general methods. In each group we further split the methods into methods that can handle data with $p \gg n$. The exception here are the wrapper methods si2 and hi2, as their ability to classify high-dimensional data depends on the chosen base learner.*
*This list only shows the most prominent methods, except for the ordinal and high-dimensional methods where the list is more detailed, as that class of methods is the main focus of this work.*

# 3 Material

In the previous chapter we discussed how to build and evaluate classifiers generally in the case of high-dimensional data when the response is on an ordinal scale. We also looked at existing methods to do such classification.

Before we introduce our own method hi2 and actually compare it with existing methods in chapter 4 we will introduce the data that we will use to do that comparison.

Both, simulated and real data, will be used and we will in this chapter first introduce the real data and present the simulation study in the second part.

## 3.1 Real Data

Data with ordinal response is surprisingly common. The reason for ordinal quantities usually lies in human scoring or grading (as the tumor regression grade TRG). But also composite measures often lead to ordinal quantities. An example is the AJCC stage which is essentially a composition of the TNM stage (Edge et al., 2011).

Four datasets are included in the classifier comparison:

- Treatment Response of Rectal Cancer Patients
  This dataset has been introduced already in the introduction as it serves as the example dataset throughout this thesis. This dataset stems from a research group that the author was part of.

- Gene Expression in Neuroblastoma
  This dataset was chosen as it is very similar in nature to the rectal cancer data but is quite well known in the scientific community.

- miRNA Expression in Breast Cancer
  From this study not only mRNA expression but also miRNA expression data is publicly available. Here, we focus on the miRNA expression as an example of other sources for high dimensional data.

- Gene Expression in B-cell Acute Lymphocytic Leukemia
  This data is available to `R` users (pre-packaged through bioconductor, Gentleman et al., 2005), which is why that dataset is extremely well studied. It is known that this data contain a lot of information and are, thus, 'easy' to classify.

The evaluation of all real datasets was done via a 10-fold cross validation, where the samples were randomly assigned to one of 10 partitions.

### *3.1.1 Gene Expression in Neuroblastoma*

These data include gene expression microarray data for 251 patients (Oberthuer et al., 2006). The full dataset is available at ArrayExpress (identifier: E-TABM-38).

As response we aim to predict the tumor stage, which was classified according to the International Neuroblastoma Staging System (INSS) in its revised version as described in Brodeur et al., 1993. Some pre-processing was performed to remove some samples: The data contain samples with stage 2 which is not part of the INSS. Also, some samples are recorded with stage 'IV-S' the place of which in the ordinal scale is unclear. So, we ignore these samples. Additionally, we ignore all samples without the staging annotation. That leaves 84 patients in this dataset.

These remaining 84 samples are annotated with one of the five tumor stages $\{1, 2a, 2b, 3, 4\}$ defined in the INSS. Figure 3.1 shows the distribution.

Expression values are measured on a custom oligonucleotide microarray and the processed data contains values for 10 155 probes.

### *3.1.2 miRNA Expression in Breast Cancer*

These data are published as part of a joint mRNA-miRNA study of patients suffering from breast cancer. The data are available through the gene expression

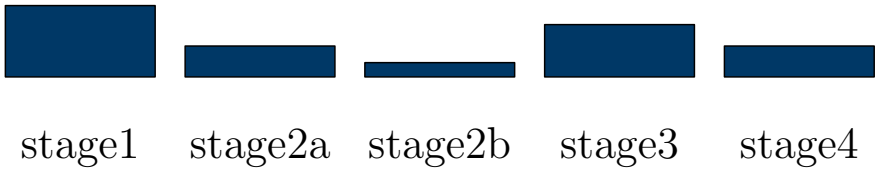



**FIGURE 3.1** *The neuroblastoma data contains information of the tumor stage according to the international neuroblastoma tumor staging system for 84 patients: 30 patients are assigned stage 1, 13 patients stage 2a, only 6 patients stage 2b, 22 patients stage 3 and 13 patients stage 4.*

omnibus data base (Edgar, Domrachev, and Lash, 2002, accession number GSE22216).

miRNA expression was measured using Illumina Human v1 MicroRNA expression beadchips which contain 735 miRNAs probes. Expression profiles are available for 207 patients.

Here, we want to use the miRNA expression to predict the tumor grade, which was assessed using the *modified method of Bloom and Richardson* (Elston and Ellis, 1991). That grading system has three levels *grade1*, *grade2*, and *grade3*.

Again we have to drop patients where the grading annotation is missing, leaving 194 patients in this dataset. Figure 3.2 shows the distribution of the tumor grades.


**FIGURE 3.2** *The available grading annotation for the breast cancer data follows the modified methods of Bloom and Richardson and knows three levels: 42 patients have been assigned tumor grade 1, 87 and 65 have been assigned tumor grade 2 and 3, respectively.*

### *3.1.3 Gene Expression in B-cell Acute Lymphocytic Leukemia*

These data include gene expression microarray data for 128 patients suffering from acute lymphoblastic leukemia (ALL), 95 patients with B-cell and 33

with T-cell leukemia (Chiaretti, Li, Gentleman, Vitale, Vignetti, et al., 2004; Chiaretti, Li, Gentleman, Vitale, Wang, et al., 2005).

We only look at the B-cell leukemia (95 patients). For these patients the tumor is staged into one of *B1*, *B2*, *B3*, *B4* depending on the expression of several antigens and immunoglobins on the leukemic cells. Again, we have to omit 5 patients for whom this staging information is missing leaving 90 patients. Figure 3.3 shows the distribution of these tumor stages.

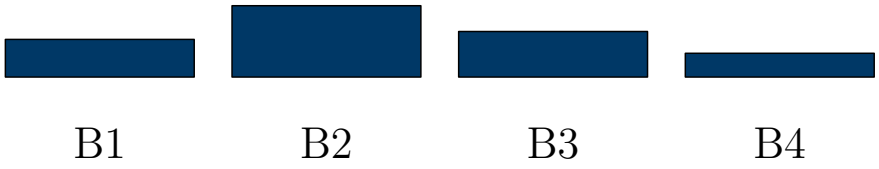



**FIGURE 3.3** *The available staging for the patients in the B-ALL data consists of four levels: the data contains 19 patients with level* B1*, 36 patients with level* B2*, 23 patients with level* B3*, and 12 patients with level* B4*.*

The expression was measured using the Affymetrix U95Av2 oligonucleotide microarrays which contain 12 626 mRNA probes.

### *3.1.4 Treatment Response of Rectal Cancer Patients*

These data have been introduced in detail already in chapter 1. To recapitulate shortly, it comprises gene expression profiles for 245 patients. Gene expression was measured using the Agilent-026652 Whole Human Genome Microarray 4x44K v2. In this analysis we focus on the 29 055 probes with unique probe ID.

As response the tumor regression grade (TRG) should be predicted using the gene expression profile. The TRG annotation is available for 231 patients. As a further preprocessing step, we merge the *TRG3* and *TRG3a* as well as *TRG3b* and *TRG4*. Figure 3.4 shows the distribution of the TRG levels.


**FIGURE 3.4** *The distribution of the response levels in the rectal cancer data is highly unbalanced. The response is the histopathological TRG and takes one of five levels. In the data there is only one patient assigned* TRG0*, while there are 31 patients with* TRG1*, 65 patients with* TRG2*, 49 patients with* TRG3 *and 85 patients with* TRG4*.*

## 3.2 Simulation Settings

Additionally to the evaluation of hi2 on several real datasets we conduct a simulation study where we are in complete control of the data and therefore know their properties. This section describes how the simulated data are generated.

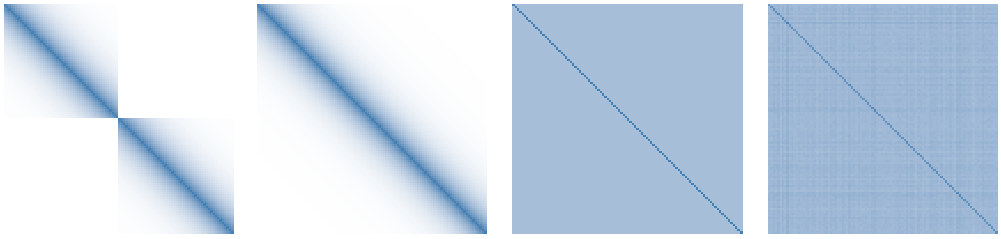
Depending on how the ordinal grouping of the samples is simulated to effect the features, we distinguish two simulation modes:

- *trend effects* and
- *plateau effects*.

For the remaining parameters of the data generation process both simulation modes will use the same base settings. In the simulation study one parameter at a time will be varied to study the effect of the aspect of the data that is governed by that particular parameter in the data generation process. Here, we first describe the base setting for the data generation.

We think of the generated features as gene expression levels similar to (normalized) microarray data. And we consider the samples represent patients with an ordinal response similar to tumor staging.

All results are generated using 500 simulation runs.



*Autoblocks* *Autoregressive* *Compound Symmetry* *Unstructured*

**FIGURE 3.5** *The four panels show samples (the first block of size 100×100) of the correlation structures that were used in the simulation. Dark colours represent high values, white represents 0.*
*The first structure we call* autoblocks*, where blocks of 50 features (genes) have an autoregressive correlation structure with parameter* $\rho = 0.9$*. The second, autoregressive correlation structure is similar but does not contain blocks. In the compound symmetry structure all off-diagonal elements are set to 0.5. And the unstructured correlation has random entries at the off-diagonal places.*

## *3.2.1 Base Settings*

### 3.2.1.1 Gene Expression

The generated data is moderately high-dimensional. The number of simulated features (genes) is set to 1000 in all settings. The values are simulated to follow a multivariate normal distribution with mean vector $\mu = (0, \ldots, 0)^\tau$ and a covariance matrix $\Sigma$.

In the base setting $\Sigma = \Sigma_{\text{autoblocks}}$ has a block structure which we call *autoblocks* where blocks of 50 genes are simulated to correlate but with no correlation between blocks. This could be interpreted as a very naive way of representing pathways, the genes of which are correlated. Of course such simulation is far from any real pathway representation as the simulated blocks are all of the same size, show all the same internal correlation structure and do not overlap, all of which is not true for real pathways. But the aim of this simulation study is not primarily to simulate biological data (as we can test the classifiers on real data) but to study properties of data that effect the classification performance.

In detail, the structure of $\Sigma_{\text{autoblocks}}$ is

$$\Sigma_{\text{autoblocks}} = \begin{pmatrix} [\mathbf{A}] & & & \\ & [\mathbf{A}] & & 0 \\ & & \ddots & \\ 0 & & & [\mathbf{A}] \end{pmatrix} \tag{3.1}$$

where each block $[\mathbf{A}] = (a_{ij})_{\substack{i=1,\ldots,50 \\ j=1,\ldots,50}}$ is a $50 \times 50$ matrix with an autocorrelation structure with parameter $\rho$, i.e.

$$a_{ij} = \rho^{|i-j|}. \tag{3.2}$$

In the base setting $\rho$ is set to 0.9.

#### 3.2.1.2 Patient Data

Data are generated for 90 samples (patients) of which 60 are used as a training set and the remaining 30 are used as test set. In each set the patients are assigned in equal numbers to one of the 5 groups {level1, level2, level3, level4, level5}.

#### 3.2.1.3 Setting Genes Differential

In order to set genes to be differential expressed between the groups, a different mean vector $\mu_l$ is used to generate the data in each group $l = 1, \ldots, 5$. 100 features (genes) are chosen randomly. We denote the index vector of chosen features with $\Delta$, so that

$$\Delta \subset \{1, \ldots, 1000\},\ |\Delta| = 100 \tag{3.3}$$

and we denote the restriction of $\mu_l$ to these positions $\Delta$ with $\mu_l|\,\Delta$.

The shape of the mean vector $\mu_l$ for $l = 1, \ldots, 5$ depends on the simulation mode (trend effect or plateau effect). Both of them generate data that generates ordered groups.

The trend effect is the simpler mode. Here, at the positions $\Delta$ the expression is shifted by some *effect size* $\delta$, so that

$$\mu_l|_\Delta = (l-1)\,\delta \tag{3.4}$$

(the other positions remain 0).

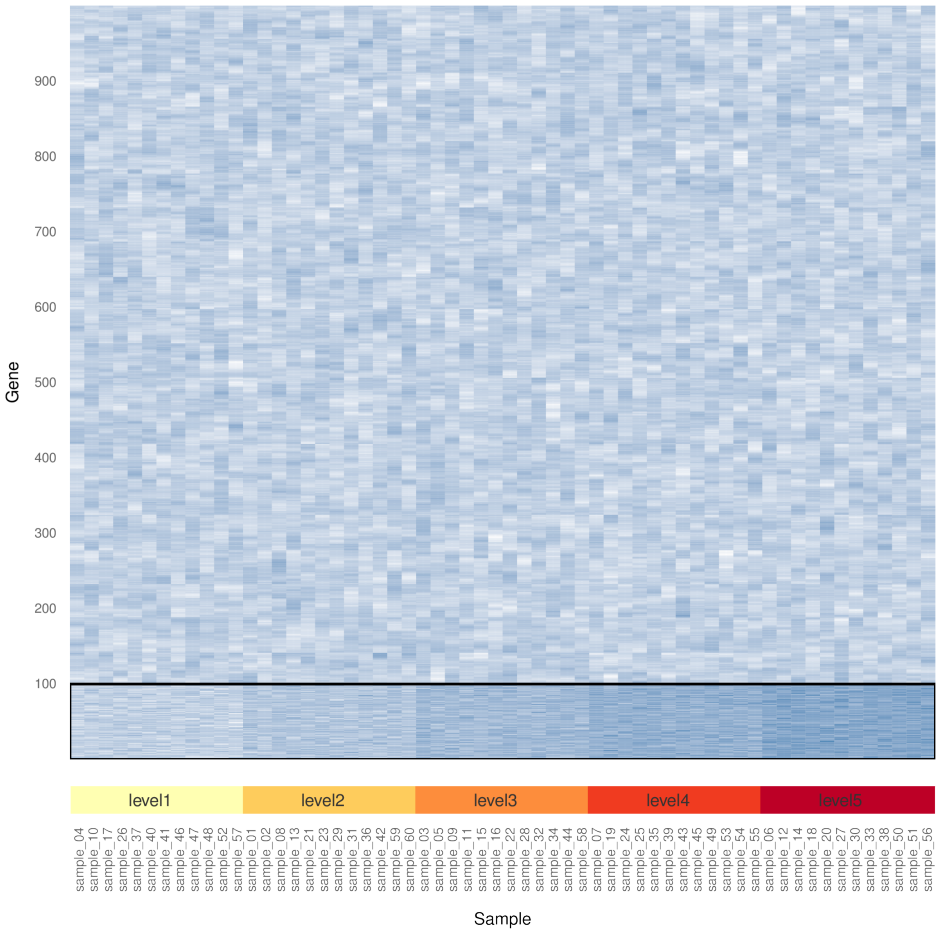



**FIGURE 3.6** *This heatmap displays one set of simulated data where the genes are displayed in the rows and the patients in the columns. The 100 differentially expressed genes are collected in the lower part and highlighted with a black box. The patients are grouped by response level in increasing order. Different levels of darkness represent the simulated expression level where white represents non-expressed genes and darker colour stands for higher expression.*
*In this setting the effect between the response levels is simulated to follow a linear trend pattern, which is visible in the bottom part that holds the differentially expressed genes.*

Figure 3.6 shows an example of a simulated data matrix where all differential genes are collected at the bottom. The differential part is marked with a black frame and the simulated trend pattern is clearly visible.

For the sake of simple notation, in the following we assume that all the divisions result in whole numbers. Otherwise some of the groupings will simply be not fully balanced, but the principle stays the same.

In the second mode (plateau effect) $\Delta$ is split further into $S$ groups $\Delta_s$, $s = 1, \ldots, S$ of the same size (if $\frac{L}{S}$ results in a whole number). The ordinal levels $\{1, \ldots, L\}$ are split into $S$ subgroups $L_s$, $s = 1, \ldots, S$ of size $\lfloor \frac{L}{S} \rfloor$. The genes in group $\Delta_s$ will stay at mean 0 in the first $(s-1)$ subgroups, will then follow a trend pattern with effect size $\delta$ in the subgroup $s$ and stay at the reached level for the remaining subgroups.

