KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Towards Declarative Statistical Inference

Gitte Vanwinckelen

Supervisor:
Prof. dr. ir. Hendrik Blockeel

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

September 2017

Towards Declarative Statistical Inference

Gitte VANWINCKELEN

Examination committee:
Prof. dr. ir. Dirk Vandermeulen, chair
Prof. dr. ir. Hendrik Blockeel, supervisor
Prof. dr. Jesse Davis
Prof. dr. Sien Moens
Prof. dr. Peter Flach
(University of Bristol)
dr. ir. Joaquin Vanschoren
(Eindhoven University of Technology)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

September 2017

Acknowledgements

At last, as I am writing these acknowledgements, the journey that is my doctoral project has come to an end. The journey has been challenging, sometimes even frustrating, but nevertheless exciting and rewarding in the end. The adventure started during my Master in Artificial Intelligence, when I met my inspiring master thesis supervisors, Kurt Driessens, Martijn van Otterlo, and Sofie Pollin. This master's thesis ignited a creative and inquisitive spark, and before I knew it, the next academic year I was writing up my first attempt at research, and submitting it to the DySPAN conference. At the same time, this also fired the starting gun for my PhD under the supervision of professor Hendrik Blockeel.

It is in that respect that I would first and foremost like to thank Hendrik for guiding me through this journey. Looking back, my skills as a researcher have matured quite a bit since the beginning. For a large part that is thanks to Hendrik, who taught me how to be critical, and how to build up consistent, logical and precise theories. Being a great writer himself, Hendrik was always rather critical of my texts, but luckily for me, at the same time he was also very patient. Consequently I learned how to keep my nose to the grindstone and keep on improving, be it incrementally and with small steps. To summarize, Hendrik taught me how to do science and how to write papers.

I would also like to thank professors Jesse Davis, Sien Moens, Joaquin Vanschoren and Peter Flach for serving on my examination committee, and professor Dirk Vandermeulen for chairing my preliminary and public PhD defenses. Additionally, I am grateful to the sponsor of my work: FWO-Vlaanderen (project G.0682.11 "Declaratieve experimentering voor automatisch leren").

The DTAI department was a very pleasant place to work thanks to the many colleagues and friends that I met there. Almost as soon as I started, I was introduced by the 'Greek' crowd to the Mediterranean time of lunch: 13h30. A big thanks to all my lunchmates throughout the years: Dimitar, Nick, Theofrastos, Jan-Tobias, Christos, Koosha, Tom, Tomas, Zubair, Jerome, and

Lilian, Raoul, Pieter, Mathy, Rula, Jesper, Marco, and Milica. The conversations we had were always entertaining and enriching. Moreover, with many of you I also had some equally interesting coffee breaks, and the occasional chess or table tennis game. The cultural diversity of the department is something that I appreciated a lot, and opened up my perspective on the world. A special thanks goes out to Dimitar, who made great iced coffee, and who was always available for a chat that cheered up my day.

I would also like to thank Nima, Wannes, Jessa and Ondrej, for making the office on the third floor almost like a second home for me during the 4.5 years spent there. An extra thanks goes out here to Wannes for helping me on my way in the beginning (and in fact also later on) with his broad technical knowledge.

Thanks to my (proof)readers for supporting me during the last mile by providing me with their detailed comments: My promoter and examination committee, Jessa, Kurt, Wannes, Ondrej, and Bert.

My PhD journey was a long one. In fact, even a bit longer than the average one. In that respect, I would like to thank my team head at KBC, Raf Geens, for being supportive, and giving me the flexibility for finishing up my text while working on his team. Additionally, I would also like to thank my other colleagues there, for creating a pleasant working environment, in a period that was rather challenging for me.

Finally, I would like to thank my family for their emotional and financial support that allowed me to attend higher education, and in the end obtain this PhD. And last but not least, my loving partner Bert: Spending my weekends with him allowed me to empty my mind and restart every week with fresh motivation.

Gitte Vanwinckelen
Heverlee, December 2017

Abstract

Wide-ranging digitalization has made it possible to capture increasingly larger amounts of data. In order to transform this raw data into meaningful insights, data analytics and statistical inference techniques are essential. However, while it is expected that a researcher is an expert in their own field, it is not self-evident that they are also proficient in statistics. In fact, it is known that statistical inference is a labor-intensive and error-prone task. This dissertation aims to understand current statistical inference practices for the experimental evaluation of machine learning algorithms, and proposes improvements where possible. It takes a small step forward towards the goal of automating the data analysis component of empirical research, making the process more robust in terms of correct execution and interpretation of the results.

Our first contribution is a synthesis of existing knowledge about error estimation of supervised learning algorithms with cross-validation. We highlight the distinction between model and learner error, and investigate the effect of repeating cross-validation on the quality of the error estimate.

Next, we focus on the evaluation of multi-instance learning algorithms. Here, instances are not labeled individually, but instead are grouped together in bags and only the bag label is known. Our second contribution is an investigation of the extent to which conclusions about bag-level performance can be generalized to the instance-level. Our third contribution is a meta-learning experiment in which we predict the most suitable multi-instance learner for a given problem.

The intricate nature of statistical inference begs the question whether this aspect of research cannot be automated. One requirement for this is the availability of a model representing all relevant characteristics of the population under study. Bayesian networks are a candidate for this, as they concisely describe the joint probability distribution of a set of random variables, and come with a plethora of efficient inference methods. Our last contribution is a theoretical proposal of a greedy hill-climbing structure learning algorithm for Bayesian networks.

Beknopte samenvatting

Doorgedreven digitalisatie resulteert in steeds grotere hoeveelheden beschikbare data. Om uit deze ruwe data nuttige inzichten te destilleren, is een goede kennis van data-analyse en statistische inferentie methodes essentieel. Terwijl we kunnen verwachten dat onderzoekers experts zijn in hun eigen vakgebied, is het niet vanzelfsprekend dat zij ook expert zijn in de statistiek. Bovendien is data-analyse een arbeidsintensieve en foutgevoelige taak. Dit proefschrift heeft als doel statistische inferentie methodes voor de experimentele evaluatie van machinale leeralgoritmes te begrijpen, en verbeteringen voor te stellen waar mogelijk. Het is een kleine stap vooruit in de richting het automatiseren van het data-analyse gedeelte van empirisch onderzoek, waardoor het proces robuuster wordt in termen van correcte uitvoering en interpretatie van de resultaten.

De eerste bijdrage is een synthese van de bestaande kennis over performantie schatting van gecontroleerde leeralgoritmes met cross-validatie. We onderscheiden de performantie van een model en van een leeralgoritme, en onderzoeken het effect van herhaalde cross-validatie op de kwaliteit van de schatting.

Vervolgens focussen we op de evaluatie van *multi-instance* leeralgoritmes. Deze algoritmes leren een model van voorbeelden die gegroepeerd zijn, en enkel het groepslabel bekend is. De tweede bijdrage is een onderzoek naar de relatie tussen de performantie van deze algoritmes op groepsniveau en op het niveau van individuele voorbeelden. De derde bijdrage is een meta-leren experiment waarin we het meest performante algoritme voor een bepaald probleem voorspellen.

Het complexe karakter van statistische inferentie roept de vraag op of dit aspect van wetenschappelijk onderzoek geautomatiseerd kan worden. Een vereiste is de beschikbaarheid van een statistisch model van de onderzochte populatie. Een Bayesiaans netwerk is geschikt omdat het bondig de gezamenlijke kansverdeling van een set stochastische variabelen beschrijft, en er efficiente inferentie methodes beschikbaar zijn. Onze laatste bijdrage is een theoretisch voorstel van een *greedy hill-climbing* algoritme voor het leren van de structuur van dit netwerk.

Contents

Chapter 1

Outline

1.1 Context

1.1.1 Machine learning

Recently, the verb 'learning' acquired a new meaning: It is no longer exclusively associated with humans and animals; but computers, or machines, are added to the list of possible subjects. It probably comes as no surprise to the reader that *machine learning* is the topic of this dissertation. This subfield of artificial intelligence is concerned with the study of algorithms that have as a goal learning patterns from data. The challenge is that these patterns should not only apply to the data from which they were extracted, but they should also generalize to unseen data.

Wide-ranging digitalization has resulted in an abundance of data in almost every industry and science, making machine learning a trending topic. Building a statistical model, however, is a challenging and complex task. The data almost always has to undergo several cleaning and preprocessing steps before being useful. Also the selection of the algorithm requires careful thought: In order to obtain a high-quality model, the inductive bias of the algorithm has to match the characteristics of the data at hand. In-depth knowledge of both machine learning and the problem setting are therefore required. The job is not finished after the model is constructed: The results still have to be analyzed and interpreted. This is the focal point of this dissertation, namely performance evaluation of statistical models.

1.1.2 Statistical inference

Since the goal in machine learning is to develop models that generalize to unseen data, the available data is considered just a sample from the population of interest. If we would repeat the experiment, we would almost certainly draw another sample from the population, and obtain slightly different results. The goal is therefore to quantify the uncertainty about the performance of the model on the population. This setup is reminiscent of frequentist philosophy. Confidence intervals or hypothesis tests are indeed very often the method of choice to analyze and present performance results. Estimation of the sample variance of the performance estimate lies at the core of this approach.

To obtain an unbiased estimate of the performance of the model, and consequently avoid overfitting, it is good practice to train and test the model on two independent datasets. This poses a dilemma: On one hand, the more training data is available, the more accurate the model will be. On the other hand, the more test data is available, the more accurate the performance estimation will be. The most widely accepted solution to this problem is the use of resampling estimators. Several variants exist, such as for instance: cross-validation, bootstrapping, or repeated hold-out. Currently, k-fold cross-validation seems to be the method of choice in machine learning research, where k can range from two to the number of instances in the dataset. Sometimes the cross-validation procedure is also repeated on random shuffles of a dataset, after which the results are averaged.

A resampling method bears a similarity with learning algorithms: Namely, its goal is to predict which model or learner will have best predictive performance on a given task. Consequently, resampling estimators can also be considered to have an inductive bias. It is perhaps because there is no clear description of the set of assumptions that this bias consists of, that the selection of the most appropriate resampling method for a given problem, and the tuning of its parameters, is still a debated topic today.

One important assumption for instance is about the scope of the problem: Do resampling estimators estimate the performance of the model that can be constructed on the available dataset, or do they estimate the performance of the underlying learning algorithm, on the population from which the available data is just a random sample? The interpretation of the results obtained by resampling, and the most appropriate statistical inference method for analysis of these results, depends entirely on the answer to this question. Investigation of this issue is one of the topics of this dissertation.

1.1.3 Performance measures

Another aspect of model performance evaluation is the choice of the performance measure. A performance measure is defined as the expected loss of a model $\mathcal{M}$ over the entire population P: $E_{x\sim P}[l(\mathcal{M}(x))]$, with the loss function l measuring the distance between a prediction and the true label of an instance. For the choice of the loss function there are many options. The function most often used for binary classification for instance is 0/1 loss, which equals zero if the label and the prediction are identical, and one otherwise. The performance measure associated with 0/1 loss is accuracy (acc). Sometimes also $1-acc$, or error, is used. Other popular performance measures for classification are 'Area Under the ROC Curve' (AUC-ROC) and the F1-score. For regression, mean squared error (MSE) is common due to its convenient mathematical properties. MSE is sometimes also used for classification algorithms that produce continuous class probabilities.

A model can score excellently on one performance measure, yet poorly on another. One should therefore think carefully about what aspects of the learning task are important. A typical example of this can be found in text classification, where the task is to assign categories to text fragments. Often the number of categories is large, whereas the number of texts that belong to a certain category is small. Almost never assigning a certain category to the text fragments will then result in an accuracy close to 100% for that category. Obviously, this does not give an accurate description of the performance of the model. This is the reason why in the field of natural language processing, the prevalent performance measures are precision, recall and the F1 score.

In this dissertation, we study the choice of performance measure in the context of multi-instance learning. As in traditional supervised learning, the goal here is to learn a model from labeled instances that can predict the label of future instances. Instances are however not labeled individually, but instead they are grouped together in bags and only the bag label is known. Besides the choice between the above discussed measures, here one also has the choice between computing the performance with respect to classifying bags, or with respect to classifying individual instances. It seems that currently most researchers choose to evaluate their models at the bag level. We investigate whether this bag-level performance of multi-instance learners can be extrapolated to instance-level performance.

1.1.4 Towards declarative statistical inference

The previous sections already gave a few examples of the challenges that are encountered during statistical model development. They demonstrate that the process is complex and requires a thorough understanding of machine learning and statistics. In fact, even for machine learning researchers themselves it is impossible to know all the available methods, their assumptions, and their advantages and disadvantages (Demsar 2006; T. G. Dietterich 1998). Moreover, issues already arise with tasks that appear rather simple at first sight.

Consider for instance the task of inferring from the error estimates on a test sample, which one of two binary classifiers performs best on a given population. Since binary classification error is defined as the proportion of examples for which the predicted label is different from the true label, a hypothesis test for binomially distributed variables is suitable for this task. While several alternative hypothesis tests exist for comparing binomial proportions, the most well-known variant relies on the approximation of the binomial error distribution by a normal distribution. This approximation is justified by the central limit theorem, which states that for n independent and identically distributed variables x_i with finite variance σ^2 and mean μ, the sampling distribution of their average $\bar{x}_n$ converges in probability almost surely to a normal distribution $\mathcal{N}(\mu, \frac{\sigma^2}{n})$ as the number of data points n goes to infinity. If the assumptions of this theorem are indeed satisfied, computing the hypothesis test is no more than applying a cookie-cutter method. It is, however, not so obvious that they are fulfilled. For instance, how do we know when the sample size is 'sufficiently large'? The most concrete, formal, answer comes from the Berry-Esseen theorem, which tells us that the rate of convergence of $\bar{x}_n$ towards μ is at least $n^{-\frac{1}{2}}$. A practical rule of thumb derived from this theorem is that the central limit theorem is valid starting from 30 instances. However, the validity of this recommendation should always be evaluated in its specific context. For variables with heavy-tailed distributions, for instance, it is known that the minimum sample size required for their average to be approximately normally distributed is significantly larger (Wilcox 2010).

Another assumption of the theorem that requires special attention is the independence of the averaged variables. In real world experiments, random variables are seldom entirely independent. The questions is then at what point the dependencies become sufficiently weak to be negligible? When evaluating a classifier on a test set, for instance, dependencies exist between the errors on the individual instances, because they originate from predictions by the same classifier. These dependencies are generally ignored, but the main reason for this is likely that one is unaware of them, rather than that this was a conscious decision.

Examples such as these beg the question whether it is appropriate to put the burden of statistical analysis entirely on the shoulders of the researcher. While to a large extent, the problems can be mitigated by involving a statistical expert, a more efficient solution would be if the researcher could formulate their hypothesis, and the remainder of the experimentation process would be executed by a designated software system. Such a software system would most likely consist of the following components: First, a declarative language in which the statistical queries can easily be formulated. Second, an inference engine that provides the execution strategies for these queries. Finally, a method to translate the declarative queries to the procedural inference engine.

In the last part of this dissertation we focus on the inference engine of such a declarative inference system. A suitable candidate for this component is a Bayesian network model. The structure of such a network encodes the independences between a set of random variables set, so that inference becomes tractable. In fact, a wide variety of fast, efficient probabilistic inference methods is available for these models. In theory, the structure of such a network could be specified by the researcher, but it is quite likely that the dependencies between the variables are unknown, or even more so, the subject of the research. Consequently, we propose that they should be *learned* automatically from the data. Since we only have a sample of the population available, however, there will be uncertainty about the correct model. In fact, multiple competing models may remain after structure learning. This is the final topic of this dissertation: A structure learning algorithm for Bayesian networks that takes into account model selection uncertainty caused by sample variance.

1.2 Dissertation statement

The high-level objective of this dissertation is the understanding and improvement of current statistical inference practices for the evaluation of supervised machine learning algorithms. It takes a small step forward towards the goal of automating the data analysis component of empirical research, making the process more robust in terms of correct execution and interpretation of the results.

1.3 Contributions

The **first contribution** of this dissertation is a synthesis of the existing knowledge about error estimation of predictive models with cross-validation. The

insights that are gathered in this work are otherwise scattered in the literature, ranging from basic statistical research, through (applied) machine learning, to bioinformatics. We found that the results of existing empirical research are sometimes conflicting, probably because of the variety of experimental settings. The choices are indeed extensive: The evaluation of models versus learners, the choice of datasets and learning algorithms, but also the choice of the parameters of cross-validation itself. Theoretical papers do not always offer more insight to the average reader: They often require expert knowledge of statistics and learning theory. In this dissertation we collect insights from all these papers and present a comprehensive and clear overview of their conclusions. These conclusions are supplemented with experiments focused on repeated cross-validation.

The **second contribution** is an investigation of the extent to which the conclusions of experimental studies of multi-instance learning algorithms evaluated in terms of bag-level performance can be generalized to the instance-level. We show theoretically that there is no one-to-one mapping between instance level and bag level accuracy. We support these results with an extensive empirical evaluation of the performance of fourteen multi-instance learners on both synthetic and real-world datasets, in terms of accuracy and AUC-ROC.

The **third contribution** of this dissertation is a meta-learning experiment in which we automatically learn the most suitable multi-instance learner for a given problem. For this aim, we extend the landmarking approach introduced by Pfahringer et al. (2000) to the multi-instance learning setting. Landmarking is the application of a set of quick and simple learning algorithms on a problem, in order to predict the performance of slower, more complex algorithms.

The **final contribution** is a Bayesian network structure learning algorithm that fits into the frequentist framework: If we would repeat the experiment, we would almost certainly draw another sample from the population leading to slightly different results. This sample uncertainty is taken into account by incorporating bootstrapping into the structure learning algorithm. The efficiency of the algorithm is increased by using a Bayesian version of the bootstrap, and applying a racing algorithm. The algorithm is a practical demonstration of how a frequentist problem can be approached from within a Bayesian framework. It is worth noting that this last contribution is purely theoretical, and experiments are still needed to confirm the viability of the approach.

1.4 Structure of the dissertation

The structure of the dissertation is as follows:

Chapter 2 introduces the process of statistical model development with a discussion of our submission to the 2014 ECML-PKDD Predictive Web Analytics Challenge. Along the way, several concepts, methods, and terminology that will be used later on in the text are introduced. This chapter is based on the following workshop paper:

> G. Vanwinckelen and W. Meert (2014). "Predicting the popularity of online articles with random forests". In: *ECML/PKDD Workshop on Predictive Web Analytics*. France, September

Chapter 3 presents a synthesis of the existing knowledge about error estimation of supervised models with cross-validation. The conclusions are supported by an experimental study of the performance of repeated cross-validation for estimating the error of a learner or a model. This chapter is based on the following workshop papers and publication:

> G. Vanwinckelen and H. Blockeel (2012). "On estimating model accuracy with repeated cross-validation". In: *Proceedings of the 21st Belgian-Dutch Conference on Machine Learning (BeneLearn)*. Belgium, May
>
> G. Vanwinckelen and H. Blockeel (2014b). "Look before you leap: Some insights into learner evaluation with cross-validation". In: *ECML/PKDD Workshop on Statistically Sound Data Mining*. France, September
>
> G. Vanwinckelen and H. Blockeel (2014c). "Look before you leap: Some insights into learner evaluation with cross-validation (Poster)". In: *Intelligent Data Analysis*. Belgium, October

In **chapter 4** we show that there is not always a one-on-one mapping between the instance-level accuracy and bag-level accuracy of multi-instance learners. It is supplemented with an extensive empirical evaluation of the bag-level and instance-level performance of multi-instance learning algorithms in terms of accuracy and AUC. The chapter is based on the following publication:

> G. Vanwinckelen, V. Tragante Do O, D. Fierens, and H. Blockeel (2014). "Instance-level accuracy versus bag-level accuracy in multi-instance learning". In: *Data Mining and Knowledge Discovery*

Chapter 5 presents an exercise in meta-learning, aiming to automatically select the best performing multiple-instance learner for a given problem. This chapter is based on the following journal paper:

> G. Vanwinckelen and H. Blockeel (2014a). "A meta-learning system for multi-instance classification". In: *ECML/PKDD Workshop on Learning from Multiple Contexts*. France, September

Chapter 6 introduces a Bayesian network structure learning algorithm that explicitly takes into account sampling variance by means of bootstrapping. This work is unpublished, but it is inspired by the proposal for a declarative experimentation system introduced in Section 1.1.4 and presented in the abstract:

> G. Vanwinckelen and H. Blockeel (2013). "A declarative query language for statistical inference". In: *ECML/PKDD Workshop on Languages for Data Mining and Machine Learning*. Czech Republic, September

1.5 Other publications

In addition to the work presented in this dissertation, the author has also published the following conference paper:

> G. Vanwinckelen, M. V. Otterlo, K. Driessens, and S. Pollin (2011). "Power control for secondary users based on distributed measurements". In: *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*. Germany, May

She also contributed to the project presented in this poster:

> G. Vanwinckelen, D. Verbeeck, W. Meert, and H. Blockeel (2013). "Optimal mobile connectivity using a practical coverage map". In: *LICT Scientific symposium on adaptivity in ICT*. Belgium, September

Chapter 2

Case study: Predictive web analytics

This chapter introduces the process of model development in machine learning by means of a case study from the 2014 ECML/PKDD discovery challenge (Vanwinckelen and Meert 2014). The topic of this challenge was predictive web analytics, more specifically the prediction of the popularity of online content. Our approach won second place for predicting the number of visitors and the number of Facebook likes of a webpage, and first place for predicting the number of tweets that mention a certain webpage.

2.1 Introduction

Fueled by prospects of better catering to user's interests, predictive web analytics is a topic that has been receiving growing interest. The underlying goal is of course increasing revenue by allowing for more focused advertising. Besides predicting the popularity of the actual content (e.g., news articles, blog posts, YouTube videos), the problem can be broadened to predicting the popularity of content on social media sites. A significant portion of internet users today depends on sites like Facebook and Twitter for connecting with content that interests them. Having a presence on social media is thus important for content providers because it increases the exposure and popularity of their articles. Understanding the interactions between social media presence and content popularity therefore contributes to the appeal of predictive web analytics. This is also the goal of the ECML/PKDD challenge: The development of a model

predicting the future number of visits and social media citations of a web page based on its past popularity. The model development process can be divided into different subtasks that are also applicable to other prediction problems. We use the ECML/PKDD challenge as a stepping-stone to describe these different tasks. We start by providing a detailed overview of the data and the prediction task.

2.2 Data and prediction task

The data in the predictive web analytics challenge consists of a collection of time series that were gathered by the real-time analytics engine Chartbeat. For each of a set of hundred websites, a collection of 600 URLs was monitored during 48 hours. Every five minutes, information was collected about the number of visitors in that interval, the number of times the URL appeared in a Twitter message, the number of times a Facebook message containing the URL was liked, and the average time a visitor was active on the page. Additionally, the website's ID, and the weekday and hour the URL was posted is available. The time recorded is presumed to be server side time but this was not explicitly stated in the original data description. It was ensured by the organizers that each URL has at least ten visits.

The prediction task is defined as follows: *Based on the time series data from the first hour, predict for each URL the total number of visitors, tweets, and likes after 48 hours.*

A secret test set was used to objectively compare the solutions of the competitors. It consisted of half of the 600 URLs for each of the hundred web domains, and contains only the data from the first hour. The other half of the data was fully disclosed to the participants for training and evaluation purposes.

The prediction of the three targets on the secret test set had to be submitted to the organizers for evaluation. The quality of the solutions is measured by the mean squared error (MSE) of $\log(x+1)$, with x one of the three targets. Note that from now on we will denote $\log(x+1)$ as $\mathrm{log1p}(x)$. Since the first ranking criterion is the MSE for the number of visitors, priority is given to this target.

2.3 Short overview of the steps in model development

This section gives a general overview of the model development process for a practical application. Each of these steps is described in more detail in the upcoming sections, with a focus on the aspects relevant to the ECML/PKDD challenge.

Data exploration The first step of the process is to obtain an understanding of the structure of the data. For this task, we make use of descriptive statistics and visualizations.

Preprocessing Raw data is often of poor quality: Missing values or erroneous information are ubiquitous. A number of preprocessing steps are therefore needed to obtain a clean dataset that yet still contains all the information needed to produce a usable model. Preprocessing encompasses a range of subtasks, including outlier detection, missing value treatment, feature scaling, selection and generation, and dimensionality reduction.

Modeling One or more machine learning algorithms are applied to the prepared data. While this is the core of the process, it is usually also the part that takes the least amount of time.

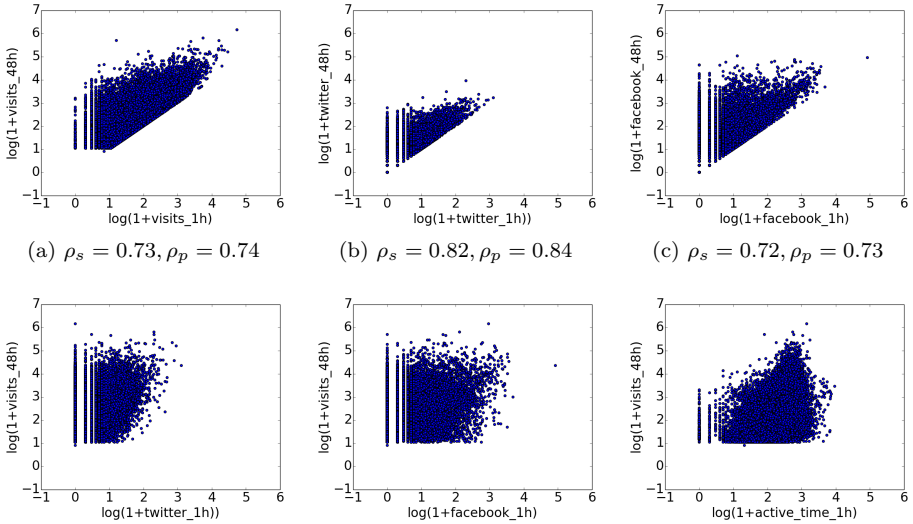
Evaluation After the models are constructed, we select one of them for the final application. Model preference depends on several factors: Is a black-box model acceptable or does the model need to be interpretable? Which performance characteristics are most important? Performance comparison is usually based on statistical inference techniques such as hypothesis testing.

2.4 Data exploration

In order to be able to make the right design choices later on, it is important to have a good understanding of the structure of the data. This is typically accomplished by visualization and descriptive statistical analysis.

2.4.1 Transformation to log space

Earlier research on online content popularity prediction found a linear relationship between the log visitor counts at two different moments in time



(a) $\rho_s = 0.73, \rho_p = 0.74$ (b) $\rho_s = 0.82, \rho_p = 0.84$ (c) $\rho_s = 0.72, \rho_p = 0.73$

(d) $\rho_s = 0.32, \rho_p = 0.36$ (e) $\rho_s = 0.32, \rho_p = 0.32$ (f) $\rho_s = 0.43, \rho_p = 0.39$

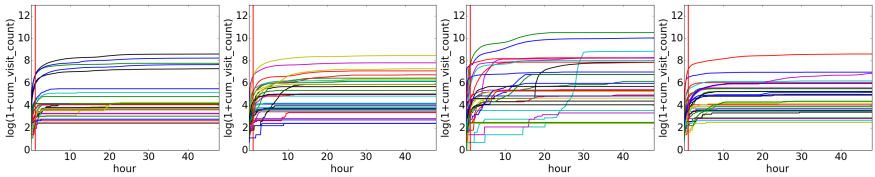
Figure 2.1: The first three scatter plots show $\text{log1p}(x_{1h})$ versus $\text{log1p}(x_{48h})$ with x either equal to *visits*, *twitter*, or *facebook*. The last three scatter plots show the $\text{log1p}(\textit{visits48h})$ versus $\text{log1p}(x_{1h})$.

(Castillo et al. 2014; Szabo and Huberman 2010). If there is also a strong linear relationship between the predictors and the target for the problem at hand, this facilitates our model choice. A log-linear model would then be an appropriate choice.

To avoid having to take the logarithm of 0, we use the log1p instead of the log transformation. Figure 2.1a plots the log1p transform of the total number of visitors after one hour, $\text{log1p}(\textit{visits1h})$, versus the total number of visitors after 48 hours, $\text{log1p}(\textit{visits48h})$. Similar scatter plots are shown for the number of tweets and likes in Figures 2.1b and 2.1c.

For each target, we also compute the Spearman rank and the Pearson correlation (respectively ρ_s and ρ_p, shown below each plot). The Spearman rank coefficient measures the monotonic relationship between two variables, while the Pearson coefficient measures the linear relationship. Both take values between -1 and 1, with values close to $|1|$ indicating a strong correlation. We are mostly interested in the linear correlation, but differences between the two correlation coefficients could provide us with additional useful information.

We find that for all the variables there is indeed a linear relationship between



(a) Website 31 (b) Website 49 (c) Website 69 (d) Website 76

Figure 2.2: The cumulative visitor count time series for 30 random URLs from four different websites.

the count after one hour and after 48 hours. The Spearman and the Pearson correlations are approximately identical, indicating that the relationship is mostly linear and there are no outliers. This can also be derived from the figures. The correlation is highest between measurements of the same variable at 1h and 48h, with the correlation between the number of tweets being strongest ($\rho_s = 0.82, \rho_p = 0.84$). This may explain why the models presented in Section 2.7 perform best for predicting the number of tweets.

The correlation between the visitor count after 48h and the other 1h variables (tweets, likes and time) is not very strong. If we would like our model to extract useful information from these variables also, a log-linear model might not be sufficient.

2.4.2 Cumulative visitor count

Figure 2.2 shows the 48-hour cumulative time series of the number of visits for 30 random URLs from four different websites. The first hour after the URL was posted is indicated by a vertical red line.