Formally, this results in

$$\mu_l|_{\Delta_s} = \begin{cases} 0, & l = 1, \ldots, (s-1)\lfloor \frac{L}{S} \rfloor + 1 \\ (l - s\lfloor \frac{L}{S} \rfloor + 1)\,\delta, & l = s\lfloor \frac{L}{S} \rfloor, \ldots, (s+1)\lfloor \frac{L}{S} \rfloor - 1 \\ \lfloor \frac{L}{S} \rfloor\,\delta, & l \geq (s+1)\lfloor \frac{L}{S} \rfloor \end{cases} \tag{3.5}$$

A sample data matrix generated using plateau effects is presented in Figure 3.7. Again, the differential genes are collected (here in their $S$ groups) in the black framed bottom part. The plateaus in each group are strongly visible.

In the base setting $\delta$ is set to 0.8.

### *3.2.2 Simulation Study*

In the simulation study presented in section 4.2.2.2 one parameter of the data generation is varied at a time to show the effect of that data property on the classification performance.

#### 3.2.2.1 Effect Size

The parameter where a direct effect on the classification performance is most obviously expected is the effect size $\delta$. The bigger the effect of the differential genes between the groups is, the easier this effect should be detected by the classifiers.

So, the first part in the simulation study looks at different effect sizes $\delta$. There are two ways to do that in the plateau effects mode. Here, the set of differentially expressed genes is divided into $S$ partitions. In case $\frac{L}{S}$ is not

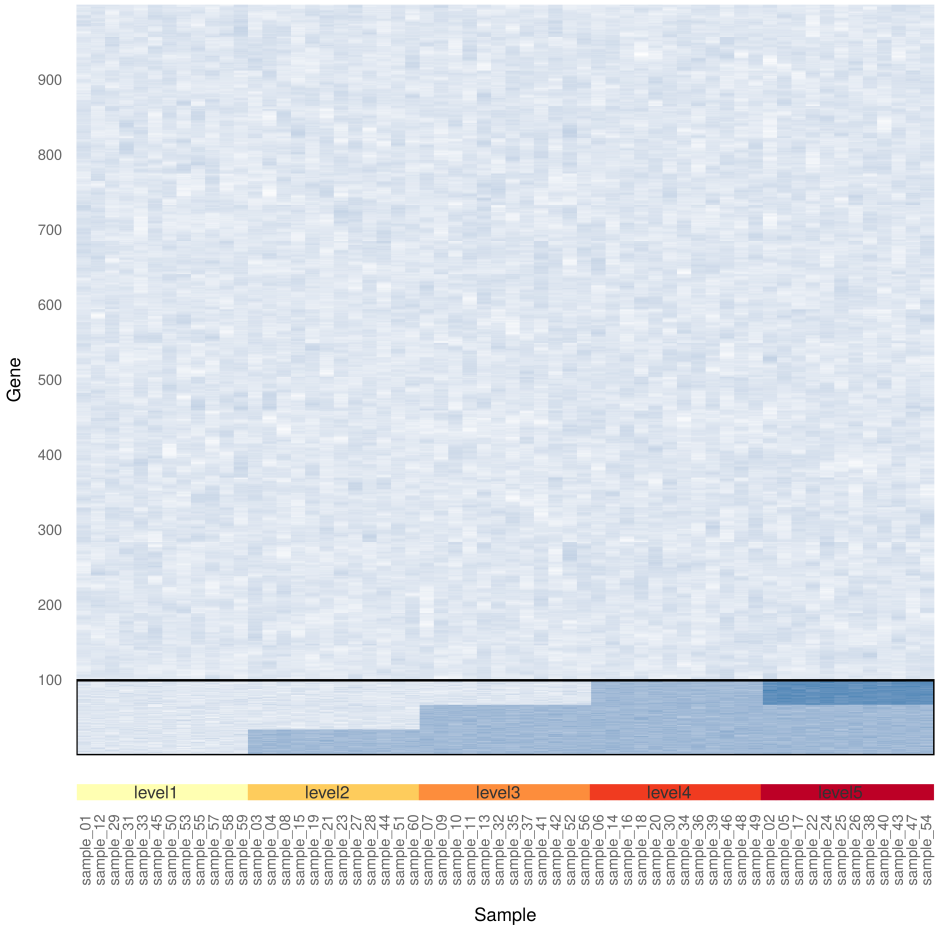



**FIGURE 3.7** *Similar to Figure 3.6 this heatmap displays one set of simulated data. Again, the genes are displayed in the rows and the patients in the columns and the 100 differentially expressed genes are collected in the lower part and highlighted with a black box. The patients are grouped by response level in increasing order. Different levels of darkness represent the simulated expression level where white represents non-expressed genes and darker colour stands for higher expression.*
*In this setting the effect between the response levels is simulated to follow what we call a plateau effect pattern. Here, the set of differentially expressed genes is partitioned into three groups, which are differentially expressed between different response levels.*

an integer, some partitions will have one more increment in an additional sample level. The effect size $\delta$ can either be *fixed* and ignore these additional increments. Or it can be *non-fixed*, i.e. divided by the number of increments in the actual partition $s = 1, \ldots, S$. In that second case, each partition $s$ gets the same effect size *delta* to split the groups it distinguishes. That means, that the effect size is not the same between all neighbouring levels $l = 1, \ldots, L$ which makes this setting also interesting. As a side effect, the overall effect in the

non-fixed $\delta$ setting is smaller than in the fixed $\delta$ setting and the values for $\delta$ are not directly comparable between the two settings.

#### 3.2.2.2 Correlation Structures

A second parameter that we can vary in simulated data is the correlation structure.

We present results for four different correlation structures:

- autoblocks,
- auto-regressive,
- compound symmetry, and
- unstructured.

Figure 3.5 displays the first $100 \times 100$ block of $\Sigma$ in each of these cases.

$\Sigma_{\text{autoblocks}}$ is the base setting and has been described already in section 3.2.1.1 (see Equation 3.1).

Very similar to $\Sigma_{\text{autoblocks}}$ is $\Sigma_{\text{autoregressive}}$ which has the same structure, but consists of one block only. This means that $\Sigma_{\text{autoregressive}} = (c_{ij})_{\substack{i=1,\ldots,1000 \\ j=1,\ldots,1000}}$ where

$$c_{ij} = \rho^{|i-j|}. \tag{3.6}$$

In the compound symmetry setting $\Sigma_{\text{compoundsymmetry}}$ has entries 1 on the diagonal and 0.5 at the off-diagonal entries.

In the unstructured case the matrix $\Sigma_{\text{unstructured}}$ is constructed starting with the absolute value of Gaussian random numbers with mean 0.1 in the upper triangle. The lower triangle is then filled to be symmetric. Then the matrix is multiplied with its transpose and scaled to 1 by dividing it by the maximal entry.

#### 3.2.2.3 Number of Levels

A last parameter that will be varied in the simulation study is the number of levels in the response. This is an interesting parameter, as it might be expected

that more groups mean more ordinal structure from which ordinal classifiers should especially benefit.

Again, there are two settings of how to increase the number of groups. In the first setting the number of groups is simply increased. As a side effect this means that the overall effect is increased as well: If we have set the effect size $\delta = 0.2$ then simulating 3 levels (in the trend mode) results in a maximal effect of 0.4 between the first and the last level, while simulating 10 levels results in that maximal effect being 1.8.

So, we present a second setting *max-*$\delta$ where we keep the maximal effect constant even when increasing the number of levels.

# 4 Results

This chapter presents first our own method hierarchical twoing – both the method and the implementation. The subsequent evaluation compares the performance of hierarchical twoing with that of some established classification algorithms, both specific ordinal and general ones. The first part of that evaluation contains a comparison of different parametrizations of hi2 (section 4.2.1). The second part (section 4.2.2) shows the results of the comparison to established methods – first on real data and second from a simulation study.

## 4.1 Hierarchical Twoing (hi2)

This section introduces in detail our method *hierarchical twoing (hi2)* which is an extension to simple twoing by Frank and Hall (see section 2.3.2.5). Hierarchical twoing is based on simple twoing but in a way combines that idea with the concepts of decision trees. Hierarchical twoing is an ensemble method and – again similar to simple twoing – is a wrapper around a binary base learner of the user's choice.

In the following subsections we will first introduce the method, then discuss the meta parameters that can be set for hi2, and finally present the implementation in the R package `hi2`.

### *4.1.1 The Method*

Hierarchical twoing inherits from simple twoing the idea to transform the $L$-level classification problem into a set of $L-1$ binary classification problems. Unlike si2, however, these $L-1$ binary classifiers or *base learners* are hierarchically ordered to form a decision tree structure. Figure 4.1 shows the example of the

rectal cancer data and two possible transformations of the 5-level problem into a tree of four 2-level problems. Of these $L-1$ base learners, the classifier $B_m$ will serve to separate the levels $1,\ldots,m$ from the levels $m+1,\ldots,L$. The tree is constructed by choosing one of the $L-1$ classifiers, say $B_r$, as the classifier at the root node. If $r=1$, the left child node is a terminal node representing the level 1. Otherwise, the left child is a subtree constructed in the same way as the whole tree, but using only the levels $1,\ldots,r$.
Analogously, the right child node is either a terminal node representing the level $L$ or a subtree constructed by only considering levels $r+1,\ldots,L$.

Presented with a sample $x$ each binary classifier $B_m$, $m=1,\ldots,L-1$ used in tree $B$ is assumed to produce class probabilities $p_{B_m}(L \mid x)$ and $p_{B_m}(R \mid x) = 1 - p_{B_m}(L \mid x)$ for its two classification results $L$ and $R$. To predict the response level $\hat{g} = B(x)$ for a sample $x$ using the tree $B$, the sample is passed down through the tree and a probability $p_B(l \mid x)$ is assigned to each level $l$, $l=1,\ldots,L$ as the product of the class probabilities of the binary base learners along the path to that level $l$. In the example given in Figure 4.1 (a) the level probability for TRG0 is simply $p_B(\text{TRG0} \mid x) = p_{B_1}(L \mid x)$ whereas the level probability for TRG1 is the product $p_B(\text{TRG1} \mid x) = p_{B_1}(R \mid x) \cdot p_{B_3}(L \mid x) \cdot p_{B_2}(L \mid x)$. If a clear decision is wanted, the predicted level $\hat{g} = B(x)$ is chosen as the level with the highest level probability $\arg\max_{l \in 1,\ldots,L} p_B(l \mid x)$.

Depending on the order in which the binary classifiers are chosen, the tree structure will be different. As an example Figure 4.1 shows two different trees for the same problem. hi2 uses all different trees that can be built for an $L$-level problem and combines the results. Thus, hi2 belongs to the class of ensemble methods. We denote the ensemble of possible trees with $\mathcal{B}$. The classification performance $w(B)$ for each tree $B \in \mathcal{B}$ is assessed by re-classifying the training data. This measure can be used as weight for the tree. If possible, the classification performance is measured with a zero-truncated and normalized (to sum up to 1) version of Kendall's $\tau$. Only when the whole training set is classified into the same class Kendall's $\tau$ is not computable and hi2 uses the accuracy instead.

There are several strategies to combine the results from the individual trees from the ensemble. Available in hi2 are the four strategies

- (weighted) average level probability,

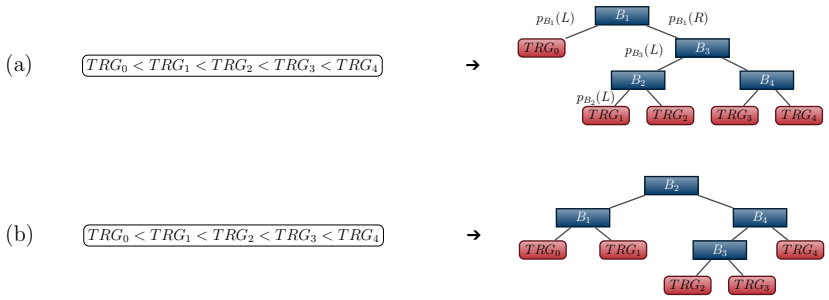



**FIGURE 4.1** *The concept of hi2 is to tranform an ordinal $L$-level classification problem into an ensemble of trees that are built using $L-1$ binary classifiers as their inner nodes. This figure shows two trees from the ensemble trained for the rectal cancer data. Each tree of the ensemble is trained and one of several aggregation strategies is applied to combine the results into an overall classification from the whole ensemble.*

- (weighted) majority vote,
- (weighted) median, and
- maximal weight.

In the last case, the ensemble of trees $\mathcal{B}$ is replaced with a single tree $B_{max} \in \mathcal{B}$ with $B_{max} = \arg\max_{B \in \mathcal{B}} w(B)$ and the classification is simply the result of the tree $B_{max}$ in the ensemble that has the best classification performance on the training data.

For the average level probability the level probabilities $p_{\mathcal{B}}(l \mid x)$ that are assigned to each level $l$, $l = 1, \ldots, L$ by the whole ensemble $\mathcal{B}$ for a sample $x$ are computed as

$$p_{\mathcal{B}}(l \mid x) = \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} p_B(l \mid x), \tag{4.1}$$

the weighted version includes the weight of the individual trees as factors:

$$p_{\mathcal{B}}(l \mid x) = \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} w(B)\, p_B(l \mid x). \tag{4.2}$$

For the majority vote the level probabilities $p_B(l \mid x)$ are replaced by the hard decisions $I(B(x) = l)$ and yield:

$$p_{\mathcal{B}}(l \mid x) = \sum_{B \in \mathcal{B}} I(B(x) = l) \tag{4.3}$$

for the unweighted and

$$p_{\mathcal{B}}(l \mid x) = \sum_{B \in \mathcal{B}} w(B)\, I(B(x) = l) \tag{4.4}$$

for the weighted version.

The median can be used to replace the sum and is especially meaningful in the ordinal setting. Again, the weights can be neglected as in

$$p_{\mathcal{B}}(l \mid x) = \underset{B \in \mathcal{B}}{\text{median}}\, I(B(x) = l) \tag{4.5}$$

or included as in

$$p_{\mathcal{B}}(l \mid x) = \underset{B \in \mathcal{B}}{\text{median}}\, w(B)\, I(B(x) = l) \tag{4.6}$$

We omit the formulas for the (weighted) median based on the scores of the individual trees.

The base learner that is used to train the internal binary classifiers $B_m$, $m = 1, \ldots, L-1$, is free to be chosen apt to the application and data at hand. For the aggregation methods based on the level probabilities the base learner obviously has to provide class probabilities. But other than that no general restrictions apply. The data, however, can impose some more restrictions. Most importantly in the context of this work high dimensional data require a base learner that can classify high dimensional data. A very simple yet robust base learner is the prevalence. If we denote the two classification outcomes of the binary base learner with $L$ and $R$, the prevalence base learner simply returns $p_{\text{prevalence}}(L) = \frac{Y^{\mathcal{T}} \in L}{|Y^{\mathcal{T}}|}$ the fraction of samples in the training data, that fall into $L$ and analogously for $R$.

When training the internal binary base learners there are two possible strategies (hi2 calls them *modes*) regarding the samples that are used. In the default *all data mode* each binary base learner is trained using all samples. That means that $B_m$ is trained using all samples with response $\leq m$ in one class and all samples with response $> m$ in the other class – regardless of the position of $B_m$ in the tree $B$.

In the second mode only the base learner at the root of the tree is trained on all data. If we assume that $B_r$ is chosen as the base learner at the root

node, the left subtree is constructed using only the samples with response $\leq r$. This way all base learners in the left subtree will only ever see a fraction of the available samples in the dataset. The same principle is repeated recursively in that subtree. Analogously, the right subtree is constructed. While this *split mode* allows the base learners to focus on the levels that have not been dealt with by the part of the tree above, the downside are the reduced datasets presented to the base learners in lower levels of the tree. This size reduction for the dataset used in lower levels of the tree is the more severe the more unbalanced the groups are.

Note also, that the computational burden is considerably larger in split mode. In the all data mode, all $L-1$ base learners can be trained upfront and re-used in each tree. This strategy is not available in split mode.

hi2 defaults to the all data mode.

As the last part in the presentation of hi2, we consider the computational complexity of the wrapper.

As hi2 is an ensemble method and computes all possible trees that can be constructed for an $L$-level classification problem, the overall complexity is dominated by the number of possible trees that have to be considered. This is in particular the case when the split mode is chosen. In the all data mode only $L-1$ base learners have to be trained and evaluated – the tree only governs which of the base learners have to be considered for any level $l$, $l = 1, \ldots, L$. But even in the all data mode the sheer number of possible trees quickly dominates the run time of the whole algorithm for problems with slightly more levels.

For shorter notation we define

$$q \coloneqq L - 1. \tag{4.7}$$

Using that the number of possible trees is a function of $q$ given by the *Catalan Numbers* (Koshy, 2009) which can be expressed as:

$$C_q = \frac{\binom{2q}{q}}{q+1} = \frac{(2q)!}{q!(q+1)!} \tag{4.8}$$

| | number of response levels | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| number of trees | 2 | 5 | 14 | 42 | 132 | 429 | 1430 | 4862 |

**TABLE 4.1** *The number of classification trees in hi2 grows with the number of response levels in the classification problem following the* Catalan Numbers*. The table shows the number of trees that hi2 has to construct for classification problems up to size 10.*
*While it is still feasible to apply hi2 for 10 class problems, computing time constraints hamper the applicability of hi2 to problems with many levels. That does not represent a real constraint, though, as with a growing number of classes, classification into ordinal classes approximates a regression problem.*

The Catalan Numbers are the sequence with A000108 in *The On-Line Encyclopedia of Integer Sequences* 2014. The first numbers for $q = 2, 3, \ldots$ are $2, 5, 14, 42, 132, 429, 1430, 4862, 16\,796, 58\,786, 208\,012, \ldots$.

Table 4.1 shows the relation of the number of levels in the classification problem to the number of possible trees. It is clear that the number of trees grows too quickly to apply hi2 to problems with many levels. As an example the *Charcot-Marie-Tooth Neuropathy Score (CMTNS)* (Shy et al., 2005; Mannil et al., 2014) has in its classical version 37 levels which would lead to 11 959 798 385 860 753 408 possible trees.

We do not consider that growing computational complexity a serious limitation, though, as for problems with that many levels regression based methods are a good choice anyway. We will present simulation results for problems with up to 10 levels, which are still unproblmatic to perform on today's desktop computers.

### *4.1.2 Parameters*

In this section we will first discuss the meta parameters introduced by hi2 and its design to pass a lot of freedom on to the user. Most of the parameters are decisions of how to run hi2, which could have been set fixed to the (now) default value. But hi2 enables the user to experiment also with the alternatives. The theoretical meaning of these parameters has been discussed already in the previous section, so we will present the practical use here.

Of more influence with regard to the performance will usually be the parameters of the base learner that is used as a plug-in classifier at the internal nodes of the trees in hi2. Thus, in the third part we will – as an example – present the influence of the number of feature to be extracted in our base learner.

But before that the base learner that is used in all the results in chapter 4 will be described in section 4.1.2.2.

#### 4.1.2.1 Meta Parameters of hi2

The most influential 'parameter' if you like is of course the choice of the binary base learner to be used. Since hi2 is merely a wrapper or scheme that enables a binary classifier to do ordinal classification, it needs to be provided with a base learner function.

The base learner is trained through hi2 and the resulting (base) classifier is called by hi2. Therefore at least two functions have to be present for any base learner: a *training function* (`base_learner_tfun`) that returns an instance of that classifier which can later be passed as argument to the *prediction function* (`base_learner_pfun`).

The strategy of how to use the samples for the training of the internal nodes of the tree is controlled by the parameter `strategy` and is set to the all data mode (`"all"`) by default.

The aggregation of classification results from all the trees in the ensemble is controlled through two parameters: `aggregation` and `use_scores`.

As discussed in the previous section `aggregation` can take one of the values

- `"majority vote"`,
- `"weighted majority vote"`,
- `"median"`,
- `"weighted median"`, or
- `"maximal weight"`.

If this parameter is not specified explicitly, `hi2` defaults to "median".

When `use_scores` is set to `TRUE` (the default), then the class probabilities of the base learners ($p_B(l \mid x)$) are used for the aggregation. When set to `FALSE` the hard predictions ($I(B(x) = l)$) are used.

The final meta parameter of hi2 is a `failing_split`. This parameter controls the behaviour of hi2 in the case that the training of the internal base learner $B_m$ fails for (at least) one $m \in 1, \ldots, L-1$. This can happen easily especially when hi2 is run in split mode, when one of the classes only contains very few samples. Some binary classifiers might fail in situations where one of the groups only contains a single sample.

hi2 can deal with that situation in two ways. It can discard the whole tree $B$ and remove it from the ensemble $\mathcal{B}$, which will happen when the parameter is set to the value `"discard"`.

Or the base learner $B_m$ can be replaced by the prevalence base learner which will always work. The parameter value for that is `"replace_prevalence"` which is also the default setting.

Except for the base learner which should be chosen carefully to work well in the data at hand, the meta parameter can be left at their default value without hesitation and are mainly implemented to allow for experiments. The results presented in chapter 4 are all based on the default values of these parameters as well.

#### 4.1.2.2 Example base learner 'limma+LDA'

We have already shortly introduced the prevalence as a simple base learner. Although that base learner will be too simple to be used as the primary base learner for most applications, it is robust in that it cannot fail and, thus, serves as a fall-back to replace failing base learners.

In this section we introduce the combination of limma (Smyth, 2005) and LDA (see section 2.3.1.1) as a multivariate classifier. This combination has proven itself to be a strong classifier (see chapter 4), which was the reason to use it also as the binary base learner in our applications of hi2 and also si2.

In that combination LDA is the actual classifier. But as we have seen in section 2.3.1.1 already, LDA is only applicable to non high-dimensional problems. Therefore, some dimension reduction has to be performed prior to

the application of LDA to a high-dimensional problem. In section 2.3.1.1 PCA and PLS are introduced as possible candidates for a dimension reduction.

Here, we use another approach to perform a feature selection in a filter step (see section 1.3) before the classification step. This approach univariately tests the association of each feature to the response. The result is a p-value assigned to each feature assessing its (univariate) association with the response.

The feature selection simply proceeds by choosing only the features with the smallest p-values. The number of features to retain is the most important parameter of the limma+LDA classifier. The next section will discuss that parameter.

To test the association between the individual feature and the response, this combination uses *limma: Linear Models for Microarray Data* which introduces an empirical Bayes method to moderate (i.e. shrink towards the pooled estimate) the standard deviation estimates across the features (Smyth, 2004). This shrinkage towards the pooled estimate results in much more stable estimates even for small numbers of samples.

Smyth shows that the moderated t-Statistic indeed follows a t-distribution and that it extends to tests of composite null hypotheses through moderated F-statistics. That means that we can perform a one-way ANOVA (internally based on such moderated F-statistic) to test for an overall effect between the feature value and the response levels using all pair comparisons.

#### 4.1.2.3 Number of Selected Features as Example of Parameters for the Base Learner

As we have seen in the previous section, the number of features to retain in the feature selection step is a key parameter of the limma+LDA classifier.

In order to find the best number here, we propose to perform a grid search over a range of possible values. An internal cross validation should be used to asses the classification performance of the full limma+LDA classifier for each of the values for the number of selected features. The best performing number assessed in the internal cross validation can then be used.

Figure 4.2 shows the results from the individual runs of that internal cross validation in the example of the miRNA Breast Cancer data (see section 3.1.2).

Interestingly, on average less selected features lead to better classification performance.

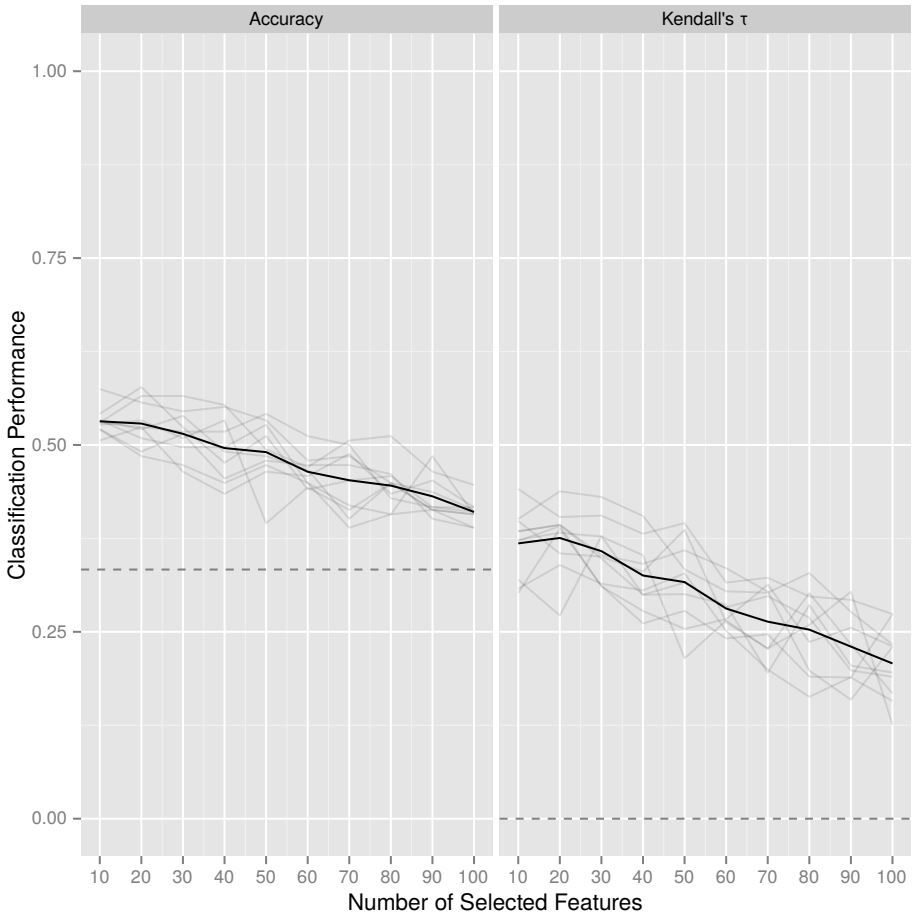



**FIGURE 4.2** *In each of the 10 folds of the cross validation used to estimate the generalization performance of hi2 on the miRNA breast cancer data, the performance of the classifier was assessed when 10, 20, ..., 100 features were selected. An internal 10 fold cross validation was applied to do this assessment. The best performing number of selected features is used in the fold at hand.*
*The grey lines show the performance measures for the different numbers of selected features in each of the 10 folds. The black lines are the mean values across the folds.*

### *4.1.3 The Implementation*

A reference implementation of hi2 is available as package for the statistical computing framework `R` (R Core Team, 2014) at `https://gitorious.org/hi2` under the GPL.

This section describes some aspects of the implementation but is not intended to be a proper documentation of the R-package. For that please refer to the included documentation.

The package can easily be installed from the repository using the devtools package (Wickham and Chang, 2014) via

```
library("devtools")
install_gitorious("hi2/hi2")
```

After loading the package with

```
library("hi2")
```

the following functions are available

- `hi2()`,
- `prevalence()`,
- `limmaPlusLDA()`, and
- `calcPossibleTrees()`.

`hi2()` is the main function of the package, which provides the implementation of the hi2 algorithm.

R provides several frameworks for object oriented programming of which S3 is the only one available in base R. S3 provides *generic-function* object orientation, which allows the definition of *generic functions* which invoke different methods depending on the class of their first argument. See the chapter "OO field guide" in Wickham, 2014, for an excellent introduction into object orientation in R.

Formally, `hi2()` is an S3 generic function and provides two methods that differ in how the data are passed:

- `hi2.default()` accepts samples and response as `data.frame` and `vector` and
- `hi2.formula()` accepts both samples and response together in one `data.frame` and makes use of R's formula interface to specify the model. This second method is implemented for convenience only and only extracts the samples and response from the full `data.frame` to be able to call the first method.

The most important arguments to `hi2()` are the data and the functions to train the base learner and to predict using the base learner. Here, the package benefits from `R` being a programming language, as that makes it easy to pass functions as arguments.

The base learner functions can be user supplied. The requrired signature for the training function is

```
base_learner_tfun(x, y, <optional arguments>)
```

where `x` is the `data.frame` containing the samples and `y` is the vector of responses.

The required signature for the prediction function is

```
base_learner_pfun(trained_base_learner, y, ...)
```

where `trained_base_learner` is an object returned by the `base_learner_tfun()` function.

Additionally, we make use of the ellipsis argument, which is available in `R`. The last argument in the signature of `hi2.default()` is literally `...`. That means that `hi2.default()` accepts arbitrary arguments that do not have to be declared in the definition of `hi2.default()`. Instead, all arguments that do not match the declared arguments will be passed on to the base learner functions. This allows for a seamless integration of new base learners into the wrapper `hi2`.

Two base learners are included in the package. These make the package usable out of the box and additionally serve as reference of how to use other base learners in hi2. The two included base learners are the prevalence classifier and the combination of limma and LDA. Both of them again make use of the `S3` framework and provide methods supporting the formula interface additionally to the plain variable passing.

Both of them are not limited to be used as (binary) base learners within the `hi2` wrapper, but constitute multi-class classifiers of their own. Thus, we took care to implement them without restriction and they might be useful even outside of hi2.

Altogether, there are three classifiers included in the `hi2` package:

- hi2

- limma + LDA
- prevalence

All of them provide a method for the generic function `predict()` which is called automatically when an object of the respective class is passed. Thus, the prediction using one of these classifiers behaves as most other classifiers available in R.

Below, we present an example of how to use the `hi2` package on the Leukemia data (see section 3.1.3). Most of the code is needed to prepare and select the data. Both training of the hi2 classifier and the prediction (these are the last two statements) are simple function calls.

```
## load the data
data("ALL")

## subset the data
## (tumor stage annotation not available for all patients)
ALL_B_levels <- c("B1", "B2", "B3", "B4")
ALL_B <- ALL[, is.element(pData(ALL)$BT, ALL_B_levels)]

## extract the data we need
y <- factor(pData(ALL_B)$BT, levels=ALL_B_levels,
        ordered=TRUE)
x <- t(exprs(ALL_B))

## select training set
idx_train <- sample(1:length(y), round(2/3*length(y)))

## train the classifier using limmaPlusLDA as base
## learner and using the top 20 genes in each feature
## selection step
pred_limmaPlusLDA <- hi2(x[idx_train,], y[idx_train],
             base_learner_tfun="limmaPlusLDA",
             feature_threshold=20)
## predict
prediction <- predict(pred_limmaPlusLDA,
              newdata=x[-idx_train,])
```

The design of `hi2` makes it easy to supply own base learner functions to be used in the `hi2` framework. As an example, the integration of SVM as base learner does not require any code for the training function, but `svm()` from the package `e1071` (Meyer et al., 2014) can be passed directly as argument to `hi2()`.

For the prediction function a small wrapper is needed, though. `hi2` passes the numeric values 0 and 1 to its base learner, and `svm()` does a regression by default in that case. So, our wrapper rounds the resulting prediction to 0 and 1 again. Additionally, the object returned by the prediction function of a base learner is expected to be of type `list` with at least the slots `$class` and `$probabilities`. The full wrapper is still a simple function:

```
predict.hi2svm <- function(object, newdata, ...)
{
  numprediction <- predict(object, newdata)

  numprediction[numprediction < 0] <- 0
  numprediction[numprediction > 1] <- 1

  classprediction <- round(numprediction)

  probs <- matrix(c(1-numprediction, numprediction),
          ncol=2)

  return(list(class=classprediction,
          probabilities=probs))
}
```

When this wrapper function is defined, using SVM inside `hi2` simply amounts to specify the parameters `base_learner_tfun` and `base_learner_pfun` as in

```
hi2(x=X, y=Y,
    base_learner_tfun = svm,
    base_learner_pfun = predict.hi2svm)
```

The last exported function from the `hi2` package is `calcPossibleTrees()`. This function is mainly of internal use and does not have to be called directly. But we decided to expose it in the package interface for the interested reader.

That function takes a vector of inner nodes and recursively generates all possible trees. In our notation the vector will be of length $q = L - 1$ and the passed vector will contain the nodes $B_1, \ldots, B_q$.