While these figures show just a small sample of URLs, the plots nevertheless hint that the characteristics of the curves may be different for each website. We observe for instance a difference in the minimum and maximum number of visitors, and in the spread of the final total visitor count.

Most growth curves are smooth and saturate fairly quickly. Some curves, however, follow a more 'erratic' growth path. Sudden bumps could be caused by the website using a model where popular stories are promoted to a front page, thus suddenly experiencing a surge in visitors because the URL gets more exposure (Lerman and Hogg 2010). Web domains that do not follow this model would not show this behavior. This type of growth poses a problem for learning, because when these bumps occur after the 12th datapoint (the boundary of

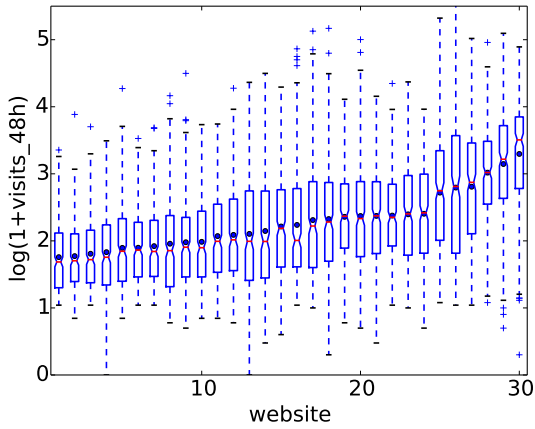


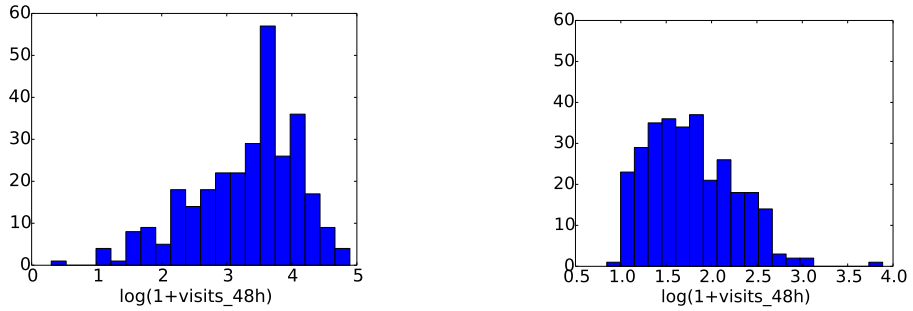
Figure 2.3: Box plots of log1p(*visits48h*) for a random selection of 30 websites, ordered by increasing average visitor count.

the 1h input data), the final visitor count will be difficult to predict. It could therefore be beneficial to remove these observations, as they introduce noise that confuses the learner.

2.4.3 Website statistics

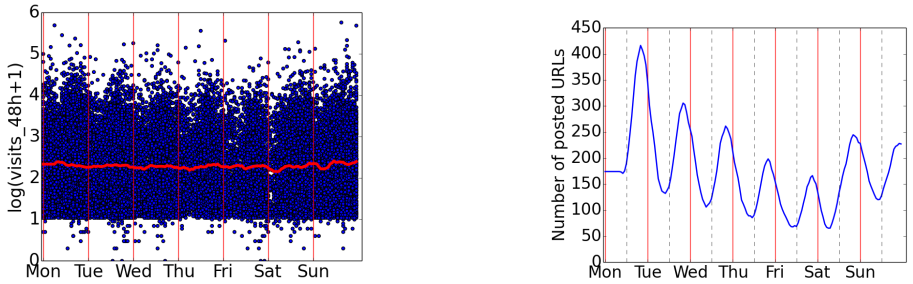
More indications that the characteristics of the web domains differ are found when constructing box plots for log1p(*visits48h*). Figure 2.3 shows the results for a random sample of 30 web domains, ordered by the mean number of visitors. The mean is indicated by a blue dot and the median by a red line. The upper and lower limits of the box represent respectively the first and third quartile. The length of the whiskers corresponds to 1.5 times the interquartile range (IQR), and outliers are indicated individually by '+'.

Figure 2.3 shows some interesting differences between the web domains: Websites with a small mean number of visitors have a median number of visitors that is typically smaller than the mean. This suggests a left skewed distribution where most URLs receive few visitors, and a few URLs receive a large number of visitors. This trend reverses for websites with a large mean number of visitors, suggesting a right skewed distribution for those web domains. This is confirmed by Figure 2.4, showing the histogram of log1p(*visits48h*) of the least popular web domain (a) (smallest mean visitor count), and that of the most popular web domain (b) (largest mean visitor count).



(a) (b)

Figure 2.4: Histograms of log1p(*visits48h*) of respectively the website with (a) the smallest mean visitor count, and with (b) the largest mean visitor count.



(a) (b)

Figure 2.5: (a) Scatter plot of log1p(*visits48h*) versus posted hour and weekday, together with the mean visitor count plotted in red. (b) Smoothed time series of the mean number of posted URLs.

We also notice a difference in the spread of the visitor counts, where unpopular web domains have a smaller visitor count spread than popular web domains. This is especially true for the visitor counts in the first quartile.

We conclude that the statistical properties of the visitor counts differ for each web domain. It would therefore be reasonable to learn a model for each domain separately.

2.4.4 Time dependence

Because the day and hour a URL was posted is known, we can investigate if the time series for the visitor count exhibits periodicity. Figure 2.5a shows a scatter plot of log1p(*visits48h*) in function of the day and time the URL was posted, together with the mean number of visitors, indicated in red. A new day starts at midnight and is indicated by a red vertical line; noon is indicated by a black dotted line. Based on previous research, we expected to find a trend in the popularity of the URLs, with for instance URLs posted at noon being more popular than URLs posted for instance at 4am (Szabo and Huberman 2010). However, no significant periodicity that would indicate such a daily trend was detected in this signal.

Another aspect is the number of URLs posted at each point in time, shown in Figure 2.5b. In this case, a trend *is* observed, which can be visualized using an exponential moving average with a window size of 6 hours. We see that regardless of the day, most URLs are posted somewhere during the evening, and few URLs are posted before noon. Furthermore, we see that most URLs are posted on Monday, and after that there is a decreasing trend with the least number of URLs posted on Friday.

These results suggest that most content providers are from the same region, whereas the readers are from various regions. When we constructed similar plots for smaller, random subsets of websites, we did notice periodicity in the visitor count in some of the subsets. This indicates that readers of a single website are in fact often from the same region. Unfortunately, the trends are obfuscated when the data from the different sites is merged due to the presence of time zones. This is another indication that it may be useful to learn a separate model for each web domain.

2.5 Data preprocessing

Preprocessing is a very extensive field and it is not the purpose of this section to give a complete overview. Instead, we focus on the techniques that are most useful in the predictive web analytics challenge. For a more detailed overview we refer the reader to the literature (Kotsiantis et al. 2006; I. H. Witten et al. 2016).

2.5.1 Data cleaning

The expression *'garbage in, garbage out'* is highly relevant in the context of machine learning model development. The quality, representational format, and size of the dataset all have a high impact on model performance. Data errors and irrelevant or redundant information create noise that makes it more difficult for an algorithm to crystallize the data into a useful signal. One of the goals of preprocessing is to deliver a clean dataset that is free of these issues.

The dataset of the predictive web analytics challenge is already of reasonable quality. It contains for instance no missing values, and it was ensured by the creators that each URL has at least ten visits. A useful data cleansing step would be to remove the time series in the dataset that show erratic behavior after the 12th data point, as shown in the cumulative visitor count Figure 2.2. When these fluctuations show no correlation with the data during the first hour, they are unpredictable and may distort the model. However, this strategy was left as future work in the current proposal.

2.5.2 Feature extraction

Besides data cleaning, preprocessing also encompasses feature selection and transformation. Feature selection can be seen as a form of dimensionality reduction and presents several advantages. It reduces the required dataset size for learning, and prevents overfitting. Feature transformation can be interpreted broadly, ranging from rescaling to generating entirely new features. The purpose is to reshape the data into a format that is suitable for the bias of a learning algorithm. For instance, algorithms that rely on distances are known to be sensitive to variables being measured on a different scale, so normalization is recommended in such cases. Generating entirely new features generally requires a thorough understanding of the problem at hand, although automatic feature generation is also an active research topic (Markovitch and Rosenstein 2002).

Feature selection and transformation are issues that require special attention in the current setting: Since the data consists of time-series, it will either have to be transformed into attribute-value data or it will have to be processed by a special-purpose learning algorithm. For simplicity we choose the former approach, i.e., treat the problem as a regular supervised learning problem. The challenge is then to extract uncorrelated features from the time series data that still contain all the required information.

In the exploratory data analysis we saw that a linear relationship exists between the log1p of the total number of visitors during the first hour and after 48 hours.

A similar effect was observed for the number of tweets, likes and active time. A first step is therefore to reduce the four original time series (visits, tweets, likes, and active time) to $\log 1p(1 + x)$, with x one of the four measured quantities.

Time series not only contain information about counts, they also have a global shape that emerges from local correlations between data points nearby in time. One way of extracting this information is to transform the time series to the frequency domain by means of the Fourier transform. Because we are interested in the growth characteristics of the visitor count instead of temporal fluctuations, we apply the transformation to the cumulative visitor count series instead of the original series.

Applying the Discrete Fourier transformation on a cumulative time series of 1 hour sampled every 5 minutes results in 12 complex frequency components, with Fourier frequencies ranging from the DC component 0Hz up to the highest frequency 3.310^{-3}Hz. The imaginary part of the coefficients contains no useful information for popularity prediction, so we only retain the moduli of the complex Fourier coefficients. Because the time series is real, the real part of the coefficients is an even function, so all the required information is available in the DC-component together with the first six moduli. Instead of a time series we now have a series ordered in the frequency domain. In order to keep the number of features low we average the coefficients: The DC component may be a useful feature by itself, as it can be interpreted as the average visitor count during the first hour. We average the fourth and fifth coefficient, and the sixth to seventh coefficient.

Figure 2.6 provides some insight into the information contained in the Fourier transform of the time series. It visualizes three cumulative visitor count curves (black) together with their Fourier approximation based on the DC-component, the harmonic with the lowest frequency, and its complex conjugate. The first seven Fourier coefficient moduli are shown below. Notice how the second time series is rather smooth. In the frequency domain this translates into a DC component that is much larger than the harmonic coefficients. The third time series on the other hand, contains a 'bump' in the time domain around the fourth data point, resulting in larger harmonic components coefficients.

Our exploratory data analysis also found a relationship between the time a URL was posted and the visitor count for subsets of web domains. Timing information should therefore also be included in the feature set. We only include a feature based on the hour the URL is posted, and not on the weekday, as often for a given website there was no data available for each weekday. The 24 hour interval can be divided into four intervals: [0h,6h), [6h,12h), [12h,18h), and [18h,24h) representing respectively night, morning, afternoon, and evening.

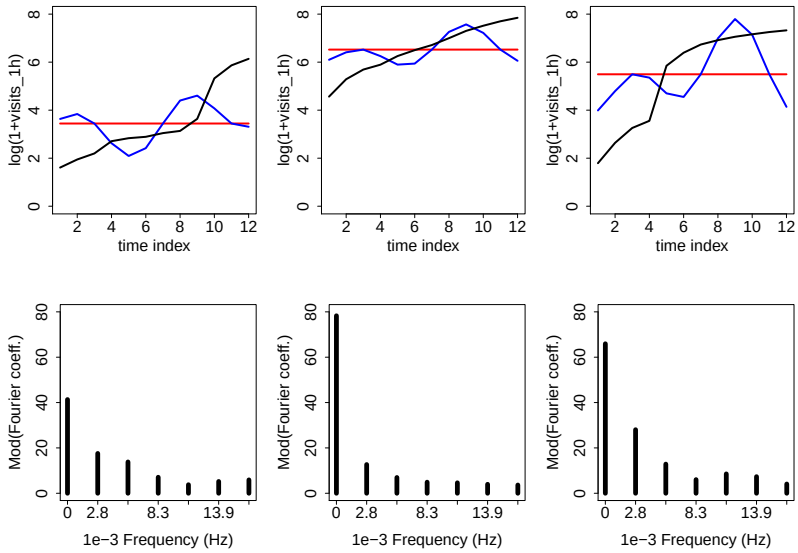


Figure 2.6: Top: Cumulative 1 hour visitor count (black), DC component of the Fourier transform (red), Fourier series consisting of the DC, the 1st harmonic, and its complex conjugate (blue). Bottom: Frequency spectrum of the Fourier transform of the time series above.

To summarize, our dataset consists of three types of features:

- The log1p of the sum of the counts for the first hour of data for each of the four original time series (visits, tweets, likes, and active time).
- Three features containing time series 'shape' information, derived from the Fourier transform of the cumulative time series for visits.
- A feature that represents the time of the day a URL is posted.

Note that we approached the problem of feature generation and selection in an intuitive, 'manual' way. Alternatively, we could have also generated and selected features in a more systematic way by first computing several alternatives and afterwards applying a search algorithm to select the best performing combinations.

2.6 Modeling

The next modeling step is to select the learning algorithm with the best matching inductive bias for the problem setting. The exploratory data analysis indicated a linear relationship between the log of past and future counts of visits, tweets and likes. It seems that a linear regression model would therefore be appropriate. As a safeguard against overfitting, we will use the lasso, also known as L1-regularization. The advantage of the lasso is that it facilitates interpreting the model, as it forces part of the coefficients to be zero. The analysis also indicated that the statistical properties of the data differ for each web domain. We could therefore learn a different model for each web domain. This is possible because the data in the secret test set stems from the same websites as those in the training set. A disadvantage of this approach though, is that the size of the training set for each model is significantly reduced as it is restricted to one out of hundred domains.

An alternative to learning multiple models is to build a single model on the full dataset and including the web domain ID as a feature. The advantage is that web domains significantly differing in characteristics from the others will naturally be modeled differently. When applying linear regression, however, this requires introducing a dummy indicator variable $D_i = I[D_i = i]$ for each ID. Without this adaptation, the web domain ID would be interpreted as a numeric variable, while it is actually a factor. Regularization is then certainly necessary, as this approach creates many extra variables.

We also test a random forest learner, as available in the R package cparty (Hothorn et al. 2006). The reason being that random forests regularly provide the top contribution in machine learning competitions (Ben 2012; González-Brenes and Matías 2011; Sculley 2012). Their success is explained by the fact that they perform well on high-dimensional problems, which are notoriously difficult to model. Furthermore, good performance can be achieved without much parameter tuning.

The inductive bias of a random forest is that the data can be modeled as a collection of decision trees, which partition the data recursively. An important advantage of decision trees is that they allow for an easy understanding of the model, as the partitioning can be graphically represented as a rooted directed tree. A disadvantage of decision trees is that they are instable with respect to changes to the training data. However, random forests solve this problem by creating many trees, each learned on a different bootstrap sample created from the original dataset. Combining the predictions of all these trees, for instance by averaging, results in a smoother decision boundary showing less variation when making small changes to the training data. Similar to the regression model, we

	Visits	Tweets	Likes
Per domain	3RHO MSE		
RF trees 50, split 3	0.90	0.47	1.92
RF trees 500, split 3	0.90	0.47	1.92
Linear regression	1.03	0.39	1.79
All domains	10KCV MSE		
RF trees 50, split 3	1.23	0.69	2.37
RF trees 500, split 3	1.19	0.68	2.41
Lasso λ 10KCV optim.	1.32	0.62	2.33
Submitted (per domain)	3RHO MSE		
RF trees 500, split all	0.76	0.35	1.50
RF trees 500, split all (test)	0.99	0.65	1.38

Table 2.1: Performance results for the random forest (RF) and (penalized) linear regression models. The number of trees in the random forest was either 50 or 500. The number of variables eligible for a node split was either 3 or all.

can construct one random forest per web domain, or alternatively construct one large random forest, possibly with the web domain ID as a feature. In theory, dummy variables are not needed here for representing these web IDs, although certain random forest implementations do require this.

2.7 Evaluation and interpretation

2.7.1 Evaluation

In this section we evaluate the performance (MSE(log1p(x_{48h}))) of the models proposed in the previous section: a lasso model and a random forest (RF), either learned per web domain or on all domains at once. In order to avoid winning the competition by overfitting, it is only allowed to submit a limited number of models to be evaluated on the secret test set. Before submitting, we therefore have to estimate model performance on the training data. This means we have to split our original dataset into training and test sets. The conventional solution for this are resampling methods. The advantages and disadvantages of this approach are discussed in detail in the next chapter. One popular resampling method is k-fold cross-validation (KCV), where the dataset is randomly divided into k equally sized, non-overlapping subsets called folds. Each of these folds is in turn used as a test set, where the remaining folds are used as a training set. the final performance estimate is the mean of the estimates on each test set. Another alternative is repeated hold-out with random

resampling (RHO). Here, the dataset is repeatedly split into train/test subsets without taking into account overlap between the different subsets.

We demonstrate both of these resampling methods in our study: Tenfold cross-validation is used for evaluating the models learned on the entire dataset, while three times repeated hold out using 1/3 sized test sets is used for evaluating the models learned per web domain. Note that using different resampling methods for these two types of models has no significant influence on our conclusions. Even if we would have used the same resampling method, the results would still be difficult to compare because of the large difference in training set sizes in both cases.

The random forest and lasso models are tested in two different settings: one model per web domain, and one model on the entire dataset. The results in Table 2.1 indicate that the domain specific models perform better. However, as already mentioned, we should be careful with comparing. First, the sizes of the training sets differ significantly in the two cases. Second, there are indications that despite our precautions, the domain specific models are still overfitting. This can be seen from the reduced performance of the random forest model on the secret test set for visits and tweets in comparison to the performance in the repeated hold-out evaluation.

The random forest learners seem to perform better than the lasso, despite the fact that we more carefully tuned the parameter of the lasso. We determined the optimal value of the penalization parameter λ by using nested tenfold cross-validation. For the model per domain, the optimal value always turned out to be 0, which amounts to simple linear regression. For the random forest, we only experimented with a forest of 50 and 500 trees. These results show that varying the number of trees does not significantly influence performance. We also experimented with either allowing all features to be chosen in node splits, or only a subset of three. The subset size was based on a rule of thumb recommending the square root of the number of features. It seems that choosing from all variables leads to the best performance.

Finally, we note that in an effort to boost performance, we also include the sum of $\log1p(\sum visits)$ and $\log1p(\sum active\_time)$ as a feature in the submitted model. This feature has a high Spearman rank correlation with respect to the visitor count (0.72), although we expect it to be highly correlated with both $\log1p(\sum visits)$ and $\log1p(\sum active\_time)$.

	var. imp. $\pm$ stdev
Visits	1.842 ± 0.144
DC Fourier	$0,304 \pm 0.026$
Fourier 2-4	0.117 ± 0.012
Fourier 5-7	0.062 ± 0.12
Tweets	0.029 ± 0.002
Visit time	0.021 ± 0.002
Post hour	0.017 ± 0.002
Likes	0.006 ± 0.001

Table 2.2: Average variable importances (10f-cv) for the eight features of the random forest constructed on the full dataset of 100 web domains.

2.7.2 Interpretation

In many applications it is important to not only have good performance, but also to understand the determining factors of a model's performance. A decision tree is easy to interpret: We can deduce which variables correlate strongest with the target based on their position in the tree. Although we can still inspect the individual trees, this interpretability is somewhat lost for a random forest. Instead, we have to rely on more sophisticated solutions such as for example computing variable importances. We give a basic description of this concept, for more details we refer to (Strobl et al. 2008).

The computation of variable importances is similar to the process of selecting a leaf to split, i.e, by means of a permutation test. First, we let every tree in the forest predict the *out-of-bag* cases. Next, we permute the input variable of interest for the out-of-bag samples, and make predictions again for these modified examples. The variable importance then equals the difference in accuracy between the original and the modified tree, averaged over all trees.

For conciseness we only discuss the variable importances for predicting the visitor count. The results presented in Table 2.2 show that, not surprisingly, the 48 hour visitor count is the most powerful feature for predicting this count. Interestingly, the most important features after that are the averaged Fourier coefficients. The other features were not found to have much predictive power.

2.8 Conclusion

This chapter walked the reader through the steps of the model development process. As an example we used the ECML/PKDD 2014 Discovery Challenge,

where the goal was to predict the popularity of a web page based on time series data from the Chartbeat web analytics engine. The approach submitted in the competition consisted of learning a random forest on a set of features derived from time series data, capturing information about the initial visitor count, the growth rate, and temporal effects.

Several other approaches that were not investigated in this chapter can still be thought of. First, algorithms specifically designed to handle time series can be investigated. Second, our data analysis suggests that some groups of websites show similar behavior, but when merging all of them together, specificities are averaged out. A solution to this problem is to learn a single model per website. The disadvantage is that this significantly decreases the size of the training set. A noteworthy alternative is to cluster websites that show similar behavior. Finally, since the goal of the challenge was to predict three correlated targets, an interesting direction might be multivariate prediction.

Chapter 3

Model evaluation with cross-validation

This chapter investigates the estimation of the prediction error of a learner or model with cross-validation. First, we make the reader aware that there are two different definitions of the prediction error. Next, we investigate which of these two error measures cross-validation estimates. Special attention is paid to leave-one-out cross-validation. Finally, we explore whether repeated cross-validation results in more reliable estimates of learner accuracy than a single cross-validation. This work contains material from the following papers: (Vanwinckelen and Blockeel 2012, 2014b,c).

3.1 Introduction

Most machine learning tasks can be addressed using multiple alternative learning methods. Empirical performance evaluation plays an important role here. The behavior of all these methods is not always theoretically well-understood, and if it is, it is important to stress the real-world implications. Therefore, almost all papers contain some form of performance evaluation, which usually estimates the quality of models resulting from the machine learning effort (for instance, predictive performance), the computational effort required to obtain these models, or other performance criteria.[1]

[1] In line with most machine learning literature, and somewhat at variance with the statistical literature, the term "model" here refers to the result of the learning effort (e.g., a specific decision tree), not to the type of model considered (e.g., "decision trees").

For predictive models, a major criterion is usually the accuracy of the predictions, or more generally, the expected "loss", using a loss function that compares the predicted values with the correct ones. Much of the research in machine learning focuses on developing better learning methods, that is, methods that are more likely to return models with a lower expected loss.

This goal statement is still somewhat vague, and can be interpreted in multiple ways. From the no-free-lunch theorems (Wolpert 1996), we know that, averaged over all possible learning tasks, all learners perform equally well, so the goal only makes sense when the set of tasks is restricted to, for instance, a specific application domain. A specific learning task can be formalized using a single population. The task is then to learn a model for this population from a dataset sampled at random from it. The following two different versions of this task can then be distinguished.

1. Given a dataset D from population P, and a set of learners, which learner learns from D the most accurate model on P?
2. Given a population P, and a set of learners, which learner is expected to yield the most accurate model on P, when given a random sample of a particular size from P?

In statistical terminology, these versions correspond to finding the learner that minimizes the conditional and unconditional error, respectively.

The first question is relevant for researchers who evaluate the learning algorithms using the same dataset that an end user will use to build a model. The second question is relevant when the end user's dataset is not available to the researcher.

Authors rarely clarify which of these two questions they try to answer when evaluating learning methods. This is often clear from the context. For instance, when testing a learning method on UCI datasets (Lichman 2013), one is clearly not interested in the models learned from these datasets, but in the behavior of the learner on other, similar learning problems, where "similar" is to be interpreted here as "learning from a dataset of similar size, sampled from a population with a distribution similar to that of the UCI datasets population[2]". On the other hand, when learning predictive models from a given protein-protein interaction network, one may well be interested in the predictive quality of these specific models.

[2]The qualification "of similar size" for the dataset is needed because the quality of a learned model depends on the size of the dataset from which it was learned (see, e.g., (Perlich et al. 2003)), and the qualification of the distribution is needed because it is well-known that no learner can be optimal for all population distributions (Wolpert 1996)

In the statistical literature, the two questions are clearly distinguished, and studied separately. However, this literature is not always very accessible to the machine learning audience; relevant information is spread over many different articles that are often quite technical. Consequently, researchers are not always aware of the distinction, and so the question is not made explicit. This carries a risk, because both questions require a different approach, and leaving the question implicit may obfuscate the fact that incorrect statistical methods are used.

When no large dataset is available, one particular statistical sampling method, k-fold cross-validation, has become a standard in machine learning. What constitutes a small dataset is not always clear and so we can find cross-validation in a wide range of research papers. The statistical properties of this estimator have been studied in detail, and in order to improve the quality of the error estimate, a number of variants have been proposed. It is for instance often advocated that in order to reduce the variance of the cross-validation estimator, the procedure should be repeated a number of times and the average over these results should be taken.

It is not clear, however, whether this recommendation actually improves the quality of the estimate. To answer this question, we would first have to understand which of the above questions cross-validation actually helps answering. The cross-validation estimate is computed on a single sample, so it is not illogical to assume it is an accurate estimator for the error of a model learned on the given sample. On the other hand, the estimate is computed as the average error of models learned on training sets that were derived from the sample, so perhaps it is an estimator for the error of the learner. The answers to these questions can be found in the statistical literature, but it is our goal to make this information more accessible.

The remainder of the chapter is organized as follows. In Section 3.2 we first provide a number of general definitions that are needed in the rest of the chapter. Section 3.3 discusses cross-validation as an estimator for the conditional error, with a specific focus on leave-one-out cross-validation. Section 3.4 then discusses the issues with using repeated cross-validation as an estimator for the unconditional error. We supplement our discussion with two experiments focused on repeated cross-validation in Section 3.5. The first experiment investigates whether repeated cross-validation estimates the error of a model or a learner. The second experiment investigates the uncertainty about selecting the winning model when comparing two models with cross-validation. We conclude in Section 3.6.

3.2 Preliminaries

3.2.1 Learning task

We focus on the setting of learning predictive functions from examples of input-output pairs. In the following, 2^S denotes the power set of S and $\mathcal{Y}^{\mathcal{X}}$ denotes the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. We formalize learning tasks as follows.

Definition 1 (predictive learning)**.** A predictive learning task is a tuple $(\mathcal{X}, \mathcal{Y}, p, T, C)$, where $\mathcal{X}$ is called the input space, $\mathcal{Y}$ is called the output space and p is a probability distribution over $\mathcal{X} \times \mathcal{Y}$. The training set is defined as $T \subseteq \mathcal{X} \times \mathcal{Y}$, and $C : \mathcal{Y}^{\mathcal{X}} \times \mathcal{P} \to \mathbb{R}$ (with $\mathcal{P}$ the set of all distributions over $\mathcal{X} \times \mathcal{Y}$) is some criterion to be optimized.

The probability distribution p is called the *population distribution*, or simply population. The instances in T are usually assumed to have been drawn independently from distribution p. Without loss of generality, we assume from here on that C is to be *minimized*.

Definition 2 (learner)**.** A learner L is a function with signature $L : 2^{\mathcal{X} \times \mathcal{Y}} \to \mathcal{Y}^{\mathcal{X}}$.

Definition 3 (performance)**.** Given a learning task $(\mathcal{X}, \mathcal{Y}, p, T, C)$, a learner L_1 has better performance than a learner L_2 if $C(L_1(T), p) < C(L_2(T), p)$.

Note that, as the goal of predictive learning is to find a model that can make predictions for instances we have not seen before, the quality criterion for the resulting model is based on the population p, not on the training set T. Differently from T, however, p is not known to the learner. It is often also not known to the researcher evaluating the method.

In the following, we assume that the output space $\mathcal{Y}$ is one-dimensional. If $\mathcal{Y}$ is a set of nominal values, the learning task is called classification; if $\mathcal{Y}$ is numerical, the task is called regression.

3.2.2 Error measures

Much of the relevant literature on the estimation of learning performance with resampling estimators focuses on regression and classification tasks, and uses error as a performance measure. A few examples are (Borra and Ciaccio 2010; Burman 1989; T. G. Dietterich 1998; Efron 1983; Hanczar and Dougherty 2010). We start with repeating some basic definitions used in that context. For

simplicity, our definitions focus on classification. They are, however, easily extensible to regression.

In the following, $\mathbf{Pr}_{x\sim p}[C]$ denotes the probability of a Boolean function C of x evaluating to true, and $\mathbf{E}_{x\sim p}[f(x)]$ denotes the expected value of $f(x)$, when x is drawn according to p. For a set T, we use the notation $T \sim p$ to denote that all elements of T are drawn independently according to p.

The concept of "error" can be defined on two levels: that of learners, and that of learned models (classifiers). The error of a classifier is defined as follows:

Definition 4 (error)**.** The error of a classifier m is the probability of making an incorrect prediction for an instance drawn randomly from the population. That is,

$$\varepsilon(m) = \mathbf{Pr}_{(\mathbf{x},y)\sim p}[m(\mathbf{x}) \neq y] \tag{3.1}$$

For learners, two types of error are typically distinguished: The conditional and the unconditional error (Hastie et al. 2001, Chapter 7).

Definition 5 (conditional error)**.** The conditional error of a learner L for a dataset T, denoted as $\varepsilon_c(L,T)$, is the error of the model that it learns from T.

$$\varepsilon_c(L,T) = \varepsilon(L(T)), \text{ with } m_T = L(T). \tag{3.2}$$

Definition 6 (unconditional error)**.** The unconditional error of a learner L at size n, denoted $\varepsilon_u(L,n)$, is the expected error of the model learned by L from a random dataset of size n. It is the mean of the conditional error $\varepsilon_c(L,T)$ taken over all datasets of size n that can be sampled from population p.

$$\varepsilon_u(L,n) = \mathbf{E}_{\{T\sim p:|T|=n\}}[\varepsilon_c(L,T)]. \tag{3.3}$$

These two different types of error are clearly related to the two different questions mentioned in the introduction. The conditional error of a learner is relevant if the dataset T used for the estimation is identical to the one that will be used by other researchers when learning predictive models for the population. The unconditional error is relevant if the dataset T used for the estimation is representative for, but not identical to, the datasets that other researchers will use. It estimates the expected performance of the learner on similar datasets (that is: datasets of the same size sampled from the same distribution), rather than its performance on the given dataset.