The return value is a matrix of which each row represents one tree by simply enumerating the inner nodes in each level of the tree from left to right.

```
calcPossibleTrees(c('B_1', 'B_2', 'B_3', 'B_4'))
```

| | | | | |
|---|---|---|---|---|
| 1 | $B_1$ | $B_2$ | $B_3$ | $B_4$ |
| 2 | $B_1$ | $B_2$ | $B_4$ | $B_3$ |
| 3 | $B_1$ | $B_3$ | $B_2$ | $B_4$ |
| 4 | $B_1$ | $B_4$ | $B_2$ | $B_3$ |
| 5 | $B_1$ | $B_4$ | $B_3$ | $B_2$ |
| 6 | $B_2$ | $B_1$ | $B_3$ | $B_4$ |
| 7 | $B_2$ | $B_1$ | $B_4$ | $B_3$ |
| 8 | $B_3$ | $B_1$ | $B_2$ | $B_4$ |
| 9 | $B_3$ | $B_2$ | $B_1$ | $B_4$ |
| 10 | $B_4$ | $B_1$ | $B_2$ | $B_3$ |
| 11 | $B_4$ | $B_1$ | $B_3$ | $B_2$ |
| 12 | $B_4$ | $B_2$ | $B_1$ | $B_3$ |
| 13 | $B_4$ | $B_3$ | $B_1$ | $B_2$ |
| 14 | $B_4$ | $B_3$ | $B_2$ | $B_1$ |

Note, that this representation does not capture the structure of the tree but requires that the structure is implicitly encoded in the labels of the nodes.

## 4.2 Evaluation

Here we turn to the evaluation of the classification performance of hi2. We present first a comparison of different parametrizations of hi2 using real data. After that we compare hi2 with established algorithms using both, real and simulated data.

In all considered data, both real and simulated, the performance of all classifiers was evaluated using the accuracy and Kendall's $\tau$. We present results from both measures side-by-side. In all cases where this is practical we indicate

(with a dashed line) the performance which one could expect to achieve when merely guessing the correct level.

### 4.2.1 Parametrizations of hi2

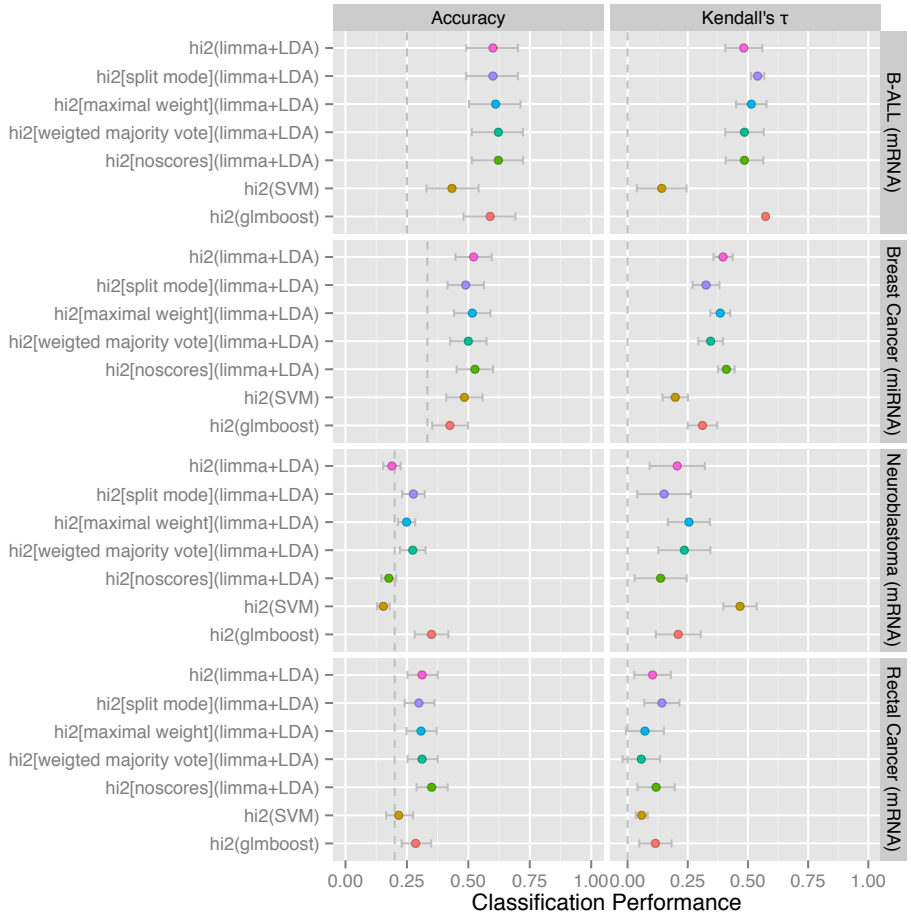



**FIGURE 4.3** *Four datasets (the four rows) were used to compare the performance of hi2 using different parameter settings. The two columns show the two performance measures accuracy and Kendall's τ, the values of which are displayed along the x axis.*
*In comparison with the default base learner 'limma+LDA' the two bottom values show the performance using SVM or regularized logistic regression (glmboost) as base learner. The other values show the performance of hi2 when changing the meta-parameters of hi2.*
*The performance using SVM as base learner is very unstable, while the performance using glmboost is comparable to the default base learner.*

As described in section 4.1.2 there are many ways to do a classification using hi2. Different base learners can be plugged in, there are two modes of how to use the available samples, and there are different aggregation methods,

to repeat the most important parameters. Figure 4.3 (and Table 4.2) compares the performance of several parametrizations of hi2 on real data.

Besides the default 'limma+LDA' base learner we present results using an SVM and glmboost as alternatives.

Additionally, the aggregation methods 'weighted majority vote' and 'maximal weight' are compared to the default 'median' method. For the 'median' method the variant *noscores* is included as well which does not use the class probabilities of the base learner but relies on the hard classification results.

And finally, hi2 with base learner 'limma+LDA' has been run in split mode in comparison to the default all data mode.

Most settings produce comparable results in all datasets and it is hard to choose a clear recommendation or to see consistent trends separating the different settings.

Most striking is the unstable behaviour when using the SVM as base learner. While the performance – when measured using Kendall's $\tau$ – is exceptionally good in the neuroblastoma data, we observe weak performance in the other datasets, especially in the B-ALL data.

Using glmboost as base learner is beneficial in the B-ALL data and the neuroblastoma data compared to the default settings 'hi2(limma+LDA)' but not advisable in the breast cancer data.

Similarly, the split mode is only in some cases favorable and shows even contradictory behaviour in the neuroblastoma data, where it is clearly more accurate than the default settings, but produces a lower Kendall's $\tau$.

The different aggregation methods are too close to see a trend preferring one over the other.

Also the *noscores* setting performs comparably to the setting using the scores in all datasets.

Overall the B-ALL data seems to be most easy to be classified correctly using both measures, accuracy and Kendall's $\tau$ – the only exception is the SVM based version of hi2. The breast cancer data, although having even less levels, is harder and the values for both, accuracy and Kendall's $\tau$, tend to be lower.

The neuroblastoma data and the rectal cancer data both have 5 levels and show opposite trends when assessed with accuracy and Kendall's $\tau$: While

the neuroblastoma data is harder to classify accurately, the classifiers tend to produce higher values for Kendall's $\tau$.

### *4.2.2 Classifier Comparison*

On all datasets, real and simulated, we evaluated the performance of hi2 with limma+LDA as base learner. Since limma+LDA can be used as a multiclass classifier on its own it is a natural comparison to do and limma+LDA is included in all comparisons. The second natural comparison is si2 with the same base learner limma+LDA, because hi2 can be seen as an extension of si2. So, si2 is part of the comparison, too.

As a null model, we include the prevalence classifier. The prevalence can serve as null model, since it does not use the feature data at all.

We include kkNN and decision trees using the twoing method (as implemented in the `rpartOrdinal` package) as established ordinal classifiers into the comparison as well.

Finally, we include as two additional references:

- a non-ordinal SVM, where we used the radial basis kernel function and a voting of all pair comparisons to handle the multi-class problems and
- a regression method, where we resorted to boosting for regularization as provided by the `mboost` package.

#### 4.2.2.1 Real Data

The results on all real datasets are presented in Figure 4.4 and Table 4.3.

Globally, the datasets are again ordered by the difficulty to classify them correctly when assessed by Kendall's $\tau$. So, there seems to be a tendency, that the B-ALL data is easier to classify than the breast cancer data. More difficult are the 5-level datasets, where the neuroblastoma data seems less difficult than the rectal cancer data.

kkNN proves to be the main exception from this trend, as it outperforms the other methods in the 5-level datasets but does not belong to the top-performers in the easier B-ALL and breast cancer data.

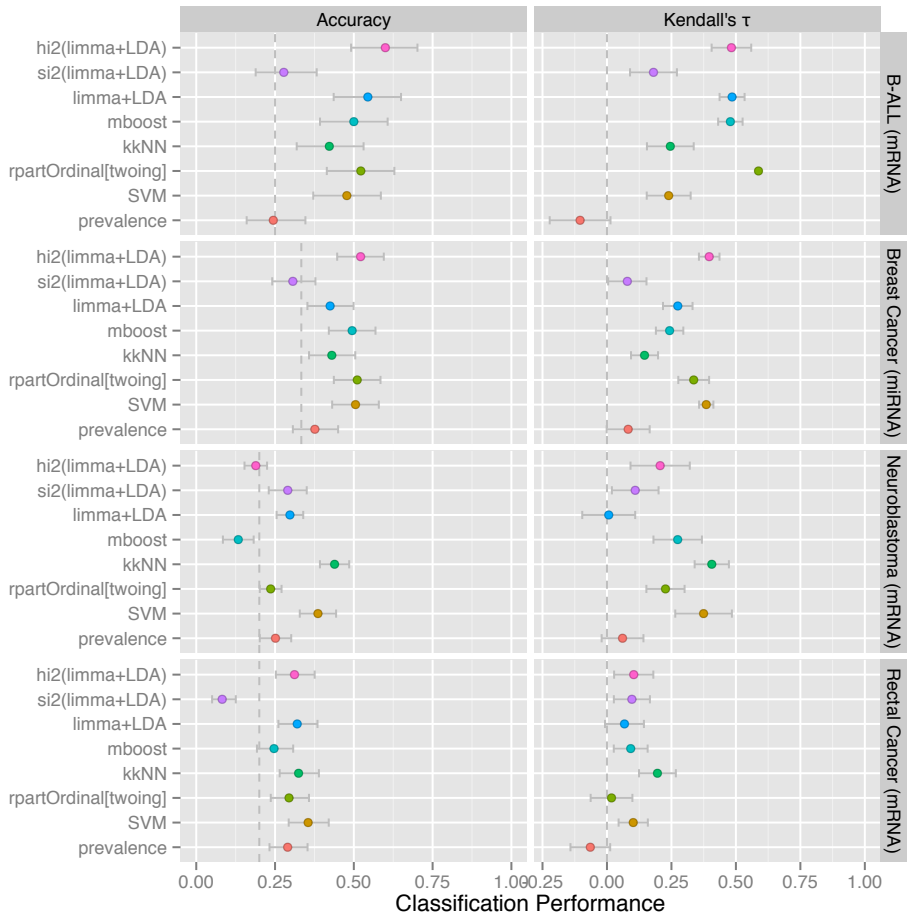



**FIGURE 4.4** *As in Figure 4.3 the x axis shows the classification performance measured using the accuracy (left panel) and Kendall's τ (right panel). Again the rows of panels display results on four different datasets.*
*The classifier prevalence is used as reference null model. Direct competitors to hi2 are si2, of which hi2 is an extension, and limma+LDA, which is used as base learner in hi2.*
*With the exception of the accuracy in the neuroblastoma data, hi2 shows consistently strong performance. In contrast, especially kkNN and SVM show mixed results: While they perform very good in the neuroblastoma data, their performance is weak on the B-ALL data.*

The overall weakest performance shows the nullmodel, the prevalence classifier, that rarely beats mere guessing. Only in the rectal cancer data its accuracy is clearly superior to guessing, but on the other hand on this data its Kendall's $\tau$ is particularly low and even negative.

For the B-ALL data, the classifiers split into two groups. hi2, limma+LDA, mboost, and rpartOrdinal form the group of good performers, which outperforms the classifiers in the second group, namely si2, kkNN, and SVM.

In the breast cancer data, hi2 and rpartOrdinal again belong to the good performers – joined by the SVM this time. limma+LDA, mboost, and to some extent kkNN show an intermediate performance, while si2 is particularly weak.

Considering the neuroblastoma data, kkNN clearly outperformes the other algorithms with SVM performing nearly as well. In terms of Kendall's $\tau$, the group of hi2, mboost, and rpartOrdinal still performs reasonable, while si2 and limma+LDA perform poorly. In terms of accuracy, the picture changes and si2 and limma+LDA still outperform mere guessing, while hi2, mboost, and rpartOrdinal perform poorly.

On the last data, the rectal cancer data, only hi2, kkNN, and SVM perform clearly superior to random guessing in both measures. But except for the outliers (kkNN in positive direction, and si2 and rpartOrdinal in the negative direction) all algorithms are close to each other.

#### 4.2.2.2 Simulation Study

The simulation study divides into two parts. The first part shows results using the simple trend pattern in the data generation, the second part shows results from data following the plateau pattern.

The second part holds more results, as we are more interested in less 'perfect' groupings than the simple trend effects.

**Trend Effects** As presented in section 3.2.1.3, every level of the response is separated by the same effect size $\delta$ in this setting, leading to a perfectly linear trend in the differential genes.

With increasing effect size $\delta$ (Figure 4.5) the performance of all classifiers improves. The main exception is the prevalence classifier which as the nullmodel does not make use of the feature data and, thus, does not profit from any effect. Also, si2 benefits surprisingly little from the increasing effect – especially in terms of accuracy.

We can see a group of rpartOrdinal, SVM, and kkNN with a moderate performance, where rpartOrdinal performs weak at low effect sizes and only catches up when the effect size reaches very high levels.

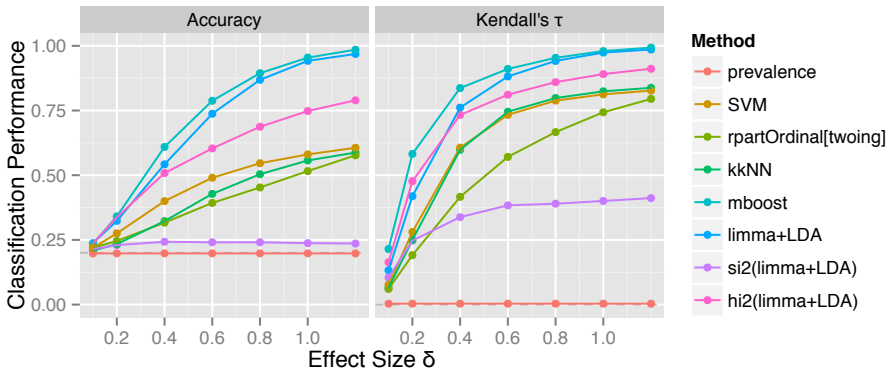



**FIGURE 4.5** *This figure shows the classification performance (on the y axis) for different effect sizes (x axis). The two panels show the accuracy (left panel) and Kendall's $\tau$ (right panel). The different algorithms are given different colours.*
*The simulated data for these simulations show a perfectly linear trend between the response levels.*
*The null model 'prevalence' performs worst and does not profit from increased effect sizes. All other algorithms perform better for larger effect sizes. Only si2 benefits surprisingly little. The linear regression based mboost has to be expected to perform very well in this trend setting.*

hi2 performs comparable to limma+LDA at small effect sizes but limma+LDA shows a superior performance for large effects. Both are still outperformed by mboost, though.

Looking at different correlation structures (Figure 4.6) we see hi2 together with limma+LDA always in the top group. Although, again both of them are outperformed by mboost – especially in terms of Kendall's $\tau$.

kkNN performs comparable to hi2 in the compound symmetry and unstructured correlations but has a weaker performance in the other settings.

**Plateau Effects** Now, we turn to plateau effects (see section 3.2.1.3) which deviate from the perfectly linear trend, while still being ordinal. That is achieved by partitioning the differential genes into $S$ groups where each group is associated with a partition of response levels and is differentially expressed between these levels while plateauing outside.

Compared with the trend effects (Figure 4.5), the regression based mboost does not outperform all other algorithms in the plateau effects setting (Figure 4.7). While mboost still shows the best Kendall's $\tau$ for small effects, it is outperformed by hi2 in situations with large effects. In terms of accuracy mboost is even outperformed by hi2 and limma+LDA.