In the remainder of this text, we focus on error as the criterion to be optimized, but it is clear that for any loss function, a distinction can be made between the conditional and unconditional version of that loss.

3.2.3 Cross-validation error estimator

As the population p is usually unknown, the true error (conditional or unconditional) cannot be computed but must be estimated using the training set T. Many different estimation methods have been proposed, but by far the most popular estimators are based on cross-validation. They all rely on the notion of empirical error:

Definition 7 (Empirical error)**.** The empirical error of a model m on a set of instances S, denoted $\hat{\varepsilon}(m, S)$, equals

$$\hat{\varepsilon}(m, S) = \frac{|\{(\mathbf{x}, y) \in S | m(\mathbf{x}) \neq y\}|}{|\{(\mathbf{x}, y) \in S\}|}.$$

In *k-fold cross-validation*, a dataset T is randomly divided into k equally sized (up to one instance) non-overlapping subsets T_i, called folds. For each fold T_i, a training set Tr_i is defined as $T \setminus T_i$, a model m_i is learned from Tr_i, and m_i's error is estimated on T_i. The mean of all these error estimates is returned as the final estimate.

Definition 8. The k-fold cross-validation estimator, denoted $\hat{\varepsilon}_{cv}(L, T)$, consists of partitioning T in k subsets $T_1, T_2, \ldots, T_k$ such that $|T_i| - |T_j| \leq 1 \; \forall i, j$, and computing

$$\hat{\varepsilon}_{cv}(L, T) = \frac{1}{k} \sum_{i=1}^{k} \hat{\varepsilon}_i(L(T \setminus T_i), T_i)$$

We call the assignment of the instances in T into k non-overlapping subsets the *partitioning* of the data. A particular partitioning of the data will be denoted by π.

If the number of folds k equals the number of instances $|T|$ in the dataset, the resampling estimator is called *leave-one-out cross-validation*. This special case is usually studied separately.

Definition 9. The leave-one-out cross-validation estimator, denoted $\hat{\varepsilon}_{loo}(L, T)$, is

$$\hat{\varepsilon}_{loo}(L, T) = \frac{1}{|T|} \sum_{i=1}^{|T|} \hat{\varepsilon}_i(L(T \setminus \{t_i\}), \{t_i\})$$

with $T = \{t_1, t_2, \ldots, t_{|T|}\}$.

Contrary to $\hat{\varepsilon}_{cv}$, which is a stochastic function, $\hat{\varepsilon}_{loo}$ is deterministic, as there is only one way to partition a set into singleton subsets.

Repeated k-fold cross-validation computes the mean of n different k-fold cross-validations on the same dataset, each time using a different random partitioning π_j.

Definition 10. The r-times repeated k-fold cross-validation estimator is

$$\hat{\varepsilon}_{rcv}(L,T) = \frac{1}{r}\sum_{j=1}^{r}\hat{\varepsilon}_{cv,\pi_j}(L,T).$$

In practice, a stratified version of these estimators is often used. In stratified cross-validation, the random folds are chosen such that the class distribution in each fold is maximally similar to the class distribution in the whole set. Note that stratification is not possible in the case of leave-one-out cross-validation.

3.2.4 Estimator quality

From the frequentist viewpoint, the error of both a learner or a model is a statistical parameter that can be estimated by repeatedly sampling p and applying cross-validation to that sample. The cross-validation estimator is thus a random variable itself, with a probability distribution called the *sampling distribution*. The quality of the estimator is described in terms of the characteristics of this sampling distribution, with a focus on mean and variance.

The difference between the mean of the sampling distribution and the parameter is called the bias of the estimator. It measures the systematic deviation of the estimator from the parameter when the data population is repeatedly sampled:

$$B(\hat{\varepsilon},\varepsilon) = \mathbf{E}[\hat{\varepsilon}] - \varepsilon.$$

The variance of the estimator measures how much it varies around its own expected value. It is defined as follows:

$$\mathrm{Var}(\hat{\varepsilon}) = \mathbf{E}[(\hat{\varepsilon} - \mathbf{E}[\hat{\varepsilon}])^2].$$

Notice that contrary to bias, variance is independent of the estimand. In our specific case, it is thus the same whether one wants to estimate the conditional or the unconditional error.

The concept of variance is not restricted to estimators only. We can for instance also define the sample variance of the conditional errors themselves $\varepsilon_c(L,T)$.

The bias and variance of the estimator are usually combined into a single quality measure, the mean squared error of the estimator:

$$MSE(\hat{\varepsilon}, \varepsilon) = \mathbf{E}[(\hat{\varepsilon} - \varepsilon)^2] = \mathrm{Var}(\hat{\varepsilon}) + B^2(\hat{\varepsilon}, \varepsilon)$$

Note that the above concepts are quite different from the bias and variance of the learner itself. It is perfectly possible that a learner with high bias and low variance (say, linear regression) is evaluated using an estimator with low bias and high variance.

3.3 Estimating the conditional error

3.3.1 Algorithmic stability

Cross-validation involves splitting the available dataset T into a training and a test set. The model m' that is learned on the training set generated by the estimator will thus differ from the model m that is learned on the complete dataset T. Typically, the training set is smaller than T, and therefore systematically produces models with a larger error. It is well known that this makes the estimator pessimistically biased with regard to both the conditional and the unconditional error (Hanczar and Dougherty 2010; Kohavi 1995).

However, cross-validation is unbiased for the unconditional error on training sets of size $n(k-1)/k$ (Bengio and Grandvalet 2004; Borra and Ciaccio 2010). Indeed, under the assumptions that the instances are i.i.d., and each sample T is drawn from the population with equal probability, the difference $\hat{\varepsilon}_{cv} - \varepsilon_c$ will on average be equally often negative as it will be positive.

The pessimistic bias for the conditional error cannot be overcome in the same way. Reducing the size of T, changes the classifier $L(T)$ and thus we are no longer considering the same learning problem. A solution for this problem is to choose a k that is large relative to n, with leave-one-out cross-validation being the most optimal choice in this respect. For simplicity, we will from now on focus on leave-one-out cross-validation, but the discussion is relevant for any large value of k.

Under certain conditions, leave-one-out cross-validation is indeed reliable to estimate the conditional error (Chapelle et al. 2002; Elisseeff, Pontil, et al. 2003; Molinaro et al. 2005). A necessary condition is that the learning problem is stable. The theory of algorithmic stability is rather technical, and multiple definitions of stability have been proposed, so we will only provide an intuitive explanation with the goal of helping the reader understand why stability is a desirable property. For technical details, we refer to (Bousquet and Elisseeff 2002; Devroye, Wagner, et al. 1980; Kearns and Ron 1999; Kumar et al. 2013).

An unstable learning problem is a problem where small changes to the training set cause large changes to the model predictions. The stability of a learning problem is a combination of both the properties of the learner and the data: When a learner generates complex models that have many tunable parameters, these models can adapt well to a specific dataset, leading to instability of the predictions with respect to changes in the training set. This contrasts with learners that makes strong assumptions about the data (their bias): The model parameters will remain more stable when changes are made to the training set, and consequently so will the predictions of the model. Stability is thus related to the bias-variance trade of the learning algorithm. A number of learning algorithms, such as regularized least-squares regression and certain types of SVMs have been proven to be stable according to the definition of stability given in (Kumar et al. 2013).

Next to the learning algorithm, the characteristics of the data also influence stability. Smaller datasets for instance make a learning problem more unstable. Also outliers in the dataset can cause a learning problem to be unstable. A toy example of an unstable learning problem is the application of a majority vote classifier on a dataset that contains 50% positive instances and 50% negative instances. When applying leave-one-out cross-validation on such as dataset, all the instances will be predicted incorrectly.

When a learning problem is unstable, the leave-one-out cross-validation estimator has *large variance* in comparison to cross-validation estimators with lower k, so it is not reliable. This is well known, but it is our impression that the meaning of this statement is not always well understood. Perhaps this is because there are several random factors that influence the cross-validation estimate, which impedes a good understanding of the concept 'variance of the cross-validation estimator'.

Another difficulty is understanding the link between the stability of the learning problem and the variance of leave-one-out cross-validation. It is indeed not self-evident how the instability of predictions on a single sample can cause the large variability of the estimator over multiple samples. The remainder of this section is therefore focused on understanding the above statement in relation to the different random factors that influence the cross-validation estimate.

3.3.2 Stochasticity of cross-validation

Most estimators considered in basic statistics are deterministic functions. Consider for instance the sample average as an estimator for the population mean: Given a sample, the average is uniquely determined. It is consequently intuitively clear that "variance" refers to the variance induced by the randomness

of the sample; that is, a different sample would result in a different average, and the variance of these estimates is what the term "variance" refers to here.

The cross-validation estimator, however, is stochastic: It depends on random choices (typically some random partitioning or resampling π of the data). Hence, even if the learner L and sample T are fixed, these estimators have a non-zero variance.

We call this variance the *internal variance* of the estimator. It is the variance of the estimator over all possible partitionings or resamplings π of the dataset T. The naming is in line with the literature, (Efron and R. Tibshirani 1997; Hanczar and Dougherty 2010; Kim 2009).

Definition 11. Internal variance of the (un)conditional error estimator

$$\mathrm{Var}_\pi(\hat{\varepsilon}(L,T)) = \mathbf{E}_\pi[(\hat{\varepsilon} - \mathbf{E}_\pi[\hat{\varepsilon}])^2]. \tag{3.4}$$

The internal variance of the leave-one-out cross-validation estimator is zero. This is because there is only one way of partitioning the dataset such that each instance becomes a test instance once. This contrasts with k-fold cross-validation, where multiple partitionings of a sample are possible and so different estimates can be obtained from that sample.

When it is stated that the leave-one-out cross-validation estimator has large variance, what is meant is the variance of the error over different samples, i.e., the *sample* variance:

$$\mathrm{Var}_T(\hat{\varepsilon}) = \mathbf{E}_T[(\mathbf{E}_T[\hat{\varepsilon}] - \hat{\varepsilon})^2] = \mathbf{E}_T[(\varepsilon_u - \hat{\varepsilon})^2]. \tag{3.5}$$

The substitution of $\mathbf{E}_T[\hat{\varepsilon}]$ by ε_u is possible because the bias of leave-one-out cross-validation is in most cases negligible. Note that the sample variance of leave-one-out cross-validation should not be confused with the variance of the individual errors for a given sample: $\mathrm{var}(e) = \mathbf{E}_i[(\mathbf{E}_i[e] - e)^2]$, with e the leave-one-out error on instance i.

3.3.3 Sample variance of leave-one-out cross-validation

In what follows, we connect the sample variance of the leave-one-out cross-validation estimator to the stability of the individual errors. We start by rewriting the sample variance as defined in 3.5 by using the 'variance sum law for dependent variables'. This law states that the variance of the sum of dependent variables equals the sum of the covariances of these variables. Since cross-validation is an average of the individual errors on a given sample, the

variance can be written as the average covariance between the individual errors:

$$\mathrm{Var}_T(\hat{\varepsilon}_{loo}) = \frac{1}{n^2}(\sum_{i=1}^{n} \mathrm{Var}_T(e_i) + \sum_{i,j=1}^{n} \mathrm{Cov}_T(e_i, e_j)), \text{ with i,j} \in T. \tag{3.6}$$

The covariance between the errors on two distinct instances e_1 and e_2 from the same sample is defined as:

$$\mathrm{Cov}_T(e_1, e_2) = \mathbf{E}_T[(\mathbf{E}_T[e_1] - e_1)(\mathbf{E}_T[e_2] - e_2)] = \mathbf{E}_T[(\varepsilon_u - e_1)(\varepsilon_u - e_2)] \tag{3.7}$$

Since the training sets overlap there is dependence between the predictions, so $\mathrm{Cov}_T(e_1, e_2)$ will contribute to $\mathrm{Var}_T(\hat{\varepsilon}_{loo})$.

The covariance between an error e_i and itself is a special case and is simply its sample variance. It can be computed as:

$$\mathrm{Var}_T(e_i) = \mathbf{E}_T[(\mathbf{E}_T[e_i] - e_i)^2] = \mathbf{E}_T[(\varepsilon_u - e_i)^2] \tag{3.8}$$

Bengio and Grandvalet (2004) proved that the variance of an individual error is the same regardless of which instance is selected. Similarly, the covariance between two randomly selected errors from the sample is the same regardless of which two instances are selected. Intuitively, this can be understood by realizing that all instances are independently sampled from the same population (i.i.d.), so they are interchangeable when computing the leave-one-out cross-validation estimate.

If we substitute the variance and covariance terms in formula 3.6 by respectively formula 3.8 and 3.7, and we take into account that the covariances of the errors from the same sample are identical, the variance of the leave-one-out cross-validation estimator can be written as:

$$\begin{aligned} \mathrm{Var}_T(\hat{\varepsilon}_{loo}) &= \frac{1}{n^2}(n\mathbf{E}_T[(\varepsilon_u - e)^2] + n(n-1)\mathbf{E}_T[(\varepsilon_u - e_i)(\varepsilon_u - e_j)]) \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{E}_T[(\varepsilon_u - e_i)(\varepsilon_u - e_j)] \end{aligned} \tag{3.9}$$

Additionally, we know that ε_u is the mean of the conditional errors ε_c over all datasets of size n. Assume for simplicity that the number of datasets is finite and equal to N. We can then write ε_u as the sum of the conditional error on the same dataset T_0 from which e_i and e_j were sampled, and a constant term which is the average of the conditional errors on the other samples:

$$\varepsilon_u = \mathbf{E}_T[\varepsilon_c] = \frac{1}{N}\sum \varepsilon_c = \varepsilon_c(T_0) + \frac{\sum_{T \setminus T_0} \varepsilon_c(T_i)}{N-1} = \varepsilon_c(T_0) + C.$$

If we plug this into formula 3.9, we find that the variance of $\hat{\varepsilon}_{loo}$ depends on the difference between the conditional error for a sample T_0 and the errors on the individual instances in that sample: $\varepsilon_c(T_0) - e_{i \in T_0}$. Actually, $\varepsilon_c(T_0)$ can be replaced by any term that is constant for a given sample. This can be another leave-one-out cross-validation error generated from T_0, or the leave-one-out cross-validation estimate $\hat{\varepsilon}_{loo}(L, T_0)$ itself; all of these are unbiased estimates of ε_u.

From this derivation we can conclude that for the variance of the leave-one-out cross-validation estimator to be small, all individual leave-one-out errors e_i on a given sample have to be as similar as possible. By definition, this is true when a learning problem is stable: The generated surrogate models will be similar to each other and to the original model. When we present the same instance to all these models, the surrogate predictions will be similar, and accordingly will the errors.

It is a pitfall to conclude from this that the variance of the leave-one-out errors on the sample that we have available, $\mathrm{Var}(e_i|T) = \mathbf{E}_i[(\mathbf{E}_i[e_i] - e_i)^2|T]$, is a measure for the stability of the learning problem. We cannot derive any information from this computation because part of the variability is caused by the differences of the test instances. What is needed is the errors of the surrogate models on the same test instance. Beleites and Salzer (2008) tried to obtain this by choosing k smaller than n, and using repeated cross-validation so that the error of different surrogate models can be compared on the same test instance.

If the stability conditions are satisfied, it is approximately correct to assume that all leave-one-out predictions are generated from the same model, one that is similar to the model build on the original dataset T. The confidence interval for the error of a binary classifier can then for instance be accurately approximated by the well-known formula $\hat{p} \pm z_\alpha \sqrt{\frac{1}{n}\hat{p}(1-\hat{p})}$, with $\hat{p}$ equal to the proportion of errors (Kearns and Ron 1999). Note that this is a well-known formula to compute a confidence interval for a classifier evaluated on an independent hold-out test set of n instances.

If the learning problem is unstable, computing $\mathrm{Var}(e_i|T)$ is not very insightful, because it is unclear what it represents; the leave-one-out errors are the result of predictions from different classifiers.

To summarize, for unstable learning problems, leave-one-out-cross-validation is an imprecise estimator for both the conditional and the unconditional error. For stable learning problems, however, leave-one-out-cross-validation or k-fold cross-validation with large k/n ratio can be used to estimate the conditional error. Unfortunately, it is not easy to know if a learning problem is stable.

3.4 Estimating the unconditional error with repeated cross-validation

In the previous section we established that cross-validation is an unbiased estimator for the unconditional error for training sets of size $n(k-1)/k$. It is often advocated to average over multiple cross-validations of the same sample to estimate the error of a learner with the goal of obtaining an estimator with lower variance. In this section we discuss whether this recommendation is indeed useful.

3.4.1 Variance decomposition

In the previous section, we introduced the concept of internal variance of the k-fold cross-validation estimator. We defined it as the variance of the cross-validation estimates over all possible partitionings π of a given dataset T.

It is often forgotten though that there is also another source of variance of the cross-validation estimator, namely the variance induced by the choice of the sample, or the *sample variance*. By applying the law of total variance, the total variance of the cross-validation estimator can be written as a combination of the internal and the sample variance:

$$\mathrm{Var}(\hat{\varepsilon}) = \mathrm{Var}_T(\mathbf{E}_\pi[\hat{\varepsilon}|T]) + \mathbf{E}_T[\mathrm{Var}_\pi(\hat{\varepsilon}|T)]. \tag{3.10}$$

Notice how the total variance is not simply the sum of the internal and the sample variance: two modifications are needed. First, because the internal variance may differ from sample to sample, we take the expected value of the internal variance over all possible samples. Second, to avoid taking the internal variance into account twice, the sample variance is computed for the expected value of the cross-validation estimator over all possible partitionings of a given sample T. This decomposition is not new, it has been mentioned in other works on cross-validation (Hanczar and Dougherty 2010; Nadeau and Bengio 1999).

For a more intuitive understanding, we illustrate the variance decomposition in Figure 3.1. The figure shows how different samples T can be drawn from a population p. On each sample the conditional error $\varepsilon_c(L,T)$ can be computed. This gives rise to the sample variance of ε_c. On the same sample we can also compute a cross-validation estimate for ε_c or ε_u. For this, multiple partitionings π into folds are possible, with each partitioning resulting in a different estimate $\hat{\varepsilon}(L,T,\pi)$. We say that the cross-validation estimator has internal variance. The sample variance of the cross-validation estimator measures how the expected

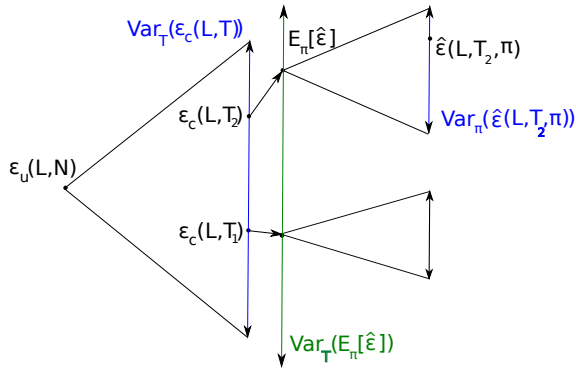


Figure 3.1: Illustration of the relationships between ε_c, ε_u, and the components of the mean squared error of the cross-validation estimator $\hat{\varepsilon}$ for these two population parameters.

value of the cross-validation estimator varies over all possible partitionings of a sample T. This quantity is not necessarily equal to the sample variance of ε_c. Finally, we also note that $\mathbf{E}_\pi[\hat{\varepsilon}(L,T)]$ is not necessarily equal to $\varepsilon_c(L,T)$; the estimator may be biased.

3.4.2 Variance estimation

It has been established in several experimental studies that repeated cross-validation indeed has smaller variance than a single cross-validation (Borra and Ciaccio 2010). As the number of repetitions over which we average the cross-validation estimates increases, the internal variance in fact converges to zero. Formula 3.10 then reduces to:

$$\mathrm{Var}(\hat{\varepsilon}) = \mathrm{Var}_T(\mathbf{E}_\pi[\hat{\varepsilon}|T]) \tag{3.11}$$

with the $\mathbf{E}_\pi[\hat{\varepsilon}|T]$ the repeated cross-validation estimate on a given sample T.

An estimator that has small variance has a number of advantages. Obviously, the obtained estimate will be more precise. But it also improves the replicability of the experiment: When setting up a new experiment on the same sample, the likelihood of obtaining the same or a similar result increases (R. Bouckaert 2004). In the end, this facilitates the reproducibility of scientific research.

The problem with repeated cross-validation is, however, that it is impossible to estimate the remaining term in formula 3.11 because we only have a single sample available. A common misconception is that the sample variance of the estimator can be approximated because several 'subsamples' are generated by the procedure. One can think of several methods to compute an estimate of the sample variance of the cross-validation estimator:

- The variance can be estimated based on the variance of the individual errors. For a binary classifier for instance, the formula $\frac{p(1-p)}{n}$ can be used, with p being the proportion of erroneous predicted instances.
- Another method is to first compute the average error for every fold, and afterwards compute the variance of these fold error estimates.
- Finally, we could use repeated cross-validation and compute the variance of the different cross-validation estimates.

One can probably think of other procedures to obtain a variance estimate, but these are all in vain: Bengio and Grandvalet (2004) proved theoretically that the variance of the cross-validation estimator cannot generally be estimated unbiasedly from a single sample Bengio and Grandvalet 2004. The main reason for this is that the generated training sets are not independent samples because they share instances. The estimation methods presented above would all lead to a liberal estimate of the variance, meaning they would all underestimate the total variance. The consequences are that confidence intervals will be too small, and the probability of making a type I error when applying a hypothesis test will be larger than the advertised significance level.

This issue is not solved by applying repeated cross-validation. In fact, this introduces even more dependencies because the test sets now also overlap[3]. We should add some nuance to the above discussion: There exist several variance estimators that are asymptotically unbiased as the sample size goes to infinity. The issue is therefore most urgent for small samples.

There is currently no consensus on a recommended method for estimating the variance of the cross-validation estimator. Consequently, there is also no recommended method for constructing a confidence interval for the learner error, or to compare learners with a hypothesis test. We briefly list a selection of proposals that have better statistical properties than the estimation methods listed above.

[3]In the appendix, we prove that when the number of repetitions goes to infinity, the formula for the variance of repeated cross-validation has the same structure as that of leave-one-out cross-validation.

In order to compare two learners, T. G. Dietterich (1998) recommended 5 times 2-fold cross-validation in combination with a modification of the paired t-test. Later, this test was fine-tuned by Alpaydm (1999). Bouckaert showed that Dietterich's test has low replicability. He argues that, ideally, the test's outcome should be independent of the sample partitioning. He proposes using ten times repeated tenfold cross-validation in combination with a t-test with adjusted degrees of freedom (R. R. Bouckaert 2003). Nadeau and Bengio propose a procedure to obtain an unbiased variance estimate for training sets of size $n/2$. However, the computation effort increases significantly in comparison to a single cross-validation, without a reduction of the variance. Grandvalet and Bengio (2006) propose a test that relies on a conservative estimate of the error correlations so that variance is always overestimated. Finally, Markatou et al. suggest a variance estimator that outperforms the estimator used by Grandvalet et al. by using additional information from the data and learning algorithm (Markatou et al. 2005).

3.5 Experiments

Our experiments try to answer the following questions:

- Does cross-validation estimate the conditional error, the unconditional error, both, or neither?
- When comparing two models learned on a specific dataset, and ignoring statistical testing, how often does cross-validation correctly identify the model with the smallest prediction error?

3.5.1 Does cross-validation estimate the conditional or unconditional error, or neither?

Our first experiment investigates whether the cross-validation estimator estimates the conditional error, the unconditional error, both, or neither. We do this by computing a cross-validation estimate $\hat{\varepsilon}(L, T)$ on a given dataset T of size 100, and comparing it to the true ε_c and ε_u. These last two would normally not be available to the researcher, but we circumvent this problem by using a very large dataset D as our population, so that we can compute all necessary quantities. The details of the experiment are described in algorithm 1:

The experiment is repeated for 100 different samples from D. ε_u is computed as the average of the conditional errors over all these samples. We follow

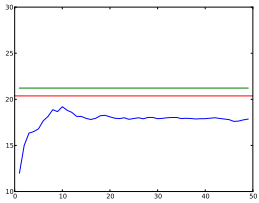
Algorithm 1 Experimental procedure

1: **Given:** A large dataset D (data population), a learner L, and a cross-validation estimator CV.
2: D is partitioned into a small dataset T of N instances and a large dataset $D \setminus T$.
3: We use T to:
4: Compute $\varepsilon_c(L, T)$ by learning a model on T and evaluating it on $D \setminus T$.
5: Compute a cross-validation estimate $\hat{\varepsilon}(L, T)$ and compare it to ε_c and ε_u.

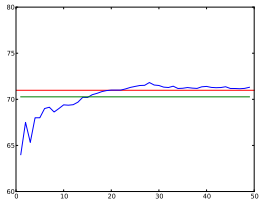
the procedure that is often used in real experiments: We compute $\hat{\varepsilon}(L, T)$ on a single dataset from the population. The only variability of the estimator therefore arises from the random partitioning of the dataset. Using repeated cross-validation instead of regular cross-validation decreases this variance. As we vary the number of cross-validation repetitions from 1 to 50, the estimator converges to an unknown value, which is hopefully ε_c or ε_u, but this is to be investigated.

As our data populations, we use the following datasets (# instances) chosen for their large size: Abalone (4177), adult (48842), king rook vs king (28056), mushroom (8124), and nursery (12960). The learning algorithms are: Naive Bayes (NB), nearest neighbors with 4 (4NN) and 10 neighbors (10NN), logistic regression (LR), the decision tree learner C4.5 (DT), and a Random Forest (RF). We also perform the experiment for a different number of folds of the cross-validation estimator, using 2-fold, 10-fold and 30-fold cross-validation. For every sample T from D we plot the model accuracy (blue) as computed by the repeated cross-validation estimator against the number of repetitions. The true conditional (green) and unconditional error (red) are also shown. Figure 3.2 presents a random selection of the results.

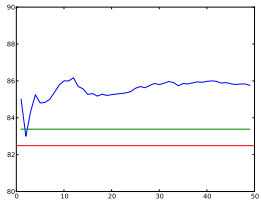
The results demonstrate that repeated cross-validation indeed decreases the internal variance $\mathrm{Var}_\pi(\hat{\varepsilon})$ so that the estimate lies closer to $\mathbf{E}_\pi[\hat{\varepsilon}_c]$. However, we also see that $\hat{\varepsilon}_{rcv}$ does not converge to the conditional error, nor to the unconditional error. The estimator clearly has a systematic deflection, $\mathbf{E}_\pi[\hat{\varepsilon}_c] - \varepsilon$, which is different for every problem. Although, averaging over ten to twenty repetitions reduces the deviation. This is not true in every setting: The tenfold cross-validation estimate obtained for C4.5 on nursery, for instance, diverges from both ε_c and ε_u. Another example is naive Bayes on nursery with 2-fold cross-validation. In this case, the estimator converges to the conditional error, but not the unconditional error.



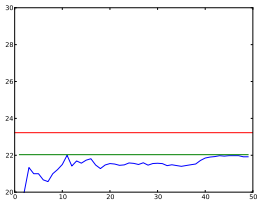
(a) 4NN on kr-vs-k, 2fcv



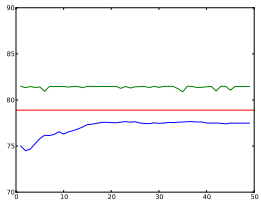
(b) 10NN on nursery, 2fcv



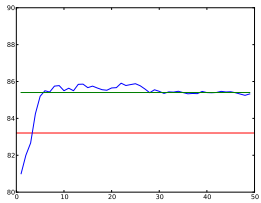
(c) C4.5 on nursery, 10fcv



(d) Random forest on kr-vs-k, 10fcv



(e) Logistic regression on adult, 10fcv



(f) Naive Bayes on nursery, 2fcv

Figure 3.2: The horizontal axis shows the number of cross-validation repetitions. The vertical axis shows the the repeated cross-validation estimator for accuracy (blue), the conditional error [accuracy] (green), and the unconditional error [accuracy] (red).

3.5.2 Comparing learners with cross-validation

In the previous experiment we established that the repeated cross-validation estimate computed on a single dataset does not consistently converge to $\hat{\varepsilon}_c$ and $\hat{\varepsilon}_u$ when increasing the number of repetitions. However, if the difference between the estimator and the parameter is similar for every learner, two learning algorithms can still be compared by means of cross-validation. This is investigated in our next experiment. We focus on estimating ε_c, but it would be interesting to extend these experiments to ε_u.

We again apply algorithm 1, but with one adjustment. Instead of one learner L, we apply step four and five on two learners L_1 and L_2, computing ε_c and $\hat{\varepsilon}$ for both learners. We perform these computations for both learners on the same datasets T with exactly the same settings for the cross-validation estimator. The resampling estimators are again 2-fold, 10-fold and 30-fold cross-validation, computed with 1 and 30 repetitions.

Based on the results for 100 samples from D, we construct a contingency table as follows, where we denote $\hat{\varepsilon}(L_i, T)$ as $\hat{\varepsilon}_i$ and $\varepsilon_c(L_i, T)$ as $\varepsilon_{c,i}$:

	$\varepsilon_{c,1} > \varepsilon_{c,2}$	$\varepsilon_{c,1} \leq \varepsilon_{c,2}$
$\hat{\varepsilon}_1 > \hat{\varepsilon}_2$		
$\hat{\varepsilon}_1 \leq \hat{\varepsilon}_2$		

Our results are presented in Tables 3.1, 3.2, and 3.3 at the back of this chapter[4]. As can be seen from these tables, 10-fold and 30-fold cross-validation outperform 2-fold cross-validation in detecting the winning model most often. Repeated cross-validation performs slightly better than regular cross-validation. This is consistent with the observations from our previous experiment, that repeated cross-validation often results in a more accurate estimate of ε_c and ε_u.