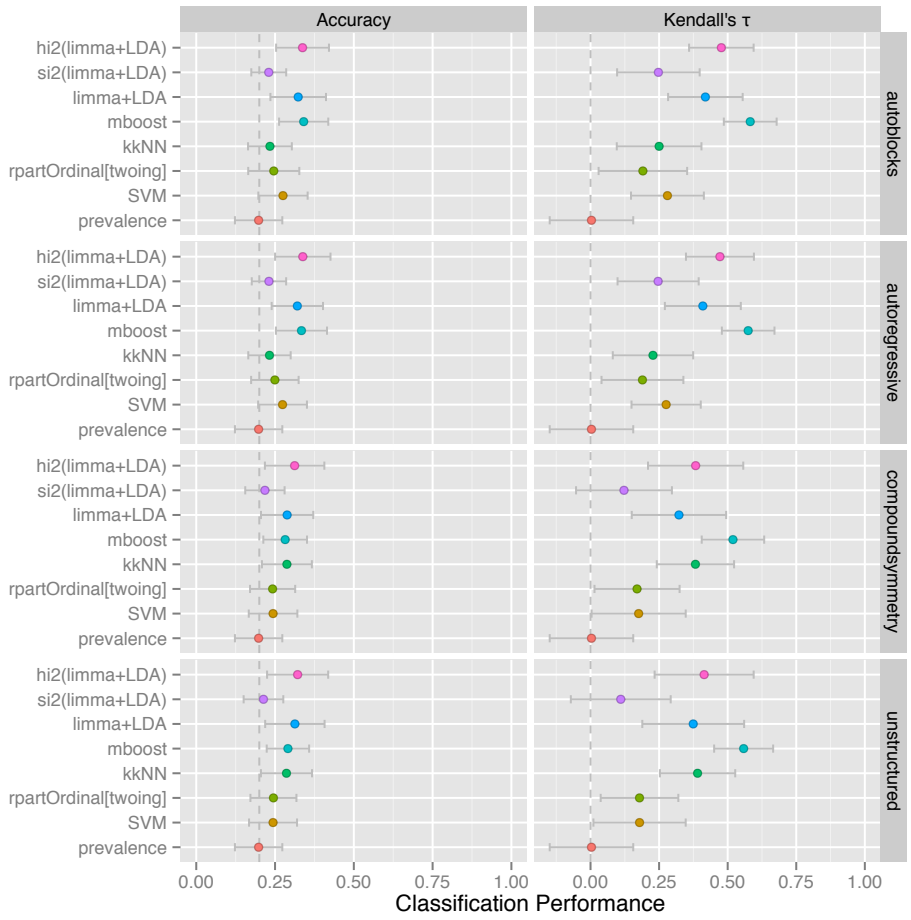



**FIGURE 4.6** *This figure shows the classification performance (on the x axis) for different correlation structures in the rows of panels. The two panel columns show the accuracy (left column) and Kendall's $\tau$ (right column).*
*The simulated data for these simulations show a perfectly linear trend between the response levels.*
*The null model 'prevalence' performs worst, while – again as expected – mboost outperforms all other algorithms. Reason is again the perfect linear trend in the data that corresponds exactly to the linear model fit by mboost. hi2 performs consistently strong, in contrast to kkNN which shows good performance only in the compound symmetry and unstructured settings.*

As with trend effects, SVM, kkNN, and rpartOrdinal cluster together at intermediate performance levels. And again, si2 gains comparably little from increased effect sizes.

hi2, limma+LDA, and mboost build the group of top performing algorithms across all different correlation structures (Figure 4.8) when measuring the performance with Kendall's $\tau$. The accuracy of mboost, however, is considerably

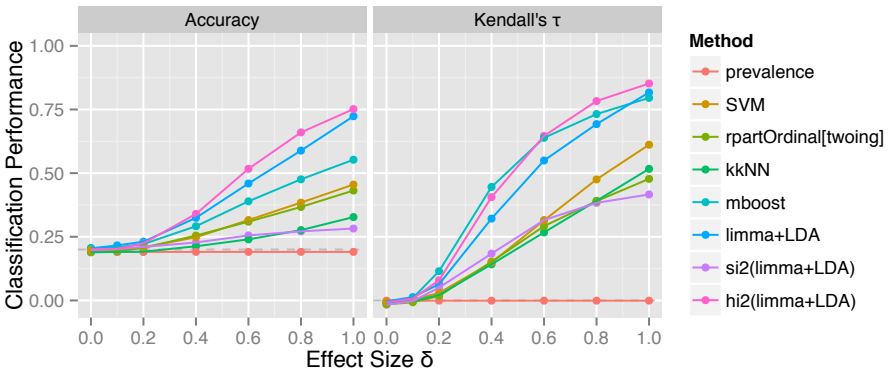



**FIGURE 4.7** *This figure is similar to Figure 4.5 and again shows the classification performance (on the y axis) for different effect sizes (x axis).*
*The difference is that the simulated data for these simulations follow a platau effects pattern between the response levels.*
*In these setting, hi2 performs comparably to mboost in terms of Kendall's $\tau$ and outperforms mboost clearly in terms of accuracy.*

lower, especially in the settings with unstructured or compound symmetry correlation structures.

The performance of kkNN in terms of Kendall's $\tau$ is – similar to the results from the trend effects setting – better in the data settings with unstructured and compound symmetry correlation structure. When measured with accuracy that effect is not visible.

Finally, we consider the effect of the number of response levels in the data. Here, all results are derived twice: First with fixed effects (Figure 4.9) and second with non-fixed effects (Figure 4.10).

As described in section 3.2.1.3 in the fixed effects setting the maximum effect between the first and the last of the $L$ response levels is $\lfloor \frac{L}{S} \rfloor \, \delta$, whereas in the non-fixed effects setting, that maximum effect is $S \, \delta$ and, thus, independent of the number of levels.

In this simulation setting we aim to avoid to model the influence of the effect size again. So, in order to avoid to increase the maximal effect with the increment of the number of response levels, we fix the maximal effect to be 1 and instead scale $\delta$ appropriately.

We first consider the fixed effects setting. In the presented case with $S = 3$ the simulated numbers of levels group into

- $\{3, 4\}$ with $\delta = 1$,

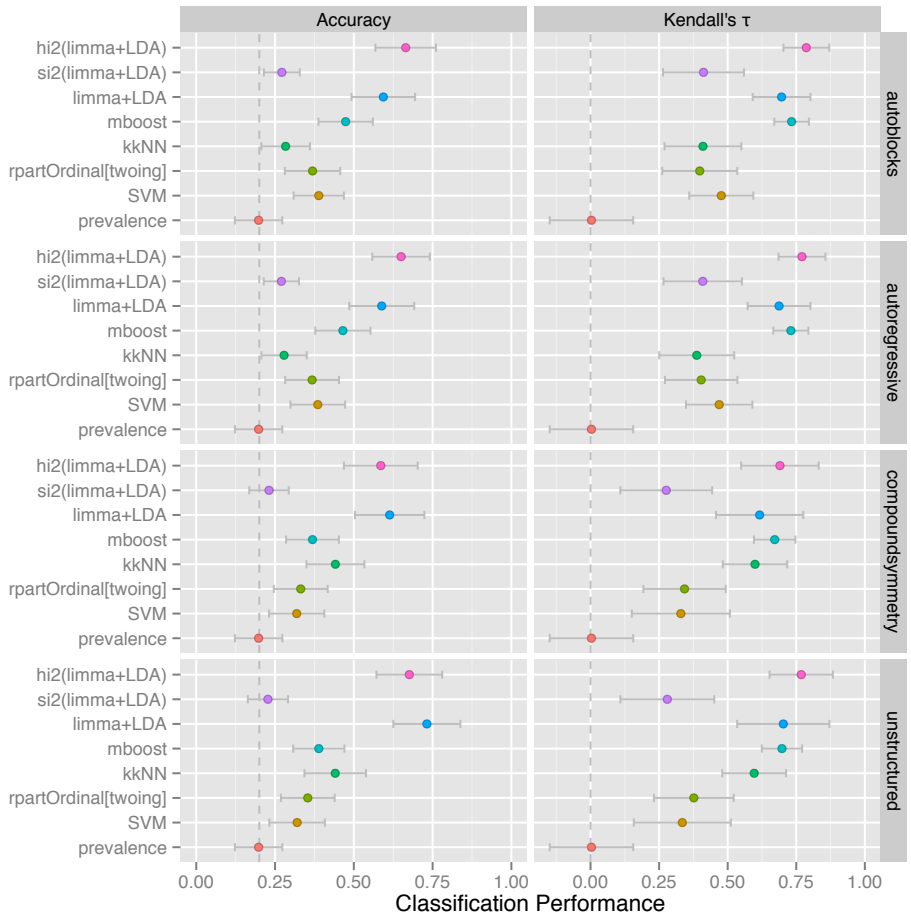



**FIGURE 4.8** *This figure shows the classification performance (on the x axis) for different correlation structures in the rows of panels. In contrast to the corresponding Figure 4.6 these results are based on data with plateau effects.*
*hi2 is the best performing algorithm across all correlation structures with limma+LDA as a strong competitor – especially when considering the accuracy. As in Figure 4.7 mboost is strong only in terms of Kendall's $\tau$, but performs comparably weak in terms of accuracy.*

- $\{5, 6, 7\}$ with $\delta = \frac{1}{2}$
- $\{8\}$ with $\delta = \frac{1}{3}$

This pattern is clearly visible in Figure 4.9. When $\delta$ stays constant, the accuracy of hi2 also stays almost constant. That same effect but to a much smaller extent can also be seen in other algorithms, especially mboost and limma+LDA.

The ordinal measure Kendall's $\tau$ can even benefit from an increased number of response levels. This happens especially for hi2 and kkNN, and a little less pronounced for mboost and limma+LDA. Overall, the classifications produced

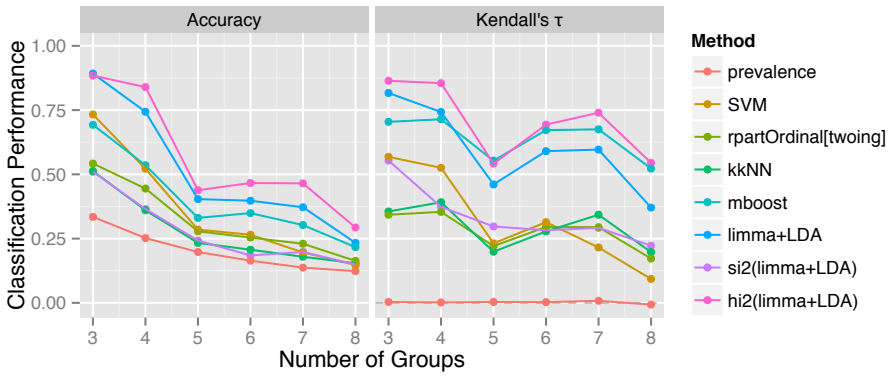



**FIGURE 4.9** *This figure shows results from the simulation when the number of response levels (displayed on the x axis) is varied between 3 and 8.*
*The interesting pattern is due to different effect sizes as explained in the main text.*
*hi2 performs again strongly in these settings and benefits a lot from increments in the number of response levels.*

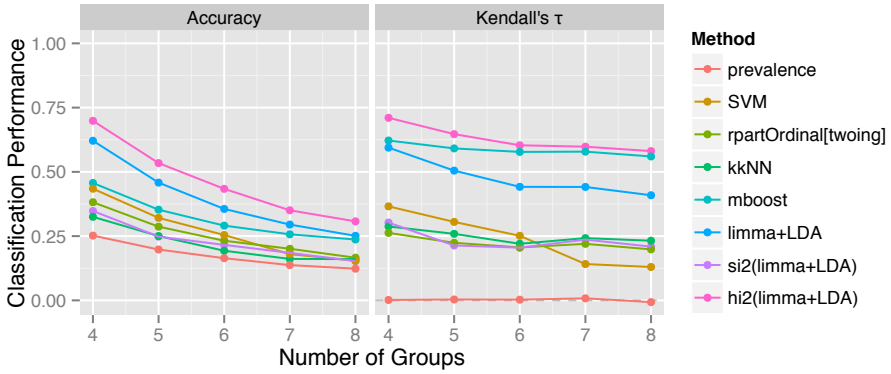



**FIGURE 4.10** *This figure shows the same simulation as Figure 4.9 but this time without changing effect size. Therefore, the striking pattern is not visible here. Instead the order of the algorithms stays the same across all numbers of response levels – main exceptions are the SVM, which has problems with many response levels, and mboost, which benefits more than the others from increments in the number of levels.*

by hi2, mboost, and limma+LDA result in the highest values for Kendall's $\tau$. Of the other algorithms SVM performs still quite strong with a small number of response levels, but shows a weak performance when the data contains more levels.

In the simulation with non-fixed $\delta$ where $\delta$ is independent of the number of response levels, no grouping effect is visible. The performance of all algorithms decreases here with an increasing number of response levels. This decrease is more pronounced when the accuracy is chosen as measure. In terms of Kendall's

$\tau$ mboost shows only a minimal decrease in performance so that in settings with more than 5 levels, hi2 and mboost result in similar values for Kendall's $\tau$. In both measures, hi2 outperformes the other algorithms. And again, hi2, mboost, and limma+LDA show a higher performance than the remaining algorithms.

| | | Accuracy | | Kendall's $\tau$ | |
|---|---|---|---|---|---|
| | | mean | se | mean | se |
| B-ALL | hi2(limma+LDA) | 0.6 | 0.109 | 0.48 | 0.077 |
| | hi2[split mode](limma+LDA) | 0.6 | 0.109 | 0.54 | 0.028 |
| | hi2[maximal weight](limma+LDA) | 0.61 | 0.109 | 0.51 | 0.063 |
| | hi2[weigted majority vote](limma+LDA) | 0.62 | 0.108 | 0.49 | 0.08 |
| | hi2[noscores](limma+LDA) | 0.62 | 0.108 | 0.49 | 0.078 |
| | hi2(SVM) | 0.43 | 0.104 | 0.14 | 0.103 |
| | hi2(glmboost) | 0.59 | 0.109 | 0.57 | |
| Breast Cancer | hi2(limma+LDA) | 0.52 | 0.074 | 0.4 | 0.04 |
| | hi2[split mode](limma+LDA) | 0.49 | 0.074 | 0.33 | 0.056 |
| | hi2[maximal weight](limma+LDA) | 0.52 | 0.074 | 0.38 | 0.041 |
| | hi2[weigted majority vote](limma+LDA) | 0.5 | 0.074 | 0.35 | 0.051 |
| | hi2[noscores](limma+LDA) | 0.53 | 0.074 | 0.41 | 0.034 |
| | hi2(SVM) | 0.48 | 0.074 | 0.2 | 0.053 |
| | hi2(glmboost) | 0.42 | 0.072 | 0.31 | 0.061 |
| Neuroblastoma | hi2(limma+LDA) | 0.19 | 0.036 | 0.21 | 0.115 |
| | hi2[split mode](limma+LDA) | 0.28 | 0.046 | 0.15 | 0.112 |
| | hi2[maximal weight](limma+LDA) | 0.25 | 0.035 | 0.25 | 0.087 |
| | hi2[weigted majority vote](limma+LDA) | 0.27 | 0.052 | 0.24 | 0.108 |
| | hi2[noscores](limma+LDA) | 0.18 | 0.03 | 0.14 | 0.108 |
| | hi2(SVM) | 0.15 | 0.025 | 0.47 | 0.069 |
| | hi2(glmboost) | 0.35 | 0.068 | 0.21 | 0.093 |
| Rectal Cancer | hi2(limma+LDA) | 0.31 | 0.059 | 0.1 | 0.076 |
| | hi2[split mode](limma+LDA) | 0.3 | 0.058 | 0.14 | 0.073 |
| | hi2[maximal weight](limma+LDA) | 0.31 | 0.059 | 0.07 | 0.079 |
| | hi2[weigted majority vote](limma+LDA) | 0.31 | 0.059 | 0.06 | 0.078 |
| | hi2[noscores](limma+LDA) | 0.35 | 0.061 | 0.12 | 0.077 |
| | hi2(SVM) | 0.22 | 0.051 | 0.06 | 0.025 |
| | hi2(glmboost) | 0.29 | 0.057 | 0.12 | 0.067 |