It is interesting to see that when one learner is not clearly better than the other, cross-validation has difficulty selecting the winning model. Consider for instance the comparison of Naive Bayes and C4.5 on adult in Table 3.1. $\varepsilon_c(C4.5)$ is smallest for more than half of the samples. However, for many samples where C4.5 wins, naive Bayes is selected by the cross-validation estimator as the winner. The opposite does not happen so often; on a sample where Naive Bayes wins, the cross-validation estimator often also selects Naive Bayes.

Let us focus on the case of 2-fold cross-validation for this problem, and compute the following conditional probabilities. By $L_1 > L_2$, we indicate that L_1 wins against L_2, i.e., the conditional error of L_1 is smaller than that of L_2.

- $P(NB > DT | CV_{NB} > CV_{DT}) = 25/62 = 0.4$
- $P(NB \leq DT | CV_{NB} > CV_{DT}) = 37/62 = 0.6$
- $P(NB > DT | CV_{NB} \leq CV_{DT}) = 7/38 = 0.18$
- $P(NB \leq DT | CV_{NB} \leq CV_{DT}) = 31/38 = 0.82$

From these estimated probabilities we see that when the cross-validation estimator indicates that naive Bayes has a smaller conditional error, this is only true in 40% of cases. When cross-validation indicates that C4.5 wins, however, we have 82% certainty that this is indeed true. The reason is perhaps that overall, C4.5 wins on most samples: 68 out of 100 samples. Therefore, changes made by the cross-validation estimator in the sample will most likely create a sample for which the decision tree wins.

[4]Because of the long runtime, 30-fold cross-validation on kr-vs-k is not included.

We can also compute the opposite conditional probabilities:

- $P(CV_{NB} > CV_{DT}|NB > DT) = 25/32 = 0.78$
- $P(CV_{NB} \leq CV_{DT}|NB > DT) = 7/32 = 0.22$
- $P(CV_{NB} > CV_{DT}|NB \leq DT) = 37/68 = 0.54$
- $P(CV_{NB} \leq CV_{DT}|NB \leq DT) = 31/68 = 0.56$

We see that when we select a sample on which we know naive Bayes wins, it is 78% certain that cross-validation will detect this. However, when we select a sample for which we know the decision tree wins, the cross-validation estimate is no better than a random guess (50% probability).

Another example is the comparison of naive Bayes and the random forest with 2-fold cross-validation on kr-vs-k (Table 3.1). Here, the random forest wins on more than half of the samples. The estimated conditional probabilities are as follows:

- $P(NB > RF|CV_{NB} > CV_{RF}) = 2/5 = 0.4$
- $P(NB \leq RF|CV_{NB} > CV_{RF}) = 3/5 = 0.6$
- $P(NB > RF|CV_{NB} \leq CV_{RF}) = 33/95 = 0.35$
- $P(NB \leq RF|CV_{NB} \leq CV_{RF}) = 62/95 = 0.65$

Again, on a sample where the random forest wins, the probability that the same conclusion is reached by the cross-validation estimator is larger than 0.5, while the opposite is true when naive Bayes wins.

- $P(CV_{NB} > CV_{RF}|NB > RF) = 2/40 = 0.05$
- $P(CV_{NB} \leq CV_{RF}|NB > RF) = 38/40 = 0.95$
- $P(CV_{NB} > CV_{RF}|NB \leq RF) = 3/65 = 0.05$
- $P(CV_{NB} \leq CV_{RF}|NB \leq RF) = 62/65 = 0.95$

Here, the results are even more extreme than in the previous example. Regardless of whether a sample is selected for which we know the random forest wins, or naive Bayes, the cross-validation estimator concludes with high probability that the random forest wins.

3.6 Conclusions

This chapter discussed a number of crucial points to take into account when estimating the error of a predictive model with cross-validation. It is motivated by the observation that, although being an essential task in machine learning research, there does not seem to be a consensus on how to perform this task.

A take-home message is that the researcher should always be clear on whether they are estimating the error of a *model*, i.e., the conditional error, or that of *learner*, i.e., the unconditional error. Estimating one or the other requires a different approach.

We saw for instance that cross-validation is not very suitable for estimating the error of a specific model, due to its pessimistic bias, which is most evident on small samples. Although leave-one-out cross-validation does not suffer from this bias, it is often also unreliable due to its large sample variance. We showed how this sample variance is a direct consequence of the instability of the generated surrogate models. While this variability is not always problematic, the circumstances under which it is not, are difficult to verify in practice.

In machine learning research, the relevant quantity is most often not the error of the specific model, but instead that of the learner. k-fold cross-validation is in principle suitable for this task, as it is an unbiased estimator for the learner error on training sets of size $n(k-1)/k$. Moreover, past experimental research has shown that its variance is acceptable when the samples are sufficiently large, and the number of folds is around ten. The estimate is, however, often presented together with a measure of its reliability, such as a confidence interval or an estimate of its variance. This is problematic because it is known that the sample variance is almost always underestimated when using a single sample. Currently, there is no consensus on the most optimal estimation method.

A common mistake is to estimate the sample variance by computing the variance of the individual errors, or that of the cross-validation errors in the individual folds. We made it clear that this *internal* variance is not a valid substitute for the sample variance. This can be seen by realizing that it converges to zero when the number of cross-validation repetitions over which is averaged increases. A performance difference between two learners can thus always be detected with (incorrect) statistical testing by using a sufficiently large number of repetitions. It should be noted though that repeated cross-validation is not a wasted computational effort. Because of the variance reduction, ten to twenty repetitions usually lead to a more reliable learner error estimate. It is a misconception, however, to believe that the estimate converges to the learner error when averaging over increasingly more repetitions. A systematic difference will remain due to bias and sample variance.

L_1, L_2	folds	2	10	30
NB-DT	adult	$\begin{bmatrix} 25 & 37 \\ 7 & 31 \end{bmatrix}$, $\begin{bmatrix} 14 & 48 \\ 1 & 37 \end{bmatrix}$	$\begin{bmatrix} 32 & 30 \\ 7 & 31 \end{bmatrix}$, $\begin{bmatrix} 28 & 34 \\ 5 & 33 \end{bmatrix}$	$\begin{bmatrix} 32 & 30 \\ 6 & 32 \end{bmatrix}$, $\begin{bmatrix} 37 & 25 \\ 4 & 34 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 8 & 8 \\ 36 & 48 \end{bmatrix}$, $\begin{bmatrix} 8 & 8 \\ 30 & 54 \end{bmatrix}$	$\begin{bmatrix} 8 & 8 \\ 29 & 55 \end{bmatrix}$, $\begin{bmatrix} 8 & 8 \\ 15 & 69 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 53 & 12 \\ 24 & 11 \end{bmatrix}$, $\begin{bmatrix} 61 & 4 \\ 27 & 8 \end{bmatrix}$	$\begin{bmatrix} 52 & 13 \\ 13 & 22 \end{bmatrix}$, $\begin{bmatrix} 52 & 13 \\ 11 & 24 \end{bmatrix}$	$\begin{bmatrix} 50 & 15 \\ 12 & 23 \end{bmatrix}$, $\begin{bmatrix} 49 & 16 \\ 9 & 26 \end{bmatrix}$
NB-4NN	adult	$\begin{bmatrix} 51 & 24 \\ 5 & 20 \end{bmatrix}$, $\begin{bmatrix} 50 & 25 \\ 3 & 22 \end{bmatrix}$	$\begin{bmatrix} 67 & 8 \\ 1 & 24 \end{bmatrix}$, $\begin{bmatrix} 69 & 6 \\ 1 & 24 \end{bmatrix}$	$\begin{bmatrix} 70 & 5 \\ 1 & 24 \end{bmatrix}$, $\begin{bmatrix} 71 & 4 \\ 1 & 24 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 22 & 15 \\ 30 & 33 \end{bmatrix}$, $\begin{bmatrix} 20 & 17 \\ 34 & 29 \end{bmatrix}$	$\begin{bmatrix} 19 & 18 \\ 20 & 43 \end{bmatrix}$, $\begin{bmatrix} 17 & 20 \\ 19 & 44 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 99 & 1 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 97 & 3 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 96 & 4 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 98 & 2 \\ 0 & 0 \end{bmatrix}$
NB-10NN	adult	$\begin{bmatrix} 34 & 34 \\ 6 & 26 \end{bmatrix}$, $\begin{bmatrix} 26 & 42 \\ 6 & 26 \end{bmatrix}$	$\begin{bmatrix} 47 & 21 \\ 3 & 29 \end{bmatrix}$, $\begin{bmatrix} 51 & 17 \\ 5 & 27 \end{bmatrix}$	$\begin{bmatrix} 59 & 9 \\ 4 & 28 \end{bmatrix}$, $\begin{bmatrix} 57 & 11 \\ 1 & 31 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 23 & 16 \\ 38 & 23 \end{bmatrix}$, $\begin{bmatrix} 20 & 19 \\ 36 & 25 \end{bmatrix}$	$\begin{bmatrix} 12 & 27 \\ 21 & 40 \end{bmatrix}$, $\begin{bmatrix} 15 & 24 \\ 18 & 43 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 99 & 0 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 99 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 95 & 4 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 97 & 2 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 94 & 5 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 95 & 4 \\ 1 & 0 \end{bmatrix}$
NB-LR	adult	$\begin{bmatrix} 9 & 25 \\ 15 & 51 \end{bmatrix}$, $\begin{bmatrix} 4 & 30 \\ 4 & 62 \end{bmatrix}$	$\begin{bmatrix} 13 & 21 \\ 12 & 54 \end{bmatrix}$, $\begin{bmatrix} 12 & 22 \\ 12 & 54 \end{bmatrix}$	$\begin{bmatrix} 15 & 19 \\ 13 & 53 \end{bmatrix}$, $\begin{bmatrix} 19 & 15 \\ 15 & 51 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 10 & 28 \\ 21 & 41 \end{bmatrix}$, $\begin{bmatrix} 10 & 28 \\ 16 & 46 \end{bmatrix}$	$\begin{bmatrix} 18 & 20 \\ 14 & 48 \end{bmatrix}$, $\begin{bmatrix} 16 & 22 \\ 11 & 51 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 27 & 4 \\ 55 & 14 \end{bmatrix}$, $\begin{bmatrix} 30 & 1 \\ 65 & 4 \end{bmatrix}$	$\begin{bmatrix} 15 & 16 \\ 23 & 46 \end{bmatrix}$, $\begin{bmatrix} 21 & 10 \\ 19 & 50 \end{bmatrix}$	$\begin{bmatrix} 14 & 17 \\ 16 & 53 \end{bmatrix}$, $\begin{bmatrix} 18 & 13 \\ 16 & 53 \end{bmatrix}$
NB-RF	adult	$\begin{bmatrix} 5 & 30 \\ 8 & 57 \end{bmatrix}$, $\begin{bmatrix} 2 & 33 \\ 4 & 61 \end{bmatrix}$	$\begin{bmatrix} 12 & 23 \\ 13 & 52 \end{bmatrix}$, $\begin{bmatrix} 8 & 27 \\ 9 & 56 \end{bmatrix}$	$\begin{bmatrix} 11 & 24 \\ 15 & 50 \end{bmatrix}$, $\begin{bmatrix} 14 & 21 \\ 11 & 54 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 2 & 3 \\ 33 & 62 \end{bmatrix}$, $\begin{bmatrix} 2 & 3 \\ 19 & 76 \end{bmatrix}$	$\begin{bmatrix} 3 & 2 \\ 20 & 75 \end{bmatrix}$, $\begin{bmatrix} 3 & 2 \\ 18 & 77 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 42 & 16 \\ 25 & 17 \end{bmatrix}$, $\begin{bmatrix} 52 & 6 \\ 32 & 10 \end{bmatrix}$	$\begin{bmatrix} 47 & 11 \\ 14 & 28 \end{bmatrix}$, $\begin{bmatrix} 48 & 10 \\ 12 & 30 \end{bmatrix}$	$\begin{bmatrix} 42 & 16 \\ 10 & 32 \end{bmatrix}$, $\begin{bmatrix} 48 & 10 \\ 10 & 32 \end{bmatrix}$

Table 3.1: Contingency tables for the comparison of naive bayes (NB) with a C4.5 decision tree (DT), 4 nearest neighbors (4NN), 10 nearest neighbors (10NN), logistic regression (LR), and a random forest (RF).

L_1, L_2	folds	2	10	30
DT-4NN	adult	$\begin{bmatrix}84 & 16\\ 0 & 0\end{bmatrix}, \begin{bmatrix}98 & 2\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}91 & 9\\ 0 & 0\end{bmatrix}, \begin{bmatrix}96 & 4\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}91 & 9\\ 0 & 0\end{bmatrix}, \begin{bmatrix}91 & 9\\ 0 & 0\end{bmatrix}$
	kr-vs-k	$\begin{bmatrix}50 & 41\\ 4 & 5\end{bmatrix}, \begin{bmatrix}56 & 35\\ 6 & 3\end{bmatrix}$	$\begin{bmatrix}55 & 36\\ 3 & 6\end{bmatrix}, \begin{bmatrix}55 & 36\\ 5 & 4\end{bmatrix}$	
	nursery	$\begin{bmatrix}95 & 5\\ 0 & 0\end{bmatrix}, \begin{bmatrix}100 & 0\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}95 & 5\\ 0 & 0\end{bmatrix}, \begin{bmatrix}100 & 0\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}96 & 4\\ 0 & 0\end{bmatrix}, \begin{bmatrix}99 & 1\\ 0 & 0\end{bmatrix}$
DT-10NN	adult	$\begin{bmatrix}50 & 38\\ 4 & 8\end{bmatrix}, \begin{bmatrix}60 & 28\\ 3 & 9\end{bmatrix}$	$\begin{bmatrix}64 & 24\\ 6 & 6\end{bmatrix}, \begin{bmatrix}64 & 24\\ 6 & 6\end{bmatrix}$	$\begin{bmatrix}62 & 26\\ 5 & 7\end{bmatrix}, \begin{bmatrix}61 & 27\\ 6 & 6\end{bmatrix}$
	kr-vs-k	$\begin{bmatrix}57 & 33\\ 6 & 4\end{bmatrix}, \begin{bmatrix}54 & 36\\ 5 & 5\end{bmatrix}$	$\begin{bmatrix}45 & 45\\ 6 & 4\end{bmatrix}, \begin{bmatrix}44 & 46\\ 6 & 4\end{bmatrix}$	
	nursery	$\begin{bmatrix}96 & 4\\ 0 & 0\end{bmatrix}, \begin{bmatrix}100 & 0\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}93 & 7\\ 0 & 0\end{bmatrix}, \begin{bmatrix}99 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}94 & 6\\ 0 & 0\end{bmatrix}, \begin{bmatrix}98 & 2\\ 0 & 0\end{bmatrix}$
DT-LR	adult	$\begin{bmatrix}1 & 7\\ 28 & 64\end{bmatrix}, \begin{bmatrix}0 & 8\\ 21 & 71\end{bmatrix}$	$\begin{bmatrix}1 & 7\\ 31 & 61\end{bmatrix}, \begin{bmatrix}2 & 6\\ 26 & 66\end{bmatrix}$	$\begin{bmatrix}4 & 4\\ 26 & 66\end{bmatrix}, \begin{bmatrix}3 & 5\\ 26 & 66\end{bmatrix}$
	kr-vs-k	$\begin{bmatrix}33 & 50\\ 6 & 11\end{bmatrix}, \begin{bmatrix}23 & 60\\ 7 & 10\end{bmatrix}$	$\begin{bmatrix}37 & 46\\ 9 & 8\end{bmatrix}, \begin{bmatrix}36 & 47\\ 8 & 9\end{bmatrix}$	
	nursery	$\begin{bmatrix}4 & 5\\ 39 & 52\end{bmatrix}, \begin{bmatrix}6 & 3\\ 40 & 51\end{bmatrix}$	$\begin{bmatrix}4 & 5\\ 15 & 76\end{bmatrix}, \begin{bmatrix}4 & 5\\ 16 & 75\end{bmatrix}$	$\begin{bmatrix}3 & 6\\ 17 & 74\end{bmatrix}, \begin{bmatrix}5 & 4\\ 22 & 69\end{bmatrix}$
DT-RF	adult	$\begin{bmatrix}2 & 5\\ 14 & 79\end{bmatrix}, \begin{bmatrix}0 & 7\\ 2 & 91\end{bmatrix}$	$\begin{bmatrix}2 & 5\\ 29 & 64\end{bmatrix}, \begin{bmatrix}0 & 7\\ 15 & 78\end{bmatrix}$	$\begin{bmatrix}2 & 5\\ 24 & 69\end{bmatrix}, \begin{bmatrix}1 & 6\\ 20 & 73\end{bmatrix}$
	kr-vs-k	$\begin{bmatrix}11 & 18\\ 28 & 43\end{bmatrix}, \begin{bmatrix}6 & 23\\ 8 & 63\end{bmatrix}$	$\begin{bmatrix}13 & 16\\ 29 & 42\end{bmatrix}, \begin{bmatrix}9 & 20\\ 21 & 50\end{bmatrix}$	
	nursery	$\begin{bmatrix}7 & 26\\ 24 & 43\end{bmatrix}, \begin{bmatrix}5 & 28\\ 7 & 60\end{bmatrix}$	$\begin{bmatrix}13 & 20\\ 24 & 43\end{bmatrix}, \begin{bmatrix}14 & 19\\ 15 & 52\end{bmatrix}$	$\begin{bmatrix}15 & 18\\ 23 & 44\end{bmatrix}, \begin{bmatrix}18 & 15\\ 22 & 45\end{bmatrix}$

Table 3.2: Contingency tables for the comparison of a decision tree (DT) with 4 nearest neighbors (4NN), 10 nearest neighbors (10NN), logistic regression (LR), and a random forest (RF).

L_1, L_2	folds	2	10	30
4NN-10NN	adult	$\begin{bmatrix} 0 & 1 \\ 7 & 92 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 0 & 99 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 11 & 88 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 7 & 92 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 7 & 92 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 8 & 91 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 32 & 30 \\ 21 & 17 \end{bmatrix}$, $\begin{bmatrix} 33 & 29 \\ 21 & 17 \end{bmatrix}$	$\begin{bmatrix} 27 & 35 \\ 18 & 20 \end{bmatrix}$, $\begin{bmatrix} 24 & 38 \\ 12 & 26 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 10 & 15 \\ 31 & 44 \end{bmatrix}$, $\begin{bmatrix} 15 & 10 \\ 30 & 45 \end{bmatrix}$	$\begin{bmatrix} 11 & 14 \\ 23 & 52 \end{bmatrix}$, $\begin{bmatrix} 11 & 14 \\ 23 & 52 \end{bmatrix}$	$\begin{bmatrix} 13 & 12 \\ 22 & 53 \end{bmatrix}$, $\begin{bmatrix} 12 & 13 \\ 24 & 51 \end{bmatrix}$
4NN-LR	adult	$\begin{bmatrix} 0 & 0 \\ 8 & 92 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 2 & 98 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 1 & 99 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 3 & 97 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 3 & 97 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 18 & 25 \\ 15 & 42 \end{bmatrix}$, $\begin{bmatrix} 19 & 24 \\ 9 & 48 \end{bmatrix}$	$\begin{bmatrix} 21 & 22 \\ 21 & 36 \end{bmatrix}$, $\begin{bmatrix} 22 & 21 \\ 16 & 41 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 0 & 0 \\ 2 & 98 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$
4NN-RF	adult	$\begin{bmatrix} 0 & 0 \\ 2 & 98 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 3 & 97 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 1 & 99 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 3 & 97 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 2 & 98 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 0 & 1 \\ 30 & 69 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 25 & 74 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 32 & 67 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 26 & 73 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 0 & 0 \\ 1 & 99 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$
10NN-LR	adult	$\begin{bmatrix} 1 & 1 \\ 25 & 73 \end{bmatrix}$, $\begin{bmatrix} 2 & 0 \\ 18 & 80 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 \\ 13 & 85 \end{bmatrix}$, $\begin{bmatrix} 2 & 0 \\ 8 & 90 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 \\ 16 & 82 \end{bmatrix}$, $\begin{bmatrix} 2 & 0 \\ 13 & 85 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 8 & 30 \\ 20 & 42 \end{bmatrix}$, $\begin{bmatrix} 13 & 25 \\ 10 & 52 \end{bmatrix}$	$\begin{bmatrix} 21 & 17 \\ 25 & 37 \end{bmatrix}$, $\begin{bmatrix} 24 & 14 \\ 21 & 41 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$
10NN-RF	adult	$\begin{bmatrix} 0 & 1 \\ 19 & 80 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 14 & 85 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 11 & 88 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 10 & 89 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 18 & 81 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 16 & 83 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 0 & 0 \\ 25 & 75 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 18 & 82 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 43 & 57 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 36 & 64 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 0 & 0 \\ 1 & 99 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 2 & 98 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 2 & 98 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$
LR-RF	adult	$\begin{bmatrix} 18 & 37 \\ 15 & 30 \end{bmatrix}$, $\begin{bmatrix} 14 & 41 \\ 19 & 26 \end{bmatrix}$	$\begin{bmatrix} 25 & 30 \\ 24 & 21 \end{bmatrix}$, $\begin{bmatrix} 24 & 31 \\ 26 & 19 \end{bmatrix}$	$\begin{bmatrix} 28 & 27 \\ 25 & 20 \end{bmatrix}$, $\begin{bmatrix} 28 & 27 \\ 28 & 17 \end{bmatrix}$
	kr-vs-k	$\begin{bmatrix} 2 & 2 \\ 47 & 49 \end{bmatrix}$, $\begin{bmatrix} 2 & 2 \\ 43 & 53 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 \\ 40 & 56 \end{bmatrix}$, $\begin{bmatrix} 2 & 2 \\ 32 & 64 \end{bmatrix}$	
	nursery	$\begin{bmatrix} 28 & 55 \\ 5 & 12 \end{bmatrix}$, $\begin{bmatrix} 24 & 59 \\ 3 & 14 \end{bmatrix}$	$\begin{bmatrix} 61 & 22 \\ 9 & 8 \end{bmatrix}$, $\begin{bmatrix} 66 & 17 \\ 12 & 5 \end{bmatrix}$	$\begin{bmatrix} 57 & 26 \\ 12 & 5 \end{bmatrix}$, $\begin{bmatrix} 70 & 13 \\ 12 & 5 \end{bmatrix}$

Table 3.3: Contingency tables for 4 nearest neighbors (4NN), 10 nearest neighbors (10NN), logistic regression (LR), and a random forest (RF).

Chapter 4

Bag- versus instance-level performance in multi-instance learning

This chapter analyzes the relationship between instance-level and bag-level accuracy of multi-instance algorithms. We demonstrate both theoretically and empirically that these two performance measures are not strongly correlated. This research has been published in the Data Mining and Knowledge Discovery journal: (Vanwinckelen, Tragante Do O, et al. 2014), which is an extension of the work presented at the Benelux Conference on Artificial Intelligence (Tragante do O et al. 2011).

4.1 Introduction

In their seminal paper on multi-instance learning, T. Dietterich et al. (1997) define the multi-instance learning task as follows: We are given a set of bags of instances. Each instance has an unknown label (positive or negative). Each bag has a known label, which is positive if and only if at least one instance in the bag is positive, and negative otherwise. The task is to learn, from this information, a classifier that can predict the label of instances, and hence also of bags.

Dietterich et al.'s article has given rise to a variety of learning approaches, as

well as some theoretical work on the problem. Although many papers refer to Dietterich et al's problem definition, the term "multi-instance learning" has gradually become used in a broader sense, namely, any kind of learning in a context where a single example is a bag of instances. The requirements that a label is associated with each instance , and that the label of a bag is positive if and only if it contains at least one positive instance, are then dropped. Some authors refer to this broader setting as "generalized multi-instance learning" as opposed to Dietterich et al.'s "standard multi-instance learning". In this chapter, when we refer to multi-instance learning, we refer to the "standard" setting.

Within standard multi-instance learning, there are two interpretations of the problem; one is that the task is to learn a classifier for bags, the other that the task is to learn a classifier for single instances. In the standard setting, these two are equivalent, so one could argue it does not matter which interpretation is used. But this is not entirely the case: as we will argue further on, it is possible that, among two classifiers, the one with best performance for bag classification has worst performance for instance classification, and vice versa.

Although Dietterich et al. explicitly stated that they wanted to learn an instance classifier, their evaluation is based on bag classification. They do not mention the reason for this, but at least two reasons are plausible. First, from a *predictive* point of view, bag classification performance is indeed relevant for the problem they focused on (classifying molecules as having the "musk" property or not), even if, from the point of view of *knowledge discovery*, the task is to identify the conditions C under which an instance is positive (which makes instance classification performance more relevant).

The second reason is more pragmatic: in the Musk datasets, the true instance labels are not known, only the bag labels are. That is, the correct solution to the multi-instance learning problem is not known, and hence cannot be compared to. This contrasts with the normal evaluation procedures used in machine learning: there, the labels of test instances are not given to the learner, but they are known to the researcher, and used to evaluate the learner's performance. In multi-instance learning, when individual instance labels are not known, a comparably reliable evaluation procedure is not available. Evaluation based on bag classification may then be the best available alternative.

Up till now, however, it has not been investigated how valid bag classification evaluation is, as an approximation to instance classification evaluation. That is, when model A is shown to have better bag classification performance on some dataset than model B, is it reasonable to assume it will also have better instance classification performance? Many papers implicitly make this assumption.

In this chapter, we evaluate that assumption for two performance measures:

Accuracy and Area Under the ROC Curve (AUC). We show that instance classification accuracy is not a monotonic function of bag classification accuracy. When ranking multi-instance classifiers according to one or the other, the correlation between them is positive, but substantially lower than one. Our experiments show that the same holds for AUC. We quantify the probability of choosing a suboptimal model when bag classification performance is substituted for instance classification performance. We also discuss what this entails in practice for some published results. Our main conclusion is that research on multi-instance classification would benefit from datasets with known instance labels, allowing for more accurate evaluation.

The chapter is structured as follows. We first review the basics of multi-instance learning and briefly discuss the mathematical relationship between instance- and bag-level classification performance (Section 4.2). Next, we discuss the literature on multi-instance learning and emphasize the different interpretations of the learning task being used, and the link with the two different evaluation settings (Section 4.3). This section primarily focuses on accuracy as a performance measure. Next, we empirically analyze the relationship between bag-level performance and instance-level performance for both accuracy and AUC (Section 4.4). Finally, we empirically assess the benefit of special-purpose multi-instance learners versus reduction to regular single-instance learning, and compare this to earlier results (Section 4.5).

4.2 Multi-instance learning: Preliminaries

4.2.1 Definition and terminology

Let $\mathcal{X}$ be the instance space and $\mathbb{B} = \{pos, neg\}$ the binary set of class labels. Standard binary classification, which we here call the *single-instance* setting, can be defined as follows. We are given a dataset D consisting of elements $(\mathbf{x}_i, f(\mathbf{x}_i))$ with $\mathbf{x}_i \in \mathcal{X}$ an instance and $f(\mathbf{x}_i) \in \mathbb{B}$ its label according to an unknown function $f : \mathcal{X} \to \mathbb{B}$. The learning task is to find the function f.

The *multi-instance* learning setting is typically defined as follows (T. Dietterich et al. 1997). We are given a dataset that consists of bags B_i of instances; the number of instances in a bag is variable. Each instance is described by a single vector $\mathbf{x}_{ij} \in \mathcal{X}$. An instance can be positive or negative, but the instance labels are not given. Instead, we are given bag labels, and we know that a bag is labeled positive if it contains at least one positive instance, and negative otherwise. From this information, we are to learn a function that can classify instances or bags.

The first interpretation of the task is that an instance classifier is learned. This corresponds to the following definition.

Definition 12 (Multi-instance learning, interpretation 1)**.**
Given: a dataset D consisting of elements $(B_i, F(B_i))$ with B_i a bag of instances $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$ and $F(B_i) \in \mathbb{B}$ its label; there exists an unknown function $f : \mathcal{X} \to \mathbb{B}$ for which it holds that

$$F(B) = \begin{cases} pos & \text{if } \exists x \in B : f(x) = pos, \\ neg & \text{otherwise.} \end{cases} \tag{4.1}$$

Find: f.

The second interpretation of the task is identical except that now the bag labeling function F is to be learned.

Definition 13 (Multi-instance learning, interpretation 2)**.**
Given: the same information as in Definition 12.
Find: F.

Both definitions are used in the literature (we present an overview later on), and both are interpretations of what we call "standard" multi-instance learning.

As said, there is also "generalized" multi-instance learning (M. Zhang 2009). Here, the task is to learn F, but contrary to standard multi-instance learning, Equation 4.1 need not hold, and the notion of an instance classifier f may not even exist. For that reason, this analysis is relevant only for standard multi-instance learning.

4.2.2 Connection between f and F

Equation 4.1 shows that f uniquely determines F. The converse is also true: given F, it must hold that

$$f(x) = F(\{x\}). \tag{4.2}$$

This follows formally from Definition 12: Since Equation 4.1 holds for all bags, it holds also for singletons, and for a singleton $\{e\}$ we have $F(e) = pos \leftrightarrow \exists x \in B : f(x) = pos \Leftrightarrow f(e) = pos$.