**TABLE 4.2** *The values in this table are visualized in Figure 4.3. Four datasets were used to compare the performance of hi2 using different parameter settings. The performance is measured using the accuracy and Kendall's $\tau$. Shown are the mean an the standard error (se) across the folds from a 10-fold cross validation.*

| | | Accuracy | | Kendall's $\tau$ | |
|---|---|---|---|---|---|
| | | mean | se | mean | se |
| B-ALL | hi2(limma+LDA) | 0.6 | 0.109 | 0.48 | 0.077 |
| | si2(limma+LDA) | 0.28 | 0.089 | 0.18 | 0.091 |
| | limma+LDA | 0.54 | 0.108 | 0.49 | 0.048 |
| | mboost | 0.5 | 0.107 | 0.48 | 0.047 |
| | kkNN | 0.42 | 0.103 | 0.25 | 0.091 |
| | rpartOrdinal[twoing] | 0.52 | 0.108 | 0.59 | |
| | SVM | 0.48 | 0.106 | 0.24 | 0.085 |
| | prevalence | 0.24 | 0.084 | -0.1 | 0.118 |
| Breast Cancer | hi2(limma+LDA) | 0.52 | 0.074 | 0.4 | 0.04 |
| | si2(limma+LDA) | 0.31 | 0.065 | 0.08 | 0.074 |
| | limma+LDA | 0.42 | 0.072 | 0.27 | 0.057 |
| | mboost | 0.49 | 0.074 | 0.24 | 0.053 |
| | kkNN | 0.43 | 0.072 | 0.15 | 0.052 |
| | rpartOrdinal[twoing] | 0.51 | 0.074 | 0.34 | 0.06 |
| | SVM | 0.51 | 0.074 | 0.38 | 0.028 |
| | prevalence | 0.38 | 0.07 | 0.08 | 0.083 |
| Neuroblastoma | hi2(limma+LDA) | 0.19 | 0.036 | 0.21 | 0.115 |
| | si2(limma+LDA) | 0.29 | 0.06 | 0.11 | 0.091 |
| | limma+LDA | 0.3 | 0.042 | 0.01 | 0.103 |
| | mboost | 0.13 | 0.049 | 0.27 | 0.094 |
| | kkNN | 0.44 | 0.046 | 0.41 | 0.066 |
| | rpartOrdinal[twoing] | 0.24 | 0.035 | 0.23 | 0.074 |
| | SVM | 0.39 | 0.058 | 0.37 | 0.11 |
| | prevalence | 0.25 | 0.05 | 0.06 | 0.081 |
| Rectal Cancer | hi2(limma+LDA) | 0.31 | 0.059 | 0.1 | 0.076 |
| | si2(limma+LDA) | 0.08 | 0.032 | 0.1 | 0.07 |
| | limma+LDA | 0.32 | 0.06 | 0.07 | 0.075 |
| | mboost | 0.25 | 0.054 | 0.09 | 0.066 |
| | kkNN | 0.32 | 0.06 | 0.2 | 0.071 |
| | rpartOrdinal[twoing] | 0.29 | 0.058 | 0.02 | 0.081 |
| | SVM | 0.35 | 0.062 | 0.1 | 0.057 |
| | prevalence | 0.29 | 0.058 | -0.06 | 0.077 |

**TABLE 4.3** *The values in this table are visualized in Figure 4.4. Four datasets were used to compare the performance of hi2 to that of other classification methods. The performance is measured using the accuracy and Kendall's $\tau$. Shown are the mean and the standard error (se) across the folds from a 10-fold cross validation.*

# 5 Discussion

With only very few exceptions throughout all settings, both real data and simulated data, the null model performs worse than the classifiers that use the gene expression data. Thus, the features are informative with respect to the outcome and using that information does indeed improve the classification performance. There are, however, differences among the classification methods.

The established algorithms for ordinal classification perform surprisingly weak. While kkNN and rpartOrdinal perform very strong in some situations, their performance is surprisingly weak in other settings. For example, kkNN is only strong in two out of four correlation structure settings (see Figure 4.8). It seems that kkNN benefits from strong correlation, as in both, the unstructured correlation and the compound symmetry, all features are correlated at least to a certain extent, whereas in contrast, the other correlation structures, autoregressive and autoblocks, contain mainly 0 entries and focus on local correlation (Figure 3.5).

Similarly, as shown in Figure 4.4 kkNN is also only strong in two out of four real datasets. On the one hand, for the rectal cancer data kkNN is the only algorithm to uncover at least some signal. But on the other hand its performance is comparably weak on the other data. There is no obvious distinction between the two pairs of data, so this might be an issue of the correlation again.

rpartOrdinal shows good performance – comparable to hi2 – on the real data, but performs very weak in all simulation settings. Most surprising, however, was the overall weak performance of si2. Apparently, when only information from maximal two of the base learners is used, the binarization of the multi-class problem looses too much information. Especially striking is the inability of si2

to benefit from large effect sizes. The pooling of response levels for the base learners apparently shadows a lot of information.

hi2 recovers from that problem with its hierarchical structure and performs consistently well. With the exception of the neuroblastoma data hi2 always belongs to the group of the top performing algorithms. Thus, we consider hi2 to be both a strong classifier and a save choice, which is not likely to perform badly.

limma+LDA without the hi2 wrapper performs very good on its own in many situations. This classifier benefits the most from large effect sizes (see Figures 4.5 and 4.7). This exemplifies the importance of good feature selection in high-dimensional problems. With large effects, limma is able to select the correct features, thereby removing all the noise variables. It should be fairly easy for the classifier to classify correctly, when it is presented only with informative features.

mboost performs comparably to hi2 in most settings. As expected in the simulation with the perfectly linear trend, this regression based method outperforms all other methods (Figures 4.5 and 4.6). However, we doubt that this setting reflects real world data very well. In the plateau effects simulations mboost still performs comparably to hi2 in terms of Kendall's $\tau$, but shows a quite low accuracy (for example in Figure 4.7). This is again expected, as a linear regression will have difficulties to fit both, a trend effect in the 'middle' and plateaus at the 'ends'.
This does not as much affect Kendall's $\tau$ since the order of the levels still will be respected. The accuracy, though, does get decreased.

The SVM shows across all settings the most unstable behaviour. In the breast cancer and neuroblastoma data the performance of the SVM is very strong, in the rectal cancer it is comparable to most others, and in the B-ALL data the SVM performs quite weakly. Also, when used as a base learner in hi2, that unstable behaviour is transferred to hi2 (see Figure 4.3).
In none of the simulation settings the SVM performs strongly. And from the simulation varying the number of response levels as shown in Figure 4.10, it is apparent that the aggregation from all binary comparisons has difficulties with many groups.

Since all algorithms – except kkNN to some extent – perform weakly on the rectal cancer data, one can assume that there is little information in the gene expression that is predictive with respect to the tumor regression grade. As this data stems from a multi-center trial it is reasonable to assume that interfering covariates add too much noise to the data.

The relative strength of the non-ordinal SVM and the not necessarily ordinal kkNN on the neuroblastoma data might suggest that the grouping of the stages in that data is not as ordinal as expected.

We want to stress that last point. While the data used in this work have been chosen to contain 'really ordinal' response levels, many seemingly ordinal quantities might not be ordinal after all. Many ordinal measures are combinations of scores and while the individual subscores are still ordinal or even continuous, the combination does not necessarily have to be ordinal. It is, therefore, advisable to always benchmark any specifically ordinal classifier against a generic multi-class algorithm in case there are doubts about the ordinality of the response.

Finally, the preferred measure of classification performance should be carefully considered. The algorithm that results in the highest value for Kendall's $\tau$ might produce predictions of low accuracy. This is, for instance, the case for the regression based mboost in the plateau effects simulation settings. There are effects that cannot be captured with the accuracy (see Figure 2.3) and there are other effects (as the comparably low accuracy of mboost in the plateau simulation, Figure 4.7, e.g.) that are not reflected in Kendall's $\tau$. Depending on the application the trend might be more important and an accurate prediction is not too important. But based on these observations, we'd argue that both measures should be considered.

# 6 Conclusions

In this thesis we have considered ordinal and high-dimensional classification problems, where the number of features $p$ is much bigger than the number of available samples $N$ and the response is measured on an ordinal scale.

Such problems arise regularly when molecular biology is applied in clinical settings, as most molecular data (such as expression screens for tens of thousands of transcripts) are high dimensional in nature and many of the clinical variables such as gradings (toxicity grades, e.g.) or stages (tumor stages, e.g.) are measured on an ordinal scale.

Both properties, the high dimensionality and the ordinal response structure, require special attention: High-dimensional problems require either algorithms that can deal with that kind of data or some rigorous feature selection or regularization. Classification into ordered categories calls for special classification algorithms that lie between multinomial classification, which ignores the ordering information, and regression methods, which over-emphasize the ordering.

As the set of algorithms tailored for ordinal response and simultaneously capable to handle high-dimensional data is small and the performance of these algorithms appears to be surprisingly low, we have developed a novel classification scheme *hierarchical twoing (hi2)* which allows to use a binary classifier as base learner and turn it into an ordinal multi-class classifier.

Some thought should be given to the evaluation of classifiers on ordinal response data. We propose to evaluate the accuracy of the classifier and to simultaneously correlate the predictions from the classifier to the true values using Kendall's $\tau$ as a non-parametric correlation measure. There are situations where each of them is able to measure a difference in performance while the other one is not.

While the number of algorithms available for ordinal classification in high dimensions is limited, there are many options, if one considers regression based methods or non-ordinal multinomial algorithms, too. We present an introduction into the state-of-the-art algorithms for ordinal classification and a comparative study of some of the most important algorithms. That comparison involves both, real world data as well as simulated settings.

The results from that study show that compared to other algorithms hi2 performs consistently strong in all data, both simulated and real. Therefore, hi2 is a good and also safe choice, as its performance is strong not only in selected settings but consistently. Although it is outperformed by other algorithms in some settings these other algorithms perform poorly in the next setting, while hi2 can be expected to produce good results in most situations.

An open source implementation of hi2 is freely available as software package for the statistical computing framework `R`. The implementation sets a high value on being easy to use and easy to extend.

# Notation

| Notation | | Meaning |
|---|---|---|
| $N$ | sample $i = 1, \ldots, N$ | Number of Samples |
| $p$ | variable $j = 1, \ldots, p$ | Number of Features (Independent Variables) |
| $L$ | level $l = 1, \ldots, L$ | Number of Levels in a Categorical Response |
| $V$ | fold $v = 1, \ldots, V$ | Number of Folds in a Cross Validation |
| $k$ | | Number of Nearest Neighbours in a kNN Classification |
| $R$ | run $r = 1, \ldots, R$ | Number of Simulation Runs |
| $X$ | | General Symbol for the Independent Variables |
| $\mathbf{X}$ | | Realizations of the independent variable for $N$ samples (i.e. $N \times p$ matrix) |
| $\mathbf{x}_i$ | | Realizations of the independent variable for sample $i$ (i.e. $p$ vector) |
| $x_{ij}$ | | Realization of the independent variable $j$ for sample $i$ |
| continuous response | categorical response | |
| $Y$ | $G$ | General Symbol for the Response (Dependent Variable) |
| $\mathbf{y}$ | $\mathbf{g}$ | Realizations of the Dependent Variable (i.e. $N$ vector) |
| $y_i$ | $g_i$ | Realization of the Dependent Variable for sample $i$ |
| $\mathcal{T} \subseteq \{1, \ldots, N\}$ | | Training Set (Set of Indices) |
| $\mathcal{S} \subseteq \{1, \ldots, N\}$ | | Test Set (Set of Indices) |

# References


Agresti, Alan (2002). "Logit Models for Multinomial Responses". In: *Categorical Data Analysis*. John Wiley & Sons, Inc., pp. 267–313 (cit. on pp. 33, 49).

— (Apr. 2010). *Analysis of Ordinal Categorical Data*. John Wiley & Sons (cit. on pp. 24, 49).

Alin, Aylin (Aug. 2009). "Comparison of PLS algorithms when number of objects is much larger than number of variables". In: *Statistical Papers* 50.4, pp. 711–720. DOI: 10.1007/s00362-009-0251-7 (cit. on p. 31).

Amit, Yali and Donald Geman (Oct. 1997). "Shape Quantization and Recognition with Randomized Trees". In: *Neural Computation* 9.7, pp. 1545–1588. DOI: 10.1162/neco.1997.9.7.1545 (cit. on p. 43).

Ananth, C V and D G Kleinbaum (1997). "Regression models for ordinal responses: a review of methods and applications." In: *International Journal of Epidemiology* 26.6, pp. 1323–1333. DOI: 10.1093/ije/26.6.1323 (cit. on p. 35).

Andersen, Anders H et al. (Apr. 2012). "Partial least squares for discrimination in fMRI data". In: *Magnetic Resonance Imaging* 30.3, pp. 446–452. DOI: 10.1016/j.mri.2011.11.001 (cit. on p. 31).

Archer, K J and V R Mas (Dec. 2009). "Ordinal response prediction using bootstrap aggregation, with application to a high-throughput methylation data set." In: *Stat Med* 28.29, pp. 3597–3610. DOI: 10.1002/sim.3707 (cit. on pp. 24, 45, 49).

Archer, K J and A A A Williams (2012). "L 1 penalized continuation ratio models for ordinal response prediction using high-dimensional datasets". In: *Statistics in Medicine*. DOI: 10.1002/sim.4484 (cit. on pp. 24, 46, 49).

Archer, Kellie J (Apr. 19, 2010). "rpartOrdinal: An R Package for Deriving a Classification Tree for Predicting an Ordinal Response". In: *Journal of Statistical Software* 34.7, pp. 1–17 (cit. on pp. 24, 45, 49).

Archer, Kellie J., Jiayi Hou, et al. (Dec. 2014). "ordinalgmifs: An R Package for Ordinal Regression in High-dimensional Data Settings". In: *Cancer Informatics* 13, pp. 187–195. DOI: 10.4137/CIN.S20806 (cit. on p. 46).

Baccianella, S, A Esuli, and F Sebastiani (Nov. 2009). "Evaluation Measures for Ordinal Regression". In: *Ninth International Conference on Intelligent Systems Design and Applications, 2009. ISDA '09*, pp. 283–287. DOI: 10.1109/ISDA.2009.230 (cit. on pp. 20, 21, 23).

Bellman, Richard E (1961). *Adaptive control processes - A guided tour*. Princeton, New Jersey, U.S.A.: Princeton University Press, p. 255 (cit. on p. 8).

Bennett, Kristin P and Colin Campbell (Dec. 2000). "Support vector machines: hype or hallelujah?" In: *SIGKDD Explor. Newsl.* 2.2, 1–13. DOI: 10.1145/380995.380999 (cit. on pp. 8, 9, 24, 38, 49).

Berkson, Joseph (Sept. 1944). "Application of the Logistic Function to Bio-Assay". In: *Journal of the American Statistical Association* 39.227, pp. 357–365. DOI: 10.1080/01621459.1944.10500699 (cit. on p. 32).

Bleckmann, Annalen et al. (Jan. 2015). "Integrated miRNA and mRNA profiling of tumor-educated macrophages identifies prognostic subgroups in estrogen receptor-positive breast cancer". In: *Molecular Oncology* 9.1, pp. 155–166. DOI: 10.1016/j.molonc.2014.07.023 (cit. on p. 2).

Boser, Bernhard E, Isabelle M Guyon, and Vladimir N Vapnik (1992). "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, pp. 144–152 (cit. on p. 38).

Boulesteix, Anne-Laure and Korbinian Strimmer (Jan. 2007). "Partial least squares: a versatile tool for the analysis of high-dimensional genomic data". In: *Briefings in Bioinformatics* 8.1, pp. 32–44. DOI: 10.1093/bib/bbl016 (cit. on p. 31).

Braun, Mikio L, Joachim M Buhmann, and Klaus-Robert Müller (2008). "On relevant dimensions in kernel feature spaces". In: *The Journal of Machine Learning Research* 9, 1875–1908 (cit. on p. 9).

Breiman, Leo (1996). "Bagging predictors". In: *Machine Learning* 24 (2). 10.1007/BF00058655, pp. 123–140 (cit. on p. 43).