Let $\mathcal{B}$ be the function that transforms an instance classifier into an equivalent bag classifier; that is, $F = \mathcal{B}(f)$ if and only if Equation 4.1 holds. Similarly, let $\mathcal{I}$ be the function that transforms a bag classifier F into an instance classifier f; that is, $f = \mathcal{I}(F)$ if and only if Equation 4.2 holds. In standard multi-instance

learning, the transformations $\mathcal{I}$ and $\mathcal{B}$ are guaranteed to be each other's inverse: $\mathcal{I}(\mathcal{B}(f)) = f$ and $\mathcal{B}(\mathcal{I}(F)) = F$.

Equation 4.1 is well-known to all multi-instance learning researchers, but the fact that Equation 4.2 trivially follows from it does not seem to have been given much consideration. It has been used implicitly in some cases; for instance, Zhou, Xue, et al. (2005) propose an algorithm for detecting regions of interest in images (Citation-ROI) that is really an application of the above rule for a particular multi-instance learner (Citation-kNN), though not presented as such.

4.2.3 Instance-level versus bag-level accuracy

From now on, we distinguish the real target concepts underlying the learning problem, denoted f and F as before, from the actual classifiers constructed by a multi-instance learner when applied on a dataset, denoted $\hat{f}$ and $\hat{F}$. Some multi-instance learners return an instance classifier $\hat{f}$; for these, we define the corresponding bag classifier as $\hat{F} = \mathcal{B}(\hat{f})$. Other multi-instance classifiers return a bag classifier; for these, we define $\hat{f} = \mathcal{I}(\hat{F})$. For the real target concepts, we have $f = \mathcal{I}(F)$ and $F = \mathcal{B}(f)$ as before.

We call a classifier *perfect* when it has zero error. Any method that learns a perfect bag classifier ($\hat{F} = F$) learns a perfect instance classifier ($\hat{f} = \mathcal{I}(\hat{F}) = \mathcal{I}(F) = f$), and vice versa. However, when learning from data, we typically do not find perfect classifiers; the learned function $\hat{f}$ (or $\hat{F}$) only approximates f (or F).

We can distinguish two accuracy measures for a classifier $(\hat{f}, \hat{F})$ on a given multi-instance dataset.

- *Bag-level accuracy* a_B is the proportion of bags for which the predicted label equals the true label. It measures how well $\hat{F}$ approximates F.
- *Instance-level accuracy* a_I is the proportion of instances for which the predicted label equals the true label. It measures how well $\hat{f}$ approximates f.

Given the one-to-one relationship between F and f (Equations 4.1 and 4.2), one might expect that there is a similar link between the corresponding accuracy measures, a_B and a_I. This is not the case, as we show next.

4.2.4 Mathematical analysis of the relationship between bag-level and instance-level accuracy

Under the simplifying assumption that instances in a bag are independent from each other, we can analyze the relationship between a_I and a_B analytically, showing that there is no one-to-one correspondence between them in this special case (and hence, not in the general case either).

Specifically, we consider the following generative process: a random number of instances are drawn randomly and i.i.d., and put in a bag; the bag is labeled positive if and only if at least one of these instances is positive. The dataset consists of a number of bags, all generated in this way. The instance distribution and bag size distribution can be any distribution.

Accuracy is a weighted average of the true positive rate TP (the proportion of positive cases that are predicted positive) and the true negative rate TN (the proportion of negative cases predicted negative). Let TP_I and TN_I be the instance-level true positive/negative rate of $\hat{f}$ over the whole instance population. The expected instance-level accuracy of $\hat{f}$ on a set of N_I instances, among which p positives and n negatives, is then

$$a_I = \frac{p \cdot TP_I + n \cdot TN_I}{N_I}. \tag{4.3}$$

The expected bag-level accuracy a_B of the corresponding $\hat{F}$ on a set of N_B bags, the elements of which are drawn i.i.d. from that population, is

$$a_B = \frac{\sum_{B_k \in \mathbf{B}^+} a_B^+(B_k) + \sum_{B_k \in \mathbf{B}^-} a_B^-(B_k)}{N_B} \tag{4.4}$$

$$a_B^-(B_k) = TN_I^{n_k}, \tag{4.5}$$

$$a_B^+(B_k) = 1 - TN_I^{n_k}(1 - TP_I)^{p_k} \tag{4.6}$$

with $\mathbf{B}^+$ and $\mathbf{B}^-$ the set of positive and negative bags, respectively, and with p_k and n_k the number of positive/negative instances in bag B_k. Equation 4.4 writes the expected bag-level accuracy as an average of the expected classification accuracy of positive and negative bags. Equation 4.5 follows from the fact that a negative bag is predicted correctly if all its instances are correctly predicted negative. Equation 4.6 follows from the fact that a positive bag is predicted correctly if it is not predicted as negative; and a bag is predicted negative when all its negative members are correctly predicted negative (probability $TN_I^{n_k}$), and all its positive members are incorrectly predicted negative (probability $(1 - TP_I)^{p_k}$).

Table 4.1: The instance- and bag-level accuracy of two classifiers, C_1 (TN_I=0.7, TP_I=0.2) and C_2 (TN_I=0.2, TP_I=0.9), for two different bag configurations.

	C_1	C_2
a_I	$\frac{6}{7} \cdot 0.2 + \frac{1}{7} \cdot 0.9 = 0.3$	$\frac{6}{7} \cdot 0.7 + \frac{1}{7} \cdot 0.2 = 0.63$
$a_{B,\text{cnf.1}}$	$\frac{1}{2}(0.7^5 + 1 - 0.7^1(1-0.2)^1) = 0.30$	$\frac{1}{2}(0.2^5 + 1 - 0.2^1(1-0.9)^1) = 0.50$
$a_{B,\text{cnf.2}}$	$\frac{1}{2}(0.7^1 + 1 - 0.7^5(1-0.2)^1) = 0.78$	$\frac{1}{2}(0.2^1 + 1 - 0.2^5(1-0.9)^1) = 0.60$

Together, Equations 4.3 to 4.6 show that, for a given a_I, there is not a single corresponding value of a_B (and vice versa): this value depends on the size of the bags B_k and the instance class distribution, which are problem-dependent, and on TP_I and TN_I, which are classifier-dependent. Hence, even on the same problem, different classifiers may exhibit a different relationship between bag-level and instance-level accuracy. Among two classifiers, one classifier may score best on one measure and worst on the other.

This is demonstrated by the following example. Consider applying two classifiers C_1 and C_2 on a dataset that is balanced on the bag level. A fraction of 1/7 of the instances are positive and 6/7 are negative. Classifier C_1 has $TN_{I,1} = 0.2$ and $TP_{I,1} = 0.9$, and C_2 has $TN_{I,2} = 0.7$ and $TP_{I,2} = 0.2$. Using Equation 4.3, we find that the instance-level accuracy of the two classifiers is respectively $a_{I,1} = 0.3$ and $a_{I,2} = 0.63$. We create two different bag configurations from these instances. In the first configuration, negative bags contain five instances and positive bags contain one positive and one negative instance. In the second configuration, negative bags contain a single instance, and positive bags contain five negative and one positive instance. Applying Equations 4.3 to 4.6 for both bag configurations, Table 4.1 shows that depending on the configuration, either $a_{B,1}$ is larger than $a_{B,2}$ or vice versa.

The discussion also shows that a_I is in a sense more 'stable' than a_B. If a model is found to have a particular a_B on some test set, it may actually achieve a significantly different bag accuracy when predicting labels of bags that are significantly larger or smaller. (As an extreme example, if in a test set all bags are singletons, a_B becomes equal to a_I). Note also that, given a class distribution and a bag size distribution, TN_I and TP_I determine the expected values of both a_I and a_B. From this point of view, they are more fundamental than a_I and a_B.

Figure 4.1 illustrates the relationship between a_I and a_B on 500 randomly

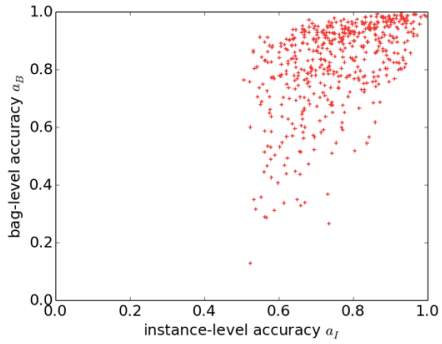


Figure 4.1: Scatter plot showing for 500 problems with a randomly chosen TP_I, TN_I, class distribution and bag size distribution, the corresponding a_I and a_B. a_I and a_B correlate positively, but do not determine each other.

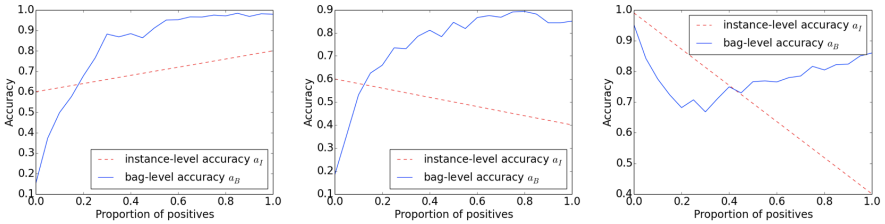


Figure 4.2: Increasing the proportion of positive instances in the instance population from 0 to 1 may affect both a_I and a_B positively (left; $TP_I = 0.8$, $TN_I = 0.6$), affect one positively and the other negatively (middle: $TP_I = 0.4$, $TN_I = 0.6$), or have a more complicated effect (right; $TP_I = 0.4$, $TN_I = 0.99$).

generated datasets.[1] The simulation shows clearly that a_I and a_B correlate positively but do not determine each other.

Figure 4.2 shows how the class distribution affects a_I and a_B, for a fixed TN_I and TP_I. Depending on the value of TN_I and TP_I, more positives in the instance population may increase both a_I and a_B (left), increase one but decrease the other (middle), or have a more complicated effect (right; here, a maximal a_B is obtained when a_I is either very high or very low).

[1]For each dataset, the size of a bag is between 1 and the maximum bag size for that dataset, which is randomly drawn from [3,10]. For each dataset, the maximum proportion of positives in a bag is drawn from [0,1]. Finally TN_I and TP_I are drawn from [0.5,1]

4.3 Literature on (standard) multi-instance learning

There is a large literature on multi-instance learning. This section is not meant to provide a complete overview, but merely to illustrate the points being made in this work regarding variation in learning tasks and evaluation measures.

4.3.1 Algorithms and applications

Many algorithms for (standard) multi-instance learning have been developed. Perhaps the simplest approach is to label each instance with the label of its bag, then apply a standard learner to this dataset to obtain $\hat{f}$. Many instances will get a positive label in the training set even if they are really negative, but the opposite will not occur; thus, multi-instance learning reduces to learning from a dataset with one-sided class noise (Blum and Kalai 1998). Such an approach was used as a "straw man" approach by T. Dietterich et al. (1997), and shown not to work well, although Ray and Craven (2005) later showed that this may depend strongly on the application.

Many authors have proposed algorithms that explicitly address the peculiarities of the multi-instance setting, such as Axis-Parallel Rectangles (T. Dietterich et al. 1997), Diverse Density (Maron and Lozano-Pérez 1998) and its extension EM-DD (Q. Zhang and Goldman 2001), neural networks (Ramon and De Raedt 2000), the k-nearest neighbor algorithm Citation-kNN (Wang and Zucker 2000), the decision tree algorithms ID3-MI (Zucker and Chevaleyre 2001) and MITI (Blockeel et al. 2005), the rule learning algorithm RIPPER-MI (Zucker and Chevaleyre 2001), the SVM algorithms MI-SVM, mi-SVM (Andrews et al. 2003) and DD-SVM (Chen and Wang 2004), the logistic regression algorithm MILR (Ray and Craven 2005) and the ensemble algorithms MI-Ensemble (Zhou and M. Zhang 2003) and MI-Boosting (Xu and Frank 2004). As the overview shows, many of these algorithms are inspired by single-instance learning algorithms.

Applications of multi-instance learning include content-based image retrieval (Andrews et al. 2003; Fu et al. 2011; Li et al. 2009; Maron and Ratan 1998; Zhou, Xue, et al. 2005), music retrieval (Mandel and Ellis 2008), protein family modeling (Tao et al. 2004) and medical applications (Fung et al. 2007) such as drug activity prediction (T. Dietterich et al. 1997). In image retrieval, many researchers have proposed the use of multi-instance learners to learn what the user's interest is. That is, given a series of pictures that are of interest to the user, presumably because they contain some object of interest (among other things), a multi-instance learner can be used to learn a function that can predict

for new pictures whether they contain the object of interest or not. Some researchers consider as a separate task the detection of the regions of interest in these pictures (Fu et al. 2011; Li et al. 2009; Zhou, Xue, et al. 2005). Using our terminology, this boils down to identifying f.

4.3.2 Learning task: Definition 12 versus Definition 13

According to Definition 12, the goal is to learn an instance classifier, i.e., an approximation $\hat{f}$ to f. According to Definition 13, the goal is to learn a bag classifier, i.e., an approximation $\hat{F}$ to F. Both interpretations are used in the literature:

- In their seminal paper about multi-instance learning, T. Dietterich et al. (1997) (p.34) state: "the goal of the machine learning algorithm will be to construct an approximation $\hat{g}$ to the *internal* function g", which in our notation is function f. This means that Dietterich et al use Definition 12. This definition is also used by Blum and Kalai (1998), who note that multi-instance learning is a special case of learning from one-sided class noise. Some theoretical results regarding PAC learning axis-parallel rectangles in this setting (Auer, Long, et al. 1998; Blum and Kalai 1998; Long and Tan 1998) also use Definition 12, as do various other papers (e.g., (Blockeel et al. 2005; Maron and Lozano-Pérez 1998)).
- The Encyclopedia of Machine Learning (Ray, Scott, et al. 2011) uses Definition 13. Many papers use this definition as well (e.g., (Fung et al. 2007; Maron and Ratan 1998; Wang and Zucker 2000; Q. Zhang and Goldman 2001; Zucker and Chevaleyre 2001)).

It seems that the use of these two different interpretations of the learning task can be explained historically. When T. Dietterich et al. (1997) introduced multi-instance learning, they considered Definition 12, i.e., learn f. In subsequent papers, there seems to be a shift in emphasis from the *learning task* to the *data format*. That is, many multi-instance papers clearly specify the format of the training data but leave it implicit whether the goal is to learn f or F. Furthermore, multi-instance classifiers are typically evaluated in terms of bag-level performance, which relates to F rather than f. This seems to have contributed to the second interpretation of the learning task (learn F) becoming more popular in recent papers.

4.3.3 Performance measure: Bag-level versus instance-level

Most papers evaluate classifiers according to bag level performance, regardless of whether Definition 12 or 13 is used. This is true for Dietterich et al, and for the large majority of cases we looked at (Andrews et al. 2003; Blockeel et al. 2005; T. Dietterich et al. 1997; Fung et al. 2007; Maron and Lozano-Pérez 1998; Ramon and De Raedt 2000; Ray and Craven 2005; Tao et al. 2004; Wang and Zucker 2000; Xu and Frank 2004; Q. Zhang and Goldman 2001; Zhou and M. Zhang 2003; Zucker and Chevaleyre 2001). The reason for this is most likely that computing a_I requires knowing the instance labels, which in most datasets are unknown. Exceptions are one paper on music information retrieval by Mandel and Ellis (2008), who explicitly state that they are interested only in a_I; a number of papers on content-based image retrieval (CBIR); and a recent theoretical study by Doran and Ray (2014).

In CBIR, two different tasks are often considered: (a) retrieving images that contain a particular type of object, and (b) identifying the region in the picture where that object occurs; this is also called ROI (region-of-interest) detection (Fu et al. 2011; Li et al. 2009; Zhou, Xue, et al. 2005). Both learning problems can be seen as multi-instance learning, with images as bags and the different regions or segments as instances. Bag-level performance is then more relevant for task (a), instance-level performance for task (b), though for task (b) an alternative measure called *success rate* is often used: The classifier indicates for every bag predicted positive one instance that is most likely to be positive; the success rate is the proportion of positive bags for which a truly positive instance is indicated. Both forms of instance-level evaluation can only be automated if individual instance labels are available, which is often not the case; for this reason, several researchers (Fu et al. 2011; Shao et al. 2008) evaluate the ROI performance not quantitatively but only *qualitatively*, by manually inspecting detected ROIs for a small subset of all images.

Doran and Ray (2014) define a number of desirable but mutually exclusive properties of MI approaches based on support vector machines, and study how these affect performance; they find that the effect differs depending on whether one considers bag classification or instance classification. Their results are consistent with ours.

In summary, with a few exceptions as mentioned above, the large majority of the literature on multi-instance learning considers bag-level evaluation only. Given that in some applications instance-level performance is important too, this raises the question: what does bag-level performance tell us about instance level performance? Or, more concretely in the case of accuracy: to what extent does a high bag-level accuracy a_B imply a high instance-level accuracy a_I?

This question has not been addressed in the multi-instance literature, where a_I and a_B are often implicitly assumed highly correlated, or, on the contrary, no relationship whatsoever is assumed (viewing ROI detection and CBIR as separate tasks). We address this question in the next section, and do the same for AUC.

4.4 Experimental analysis of the relationship

As shown in Sections 4.2.4 and 4.3, a_I and a_B cannot be expressed as a function of each other, let alone a monotonic function, and yet multi-instance classifiers are often evaluated based on bag-level performance, even when instance classification is desired. The question is then, how relevant such an evaluation is in practice. We investigate this empirically. While our theoretical analysis only covered accuracy as a performance measure, our empirical analysis also includes a comparison between instance-level AUC_I and bag-level AUC_B. Multi-instance datasets are typically imbalanced on the instance-level, with many more negative instances than positive ones, so in these cases AUC may be a more appropriate performance measure than accuracy.[2]

The first question we wish to address is: *When comparing multiple classifiers on the same dataset, to what extent does a high bag-level accuracy imply a high instance-level accuracy, and vice versa?* We will quantify this in two ways. First, the rank correlation $\rho(a_I, a_B)$ between the instance-level and bag-level accuracy of multiple classifiers on a given dataset can be computed. Second, the probability that a classifier scores higher on instance-level accuracy, given that it scores higher on bag-level accuracy, $P(a_I(C_1) > a_I(C_2) | a_B(C_1) > a_B(C_2))$, with C_i randomly chosen classifiers. While the first is a standard measurement for comparing rankings, the second is of practical importance: When we want to select, among two classifiers, the one with highest a_I, but instead choose the one with highest a_B (or vice versa), what is the probability that we have chosen the correct one?

The comparison of multiple classifiers on the same dataset is of practical importance because it is frequently encountered in experimental evaluation. However, in order to understand the influence of the choice of the learner and the characteristics of the dataset on the relationship between bag- and instance-level performance, we ask a second question: *When evaluating a particular learner on multiple datasets, to what extent does a high bag-level accuracy imply a high instance-level accuracy, and vice versa?* We explore this question by

[2]Another solution to this problem is to use weighted accuracy: $0.5(\text{TP} + \text{TN})$. We also computed this performance measure; this did not give substantially different results.

constructing a scatter plot for each learner, plotting instance-level performance versus bag-level performance.

4.4.1 Experimental setup

Datasets.

In order to evaluate instance-level performance, we need multi-instance datasets that include not only the bag labels, as usual, but also the true instance labels. Most of the common benchmark multi-instance (MI) datasets, such as Musk (T. Dietterich et al. 1997), only contain bag labels and hence cannot be used for our purpose. MI datasets with known instance labels seem to be rare and are often not publicly available. Partly for that reason, we use a mix of real-world and semi-synthetic datasets. The latter are datasets that are based on a real-world single-instance dataset, but turned into a multi-instance dataset by grouping instances into bags and assigning labels to these bags according to the standard MI assumption.

Real-world: SIVAL. The SIVAL repository (Settles et al. 2008)[3] is from the area of content-based image retrieval (CBIR). It contains 1500 images where each image contains one out of 25 different complex objects. The images are partitioned into 31 or 32 segments, i.e., the instances. An instance in this dataset consists of 30 features expressing color and texture information about the image segment and its four closest neighbors. The repository contains 25 multi-instance datasets where in turn each of the 25 objects is considered positive, while the other 24 objects are considered negative. Each of the 25 constructed datasets sets consists of 60 positive and 60 negative bags. In positive bags, the percentage of positive instances varies from 3.1% to 90.6%, with an average of 25.5%.

Semi-synthetic: Text categorization. We use 20 MI text categorization datasets extracted by Settles et al. (2008) from the 20 newsgroups corpus.[4] Each dataset contains 100 bags and is balanced on the bag level. The size of the bags (number of instances) varies from 8 to 84, with an average of 40. A bag is a collection of short texts from the newsgroups. In positive bags, the percentage of positive instances varies from 2% to 7%, with an average of 3.6%. This is a high-dimensional dataset; each instance is characterized by 200 TFIDF features. The bags were artificially created by in turn considering one newsgroup as positive, while taking the other newsgroups as negative and i.i.d.

[3]Data available on `http://pages.cs.wisc.edu/~bsettles/amil/`.
[4]Data available on `http://lamda.nju.edu.cn/data_MItext.ashx`.

sampling texts from the newsgroups such that around 3% of the texts in a bag are from the positive category.

Semi-synthetic UCI datasets. We have constructed some more semi-synthetic MI datasets from five source datasets taken from the UCI repository (Lichman 2013): Adult (Kohavi 1996), Pima Indians Diabetes (Smith et al. 1988), Spam (Cranor and LaMacchia 1998), Tic-Tac-Toe (Aha 1990) and Blood Transfusion Service Center (Yeh et al. 2009). We selected these datasets because they are imbalanced, which is useful for constructing MI datasets, since MI datasets that are balanced on the bag-level contain more negative than positive instances. All datasets are binary classification problems; we kept the labels unchanged, except for Tic-Tac-Toe, where we inverted the instance labels in order to have a majority of negatives.

We have constructed multiple MI datasets by choosing instances i.i.d. and grouping them into bags, controlling for bag size and positive/negative ratio in positive bags. We say that a MI dataset is in bag-configuration 'X/Y' if each bag in the dataset contains Y instances, and each positive bag contains X positive instances. For each source dataset, we use MI datasets in configurations 1/2, 1/3 and 2/3; for the two largest (Adult and Spam) we additionally use 1/4, 2/4, 1/5, 2/5, 1/10 and 2/10. All MI datasets are balanced on the bag level: 50% of the bags is positive and 50% is negative. To construct a MI dataset in a particular bag-configuration, we randomly sampled the required number of positive instances and negative instances from the respective source dataset. The number of bags in each MI dataset is the highest possible number for which we can sample without replacement before exhausting the source dataset, except for Adult, where for computational reasons we retained only 1200 bags for each label, randomly chosen.

Table 4.2 gives an overview of all the resulting MI datasets. The UCI datasets are complementary to the SIVAL and newsgroup datasets in the sense that they have large training set sizes, few features and relatively small bag sizes.

Learning algorithms.

We have performed experiments with fourteen MI algorithms available in the Weka data mining tool (I. Witten and Frank 2005): MIDD (Diverse Density) (Maron and Lozano-Pérez 1998), MIEMDD (Expectation-Maximization Diverse Density) (Q. Zhang and Goldman 2001), MDD (Modified Diverse Density with collective assumption) (Dong 2006), MISVM, which is a Weka implementation of the maximum pattern margin formulation mi-SVM by (Andrews et al. 2003), MIOptimalBall (Auer and Ortner 2004), MILR (Logistic Regression) (Ray and Craven 2005), logistic regression with the arithmetic mean model, referred

Table 4.2: Multi-instance dataset properties: Number of bags, number of instances per bag and percentage of positive instances per bag. The percentage of positive bags is 50% for all datasets. ('Ad.': Adult, 'TTT': Tic-Tac-Toe, 'Db.': Diabetes, 'Sp.': Spam, 'Tr.': Transfusion)

Dataset	#bags	$\frac{\#\text{inst.}}{\text{bag}}$	$\frac{\%\text{pos.}}{\text{bag}}$	Dataset	#bags	$\frac{\#\text{inst.}}{\text{bag}}$	$\frac{\%\text{pos.}}{\text{bag}}$
Ad. 1/2	1200	2	50	Sp. 1/2	1858	2	50
Ad. 1/3	1200	3	33.3	Sp. 1/3	1114	3	33.3
Ad. 1/4	1200	4	25	Sp. 1/4	796	4	25
Ad. 1/5	1200	5	20	Sp. 1/5	618	5	20
Ad. 1/10	1200	10	10	Sp. 1/10	292	10	10
Ad. 2/3	1200	3	66.6	Sp. 2/3	1392	3	66.6
Ad. 2/4	1200	4	50	Sp. 2/4	928	4	50
Ad. 2/5	1200	5	40	Sp. 2/5	696	5	40
Ad. 2/10	1200	10	20	Sp. 2/10	308	10	20
TTT 1/2	416	2	50	Tr. 1/2	356	2	50
TTT 1/3	250	3	33.3	Tr. 1/3	226	3	33.3
TTT 2/3	312	3	66.6	Tr. 2/3	178	3	66.6
Db. 1/2	332	2	50	SIVAL	1500	[31, 32]	[3.1, 90.6]
Db. 1/3	198	3	33.3	Text	100	[8, 84]	[2, 7]
Db. 2/3	248	3	66.6				

to as MILRC from now on (Dong 2006), MIRI (Bjerring and Frank 2011), AdaBoost.M1 (Freund and Schapire 1995) with a Multi Instance Tree Inducer (MITI) as base classifier (Blockeel et al. 2005), Citation-kNN (Wang and Zucker 2000), TLD (Two-Level Distribution) (Xu 2003), SimpleMI with the J48 classifier (Dong 2006), MIWrapper with the J48 classifier (Frank and Xu 2003), and MISMO, which is a Weka implementation of the normalized set kernel (NSK) by Gärtner et al. (2002).

The parameter settings are as follows. MISMO uses the radial basis function (RBF) kernel with γ equal to 0.01 and the regularization parameter C equal to 1. MISVM uses the linear kernel with the regularization parameter C being 1. We also ran MISVM with the RBF kernel but found that this kernel did not lead to good performance on the newsgroup and SIVAL datasets. For MILR, the ridge coefficient equals 10^{-6}. For Citation-kNN the number of citations and references both equal 5.

Section 4.5 also compares a number of single instance algorithms to the corresponding multi-instance variants. Table 4.9 gives an overview of the corresponding algorithms. For logistic regression the ridge coefficient equals 10^{-6}. For Nearest neighbors the number of neighbors is 5. We also apply SMO

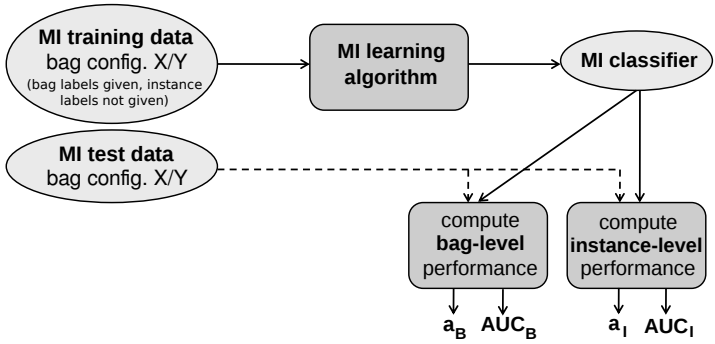


Figure 4.3: Overview of the experimental setup.

with an RBF kernel with γ and C equal to respectively 0.01 and 1. Finally, for J48 the confidence factor is 0.25 and the minimum number of instances per leaf 2.

Note that the algorithmic performances could be improved with parameter optimization. However, this would not influence the observed relative differences between instance-level and bag-level results. Due to computational limitations, we therefore limited the experiments to default parameter values.

Setup.

Figure 4.3 summarizes the experimental setup. We measure the bag-level performance and instance-level performance of each MI learning algorithm on each MI dataset with accuracy (respectively a_B and a_I) and AUC (AUC_B and AUC_I), using 10-fold bag-level stratified cross-validation.

4.4.2 Results

Rank correlations.

First, we have computed for each dataset the Spearman rank correlation $\rho(a_I, a_B)$ between instance- and bag-level accuracy of the different methods:

$$\rho(a_I, a_B) = \frac{\sum_j (r_I^j - \bar{r}_I)(r_B^j - \bar{r}_B)}{\sqrt{\sum_j (r_I^j - \bar{r}_I)^2 \sum_j (r_B^j - \bar{r}_B)^2}},$$

where r_I^j is the rank of the j-th learning algorithm in terms of instance-level accuracy ($j = 1 \ldots 10$), $\bar{r}_I$ is $\sum_{j=1}^{10} r_I^j/14$, and r_B^j and $\bar{r}_B$ are defined similarly in terms of bag-level accuracy. The Spearman correlation is 1 if a_I and a_B yield exactly the same ranking, -1 if they yield exactly opposite rankings, and 0 for independent rankings. The same formula can be used for AUC.

For each data category (SIVAL, newsgroup or UCI), we show three representative results. For evaluation in terms of accuracy, these are shown respectively in Figures 4.4, 4.5 and 4.6. For each dataset, a plot visualizes the agreement between the instance-level and bag-level ranking: For the 14 learners, the a_I and a_B they resulted in (for that particular dataset) are ranked, and each a_I is connected with the corresponding a_B. The extent to which the lines cross is a visual indication of the disagreement between the two measures. Overall, the results show that, while there is clearly some correlation between a_I and a_B, it is relatively weak. It happens quite often that a learner performs well in terms of a_B but poorly in terms of a_I, and vice versa (many crossing lines).

Figures 4.7, 4.8 and 4.9 show the results of the similar analysis for AUC. Here we observe that the rank correlations are highest for the UCI datasets and lowest for the SIVAL datasets. For the UCI datasets, $\rho(\mathrm{AUC}_I, \mathrm{AUC}_B)$ is often close enough to one to consider AUC_I a good approximation of AUC_B, and vice versa. For the newsgroup and SIVAL datasets this is not the case.

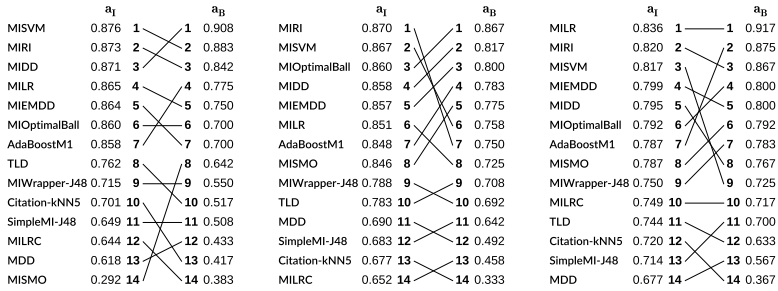
The results for the remaining datasets are similar and are available upon request to the authors.

Probabilistic analysis.

As an alternative to the use of rank correlations, we also perform a probabilistic analysis. We compute a 95% Wilson confidence interval for $P(a_I(C_1) > a_I(C_2) | a_B(C_1) > a_B(C_2))$, i.e., the probability that classifier C_1 outperforms classifier C_2 in terms of instance-level accuracy, given that it outperforms it in terms of bag-level accuracy. We use $P_{I|B}$ as shorthand for this probability. Following the usual way of estimating conditional probabilities, $P_{I|B}$ is estimated as

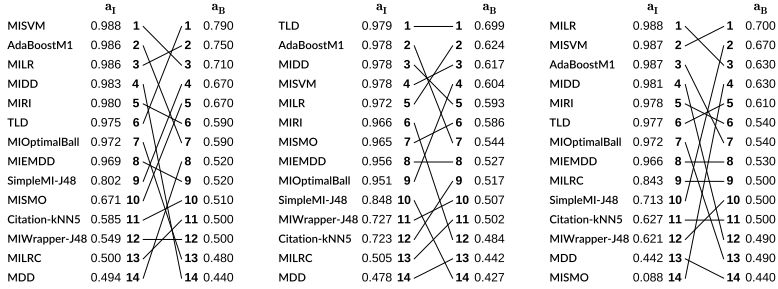
$$\hat{P}_{I|B} = \frac{\sum_{C_1} \sum_{C_2 \neq C_1} \mathbb{I}\big(a_I(C_1) > a_I(C_2) \;\;\&\;\; a_B(C_1) > a_B(C_2)\big)}{\sum_{C_1} \sum_{C_2 \neq C_1} \mathbb{I}\big(a_B(C_1) > a_B(C_2)\big)},$$

where the sums range over the different classifiers learned by our 14 learners on the considered dataset, and $\mathbb{I}(c)$ is 1 if c is true and 0 otherwise. Intuitively, $P_{I|B}$ indicates how well observations in terms of a_B are expected to transfer to a_I. The converse probability, $P_{B|I} = P(a_B(C_1) > a_B(C_2) | a_I(C_1) > a_I(C_2))$, is estimated similarly. Similar formulas can also be used for AUC.



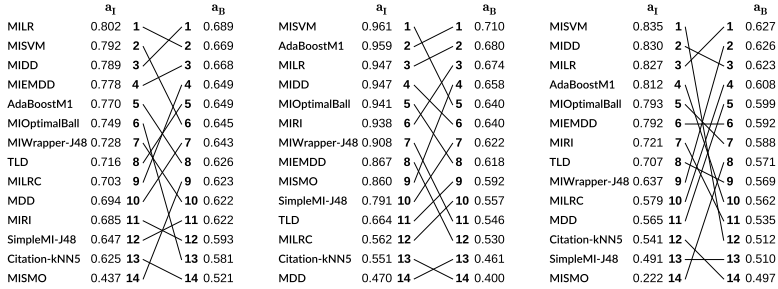
(a) ρ=0.84 (b) ρ=0.80 (c) ρ=0.79

Figure 4.4: *Accuracy Rankings on SIVAL data:* instance-level accuracy a_I versus bag-level accuracy a_B for the SIVAL datasets. ρ is the resulting rank correlation. (a) ajaxorange, (b) wd40can, (c) greenteabox.



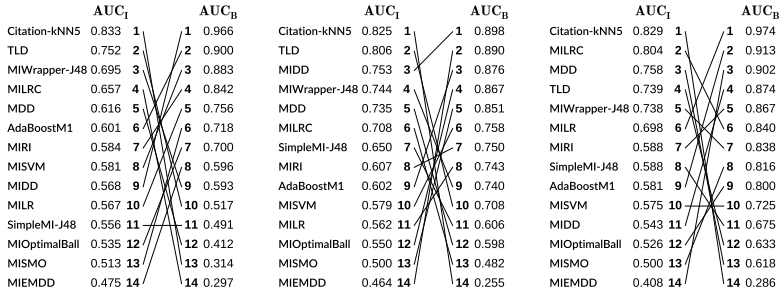
(a) ρ=0.41 (b) ρ=0.73 (c) ρ=0.31

Figure 4.5: *Accuracy Rankings on newsgroup data:* (a) alt atheism, (b) comp graphics, (c) comp os ms-windows misc.



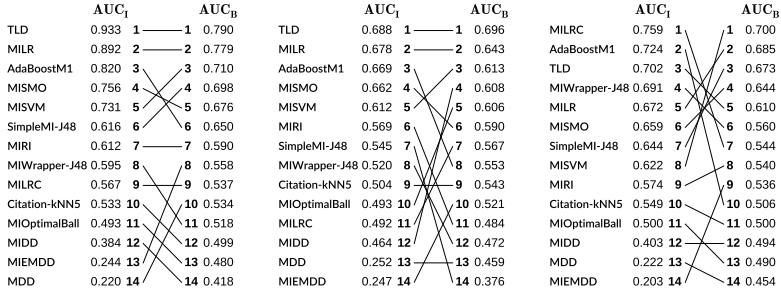
(a) ρ=0.64 (b) ρ=0.73 (c) ρ=0.24

Figure 4.6: *Accuracy rankings on UCI data:* (a) Diabetes 1/2, (b) Spam 1/10, (c) Transfusion 1/3.



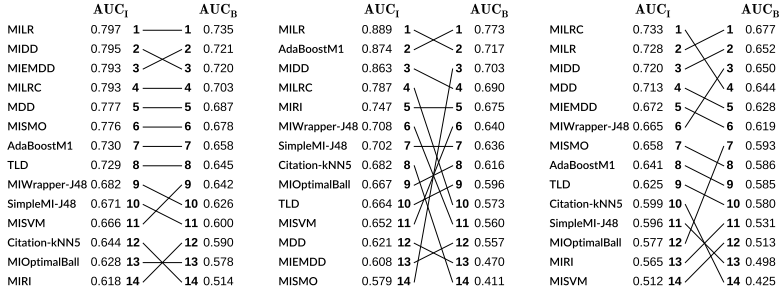
(a) ρ=-0.41 (b) ρ=-0.56 (c) ρ=-0.43

Figure 4.7: *AUC Rankings on SIVAL data:* (a) ajaxorange, (b) wd40can, (c) greenteabox.



(a) ρ=0.82 (b) ρ=0.46 (c) ρ=0.52

Figure 4.8: *AUC Rankings on newsgroup data:* (a) alt atheism, (b) comp graphics, (c) comp os ms-windows misc.



(a) ρ=0.96 (b) ρ=0.45 (c) ρ=0.83

Figure 4.9: *AUC rankings on UCI data:* (a) Diabetes 1/2, (b) Spam 1/10, (c) Transfusion 1/3.

Table 4.3: *Probabilistic analysis on SIVAL for accuracy.* 95% confidence intervals for the probability that a_I is higher, given that a_B is higher ($\hat{P}_{I|B}$).

Dataset	$\hat{P}_{I\|B}$	Dataset	$\hat{P}_{I\|B}$
ajaxorange	$[0.74, 0.90]$	dataminingbook	$[0.75, 0.90]$
apple	$[0.47, 0.67]$	rapbook	$[0.40, 0.60]$
banana	$[0.50, 0.70]$	feltflowerrug	$[0.63, 0.81]$
bluescrunge	$[0.59, 0.78]$	translucentbowl	$[0.64, 0.82]$
dirtyworkgloves	$[0.55, 0.74]$	greenteabox	$[0.72, 0.88]$
juliespot	$[0.59, 0.78]$	cardboardbox	$[0.57, 0.76]$
checkeredscarf	$[0.49, 0.69]$	dirtyrunningshoe	$[0.27, 0.47]$
wd40can	$[0.73, 0.89]$	largespoon	$[0.38, 0.58]$
candlewithholder	$[0.71, 0.87]$	goldmedal	$[0.58, 0.77]$
glazedwoodpot	$[0.45, 0.66]$	spritecan	$[0.67, 0.84]$
cokecan	$[0.76, 0.91]$	stripednotebook	$[0.70, 0.86]$
smileyfacedoll	$[0.77, 0.91]$	fabricsoftenerbox	$[0.69, 0.86]$

Note that when no two classifiers have the same rank, the denominator equals 0.5 and $P_{I|B}$ and $P_{B|I}$ are identical. In fact, $\hat{P}_{I|B}$ and $\hat{P}_{B|I}$ are nearly identical for all datasets. Therefore, we only show the confidence intervals for $\hat{P}_{I|B}$.

Tables 4.3, 4.4, and 4.5 show the results for accuracy, and Tables 4.6, 4.7, and 4.8 for AUC. Note that if a_I and a_B were unrelated to each other, we would have $\hat{P}_{I|B} = \hat{P}_{B|I} = 0.5$, while if a_I and a_B always agreed with each other, we would have $\hat{P}_{I|B} = \hat{P}_{B|I} = 1.0$. This means that the probabilities that we measured indicate that there is some agreement between a_I and a_B, but it is far from consistent. Choosing the best bag classifier (among two given classifiers) quite often does not yield the best instance classifier, and vice versa. These conclusions are consistent with our previous rank analysis, and quantify the practical implications.

4.4.3 Experimental analysis of the relationship between bag level and instance level accuracy over multiple datasets

This section addresses the following question: Given a multi-instance algorithm, how do instance-level and bag-level performance correlate with each other over multiple datasets. By studying this relationship for a specific learner, we learn about the effect of the inductive bias of that learner on the correlation between instance- and bag-level performance.

Table 4.4: *Probabilistic analysis on newsgroup for accuracy.* 95% confidence intervals for the probability that a_I is higher, given that a_B is higher ($\hat{P}_{I|B}$).

Dataset	$\hat{P}_{I\|B}$	Dataset	$\hat{P}_{I\|B}$
alt atheism	$[0.53, 0.73]$	rec sport hockey	$[0.77, 0.91]$
comp graphics	$[0.66, 0.83]$	sci crypt	$[0.56, 0.75]$
comp os ms-windows misc	$[0.53, 0.73]$	sci electronics	$[0.60, 0.79]$
comp sys ibm pc hardware	$[0.56, 0.75]$	sci med	$[0.65, 0.82]$
comp sys mac hardware	$[0.63, 0.81]$	sci space	$[0.70, 0.86]$
comp windows x	$[0.67, 0.84]$	soc religion christian	$[0.56, 0.75]$
misc forsale	$[0.43, 0.63]$	talk politics guns	$[0.52, 0.72]$
rec autos	$[0.70, 0.87]$	talk politics mideast	$[0.62, 0.80]$
rec motorcycles	$[0.65, 0.82]$	talk politics misc	$[0.39, 0.59]$
rec sport baseball	$[0.64, 0.82]$	talk religion misc	$[0.44, 0.64]$

Table 4.5: *Probabilistic analysis on UCI data for accuracy.* 95% confidence intervals for the probability that a_I is higher, given that a_B is higher ($\hat{P}_{I|B}$).

Dataset	$\hat{P}_{I\|B}$	Dataset	$\hat{P}_{I\|B}$
Adult 1/2	$[0.73, 0.89]$	Spam 2/3	$[0.70, 0.86]$
Adult 1/3	$[0.70, 0.86]$	Spam 1/4	$[0.75, 0.90]$
Adult 2/3	$[0.63, 0.81]$	Spam 2/4	$[0.77, 0.91]$
Adult 1/4	$[0.64, 0.82]$	Spam 1/5	$[0.80, 0.93]$
Adult 2/4	$[0.64, 0.82]$	Spam 2/5	$[0.71, 0.87]$
Adult 1/5	$[0.60, 0.79]$	Spam 1/10	$[0.71, 0.87]$
Adult 2/5	$[0.65, 0.83]$	Spam 2/10	$[0.71, 0.87]$
Adult 1/10	$[0.57, 0.76]$	Tictactoe 1/2	$[0.68, 0.85]$
Adult 2/10	$[0.61, 0.80]$	Tictactoe 1/3	$[0.68, 0.85]$
Diabetes 1/2	$[0.64, 0.82]$	Tictactoe 2/3	$[0.72, 0.88]$
Diabetes 1/3	$[0.49, 0.69]$	Transfusion 1/2	$[0.45, 0.65]$
Diabetes 2/3	$[0.63, 0.81]$	Transfusion 1/3	$[0.47, 0.67]$
Spam 1/2	$[0.77, 0.91]$	Transfusion 2/3	$[0.44, 0.65]$
Spam 1/3	$[0.77, 0.91]$		

Algorithms

We first briefly introduce how each algorithm learns a classifier. We partition the algorithms in two groups, inspired by the taxonomies introduced by Amores (2013) and Foulds and Frank (2010).

The first group of algorithms, which we will call *standard MI algorithms*, learn an instance classifier f and follow the standard MI assumption, meaning they learn an instance classifier that classifies a bag as positive if at least one of the

Table 4.6: *Probabilistic analysis on SIVAL for AUC.* 95% confidence intervals for the probability that AUC_I is higher, given that AUC_B is higher ($\hat{P}_{I|B}$).

Dataset	$\hat{P}_{I\|B}$	Dataset	$\hat{P}_{I\|B}$
ajaxorange	$[0.29, 0.49]$	dataminingbook	$[0.52, 0.72]$
apple	$[0.37, 0.57]$	rapbook	$[0.57, 0.76]$
banana	$[0.34, 0.54]$	feltflowerrug	$[0.30, 0.50]$
bluescrunge	$[0.42, 0.62]$	translucentbowl	$[0.46, 0.66]$
dirtyworkgloves	$[0.37, 0.57]$	greenteabox	$[0.24, 0.43]$
juliespot	$[0.52, 0.72]$	cardboardbox	$[0.34, 0.54]$
checkeredscarf	$[0.48, 0.68]$	dirtyrunningshoe	$[0.45, 0.65]$
wd40can	$[0.19, 0.37]$	largespoon	$[0.35, 0.55]$
candlewithholder	$[0.37, 0.57]$	goldmedal	$[0.44, 0.64]$
glazedwoodpot	$[0.47, 0.67]$	spritecan	$[0.29, 0.49]$
cokecan	$[0.25, 0.44]$	stripednotebook	$[0.45, 0.65]$
smileyfacedoll	$[0.34, 0.54]$	fabricsoftenerbox	$[0.15, 0.32]$

Table 4.7: *Probabilistic analysis on newsgroup for AUC.* 95% confidence intervals for the probability that AUC_I is higher, given that AUC_B is higher ($\hat{P}_{I|B}$).

Dataset	$\hat{P}_{I\|B}$	Dataset	$\hat{P}_{I\|B}$
alt atheism	$[0.75, 0.90]$	rec sport hockey	$[0.66, 0.83]$
comp graphics	$[0.56, 0.75]$	sci crypt	$[0.77, 0.91]$
comp os ms-windows misc	$[0.56, 0.75]$	sci electronics	$[0.65, 0.83]$
comp sys ibm pc hardware	$[0.64, 0.82]$	sci med	$[0.77, 0.91]$
comp sys mac hardware	$[0.72, 0.88]$	sci space	$[0.81, 0.94]$
comp windows x	$[0.82, 0.95]$	soc religion christian	$[0.73, 0.89]$
misc forsale	$[0.80, 0.93]$	talk politics guns	$[0.71, 0.87]$
rec autos	$[0.67, 0.84]$	talk politics mideast	$[0.73, 0.89]$
rec motorcycles	$[0.80, 0.93]$	talk politics misc	$[0.67, 0.84]$
rec sport baseball	$[0.82, 0.95]$	talk religion misc	$[0.70, 0.86]$

instances is classified as positive. No further information about the properties of the bag as a whole is used. This group consists of the following seven algorithms: MIDD, MIEMDD, MISVM, MIOptimalBall, MILR, MIRI, and AdaBoost.M1-MITI.

The remaining seven algorithms depart from the standard MI assumption in different ways. While different subcategories can be defined to group them, for our purposes it is sufficient to label them as *non-standard MI algorithms.* MIWrapper-J48 transforms the multi-instance problem into a single-instance problem by applying the bag label to every instance. It assumes that every instance in a bag contributes equally to the bag label, contrary to the standard

Table 4.8: *Probabilistic analysis on UCI data for AUC.* 95% confidence intervals for the probability that AUC_I is higher, given that AUC_B is higher ($\hat{P}_{I|B}$).

Dataset	$\hat{P}_{I\|B}$	Dataset	$\hat{P}_{I\|B}$
Adult 1/2	[0.81, 0.94]	Spam 2/3	[0.72, 0.88]
Adult 1/3	[0.71, 0.87]	Spam 1/4	[0.61, 0.80]
Adult 2/3	[0.80, 0.93]	Spam 2/4	[0.61, 0.80]
Adult 1/4	[0.76, 0.91]	Spam 1/5	[0.58, 0.77]
Adult 2/4	[0.78, 0.92]	Spam 2/5	[0.59, 0.78]
Adult 1/5	[0.73, 0.89]	Spam 1/10	[0.57, 0.76]
Adult 2/5	[0.72, 0.88]	Spam 2/10	[0.51, 0.71]
Adult 1/10	[0.63, 0.81]	Tictactoe 1/2	[0.78, 0.92]
Adult 2/10	[0.64, 0.82]	Tictactoe 1/3	[0.76, 0.91]
Diabetes 1/2	[0.86, 0.97]	Tictactoe 2/3	[0.86, 0.97]
Diabetes 1/3	[0.72, 0.88]	Transfusion 1/2	[0.80, 0.93]
Diabetes 2/3	[0.76, 0.91]	Transfusion 1/3	[0.75, 0.90]
Spam 1/2	[0.75, 0.90]	Transfusion 2/3	[0.80, 0.93]
Spam 1/3	[0.68, 0.85]		

MI assumption, where a single positive instance makes a bag positive. SimpleMI-J48, MILRC, and MDD map the instances in a bag to a single feature vector by averaging each feature over these instances. Learning a classifier is based on aggregate information about a bag, instead of on the individual instances. Finally, Citation-kNN, MISMO, and TLD use knowledge about the bag as a whole to learn a classifier. Citation-kNN and MISMO use respectively a distance or kernel function defined over the *bag*, and TLD learns a classifier from both the distributional properties of the instances as well as the bags.

While this work focuses on the standard MI problem, we also included algorithms that do not follow this assumption. One might argue that these algorithms are not designed to classify instances. However, a distinction should be made between the problem definition (standard MI learning) and the solving technique (the learner). As our experiments show, some of the non-standard MI algorithms are indeed suitable for classifying instances and sometimes even outperform the standard MI algorithms on this task. An additional argument for including these algorithms is that researchers may not always be aware of the assumptions an algorithm makes and may compare different types of algorithms. Our experiments thus reflect a realistic situation.

Results

The theoretical relationship between a_I and a_B was demonstrated in Figure 4.1 in section 4.2.4 with a scatter plot showing a_I and a_B on a set of random artificial MI problems. We now experimentally construct similar scatter plots for each of our fourteen learners, indicating for each individual result the category of the dataset: SIVAL (red square), newsgroup (green circle), or UCI (blue triangle). Figure 4.10 shows the results measured in accuracy and Figure 4.11 in AUC. We summarize our most important findings.

First, we see that *the domain of the dataset is an important factor for determining both instance- and bag-level performance.* In all figures, the results are clearly clustered according to category.

Second, we see that, *despite assuming the existence of instance labels, standard MI algorithms do not always have good instance-level performance.* For instance, Figure 4.11 shows that for the SIVAL datasets, AUC_I is often smaller than AUC_B. Inspecting the individual predictions showed that negative instances are almost always predicted correctly, but positive instances are not necessarily. It has been observed before by Liu et al. (2012) that, while standard MI algorithms assign instance labels, they focus on maximizing bag-level performance. The classifiers are thus often good at identifying the most positive instance in a bag, but not at finding all positive instances in a bag.

Since in our datasets the majority of instances are negative, bad performance on the positive instances does not necessarily translate into smaller accuracy. This is clearly visible in Figures 4.10 and 4.11 for some algorithms on the newsgroup datasets: While AUC_I is small, a_I is always close to one. A skewed instance label distribution is common for multi-instance data. Therefore, *accuracy may not always be the most appropriate performance measure for multi-instance problems.* Instead, AUC or weighted accuracy could be used. It also demonstrates that the performance measure influences the correspondence between instance- and bag-level performance. For instance, for the SIVAL datasets we often see that for the standard MI algorithms a_I is larger than a_B, whereas AUC_I is smaller than AUC_B.

We focus the remaining part of our discussion on the non-standard MI algorithms. Since these algorithms do not adhere to the standard MI assumption, one might expect that they will be worse at classifying instances.

An example is MISMO on the SIVAL datasets. It can be seen from Figure 4.10 that instance-level accuracy varies over a wide range. MISMO tends to either predict all instances in the test set as positive, or negative. This disagreement between instance- and bag-level accuracy of MISMO has been noted before by

Doran and Ray (2014), who showed that the set kernel used by MISMO can separate bags even when the corresponding instance kernel cannot separate the instances. It seems that for the SIVAL datasets, the classifier has learned to separate bags based on overall properties of the bags, whereas, judging from the higher AUC_I, for the newsgroup and UCI datasets individual properties of the instances are used. This again demonstrates the importance of the characteristics of the datasets in the relationship between instance- and bag-level performance.

A non-standard MI algorithm that *can* classify instances well is Citation-kNN. The scatter plot shows that on most UCI and SIVAL datasets, AUC_I is higher than AUC_B.

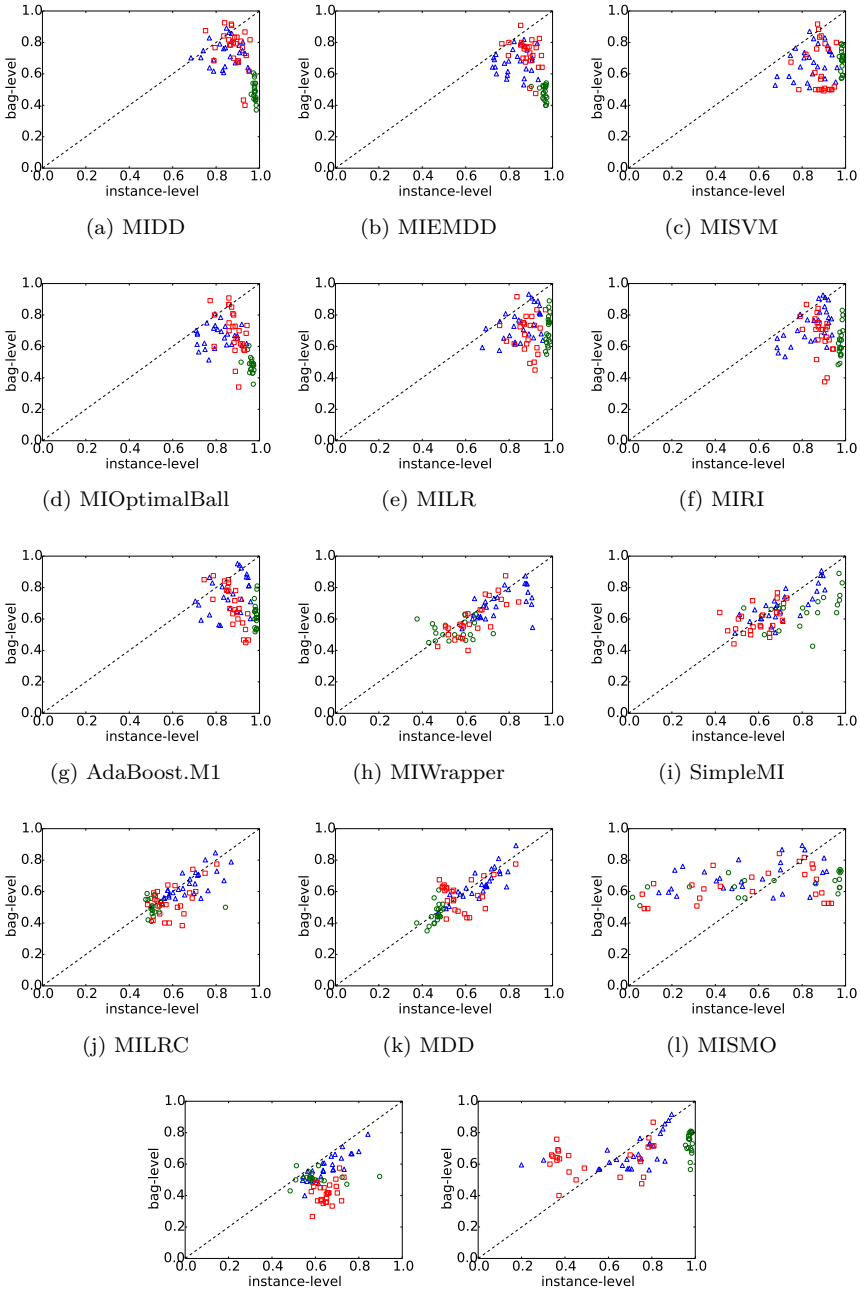
Finally, we observe that the distance based approaches, i.e., Citation-kNN, MIOptimalBall and the Diverse Density variants, do not perform well on the newsgroup datasets, both in terms of bag- and instance-level AUC. For MDD and MIEMDD, the AUC for the newsgroup datasets is even smaller than 0.5. By inspecting the individual predictions, we saw that not a single positive instance is classified positive, whereas a small number of negative instances are incorrectly classified as positive. This is explained by the presence of many irrelevant attributes (Dooly et al. 2003), and the high dimensionality of the data, which makes distance less effective as a measure of similarity.

From this discussion we conclude that the relationship between instance- and bag-level performance is determined by the domain of the dataset, the learner assumptions, and the performance measure. However, even if the correlation between, for instance, a_I and a_B, were perfectly linear for each classifier, the regression slope could still differ, resulting again in different rankings of classifiers in terms of instance- and bag-level accuracy.

Note that these results are consistent with the results in Section 4.4.2. For instance, we see in Figure 4.10 that AdaBoost.M1 consistently has better a_I than a_B on the newsgroup datasets, and that this is again observed in the rankings of the fourteen learners on the separate datasets in Figure 4.5.

4.5 Comparison of multi-instance and single-instance learning algorithms

In the multi-instance literature, experimental studies have mostly been conducted in terms of bag-level performance (Andrews et al. 2003; Blockeel et al. 2005; T. Dietterich et al. 1997; Fung et al. 2007; Maron and Lozano-Pérez 1998; Ramon and De Raedt 2000; Ray and Craven 2005; Tao et al. 2004; Wang and Zucker 2000; Xu and Frank 2004; Q. Zhang and Goldman 2001; Zhou and



(a) MIDD (b) MIEMDD (c) MISVM

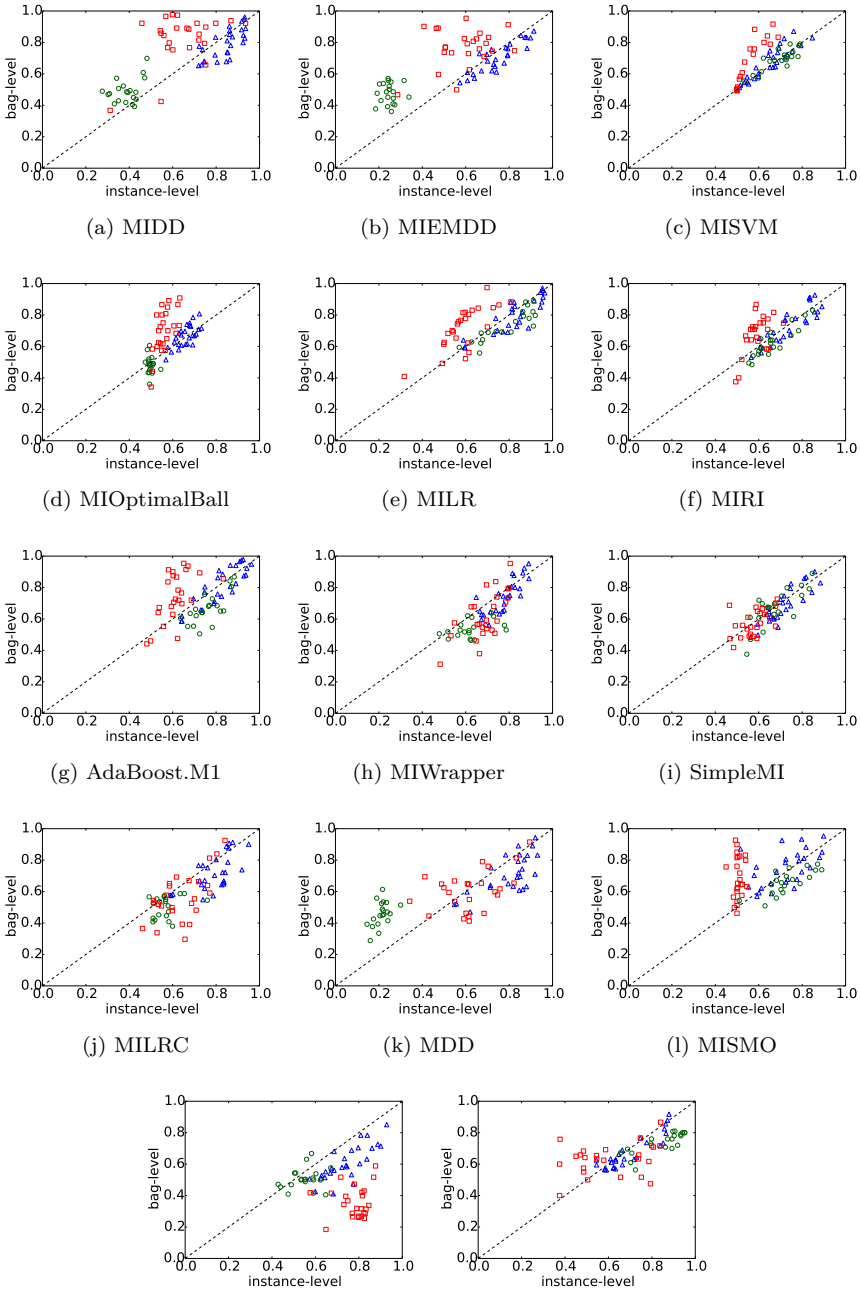
(d) MIOptimalBall (e) MILR (f) MIRI

(g) AdaBoost.M1 (h) MIWrapper (i) SimpleMI

(j) MILRC (k) MDD (l) MISMO

(m) CitationKNN (n) TLD

Figure 4.10: Bag-level versus instance-level accuracy for each classifier over the UCI (blue triangle), newsgroup (green circle), and SIVAL (red square) datasets.