— (2001). "Random Forests". In: *Machine Learning*, pp. 5–32 (cit. on p. 43).

Breiman, Leo et al. (1984). *Classification and Regression Trees*. Statistics/Probability Series. Belmont, California, U.S.A.: Wadsworth Publishing Company (cit. on pp. 9, 24, 41, 45, 47, 49).

Brodeur, G M et al. (1993). "Revisions of the international criteria for neuroblastoma diagnosis, staging, and response to treatment". In: *Journal of Clinical Oncology* 11.8, pp. 1466–1477 (cit. on p. 52).

Brügmann, Anja and Boe S Sorensen (June 2011). "Identifying responders to trastuzumab therapy in breast cancer". In: *Future Oncology* 7.6, pp. 767–773. DOI: 10.2217/fon.11.44 (cit. on p. 2).

Bühlmann, Peter and Torsten Hothorn (Nov. 2007a). "Boosting Algorithms: Regularization, Prediction and Model Fitting". In: *Statistical Science* 22.4. Mathematical Reviews number (MathSciNet) MR2420454, pp. 477–505. DOI: 10.1214/07-STS242 (cit. on pp. 24, 49).

— (Nov. 2007b). "Boosting Algorithms: Regularization, Prediction and Model Fitting". In: *Statistical Science* 22.4, pp. 477–505. DOI: 10.1214/07-STS242 (cit. on p. 28).

Cardoso, Jaime S and Ricardo Sousa (Dec. 2011). "Measuring the Performance of Ordinal Classification". In: *International Journal of Pattern Recognition and Artificial Intelligence* 25.08, pp. 1173–1195. DOI: 10.1142/S0218001411009093 (cit. on p. 23).

Cerňak, Miloš (2012). "A Comparison of Decision Tree Classifiers for Automatic Diagnosis of Speech Recognition Errors". In: *Computing and Informatics* 29.3, pp. 489–501 (cit. on p. 44).

Chang, Chih-Chung and Chih-Jen Lin (May 2011). "LIBSVM: A Library for Support Vector Machines". In: *ACM Trans. Intell. Syst. Technol.* 2.3, 27:1–27:27. DOI: 10.1145/1961189.1961199 (cit. on p. 40).

Chiaretti, Sabina, Xiaochun Li, Robert Gentleman, Antonella Vitale, Marco Vignetti, et al. (2004). "Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival". In: *Blood* 103.7, pp. 2771–2778. DOI: 10.1182/blood-2003-09-3243 (cit. on p. 54).

Chiaretti, Sabina, Xiaochun Li, Robert Gentleman, Antonella Vitale, Kathy S Wang, et al. (Oct. 2005). "Gene Expression Profiles of B-lineage Adult Acute Lymphocytic Leukemia Reveal Genetic Patterns that Identify Lineage

Derivation and Distinct Mechanisms of Transformation". In: *Clinical Cancer Research* 11.20, pp. 7209–7219. DOI: 10.1158/1078-0432.CCR-04-2165 (cit. on p. 54).

Chou, Roger et al. (Oct. 2007). "Diagnosis and Treatment of Low Back Pain: A Joint Clinical Practice Guideline from the American College of Physicians and the American Pain Society". In: *Annals of Internal Medicine* 147.7, pp. 478–491. DOI: 10.7326/0003-4819-147-7-200710020-00006 (cit. on p. 1).

Chu, Wei and S Sathiya Keerthi (Feb. 13, 2007). "Support Vector Ordinal Regression". In: *Neural Computation* 19.3, pp. 792–815. DOI: 10.1162/neco.2007.19.3.792 (cit. on pp. 24, 41, 49).

Clopper, C J and E S Pearson (Dec. 1934). "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial". In: *Biometrika* 26.4, pp. 404–413. DOI: 10.2307/2331986 (cit. on p. 17).

Cohen, Jacob (Apr. 1960). "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20.1, pp. 37–46. DOI: 10.1177/001316446002000104 (cit. on p. 21).

— (1968). "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit." In: *Psychological Bulletin* 70.4, pp. 213–220 (cit. on p. 21).

Cortes, Corinna and Vladimir N Vapnik (Sept. 1995). "Support-Vector Networks". In: *Machine Learning* 20.3, pp. 273–297. DOI: 10.1023/A:1022627411411 (cit. on p. 39).

Cover, T M and P E Hart (1967). "Nearest neighbor pattern classification". In: *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, pp. 13–21 (cit. on pp. 8, 36, 49).

Cox, Christopher (Mar. 1988). "Multinomial regression models based on continuation ratios". In: *Statistics in Medicine* 7.3, pp. 435–441. DOI: 10.1002/sim.4780070309 (cit. on pp. 24, 49).

Cramer, J S (2003). *Logit Models from Economics and Other Fields*. Cambridge: Cambridge University Press (cit. on p. 32).

Crammer, Koby and Yoram Singer (2002). "On the algorithmic implementation of multiclass kernel-based vector machines". In: *The Journal of Machine Learning Research* 2, pp. 265–292 (cit. on p. 40).

Cruz-Ramirez, M et al. (2011). "A preliminary study of ordinal metrics to guide a multi-objective evolutionary algorithm". In: *2011 11th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 1176–1181. DOI: 10.1109/ISDA.2011.6121818 (cit. on p. 23).

Dal Moro, F et al. (2006). "A novel approach for accurate prediction of spontaneous passage of ureteral stones: Support vector machines". In: *Kidney International* 69.1, pp. 157–160. DOI: 10.1038/sj.ki.5000010 (cit. on p. 38).

Deng, Li (2014). *Deep Learning: Methods and Applications*. Vol. 7. 3-4, pp. 197–387. DOI: 10.1561/2000000039 (cit. on p. 38).

DeVita, Vincent T Jr. (2002). "A Perspective on the War on Cancer". In: *Cancer Journal* 8.5, pp. 352–356 (cit. on p. 1).

Dobbin, Kevin K and Richard M Simon (2011). "Optimally splitting cases for training and testing high dimensional classifiers". In: *BMC medical genomics* 4.1, p. 31 (cit. on p. 16).

Domeniconi, Carlotta, Jing Peng, and Dimitrios Gunopulos (2002). "Locally adaptive metric nearest-neighbor classification". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.9, 1281–1285 (cit. on p. 9).

Duan, Kai-Bo and S Sathiya Keerthi (Jan. 2005). "Which Is the Best Multiclass SVM Method? An Empirical Study". In: *Multiple Classifier Systems*. Ed. by Nikunj C Oza et al. Lecture Notes in Computer Science 3541. Springer Berlin Heidelberg, pp. 278–285 (cit. on p. 40).

Dworak, O, L Keilholz, and A Hoffmann (Mar. 1997). "Pathological features of rectal cancer after preoperative radiochemotherapy". In: *International Journal of Colorectal Disease* 12.1, pp. 19–23. DOI: 10.1007/s003840050072 (cit. on pp. 6, 7).

Edgar, Ron, Michael Domrachev, and Alex E Lash (Jan. 2002). "Gene Expression Omnibus: NCBI gene expression and hybridization array data repository". In: *Nucleic Acids Research* 30.1. PMID: 11752295, pp. 207–210. DOI: 10.1093/nar/30.1.207 (cit. on pp. 7, 53).

Edge, Stephen et al., eds. (Nov. 2011). *AJCC Cancer Staging Manual*. 7th Edition. Boston: Springer (cit. on p. 51).

Efron, B (Jan. 1979). "Bootstrap Methods: Another Look at the Jackknife". In: *The Annals of Statistics* 7.1, pp. 1–26. DOI: 10.1214/aos/1176344552 (cit. on p. 17).

Efron, B and R Tibshirani (Feb. 1986). "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy". In: *Statistical Science* 1.1. Mathematical Reviews number (MathSciNet) MR833275, pp. 54–75. DOI: 10.1214/ss/1177013815 (cit. on p. 18).

Efron, Bradley (1983). "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation". In: *Journal of the American Statistical Association* 78.382, pp. 316–331. DOI: 10.1080/01621459.1983.10477973 (cit. on p. 18).

Efron, Bradley and Robert Tibshirani (June 1997). "Improvements on Cross-Validation: The .632+ Bootstrap Method". In: *Journal of the American Statistical Association* 92.438, p. 548. DOI: 10.2307/2965703 (cit. on p. 18).

Elston, C W and I O Ellis (Nov. 1991). "Pathological prognostic factors in breast cancer. I. The value of histological grade in breast cancer: experience from a large study with long-term follow-up". In: *Histopathology* 19.5. PMID: 1757079, pp. 403–410 (cit. on p. 53).

Fernández-Delgado, Manuel et al. (2014). "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?" In: *Journal of Machine Learning Research* 15, pp. 3133–3181 (cit. on p. 44).

Fienberg, Stephen E (1980). *The Analysis of Cross-classified Categorical Data*. 2nd edition. Cambridge, MA: M.I.T. Press (cit. on p. 36).

Fisher, R A (Sept. 1936). "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.2, pp. 179–188. DOI: 10.1111/j.1469-1809.1936.tb02137.x (cit. on pp. 24, 28, 49).

Fix, E and J L Hodges (1951). *Discriminatory analysis, nonparametric discrimination: Consistency properties*. Tech. rep. 4. Project Number 21-49-004. Randolph Field, Texas: US Air Force School of Aviation Medicine (cit. on pp. 8, 36, 49).

Flicek, P et al. (Nov. 2012). "Ensembl 2013". In: *Nucleic Acids Research* 41.D1, pp. D48–D55. DOI: 10.1093/nar/gks1236 (cit. on p. 7).

Frank, Eibe and Mark Hall (2001). "A simple approach to ordinal classification". In: *Proc 12th Europ Conf on Machine Learning*. Springer, pp. 145–156 (cit. on pp. 25, 47–49, 63).

Friedman, Jerome H, Trevor Hastie, and Rob Tibshirani (Feb. 2, 2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1, pp. 1–22 (cit. on pp. 24, 49).

Galimberti, Giuliano, Gabriele Soffritti, and Matteo Di Maso (May 17, 2012). "Classification Trees for Ordinal Responses in R: The rpartScore Package". In: *Journal of Statistical Software* 47.10 (cit. on pp. 24, 45, 49).

Gentleman, Robert et al. (2005). *Bioinformatics and Computational Biology Solutions Using R and Bioconductor (Statistics for Biology and Health)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. (cit. on p. 52).

Ghadimi, Michael B (2014). *Home - Klinische Forschergruppe 179*. URL: `http://www.kfo179.de/` (visited on 02/05/2014) (cit. on p. 5).

Guyon, Isabelle and André Elisseeff (2003). "An introduction to variable and feature selection". In: *The Journal of Machine Learning Research* 3, 1157–1182 (cit. on p. 9).

Hall, J M et al. (Dec. 1990). "Linkage of early-onset familial breast cancer to chromosome 17q21". In: *Science (New York, N.Y.)* 250.4988. PMID: 2270482, pp. 1684–1689 (cit. on p. 2).

Hanash, Samir M, Sharon J Pitteri, and Vitor M Faca (Apr. 2008). "Mining the plasma proteome for cancer biomarkers". In: *Nature* 452.7187, pp. 571–579. DOI: `10.1038/nature06916` (cit. on p. 3).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer (cit. on pp. 8, 11, 17, 24, 26, 49).

Hawkins, Douglas M, Subhash C Basak, and Denise Mills (Mar. 2003). "Assessing Model Fit by Cross-Validation". In: *Journal of Chemical Information and Computer Sciences* 43.2, pp. 579–586. DOI: `10.1021/ci025626i` (cit. on p. 16).

Heath, David (1992). "A Geometric Framework for Machine Learning". PhD thesis. Baltimore, Maryland: Johns Hopkins University (cit. on p. 42).

Hechenbichler, Klaus and Klaus Schliep (2006). "Weighted k-nearest-neighbor techniques and ordinal classification". In: *Discussion Paper 399*. Vol. 399. SFB 386 (cit. on pp. 24, 37, 49).

Ho, Tin Kam (1995). "Random Decision Forests". In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*. ICDAR '95. Washington, DC, USA: IEEE Computer Society, pp. 278–282 (cit. on p. 43).

Hofner, Benjamin et al. (Feb. 2014). "Model-based boosting in R: a hands-on tutorial using the R package mboost". In: *Computational Statistics* 29.1-2, pp. 3–35. DOI: 10.1007/s00180-012-0382-5 (cit. on p. 28).

Jiang, Wenyu, Sudhir Varma, and Richard Simon (Jan. 2008). "Calculating Confidence Intervals for Prediction Error in Microarray Classification Using Resampling". In: *Statistical Applications in Genetics and Molecular Biology* 7.1. DOI: 10.2202/1544-6115.1322 (cit. on p. 17).

Kendall, M G (June 1938). "A New Measure of Rank Correlation". In: *Biometrika* 30.1/2, p. 81. DOI: 10.2307/2332226 (cit. on p. 21).

Kendall, Maurice G (1975). *Rank correlation methods*. London and High Wycombe: Charles Griffin, London (cit. on p. 21).

Keprta, Stanislav (1996). "Non-binary classification trees". In: *Statistics and Computing* 6.3, pp. 231–243 (cit. on p. 43).

Kohavi, Ron (1995). "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*. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1137–1143 (cit. on p. 17).

Koshy, Thomas (2009). *Catalan numbers with applications*. Oxford; New York: Oxford University Press (cit. on p. 67).

Kozomara, A and S Griffiths-Jones (Nov. 2013). "miRBase: annotating high confidence microRNAs using deep sequencing data". In: *Nucleic Acids Research* 42.D1, pp. D68–D73. DOI: 10.1093/nar/gkt1181 (cit. on p. 8).

Kpotufe, Samory (2011). "k-NN regression adapts to local intrinsic dimension". In: *arXiv preprint arXiv:1110.4300* (cit. on p. 9).

Kuhn, Max and Kjell Johnson (2013). *Applied Predictive Modeling*. New York: Springer. DOI: 10.1007/978-1-4614-6849-3 (cit. on pp. 9, 31, 44).

Lee, Yoonkyung, Yi Lin, and Grace Wahba (2001). "Multicategory support vector machines". In: *Proceedings of the 33rd Symposium on the Interface*. Citeseer (cit. on p. 40).

Leha, Andreas, Klaus Jung, and Tim Beißbarth (2013). "Utilization of ordinal response structures in classification with high-dimensional expression data". In: *German Conference on Bioinformatics 2013*. Ed. by Tim Beißbarth et al. Vol. 34. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany:

Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 90–100. DOI: 10.4230/OASIcs.GCB.2013.90 (cit. on pp. 24, 25, 49).

Lin, Hsuan-Tien and Ling Li (Feb. 2012). "Reduction from Cost-Sensitive Ordinal Ranking to Weighted Binary Classification". In: *Neural Computation* 24.5, pp. 1329–1367. DOI: 10.1162/NECO_a_00265 (cit. on p. 23).

Lister, Ryan et al. (Oct. 2009). "Human DNA methylomes at base resolution show widespread epigenomic differences". In: *Nature* 462.7271, pp. 315–322. DOI: 10.1038/nature08514 (cit. on p. 8).

Liu, Ivy and Alan Agresti (June 2005). "The analysis of ordered categorical data: An overview and a survey of recent developments". In: *Test* 14.1, pp. 1–73. DOI: 10.1007/BF02595397 (cit. on p. 35).

Loh, Wei-Yin and Nunta Vanichsetakul (Sept. 1988). "Tree-Structured Classification Via Generalized Discriminant Analysis". In: *Journal of the American Statistical Association* 83.403, p. 715. DOI: 10.2307/2289295 (cit. on p. 42).

Mannil, Manoj et al. (Aug. 2014). "Selected items from the Charcot-Marie-Tooth (CMT) Neuropathy Score and secondary clinical outcome measures serve as sensitive clinical markers of disease severity in CMT1A patients". In: *Neuromuscular Disorders*. DOI: 10.1016/j.nmd.2014.06.431 (cit. on p. 68).

Martin, Todd M et al. (Oct. 2012). "Does Rational Selection of Training and Test Sets Improve the Outcome of QSAR Modeling?" In: *Journal of Chemical Information and Modeling* 52.10, pp. 2570–2578. DOI: 10.1021/ci300338w (cit. on p. 16).