(a) MIDD (b) MIEMDD (c) MISVM

(d) MIOptimalBall (e) MILR (f) MIRI

(g) AdaBoost.M1 (h) MIWrapper (i) SimpleMI

(j) MILRC (k) MDD (l) MISMO

(m) CitationKNN (n) TLD

Figure 4.11: Bag-level versus instance-level AUC for each classifier over the UCI (blue triangle), newsgroup (green circle), and SIVAL (red square) datasets.

Table 4.9: Multi-instance (MI) algorithms and their corresponding single-instance (SI) algorithm.

MI algorithm	SI algorithm
MIWrapper-J48 (Frank and Xu 2003)	J48 (Quinlan 2003)
SimpleMI-J48 (Dong 2006)	J48 (Quinlan 2003)
Citation-kNN (Wang and Zucker 2000)	kNN (Aha et al. 1991)
MILR (Ray and Craven 2005)	Logistic (Cessie and Houwelingen 1992)
MISMO (Xu 2003)	SMO (Platt 1999)
AdaBoost.M1-MITI (Freund and Schapire 1995), (Blockeel et al. 2005)	AdaBoost.M1-J48 (Freund and Schapire 1995), (Quinlan 2003)

M. Zhang 2003; Zucker and Chevaleyre 2001). Now that we have established that the correlation between instance-level and bag-level performance is weak, the question is whether the conclusions from these studies also apply in terms of instance-level performance or not. Below we investigate one particular aspect of this question, namely: *when evaluated in terms of instance-level performance, do multi-instance learning algorithms indeed perform better on multi-instance problems than single-instance learning algorithms do?* Similar to (Ray and Craven 2005), we limit our study to AUC.

The context of this question is that multi-instance (MI) learning can be tackled with single-instance (SI) algorithms: simply label each instance with the label of its bag and apply a standard SI learning algorithm to the resulting dataset. Consequently, many instances will have a positive label even if they are really negative. The opposite will not occur, so we get a dataset with one-sided class noise. The question is then how well SI learning performs on this dataset, as compared to MI learning on the original MI dataset. This has been investigated in the literature with mixed results: T. Dietterich et al. (1997) found the SI approach not to work well, while Ray and Craven (2005) provide a more complex picture, showing that this depends on the data and the actual learner used. Both papers measured performance on the bag-level only. Here, we investigate the same question but in terms of instance-level performance.

The experimental setup is similar to that of the previous section, although here we only use a selection of six MI learning algorithms that are based on some SI algorithm. These six algorithm pairs are listed in Table 4.9. On each dataset, we compare the instance-level AUC of each MI algorithm with that of its corresponding SI algorithm. A scatter plot is constructed for each pair of learners, plotting the SI and MI AUC of the learner on each dataset. The results are shown in Figure 4.12.

In many cases the MI algorithms do not clearly outperform their SI counterparts. This is for instance the case for SimpleMI-J48 and MIWrapper. Both these algorithms are based on the application of J48 on a single-instance dataset derived from the original multi-instance dataset. SimpleMI does this by averaging the features over all instances in a bag. MIWrapper, similar to our own conversion of the multi-instance datasets, assigns the bag label to each instance in the bag, after which it also reweighs the instances so that each bag has the same weight independent of the number of instances in it. However, in our experiments this extra step did not influence the performance much because except for the text datasets all bags have more or less the same size. Consequently, the difference between the performance of MIWrapper-J48 and J48 is very small. Moreover, for the text datasets, which have varying bag sizes, the reweighing does not always lead to improved performance of MIWrapper-J48.

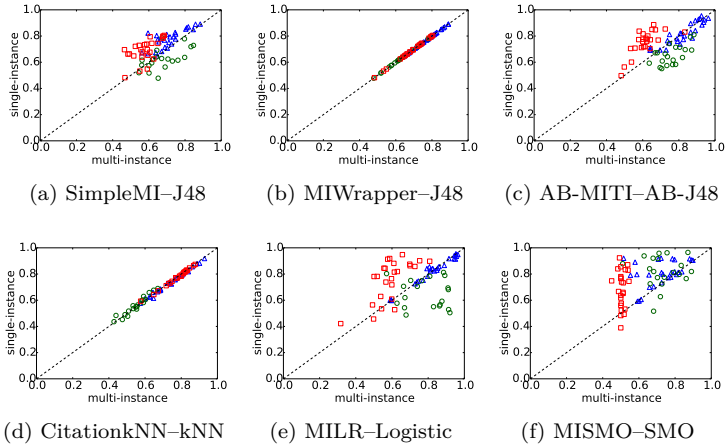
SimpleMI-J48 outperforms J48 only for the text datasets. We already saw in Section 4.4.3 that SimpleMI-J48 has good performance for the text datasets, both for bag- and instance-level classification. The fact that the text data has many irrelevant attributes with a value close to zero, explains why averaging works well here.

AdaBoost.M1-MITI clearly outperforms J48 for the text datasets, and it has a slight advantage on the UCI data. On the SIVAL datasets, AdaBoost.M1-J48 mostly wins against AdaBoost.M1-MITI.

There is almost no difference between the performance of Citation-kNN and single-instance kNN. With only a single instance in a test bag, identifying references with the minimal Hausdorff distance is the same as finding the nearest neighbor in the training set. Furthermore, the results show that the closest citations are mostly identical to the closest references and using them does not make a big difference in performance.

Lastly, MI-Logistic regression (MILR) often wins against its single-instance variant for the text and UCI datasets The same goes for MISMO, which we also observed not to work well for instance-level classification on the SIVAL datasets.

These results supplement Ray and Craven's conclusions, showing again a mixed picture. Whether an MI learner outperforms its SI counterpart depends on the domain of the dataset in combination with the type of learner. It happens frequently that an SI learner performs almost as well as an MI learner, or even outperforms it.



(a) SimpleMI–J48 (b) MIWrapper–J48 (c) AB-MITI–AB-J48

(d) CitationkNN–kNN (e) MILR–Logistic (f) MISMO–SMO

Figure 4.12: Instance-level AUC of single-instance versus multi-instance algorithms. ('AB' stands for AdaBoost.M1)

4.6 Conclusions

In the current literature on multi-instance learning, almost all empirical evaluations use bag-level classification performance. However, in a multi-instance setting, the task of classifying instances is often interesting by itself as well. but very little is known about how well methods typically perform on this task, and to what extent conclusions of experimental studies in terms of bag-level performance can be generalized to the instance-level. The aim of this chapter was to investigate this.

First, we examined the correlation between instance-level performance and bag-level performance of a set of classifiers, when ranking them according to one or the other. We find that this correlation is relatively weak, both in terms of accuracy and AUC. The "best" bag classifiers do not necessarily yield the best (or even good) instance classifiers. Our experiments are consistent with our theoretical analysis, and additionally demonstrate that the strength of the correlation varies with the specific problem, i.e., the domain of the dataset and the chosen performance measure.

Second, we examined the relationship between instance-level and bag-level performance of a single classifier over multiple datasets. We again found a mixed picture where the correspondence between instance- and bag-level performance of a learner on a give dataset depends on the the data domain, the performance

measure (accuracy or AUC), and the learner in question. These results provide some explanation for the difference in rankings of a set of classifiers in terms of bag- and instance-level performance.

Finally, we also investigated whether multi-instance learning algorithms perform better than their single-instance counterparts (on one-sided noisy data) in terms of instance-level AUC. We found that single-instance algorithms often outperform multi-instance algorithms, but this again depends on the specific problem.

Given the relatively weak correlation between bag-level and instance-level performance, we conclude that it is advisable for researchers proposing novel multi-instance algorithms to test their methods on both levels, or to state explicitly which level is of interest to them, and evaluate according to that level. Moreover, our experimental results showed that multi-instance classifiers are very sensitive to the context in which they are used, and this should be taken into account when evaluating multi-instance algorithms.

Chapter 5

A meta-learning system for multi-instance classification

In this chapter we investigate whether we can predict which multi-instance learning algorithm performs best on a given dataset according to a chosen performance measure. The work is based on the workshop paper (Vanwinckelen and Blockeel 2014a).

5.1 Introduction

Machine learning is largely an empirical science. When a researcher develops a new learning algorithm, they typically evaluate it by comparing its performance with that of existing algorithms on a collection of datasets. From these experiments, we try to understand which types of problems are suitable for a certain algorithm, and which are not. A more systematic approach to understand the inductive bias of different learners is to derive meta-characteristics from each dataset, and learn a model that can predict which learner best suits which dataset. This is the purpose of meta-learning. Several such studies have been conducted in the past. For an overview, we refer to (Giraud-Carrier 2008; Vilalta and Drissi 2002).

One subfield of machine-learning is multi-instance learning. The term multi-instance learning has been used with slightly different meanings in the past. Here, we use it in the sense of what is sometimes called "generalized multi-instance learning". Instances are organized into bags that are labeled positive

or negative; the number of instances in a bag is not fixed. The task is to learn a function that predicts the label of a bag. Due to the variable size of bags, a bag cannot be represented as a single vector without loss of information; multi-instance learners somehow have to handle this complication.

Several types of multi-instance learners have been proposed in the past, and they vary quite strongly in terms of the assumptions they make. This begs the question whether one can predict, using a meta-learning approach, which methods are suitable for which data sets. In this chapter, we present preliminary work in this direction. The contributions are as follows.

First, we propose a number of dataset descriptors that are specific to the multi-instance setting, and evaluate their relevance experimentally.

Second, we propose and evaluate two landmarking approaches for multi-instance classification. The term landmarking was first introduced by Pfahringer et al. (2000) for regular 'single-instance' learning. It refers to running a number of computationally cheap classifier systems on a dataset, and recording the behavior of these systems, in the hope that this will provide information regarding what methods (including much more expensive ones) are suitable for this dataset. As multi-instance methods tend to be computationally expensive by nature, we use single-instance learners for landmarking; this implies that the multi-instance datasets somehow have to be turned into (necessarily non-equivalent) single-instance datasets. We investigate two different methods for doing so.

The remainder of the chapter is structured as follows. In Section 2, we introduce terminology on multi-instance learning. In Section 3, we present our meta-learning approach. In Section 4, we describe our meta-dataset, including the multi-instance learners and the datasets on which it is based. In Section 5, we report experimental results, and in Section 6 we conclude.

5.2 Definition and terminology

Let $\mathcal{X}$ be the instance space and $\mathbb{B} = \{pos, neg\}$ the binary set of class labels. Standard binary classification, which we here call the *single-instance* setting, can be defined as follows. We are given a dataset D consisting of elements $(\mathbf{x}_i, f(\mathbf{x}_i))$ with $\mathbf{x}_i \in \mathcal{X}$ an instance and $f(\mathbf{x}_i) \in \mathbb{B}$ its label according to an unknown function $f : \mathcal{X} \to \mathbb{B}$. The learning task is to find the function f.

The generalized *multi-instance* learning setting is defined as follows. We are given a dataset that consists of bags B_i of instances where each bag has a label; the number of instances in a bag is variable. Each instance is described by a single vector $\mathbf{x}_{ij} \in \mathcal{X}$. There exists a relationship between the properties of the

instances in a bag and its bag label, but this relationship is unknown. From this information, we are to learn a function that can classify bags.

The *meta-learning task* we consider is defined as follows. We are given information about a collection of multi-instance classification tasks, consisting of the evaluation of different multi-instance learners $l_1, l_2, ...$ on a set of datasets $D_1, D_2, ...$ For each of these tasks we also have an estimate of the performance of the resulting classifier $l(D)$ in terms of the Area Under the ROC Curve (AUC). From this information we want to predict which learner to apply when presented with a new dataset. We therefore construct a meta-dataset D_m by extracting various properties from the original multi-instance learning tasks. Each instance in D_m consists of the extracted properties of one multi-instance task D, and is labeled with the multi-instance learner l that achieved the best performance on D. This results in a new standard supervised learning problem.

5.3 Our approach

Extracting statistical and information theoretic properties from the original datasets is one approach of constructing a meta-dataset. Examples of such properties include number of features, number of classes, ratio of examples to features, correlation between features and target concept, number of nominal attributes (Giraud-Carrier 2008). For multi-instance learning, these properties can be extended with statistical information about the number of instances in a bag.

Another approach which was first introduced by Pfahringer et al. (2000) and is reported to have stronger predictive power than statistical properties, is landmarking. A landmarker is a fast and cheap learner that indirectly gives us information about the properties of a dataset.

Applying multi-instance algorithms can be computationally expensive, therefore we first derive new single-instance datasets from the original multi-instance datasets, on which we will then apply a set of landmarkers. Whether these new classification tasks are equivalent with the original ones depends on the properties of the original multi-instance data and learner assumptions. We derive the new datasets in two different ways:

1. We label each instance with the label of its bag. Many instances will have a positive label in the dataset even if they are really negative. The opposite will not occur, so we get a dataset with one-sided class noise. This approach corresponds with the standard multi-instance assumption

that a bag is labeled positive if at least one of the instances in that bag is labeled positive.

2. We map the instances in a bag to a single feature vector by averaging each feature over these instances. We now learn a classifier based on aggregate information about a bag, instead of on the individual instances. This approach corresponds with a type of collective assumption where every instance contributes to the bag label.

As landmarkers we chose four learners with reasonably different biases:

- A decision stump based on the attribute that maximizes the gain-ratio
- Naive Bayes
- Nearest neighbors with one neighbor
- Logistic regression

5.4 The meta-learning dataset

The meta-learning dataset is based on the evaluation of fourteen multi-instance learners on datasets from three different domains in terms of AUC. These evaluations have been performed in the context of related work on multi-instance learning (Vanwinckelen, Tragante Do O, et al. 2014). The description of the datasets and multi-instance learners is also adopted from this text. Reusing experiments allows for easy investigation of the behavior of learning algorithms under different conditions, an idea that has been put forward before by Vanschoren and Blockeel (2008) with the introduction of an experiment database.

5.5 Multi-instance learner performance

We measure the AUC of each MI learning algorithm on each MI dataset. For the SIVAL datasets we perform 20 independent runs for each image class and average these results. For each run 20 randomly drawn positive bags and 20 randomly drawn negative bags were selected for training. For the text datasets and the UCI datasets we use 10-fold cross-validation.

Figure 5.1 presents the global rankings of the fourteen learners taken over all datasets from each category (SIVAL, text and UCI datasets) with a critical

difference (CD) diagram as described by Demsar (2006). This CD diagram is obtained by computing a ranking of the algorithms for each dataset and afterwards computing average ranks. A Friedman test is used to test if the performance difference between the algorithms is statistically significant at $p = 0.05$. If this is the case, we proceed with a post hoc Nemenyi test to find pairwise significant performance differences between the algorithms. The mean rank of each algorithm over all datasets of a given category is indicated on the horizontal axis. The highest rank corresponds to the best performance. Algorithms that are connected do not have significantly different performance.

The diagrams make clear that the AUC varies over a wide range. The overall rankings depend on the characteristics of the datasets and is different for each domain of datasets. The top three algorithms for the UCI datasets are MIDD, MILR, and AdaBoost.M1 with MITI as base learner. The classifier that is ranked highest most often is AdaBoost.M1-MITI, for 44.4% of the datasets. For the newsgroup datasets MILR is ranked highest most often, on 50% of the datasets. TLD is also a top performer which it was not for the UCI datasets. This algorithm models the class conditional probability distributions of the features. An approach that is appropriate for the text data where the features actually represent word frequencies, an approximation of word probability. We can see that the distance based approaches such as the different versions of Diverse Density (MDD, MIEMDD and MIDD), MIOptimalBall, and CitationKNN do not perform well on the text datasets. This is explained by the high dimensionality of the datasets (200 features). Finally, on the SIVAL datasets MIDD has the highest AUC most often, on 44.0% of the datasets.

5.6 Experiments

5.6.1 Experimental setup

In total we evaluated fourteen learners, which results in a multi-class meta-learning problem. However, treating the problem as such did not result in any useful model. The meta-properties that can distinguish between the multi-instance learners are different for each algorithm. We therefore convert the problem into a set of binary classification problems by predicting for each possible combination of two learners which one will win. With fourteen learners, there are 91 pairs of classifiers. As a meta-learner we chose an unpruned CART decision tree learner with a maximum depth of two to avoid overfitting[1]. We evaluated the meta-model in terms of accuracy.

[1]Decision tree pruning is currently unsupported in the used toolbox scikit-learn 0.14.

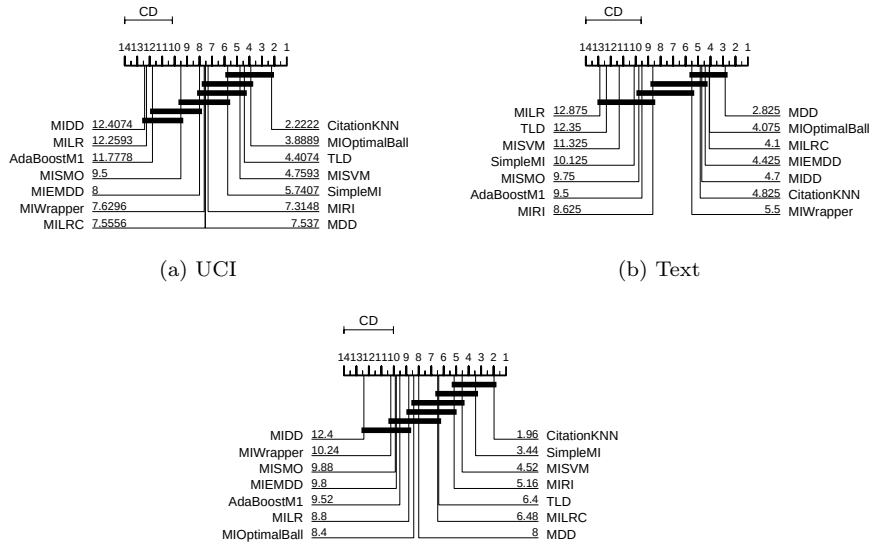


Figure 5.1: *Critical difference diagrams* for the global ranking of all learning algorithms in terms of AUC aggregated over all UCI, text, or SIVAL datasets

Because we have three categories of datasets of which the properties are very different, we performed experiments for each category of datasets and evaluated the meta-model with leave-one-out cross-validation.

5.6.2 Results: UCI datasets

In Section 5.3 we discussed a number of statistical and information-theoretic meta-properties that can be extracted from the multi-instance datasets. However, because we have datasets from three different domains, where in each domain most of these properties are very similar, we initially only use the number of features an instance has, and the noise level of the dataset. Figure 5.2 shows the results for the evaluation of the decision tree model learned from the UCI meta-dataset. The figure compares the predictive accuracy of the meta-model with that of a classifier that always predicts the majority class (which corresponds to always using the multi-instance learner that is best on average). Blue circles represent classifier pairs for which the meta-model has highest accuracy, red circles for which the majority class predictor does. The

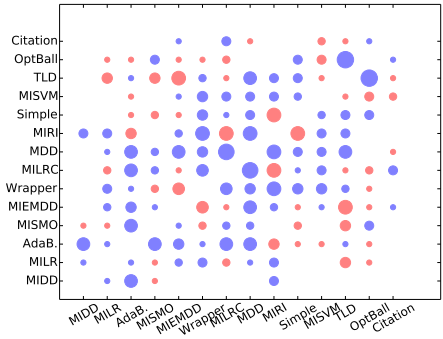


Figure 5.2: Comparison of a majority class predictor with a decision tree meta-model for the UCI datasets. Meta-properties are the number of features of an instance and noise level of the dataset.

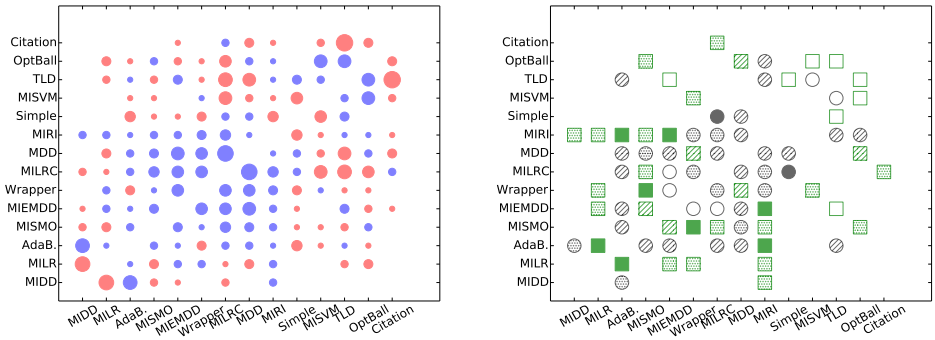


Figure 5.3: (a) Comparison of a majority class predictor with a decision tree meta-model based on landmarking for the UCI datasets. (b) Landmarkers with highest gain-ratio for classifier pairs where the meta-model performs best.

area of the circle is proportional to the difference in accuracy between the two models.

The high accuracy of the meta-model on many classifier pairs in comparison to that of the majority classifier shows that the number of features and the noise level are useful properties for determining the most performance multi-instance learner. Investigating the decision trees, we see that the number of features is

most often the determining factor in predicting the winning classifier. Each of the five source datasets (Adult, Diabetes, Spam, Tic-tac-toe, and Transfusion) from which the multi-instance datasets were constructed has a different number of features. This means that the meta-model is mostly learning to distinguish between these five dataset types.

In a next experiment, we predict the learner with the highest AUC based exclusively on the eight landmarking properties defined in Section 5.3. Figure 5.3a shows the results for this experiment. We observe that the landmarking approach has worse performance than the previous approach. Although there are a few cases where the landmarking model performs best. Examples are (MILR,MIDD), (MIRI,MIEMDD), (MIRI,MILRC), and (TLD,CitationKNN).

Figure 5.3b shows the importance of the different landmarkers by showing for each pair of classifiers the landmarker that was selected as the root node of the decision tree meta-model when trained on the complete meta-dataset, i.e., the landmarker with the highest information gain ratio. The symbols are defined as follows. Green landmarkers are computed on the averaged single-instance datasets, and gray landmarkers on the one-sided noisy single-instance datasets. The decision stump, naive Bayes, nearest neighbors, and logistic regression classifier are respectively identified by a empty, colored, hatched (//), and dotted symbols (.). We only show the landmarkers where the meta-model outperformed the majority class predictor. From this figure, we see that the landmarker with the highest gain ratio often changes from one classification pair to the other.

5.6.3 Results: Text datasets

Exclusively based on number of features and the training set size, we cannot make predictions for the text and SIVAL datasets because these properties are the same for each dataset from these domains. We therefore employ landmarkers again. As can be seen from Figure 5.4a, our meta-model does not perform very well on the text datasets. In most cases it is better to predict the multi-instance algorithm that performs best on the majority of text datasets. Figure 5.4b again shows the landmarkers having the highest gain-ratio for the cases where the meta-model wins.

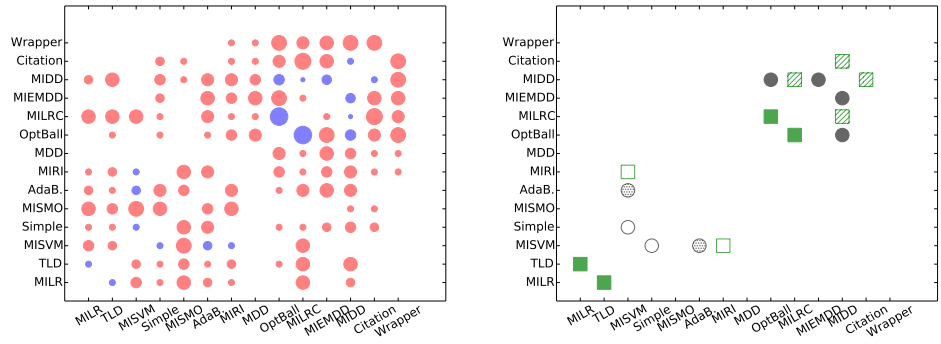


Figure 5.4: (a) Comparison of a majority class predictor with a decision tree meta-model based on landmarking for the text datasets. (b) Landmarkers with highest gain-ratio for classifier pairs where the meta-model performs best.

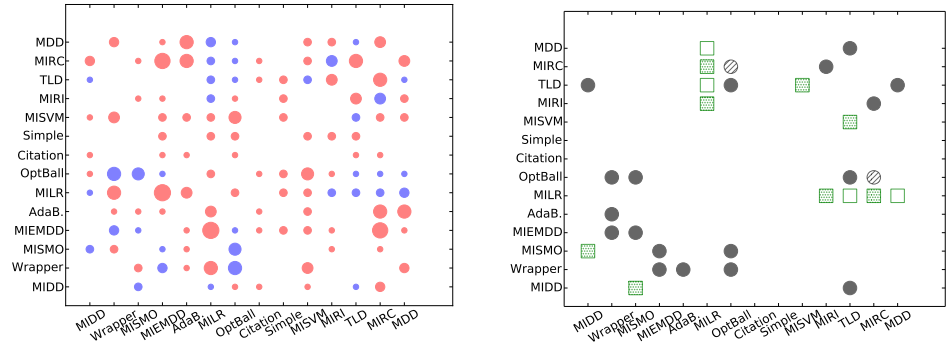


Figure 5.5: (a) Comparison of a majority class predictor with a decision tree meta-model based on landmarking for the SIVAL datasets. (b) Landmarkers with highest gain-ratio for classifier pairs where the meta-model performs best.

5.6.4 Results: SIVAL datasets

For the SIVAL datasets our meta-model based on landmarking again did not outperform the majority class predictor in many cases, as can be seen from Figure 5.5a. Regarding the most important landmarkers, Figure 5.5b shows that for the SIVAL datasets this is frequently naive Bayes, trained on one-sided

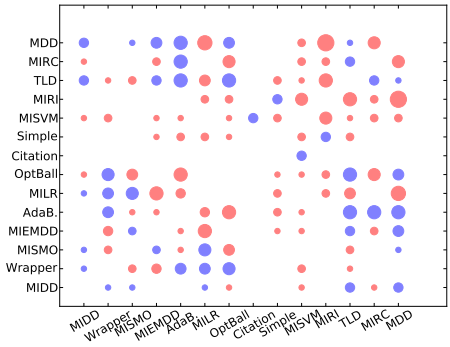


Figure 5.6: Comparison of a majority class predictor with a meta-model based on the mean and variance of the percentage of positive instances in a bag (SIVAL data).

noisy data. This is in contrast with the UCI datasets, where this landmarker was selected only once for (SimpleMI,MILRC).

As an alternative, we therefore investigated if the distribution of positive instances in a bag has any predictive power. Our meta-properties in this case are the average percentage of positive instances in a bag (an indication of the noise level of the dataset), and the variance of the percentage of positive instances over all bags in a given dataset. Note that this information would not be available in a generalized multi-instance setting. Nevertheless, this is an interesting property to investigate. Figure 5.6 shows the results of this experiment. As can be seen, the distribution of positive instances in a bag influences the performance of the multi-instance learners. Inspecting the learned decision trees, we see for example that for MDD, MILRC, TLD, and MIWrapper, the decision tree learns that AdaBoost.M1-MITI outperforms these classifiers on datasets with a large percentage of positive instances, i.e., a low noise level. It is known that for regular supervised learning, AdaBoost is prone to overfitting on noisy datasets. This also appears to be the case for multi-instance learning.

5.7 Conclusions

In recent years, several algorithms have been proposed specifically for multi-instance learning. In this chapter, we investigated if we can predict which algorithm is most suitable for a given multi-instance dataset. We experimented with extending the landmarking approach introduced by Pfahringer et al. (2000)

to the multi-instance setting. We found that which meta-features have best predictive performance depends on the domain of the datasets, and the multi-instance learners that are compared. A meta-model that was learned on one domain does not necessarily transfer to another domain. These observations have consequences for empirical research on multi-instance learning. They illustrate that it is insufficient to evaluate multi-instance learners on datasets from a single problem domain. Instead, evaluation should include datasets from several domains, or otherwise the domain selection introduces a bias.

A weak point of this study is the limited availability of meta-data: Each observation in the meta-learning dataset is a 'costly' run of an algorithm on a dataset. As a result of the small sample size, performance evaluation was based on leave-on-out cross-validation. However, as we discussed in chapter 3 the estimates obtained by this resampling method typically have large sample variance, so they are not very reliable. In our study we restricted ourselves to the performance results obtained in chapter 4. It would be worthwhile to extend the meta-learning dataset with more performance results, for instance on synthetic datasets.