McCullagh, P and John A Nelder (Aug. 1989). *Generalized Linear Models, Second Edition*. 2 edition. Boca Raton: Chapman and Hall/CRC (cit. on p. 36).

McCullagh, Peter (1980). "Regression models for ordinal data". In: *Journal of the Royal Statistical Society, Series B* 42.2, pp. 109–142 (cit. on p. 35).

McFadden, Daniel (1973). "Frontiers in Econometrics". In: ed. by Zarembka P. New York: Academic Press. Chap. Conditional Logit Analysis of Qualitative Choice Behavior, pp. 105–142 (cit. on pp. 33, 49).

Mehta, Sunali et al. (Mar. 2010). "Predictive and prognostic molecular markers for cancer medicine". In: *Therapeutic Advances in Medical Oncology* 2.2. PMID: 21789130 PMCID: PMC3126011, pp. 125–148. DOI: 10.1177/1758834009360519 (cit. on p. 2).

Mercer, J (Jan. 1909). "Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 209.441-458, pp. 415–446. DOI: 10.1098/rsta.1909.0016 (cit. on p. 40).

Meyer, David et al. (2014). *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package version 1.6-4 (cit. on p. 76).

Mihaylova, Iglika et al. (Aug. 2011). "Degree of tumor regression after preoperative chemo-radiotherapy in locally advanced rectal cancer—Preliminary results". In: *Reports of Practical Oncology and Radiotherapy* 16.6, pp. 237–242. DOI: 10.1016/j.rpor.2011.06.008 (cit. on p. 5).

Molinaro, Annette M, Richard Simon, and Ruth M Pfeiffer (Aug. 2005). "Prediction error estimation: a comparison of resampling methods". In: *Bioinformatics* 21.15. PMID: 15905277, pp. 3301–3307. DOI: 10.1093/bioinformatics/bti499 (cit. on p. 16).

Murphy, Patrick M and Michael J Pazzani (1994). "Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction". In: *arXiv preprint cs/9403101* (cit. on p. 43).

Nana-Sinkam, S P and C M Croce (Jan. 2013). "Clinical Applications for microRNAs in Cancer". In: *Clinical Pharmacology & Therapeutics* 93.1, pp. 98–104. DOI: 10.1038/clpt.2012.192 (cit. on p. 3).

Natarajan, Sundar et al. (2007). "Agreement Between Two Ratings with Different Ordinal Scales". In: *Advances in Statistical Methods for the Health Sciences*. Ed. by N Balakrishnan et al. Statistics for Industry and Technology. Birkhäuser Boston, pp. 139–148. DOI: 10.1007/978-0-8176-4542-7_9 (cit. on pp. 20, 21).

Naumov, G E (1991). "NP-completeness of problems of construction of optimal decision trees". In: *Soviet Physics: Doklady* 36.4, pp. 270–271 (cit. on p. 42).

Noether, G E (1981). "Why Kendall Tau?" In: *Teaching Statistics* 3.2, pp. 41–43. DOI: 10.1111/j.1467-9639.1981.tb00422.x (cit. on p. 21).

Nørregaard Jensen, Ole (Feb. 2004). "Modification-specific proteomics: characterization of post-translational modifications by mass spectrometry". In: *Current Opinion in Chemical Biology* 8.1, pp. 33–41. DOI: 10.1016/j.cbpa.2003.12.009 (cit. on p. 8).

Nowsheen, Somaira et al. (Dec. 2012). "Molecular markers for cancer prognosis and treatment: have we struck gold?" In: *Cancer letters* 327.1-2. PMID: 22120674, pp. 142–152. DOI: 10.1016/j.canlet.2011.11.022 (cit. on pp. 2, 3).

Oberthuer, André et al. (2006). "Customized Oligonucleotide Microarray Gene Expression-Based Classification of Neuroblastoma Patients Outperforms Current Clinical Risk Stratification". In: *Journal of Clinical Oncology* 24.31, pp. 5070–5078. DOI: 10.1200/JCO.2006.06.1879 (cit. on p. 52).

Olshansky, S Jay et al. (2005). "A Potential Decline in Life Expectancy in the United States in the 21st Century". In: *New England Journal of Medicine* 352.11. PMID: 15784668, pp. 1138–1145. DOI: 10.1056/NEJMsr043743 (cit. on p. 1).

Ortega, J M and W C Rheinboldt (Jan. 2000). *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics (cit. on pp. 33, 34).

Paik, Soonmyung, Steven Shak, et al. (2004). "A Multigene Assay to Predict Recurrence of Tamoxifen-Treated, Node-Negative Breast Cancer". In: *New England Journal of Medicine* 351.27. PMID: 15591335, pp. 2817–2826. DOI: 10.1056/NEJMoa041588 (cit. on p. 2).

Paik, Soonmyung, Gong Tang, et al. (Aug. 2006). "Gene Expression and Benefit of Chemotherapy in Women With Node-Negative, Estrogen Receptor–Positive Breast Cancer". In: *Journal of Clinical Oncology* 24.23. PMID: 16720680, pp. 3726–3734. DOI: 10.1200/JCO.2005.04.7985 (cit. on p. 2).

Pérez-Enciso, Miguel and Michel Tenenhaus (May 2003). "Prediction of clinical outcome with microarray data: a partial least squares discriminant analysis (PLS-DA) approach". In: *Human Genetics* 112.5-6, pp. 581–592. DOI: 10.1007/s00439-003-0921-9 (cit. on p. 31).

Piccarreta, Raffaella (Jan. 2004). "Ordinal Classification Trees Based on Impurity Measures". In: *Advances in Multivariate Data Analysis*. Ed. by Prof Dr Hans-Hermann Bock, Prof Marcello Chiodi, and Prof Antonino Mineo. Studies in Classification, Data Analysis, and Knowledge Organization. Springer Berlin Heidelberg, pp. 39–51 (cit. on p. 46).

— (2008). "Classification trees for ordinal variables". In: *Computational Statistics* 23 (3). 10.1007/s00180-007-0077-5, pp. 407–427 (cit. on p. 46).

Press, William H et al. (Sept. 2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3 edition. Cambridge, UK ; New York: Cambridge University Press (cit. on p. 34).

Quinlan, J R (Mar. 1986). "Induction of decision trees". In: *Machine Learning* 1.1, pp. 81–106. DOI: 10.1007/BF00116251 (cit. on pp. 9, 24, 49).

Quinlan, J Ross (1993). *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (cit. on pp. 9, 44).

Quinn, Jennifer E et al. (Dec. 2007). "BRCA1 mRNA Expression Levels Predict for Overall Survival in Ovarian Cancer after Chemotherapy". In: *Clinical Cancer Research* 13.24. PMID: 18094425, pp. 7413–7420. DOI: 10.1158/1078-0432.CCR-07-1083 (cit. on p. 2).

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria (cit. on pp. 4, 72).

Rännar, Stefan et al. (Mar. 1994). "A PLS kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm". In: *Journal of Chemometrics* 8.2, pp. 111–125. DOI: 10.1002/cem.1180080204 (cit. on p. 31).

Rödel, Claus et al. (Dec. 2005). "Prognostic Significance of Tumor Regression After Preoperative Chemoradiotherapy for Rectal Cancer". In: *Journal of Clinical Oncology* 23.34. PMID: 16246976, pp. 8688–8696. DOI: 10.1200/JCO.2005.02.1329 (cit. on p. 7).

Rokach, Lior and Oded Maimon (2008). *Data Mining with Decision Trees: Theory and Applications*. River Edge, NJ, USA: World Scientific Publishing Co., Inc. (cit. on p. 41).

Rubbia-Brandt, L et al. (Feb. 2007). "Importance of histological tumor response assessment in predicting the outcome in patients with colorectal liver metastases treated with neo-adjuvant chemotherapy followed by liver surgery". In: *Annals of Oncology* 18.2. PMID: 17060484, pp. 299–304. DOI: 10.1093/annonc/mdl386 (cit. on p. 7).

Ruczinski, Ingo, Charles Kooperberg, and Michael LeBlanc (Sept. 2003). "Logic Regression". In: *Journal of Computational and Graphical Statistics* 12.3, pp. 475–511. DOI: 10.1198/1061860032238 (cit. on p. 47).

Saeys, Yvan, Iñaki Inza, and Pedro Larrañaga (Oct. 2007). "A review of feature selection techniques in bioinformatics". In: *Bioinformatics* 23.19. PMID:

17720704, pp. 2507–2517. DOI: 10.1093/bioinformatics/btm344 (cit. on p. 9).

Salazar, Ramon et al. (Nov. 2010). "Gene Expression Signature to Improve Prognosis Prediction of Stage II and III Colorectal Cancer". In: *Journal of Clinical Oncology*. PMID: 21098318, JCO.2010.30.1077. DOI: 10.1200/JCO.2010.30.1077 (cit. on p. 2).

Sánchez-Monedero, J et al. (2013). "Exploitation of pairwise class distances for ordinal classification". In: *Neural computation* 25.9, 2450–2485 (cit. on pp. 21, 24, 25, 46, 49).

Sauer, Rolf et al. (Jan. 2012). "Preoperative Versus Postoperative Chemoradiotherapy for Locally Advanced Rectal Cancer: Results of the German CAO/ARO/AIO-94 Randomized Phase III Trial After a Median Follow-Up of 11 Years". In: *Journal of Clinical Oncology* 30.16, pp. 1926–1933. DOI: 10.1200/JCO.2011.40.1836 (cit. on p. 5).

Schmidhuber, Juergen (Apr. 2014). "Deep Learning in Neural Networks: An Overview". In: *arXiv:1404.7828 [cs]*. arXiv: 1404.7828 (cit. on p. 38).

Serrano-Cinca, Carlos and Begoña Gutiérrez-Nieto (Feb. 2013). "Partial Least Square Discriminant Analysis for bankruptcy prediction". In: *Decision Support Systems* 54.3, pp. 1245–1255. DOI: 10.1016/j.dss.2012.11.015 (cit. on p. 31).

Shashua, Amnon and Anat Levin (2002). "Ranking with large margin principle: Two approaches". In: *Proceedings of Advances in Neural Information Processing Systems* (cit. on p. 40).

Shy, M E et al. (Apr. 2005). "Reliability and validity of the CMT neuropathy score as a measure of disability." In: *Neurology* 64.7, pp. 1209–1214. DOI: 10.1212/01.WNL.0000156517.00615.A3 (cit. on p. 68).

Smith, Caitlin (July 2013). "Cancer biology: Cancer shows strength through diversity". In: *Nature* 499.7459, pp. 505–508. DOI: 10.1038/499505a (cit. on p. 2).

Smola, Alex J and Bernhard Schölkopf (Aug. 2004). "A Tutorial on Support Vector Regression". In: *Statistics and Computing* 14.3, pp. 199–222. DOI: 10.1023/B:STCO.0000035301.49549.88 (cit. on p. 40).

Smyth, Gordon K (2004). "Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments". In: *Statistical*

*Applications in Genetics and Molecular Biology* 3 (1). DOI: 10.2202/1544-6115.1027 (cit. on p. 71).

Smyth, Gordon K (2005). "limma: Linear Models for Microarray Data". In: *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Ed. by Robert Gentleman et al. Statistics for Biology and Health. New York: Springer-Verlag. Chap. 23, pp. 397–420. DOI: 10.1007/0-387-29362-0_23 (cit. on p. 70).

Storey, John D and Robert Tibshirani (May 2003). "Statistical significance for genomewide studies". In: *Proceedings of the National Academy of Sciences* 100.16, pp. 9440–9445. DOI: 10.1073/pnas.1530509100 (cit. on p. 27).

Sun, Bing-Yu et al. (June 2010). "Kernel Discriminant Learning for Ordinal Regression". In: *IEEE Transactions on Knowledge and Data Engineering* 22.6, pp. 906–910. DOI: 10.1109/TKDE.2009.170 (cit. on pp. 35, 49).

't Veer, Laura J van et al. (Jan. 2002). "Gene expression profiling predicts clinical outcome of breast cancer". In: *Nature* 415.6871, pp. 530–536. DOI: 10.1038/415530a (cit. on p. 2).

*The On-Line Encyclopedia of Integer Sequences* (2014). published electronically at http://oeis.org (cit. on p. 68).

Therneau, Terry, Beth Atkinson, and Brian Ripley (2014). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-8 (cit. on p. 43).

Tonello, L, F Vescini, and R Caudarella (2007). "Support vector machines versus artificial neural network: Who is the winner?" In: *Kidney International* 71.1, pp. 84–84. DOI: 10.1038/sj.ki.5001907 (cit. on p. 38).

Tsoi, Ah Chung and R A Pearson (1991). "Comparison of Three Classification Techniques: CART, C4. 5 and Multi-Layer Perceptrons". In: *Advances in neural information processing systems*, pp. 963–969 (cit. on p. 44).

Tu, P-L. and J-Y. Chung (Nov. 1992). "A new decision-tree classification algorithm for machine learning". In: *, Fourth International Conference on Tools with Artificial Intelligence, 1992. TAI '92, Proceedings*, pp. 370–377. DOI: 10.1109/TAI.1992.246431 (cit. on p. 43).

Tutz, Gerhard (May 1991). "Sequential models in categorical regression". In: *Computational Statistics & Data Analysis* 11.3, pp. 275–295. DOI: 10.1016/0167-9473(91)90086-H (cit. on p. 36).

Vanwinckelen, Gitte and Hendrik Blockeel (2012). "On estimating model accuracy with repeated cross-validation". In: *BeneLearn 2012: Proceedings*

*of the 21st Belgian-Dutch Conference on Machine Learning*, 39–44 (cit. on p. 17).

Vapnik, Vladimir N (1995). *The Nature of Statistical Learning Theory*. N.Y.: Springer (cit. on pp. 38, 49).

Walker, Strother H and David B Duncan (June 1967). "Estimation of the probability of an event as a function of several independent variables". In: *Biometrika* 54.1-2, pp. 167–179. DOI: 10.1093/biomet/54.1-2.167 (cit. on p. 35).

Welch, B L (July 1939). "Note on Discriminant Functions". In: *Biometrika* 31.1/2, p. 218. DOI: 10.2307/2334985 (cit. on p. 28).

Wickham, Hadley (Sept. 2014). *Advanced R*. 1 edition. The R Series. available online at http://adv-r.had.co.nz/. Boca Raton, FL: Chapman and Hall/CRC, p. 476 (cit. on p. 73).

Wickham, Hadley and Winston Chang (2014). *devtools: Tools to make developing R code easier*. R package version 1.6.0.9000 (cit. on p. 73).

Willand, Ilka, ed. (Nov. 2014). *Statistisches Jahrbuch 2014*. Wiesbaden: Statistisches Bundesamt (cit. on p. 1).

Wold, Henry (1966). "Multivariate Analyses". In: ed. by Krishnaiah P. New York: Academic Press. Chap. Estimation of Principal Components and Related Models by Iterative Least squares, pp. 391–420 (cit. on p. 31).

Wold, S, H Martens, and H Wold (Jan. 1983). "The multivariate calibration problem in chemistry solved by the PLS method". In: *Matrix Pencils*. Ed. by Bo Kågström and Axel Ruhe. Lecture Notes in Mathematics 973. Springer Berlin Heidelberg, pp. 286–293 (cit. on p. 31).

Wolf, Bethany J, Elizabeth H Slate, and Elizabeth G Hill (Feb. 2015). "Ordinal Logic Regression: A classifier for discovering combinations of binary markers for ordinal outcomes". In: *Computational Statistics & Data Analysis* 82, pp. 152–163. DOI: 10.1016/j.csda.2014.08.013 (cit. on p. 47).

Yee, Thomas W (2010). "The VGAM package for categorical data analysis". In: *Journal of Statistical Software* 32.10, 1–34 (cit. on p. 36).

Zadran, Sohila, F Remacle, and R D Levine (Nov. 2013). "miRNA and mRNA cancer signatures determined by analysis of expression levels in large cohorts of patients". In: *Proceedings of the National Academy of Sciences* 110.47. PMID: 24101511, pp. 19160–19165. DOI: 10.1073/pnas.1316991110 (cit. on p. 3).

Ziech, Dominique et al. (Nov. 2010). "The role of epigenetics in environmental and occupational carcinogenesis". In: *Chemico-Biological Interactions* 188.2, pp. 340–349. DOI: 10.1016/j.cbi.2010.06.012 (cit. on p. 3).