Chapter 6

Bayesian network structure learning in the presence of sampling variance

6.1 Introduction

When we perform an experiment in machine learning, we do so with the data that is available. However, most of the time we are not really interested in the results for that particular dataset. Instead, we are interested in the results for a certain data population from which that dataset is a sample.

Consider the following example. In his 1936 paper, Fisher introduced the famous 'iris' dataset that contains measurements from 50 iris flowers categorized into three subspecies (Fisher 1936). With the available data, we could learn a model to predict the subspecies on 2/3 of the observations, and use the other 1/3 to estimate the prediction error of the learned model. In this case, we are estimating the prediction error of the model learned on that specific dataset. It is more likely, however, that we are interested in the performance of the learner on any random sample of 33 iris flowers of these three subspecies, instead of the specific sample available. In that case, the performance of the learner will differ depending on which sample we have available. Part of the problem then becomes how to quantify this sample variance.

Bayesian network learning has a similar setup: We have a data sample available from which we can learn a structure and estimate the conditional probabilities

of the variables. However, had we drawn another sample, it would have likely resulted in a slightly different network. This model selection uncertainty, or sample variance, is usually not taken into account in the learning process. Instead, the focus usually lies on learning a single optimal network structure with fixed parameter values. It is only after the network has been learned, in the inference phase, that the probabilistic character of data is again taken into account.

The goal of this chapter is to develop a structure learning algorithm that takes into account model selection uncertainty because of sample variance. In Section 6.2, we explain how this sample variance is simulated by means of bootstrapping. By treating the problem as a statistical decision problem we arrive at a bagging algorithm, as described in Section 6.4.1. As a secondary result, we expect that this algorithm will result in higher quality structures than competing structure learning algorithms. Although this chapter does not include experiments, the latter follows from the related work of Elidan (2011) (see Section 6.6).

Our starting point is frequentist in nature: We try to infer a model structure from limited available data that can be repeatedly sampled from a population. This entails that if we could learn the model on the entire data population there would be no uncertainty left. Section 6.4.2 explains how this frequentist problem setting can also be fitted into a Bayesian framework. We make use of this insight to increase the computational efficiency of the algorithm. We are able to reduce the computational effort even more by making use of the statistical racing technique explained in Section 6.4.3. Finally, in Section 6.5 we explain how we can perform inference when more than one network structure remains after racing.

6.2 Bootstrapping

Our goal is to find a set of Bayesian network structures that most faithfully represents the correlations between the variables in a data population from which we only have one sample available. Network structure quality can be measured by several scoring functions such as BDeu, MDL, BIC or AIC. Studying the differences between these scoring functions is not in scope of this work; any of these can be chosen.

When there is sample uncertainty, the score of a network will be a random variable with a probability distribution over all possible samples from the population at hand. Since we only have partial information about the data population (a sample), the sampling distribution of the scores of a network is

unknown, but we can approximate it by means of bootstrapping. The underlying idea of this technique is that the sampling distribution of a statistic (the network score) can be approximated by the distribution of estimates computed on data samples that are obtained by repeatedly sampling the original dataset with replacement. A detailed introduction to bootstrapping is (Efron and R. J. Tibshirani 1994).

One approach of using bootstrapping to simulate the sampling variance of the network structures, is to learn an optimal network structure independently on each bootstrap sample, as was done in (Friedman, Goldszmidt, et al. 1999). This method has two disadvantages though: First, learning the optimal structure of a Bayesian network is an NP-hard problem, meaning that, at least for now, exact algorithms always have exponential time complexity (Chickering 1996). Learning a structure on each bootstrap sample is thus computationally expensive. In comparable problem settings, the number of bootstrap samples is usually set to a 100 or more (Elidan 2011; Friedman, Goldszmidt, et al. 1999). Second, by conducting each search independently we are not taking into account the distributional properties of the score, which may provide useful information for the search.

As an alternative, we propose a probabilistic structure learning algorithm that uses all the bootstrap samples in a single procedure. Based on a decision theoretic approach, we identify which properties of the score distribution should be taken into account. The output of the procedure is a set of network structures for which we are confident that they have a high score for the given data population.

6.3 Greedy hill-climbing

Formally, a Bayesian network is a directed acyclic graph $G(V, E)$ that represents a joint probability distribution over set of random variables $V = \{X_1, ..., X_m\}$ by factoring it as a product of local conditional probability distributions. Each variable X_i is represented by a node, and each dependency between two variables X_i and X_j is indicated by a directed edge $E(i, j)$. We will call the set of variables on which a node X_i is dependent the parents Pa_i of X_i. In this work, we only consider distributions of discrete variables. We denote the number of states of a variable X_i as r_i. The local conditional probability distributions are modeled as conditional probability tables (CPTs). The structure learning task can be described as finding the optimal Bayesian network structure with respect to a dataset D consisting of N observations.

Structure learning methods can be classified into two broad categories: Constraint based and search-and-score algorithms. Our approach fits into

the latter category: The algorithm explores the states-space of structures to find the structure that maximizes a predefined scoring function. The number of possible network structures grows exponentially in the number of variables so it is impossible to score every possible structure. Search algorithms usually make use of a heuristic to restrict the search space so that only the most promising candidates are visited, and getting stuck in local minima is avoided.

The focus of this work lies on how to adapt such an algorithm to handle sample variance. Our proposal can be used in combination with several existing search-and-score algorithms, but for demonstration purposes we have chosen to adapt the greedy hill-climbing algorithm. The reason is that it is used by many authors as a baseline for newly developed algorithms because of its relatively simple setup and yet good performance. We provide a short description of the original algorithm, before discussing our adaptations in Section 6.4.

Greedy hill-climbing starts with one specific network. This can be a random network, the empty network or a network that includes prior structure knowledge. The heuristic used by greedy hill-climbing is that network quality can be improved by only changing a small part of the graph at the time, i.e., making a local change. Such a change is restricted to one edge addition, deletion or reversal for each node. Usually there is also a maximum put on the number of parents a node can have. The quality of an adaptation is evaluated by means of a scoring function. In this work we choose the popular Minimum Description Length (MDL) function.

If we denote the number of observations consistent with Pa_i=pa_i as N_{pa_i}, and the number of observations consistent with $\{\mathrm{Pa}_i{=}\mathrm{pa}_i \wedge X_i = x_i\}$ as N_{x_i,pa_i}, then the MDL of a graph G can formally be written as:

$$MDL(G) = \sum_{i=1}^{m} MDL(X_i|Pa_i) \tag{6.1}$$

$$MDL(X_i|Pa_i) = H(X_i|Pa_i) + \frac{logN}{2} K(X_i|Pa_i) \tag{6.2}$$

$$H(X_i|Pa_i) = -\sum_{x_i,pa_i} N_{x_i,pa_i} log \frac{N_{x_i,pa_i}}{N_{pa_i}} \tag{6.3}$$

$$K(X_i|Pa_i) = (r_i - 1)\Pi_{pa_i} r_l \tag{6.4}$$

Several alternative scoring functions exist, but they all have two important properties in common: First, the score is decomposable, meaning that it can be expressed as the sum of the scores of the different nodes. This allows to evaluate local changes quickly because the entire score does not have to be recomputed

each time. Second, the computation of the score can be reduced to computing a set of sufficient statistics. For discrete networks, these are the counts N_{pa_i} and N_{x_i,pa_i}.

In order to avoid getting stuck in a local minimum, a tabu list is maintained. This is a list of the k most recently visited structures so that we can avoid revisiting these. If the algorithm would still get stuck, a number of random edge changes can be performed in order to escape the minimum. If, after a certain number of steps, there is no more score improvement the search ends.

The greedy hill-climbing algorithm can be improved even further in multiple ways, for instance by integrating the sparse candidate algorithm of Friedman, Nachman, et al. (1999). Such additional, more advanced improvements are currently not in scope of our work.

A practical implementation of greedy hill-climbing requires setting a number of parameters that influence algorithmic performance. We propose to use the same parameter settings as Teyssier and Koller (2012): These authors found that starting with an empty network resulted in good performance. The optimal size of the tabu list and the number of random moves between each restart was determined by systematic search. The number of moves without improvement before restarting was selected to be the same as the size of the tabu list. The number of candidate parents per node was chosen between 10 and 30, depending on the size of the data set and the number of nodes in the network to be discovered.

6.4 Structure learning with sample variance

6.4.1 Rationale

Without sample variance, it is straightforward to compare the scores of two network structures, as they are deterministic. In the current setting, however, we have to compare the probability distributions of the scores of the structures. The change that leads to the largest score improvement 'over all bootstrap samples' with respect to the current structure should be accepted. The question then is which properties of the score sampling distribution should be involved in this comparison.

The answer is simpler than expected and can be found in decision theory. One of the central questions in this field is the optimal choice of an agent between actions when the outcomes of those actions are uncertain. It is straightforward to see the correspondence with our problem: We have an imaginary agent

moving through the state-space of structures in search of the optimal structure with respect to the MDL score. An action corresponds to choosing a certain network structure, and the outcome of that action (the score) is uncertain because of sample variance.

In the context of game theory and economics, Von Neumann and Morgenstern (2007) showed that when an agent is rational, the optimal decision for the above problem is the decision that maximizes expected utility. A utility function is a very general concept that measures the value that an agent attaches to a certain outcome. In economics, value judgments are often subjective and rooted in the psychology of an agent. In our setting, however, utility is based on probability theory and Occam's razor, as it is simply the MDL score of a network. The expected utility is the weighted average of the utility of each of the possible outcomes, where the weight is the probability that the act will lead to that outcome. Each bootstrap sample is assumed to be equally likely possible so the weights are uniform.

An alternative way of arriving at expected utility maximization is by an interpretation of the law of large numbers (Feller 1968): Imagine that the agent would have to make the same choice repeatedly; each setting would be independent from the others and the scores in each setting would be identically distributed. Informally stated, the weak law of large numbers then tells us that the average amount of 'utility' gained per trial over the long run is overwhelmingly likely to be close to the expected value of an individual trial. This means that if we maximize expected utility, we are maximizing gains in the long term. Recall from the introduction that understood sample variance as originating from a repeated experiment, so the above is exactly what we are interested in.

The law of large numbers transforms an average over a series of repeated experiments in an expected value of a variable that is defined for a choice that in principle only needs to made once. It thus gives us a handle on how to make a decision in such a one-time situation.

To conclude, in order to navigate through the state-space of network structures with the greedy hill-climbing algorithm, we use as a scoring function the mean score over bootstrap samples. This approach must sound familiar to the machine learning audience, as it is the bagging approach described by Breiman (1996). Bagging is known to produce high quality models in noisy settings because it reduces the variance of a learner, thereby preventing overfitting. Our discussion provides a novel perspective on bagging: It can be viewed as a useful summary of the information contained in the sampling distribution of the utility function of a learner. Under the assumption of rationality, maximizing the bagged score is the optimal decision strategy in the presence of sample variance.

6.4.2 Bayesian bootstrap

Computing the network scores on each of the bootstrap samples requires keeping track of the counts on each of those samples. In order to avoid this computationally expensive task, we propose to use the Bayesian bootstrap introduced by Rubin et al. (1981) as an alternative for the traditional bootstrap procedure described in Section 6.2.

Recall that in bootstrapping, we randomly resample a dataset D with n observations with replacement. In theory, an observation can thus be present in a bootstrap sample anywhere from 0 to n times. The distribution of these counts is a multinomial, with the number of trials being equal to n, and the event probabilities all being $\frac{1}{n}$.

An alternative perspective is that we keep the observations in the dataset fixed, but sample a weight for each observation from the set $\{0, \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, ...1\}$ such that the sum of the weights equals 1. The weights can then be interpreted as normalized counts from the multinomial described above.

Notice that the set of weights is discrete and the granularity is determined by the size of the dataset. From this viewpoint, it is natural to smooth the weights so that they can be anywhere between 0 and 1. This is what the Bayesian bootstrap does, by sampling the weights from a Dirichlet distribution $\text{Dir}(\alpha_1, ..., \alpha_n)$ with $\alpha_i = 1$ for $i \in [1, n]$. Rubin et al. (1981) showed that this corresponds with using as a prior over the observations a symmetric Dirichlet distribution with $\alpha = 0$, also known as the Haldane prior. After seeing the data, the posterior distribution of the weights becomes $\text{Dir}(1,...,1_n)$.

It is interesting to note that the $\text{Dir}(1,...,1_n)$ distribution is actually known itself as an uninformative prior, namely the Bayes-Laplace prior. It is yet another interpretation of ignorance about the weights of the observations than the Haldane prior, which is based on the principle of insufficient reason introduced by Bayes and Price (1763) and Laplace (1840). In the words of Peterson (2017), the principle theorizes that: *"If one has no reason to think that one state of the world is more probable than another, then all states should be assigned equal probability"*. In the Bayesian framework this means that every combination of weights has equal probability. If we ignore the discrete character of the weights of the frequentist bootstrap, it indeed corresponds to each bootstrap sample being equally likely.

If we now concretely want to compute the score of a network with the Bayesian bootstrap, equation 6.3 in Section 6.3 has to be modified. Since each observation receives a weight between zero and one, we replace the counts N_{pa_i} by $\sum w_{pa_i}$, i.e., the sum of the weights of the number of observations consistent with

Pa_i=pa_i. We do the same for N_{pa_i,x_i}. Equation 6.3 becomes:

$$H(X_i|Pa_i) = -\sum_{x_i,pa_i}\sum w_{x_i,pa_i} \cdot log\frac{\sum w_{x_i,pa_i}}{\sum w_{pa_i}} \tag{6.5}$$

For each of the counts N_{pa_i} and N_{pa_i,x_i}, we now keep track of the sum of the weights consistent with these variable instantiations. When new weights are sampled, the counts stay the same, and only their weights are updated.

The final score of a network is the average over all B weight set samples:

$$MDL_{bag}(G) = \frac{1}{B}\sum_{j=1}^{B}\sum_{i=1}^{m} H_j(X_i|Pa_i) + \frac{logN}{2}K_j(X_i|Pa_i) \tag{6.6}$$

6.4.3 Racing

It is impossible to generate all possible weight combinations from the Dirichlet distribution for the dataset at hand. Consequently, the expected value of the bootstrap distribution of the scores of a structure is unknown. It is, however, well approximated by the average score of the available bootstrap samples. By the weak law of large numbers we know that the margin of error for the expected value can be reduced to an arbitrary small constant as long as we average over a sufficiently large number of bootstrap samples. Unfortunately, the exact number of bootstrap samples to achieve an acceptable approximation quality is difficult to determine. To be on the safe side, we can choose a large number, but this wastes computational resources. Furthermore, for some structures it may be clear very quickly that they are not going to be selected because their average score differs substantially from the best average score. For these structures, it is expected that less bootstrap samples are needed to reject them in comparison to structures with higher scores. In conclusion, ideally, we do not want to generate more bootstrap samples than is strictly necessary. This is the objective of sequential sampling, also called *racing algorithms*.

In the proposed racing technique, we compute confidence intervals for the bootstrap scores to eliminate non-promising candidate structures as soon as possible. When the upper bound of such a confidence interval is smaller than the lower bound of the confidence interval of the best structure (the structure with the confidence interval that has the largest upper-bound), the former structure can be eliminated. As soon as one of the confidence intervals lowest bounds is larger than any of the other confidence intervals upper bounds, we know that this structure is significantly better than all the other structures so we can replace the current network structure by that one. Note that the width of

the confidence intervals should be adapted to account for multiple comparisons. For instance by using the Bonferroni correction.

Given a sufficient number of bootstrap samples, the central limit theorem justifies computing a t-distribution based confidence interval for the the average score of a structure. To ensure that the average scores indeed approximately follow a t-distribution, we can for instance start with at least 15 weight samples. The number of weight samples can be incremented either by one, or stepwise. This should be investigated experimentally. Another parameter of the racing algorithm is the significance level, or the nominal coverage probability, of the confidence intervals. Finally, in order to ensure that the algorithm always terminates we should also set a maximum on the number of bootstrap weight samples. If no statistically significant better candidate structure than the currently best structure can be determined after the maximum number of samples, greedy hill climbing can either terminate, or a number of random edge changes can be made. After the structure learning algorithm terminates, the parameters θ_i of the Bayesian network can be learned by means of maximum likelihood estimation, which, in the case of discrete variables, boils down to setting them to the estimated conditional probability estimates based on the ratio of the counts N_{x_i,pa_i} and N_{pa_i} available from the dataset.

6.5 Model selection uncertainty quantification

6.5.1 Introduction

Once the structure and parameters of the Bayesian network are learned, it can be used to answer probabilistic queries about the data population it models. A typical query is for instance the computation of the marginal probability $P(\mathbf{X})$ of a subset $\mathbf{X}$ of variables in the network. Another example is the conditional probability $P(\mathbf{X}|\mathbf{E} = \mathbf{e})$), taking into account evidence about the values of another subset of variables $\mathbf{E}$. For generality, we will hereafter denote these types of probabilities as θ.

Several specialized methods have been developed for efficiently inferring the probabilities θ from a known network (Neapolitan et al. 2004). It is possible, however, that the greedy hill-climbing algorithm returns multiple Bayesian network structures, whose scores cannot statistically be distinguished from each other. It is indeed well known that, when the sample size is relatively small in comparison to the hypothesis space, several alternative graph structures can be supported by the data. I.e., these structures receive a high goodness-of-fit score, or have a high posterior probability (Dash and Cooper 2004; Friedman

and Koller 2003). This is a problem of practical importance as small sample sizes are common in fields such as medicine, psychology and biology. In this section we explain how we can use such a set of networks to answer probabilistic queries about the variables.

6.5.2 Bayesian model averaging

A widely accepted solution when being confronted with several competing models is to use Bayesian Model Averaging (BMA) (Burnham and Anderson 2003; Raftery et al. 2005) . This method extends Bayes' theorem for updating probabilities in the light of new evidence (the data D) to the model selection phase. As shown in formula 6.7, the posterior probability of a model G_i equals the likelihood of the data given that model $P(D|G_i)$, multiplied by the prior probability of the model $P(G_i)$. To obtain a probability, a normalization factor $P(D)$ is applied. However, when the purpose is only to compare models relative to each other, this factor can be left out.

$$P(G_i|D) = \frac{P(D|G_i)P(G_i)}{P(D)} \tag{6.7}$$

The final BMA prediction $\hat{\bar{\theta}}$ is then the weighted average of the predictions $\hat{\theta}_i$ over all k individual models G_i, where the weights are equal to the posterior model probabilities $P(G_i|D)$: $\hat{\bar{\theta}} = \sum_{i=1}^{k} w_i \hat{\theta}_i$.

When the number of possible models is very large models, as is usually the case when learning the structure of a Bayesian network, small weights can be disregarded to reduce the computational cost. This will however require renormalization of the weights as follows:

$$\hat{P}(G_i|D) = \frac{\hat{P}(G_i, D)}{\sum_{G_i \in \mathcal{G}} \hat{P}(G_i, D)}, \tag{6.8}$$

with $\mathcal{G}$ the set of k selected models.

In the current setting, we retain the k best network structures whose scores cannot statistically be distinguished from each other. Consequently, $\hat{P}(G_i|D)$ will be $1/k$ for each of the selected networks. Note that this is likely a very coarse approximation of the true probabilities $P(G_i|D)$.

6.5.3 Variance estimation

Bayesian model averaging also provides a method for estimating the variance of a BMA prediction $\hat{\theta}$ as follows:

$$\mathrm{var}(\hat{\bar{\theta}}) = \sum_{i=1}^{k} w_i[(\hat{\theta}_i - \hat{\bar{\theta}})^2 + \mathrm{var}(\hat{\theta}_i|G_i)], \text{ with} \tag{6.9}$$

- $w_i = \hat{P}(G_i|D)$ is the estimated posterior probability of network G_i
- $\hat{\theta}_i$ is the estimate of the conditional probability θ_i inferred from G_i
- $\hat{\bar{\theta}} = \sum_{i=1}^{k} w_i\hat{\theta}_i$ is the BMA estimate of θ_i.

The formula, borrowed from Burnham and Anderson (2004), is the sum of two terms: The first term is the variance of the mean prediction over the retained network structures. The second is the weighted expected variance of the estimates conditional on one of the structures, G_i, being the best. The second term would be zero if the parameter estimates in a network would always be exact, but we will now argue that this is not the case, and also explain how to estimate $\mathrm{var}(\hat{\theta}_i|G_i)$:

The traditional approach to learning the parameters of a Bayesian network is maximum likelihood estimation (MLE), i.e., choosing the parameter estimates that maximize the likelihood of the data D given the network structure G. Recall that in the case of discrete variables, the MLE estimate $\hat{\theta}_i$ of a parameter equals the ratio of the counts N_{x_i,pa_i} and N_{pa_i}. Since we assume the data D is only a sample from a population, any inferred probability estimates $\hat{\theta}_i$ from G_i are random variables that are a function of that sample.

It has been proven that the sampling distribution of the MLE estimator is asymptotically normally distributed, provided certain regularity conditions are satisfied. An estimate of the variance of the sampling distribution of $\hat{\theta}_i$ in the case of binary variables then for instance equals $\hat{\theta}_i(1-\hat{\theta}_i)/n$, with n the size of D. Moreover, $\hat{\theta}_i$ is an unbiased estimate of the mean of this distribution.

Each MLE parameter estimate $\hat{\theta}_i$ is an independent asymptotically normally distributed variable. The independence follows from the fact that a Bayesian network factorizes the joint probability distribution of a set of variables into a product of conditional probabilities, one for each variable. Each of these terms can again be decomposed by making use of the structure of the CPTs.

6.6 Related work

The work most closely related to ours is that of Elidan (2011), who uses a bagged log-likelihood score to find an optimal Bayesian network structure for a given dataset. His starting point is the traditional view in machine learning that bagging reduces the variance of a learner, and that it therefore improves structure learning on small sample sizes. He also proposes an EM extension of his approach for the case of incomplete data. The end result is a single network structure that can be used for inference and prediction.

Our goal on the other hand is to take into account sample variance when learning a Bayesian network structure. We approached the problem from the perspective of optimal decision taking in the presence of uncertainty. Additionally, the methodology in both papers differs: While Elidan uses actual bootstrap samples, we use the Bayesian bootstrap. We find that Bayesian bootstrapping facilitates score computation. Furthermore, it is known to have slightly smaller variance than traditional bootstrapping (Clyde and Lee 2001). We use a racing algorithm to determine the optimal number of bootstrap samples.

There are a few exceptions to deterministic structure learning algorithms. Friedman, Goldszmidt, et al. (1999), for instance, study the robustness of network features based on DAGs by means of learning different network structures on each bootstrap sample. Similar to our work, the authors assume there is a single 'true' but unknown Bayesian network. Their goal is to assess the confidence that certain structural elements are present in that network. Their method can also be used to quantify uncertainty about inferred probabilities. It does not directly lead to a single network structure, but they propose to use the confidence in the network features for guiding the search process for the optimal network structure.

Another structure learning algorithm that takes uncertainty into account is that of Eaton and Murphy (2012), they rely on Bayesian model averaging, and use an MCMC method to sample from the space of DAGs.

Racing algorithms for model selection have a long history and have been used before in a hill-climbing procedure. One of the first authors to investigate racing for this goal is Greiner (1996), who introduces the PALO probabilistic hill-climbing approach. He uses the Hoeffding bound to determine statistical differences between scores. The advantage of the Hoeffding inequality is that it is distribution independent. The disadvantage is that it is quite a conservative bound.

Hulten and Domingos (2002) also use Hoeffding races for Bayesian network structure learning. They make use of the decomposability of the structure score

to be able to parallelize the racing process. Both Hulten and Domingos (2002) and Greiner (1996) use racing to determine the size of a single sample, whereas we use it to determine the number of bootstrap samples.

6.7 Conclusions and future work

One of the goals of machine learning is to design learning algorithms that can accurately capture the relationships between different attributes in a dataset. While it is almost never explicitly stated, what is most often meant by 'a dataset' is a random sample from a given population, instead of a specific dataset. The goal of our work is to explicitly take this sampling variance into account in the Bayesian network structure learning process. The obtained network structure should thus be optimal with respect to the data population.

Our approach uses a greedy hill-climbing algorithm to explore the space of network structures. Sample uncertainty is taken into account in this search process by using a general principle from decision theory. Namely, that when facing a decision under uncertainty the rationally optimal decision is to choose the action with the highest expected utility. Concretely, this results in maximizing the average score over the bootstrap samples, also known as bagging. Our work thus puts bagging into a new perspective: Maximizing the bagged score is an optimal decision strategy in the presence of sample variance.

We suggest two methods to improve the computational efficiency of our algorithm. First, because we use the Bayesian bootstrap, we do not have to compute the sufficient statistics of the MDL score for each bootstrap sample. Instead, we only compute the statistics once for the original sample and update their weights. Second, we apply a racing algorithm to determine the minimally required number of bootstrap samples to decide on the optimal network change when hill-climbing. While this does not improve the asymptotic time complexity of the algorithm, it does improve practical runtime.

Experiments would be needed to confirm the viability of our approach. However, as discussed in Section 6.6, both bagging and racing have been proven to be effective in related research so we expect positive confirmation of our proposals. It would be interesting to verify whether bagging with the Bayesian bootstrap results in better network structures than bagging with the regular bootstrap.

Our work can be extended in several directions. First, the Bayesian bootstrap uses the Haldane prior as the probability distribution of the weights of the observations. It would be interesting to investigate the effect of other objective

priors on the learning algorithm. Such research could also result in alternatives to bootstrapping.

Finally, we have three proposals for improving the racing algorithm. The first proposal is with respect to the multiple comparison correction. While the Bonferroni correction is an obvious choice, it is known that this leads to conservative confidence intervals. Ramdas and Balsubramani (2015) show that tighter intervals can be constructed by making use of the dependencies between iterations of a sequential sampling procedure. They only discuss the two-sample case, but it should be possible to extend this method to multiple comparisons.

Second, the racing algorithm accepts a change as soon as it is a statistically significant improvement over the current structure, disregarding all other structures in the race. Several structures may, however, be better than the current one, so it could be interesting to explore local changes to each of these structures in parallel.

Finally, if more than one structure change leads to approximately the same average score improvement, we can decide which structure to choose by taking into account the trade-off between exploration of the state-space and exploitation of known qualitative solutions. If we intend to 'exploit', we prefer the change for which we are most certain that the expected improvement is achieved. On the other hand, if we are stuck in a local minimum, or the improvements are small, we can 'explore' by choosing the structure change with the highest uncertainty about its score.

Chapter 7

Conclusion

This chapter summarizes the contributions presented in this dissertation and discusses a number of directions for future work.

7.1 Summary of contributions

Recent advances in computer science and electronics have made it possible to capture increasingly larger amounts of data in all fields of science and industry. In order to gain insights from this raw data, a thorough knowledge of statistics is often required. While it is expected that a researcher or engineer is an expert in their own field, it is not self-evident that that person is also proficient in statistics, which is a vast and complex field by itself. Although machine learning is a field that is closely related to statistics, we find that it is affected by the same problems. This dissertation aims to better understand current statistical inference practices in machine learning and proposes improvements were needed. The focus lies on the evaluation of supervised machine learning algorithms.

7.1.1 Statistical inference with cross-validation

The first contribution of this dissertation is a synthesis of the existing knowledge about error estimation of predictive models with cross-validation. While the presented insights are available in the literature, they are often scattered over different papers or even over different research fields. Moreover, the theoretical papers on the topic often require expert knowledge of statistics and

learning theory. This leads to a catch-22 situation, in which researchers in need of this statistical knowledge cannot acquire it, because of their lack of statistical knowledge. Empirical research does not always offer a solution, as the conclusions of different papers can be conflicting because of the variety of possible experimental settings. We have collected insights from the literature and presented them in a clear and comprehensive way.

We started from the insight that a model *learned on a given training set* should be evaluated differently than a learning algorithm *applied on a random sample from a given population.* Since cross-validation generates alternative training sets from the original data, it is not very suitable for evaluating specific models. The difference with the original training set can be minimized by choosing leave-one-out cross-validation, but unfortunately the results of this method are unreliable when dealing with unstable learning problems. While there are several theoretical definitions of unstable learning problems, definitions that can be used in practice are currently lacking.

We found confirmation in the literature that the current practice of evaluating learners with repeated cross-validation, with k typically around ten, indeed results in reliable performance estimates. The issue, however, is that these estimates are often presented together with a measure of confidence, while it has been proven that such a measure can not reliably be estimated based on a single sample. A common mistake is to estimate the sample variance of the error of a learner by computing the variance of the individual errors, or that of the cross-validation errors on the individual folds. These estimates, however, have no connection with the true sample variance. We demonstrated this by showing that the variance of repeated cross-validation goes to zero when the number of repetitions over which is averaged increases. A performance difference between two learners can therefore always (incorrectly) be detected with statistical testing by using a sufficiently large number of repetitions.

It should be noted that repeated cross-validation is not a waste of computational effort. The sample variance of this estimator is usually smaller than that of a single cross-validation, leading to more reliable and reproducible error estimates. It is a misconception, though, that increasing the number of repetitions causes the estimate to converge to the true learner error. A systematic difference will always remain, due to the selection of a single sample.

7.1.2 Multi-instance learning

Our second contribution is in the area of multi-instance learning, a special type of semi-supervised learning. Similar to supervised learning, the goal is to learn a model from labeled instances that can predict the label of future instances. The

difference is that instances are not labeled individually, but instead they are grouped together in bags and only the bag label is known. We determined to what extent conclusions of experimental studies where performance evaluation happens on the bag level can be generalized to the instance level. We showed theoretically that there is no one-on-one mapping between instance- and bag level accuracy. Our results are supported by an extensive empirical evaluation of the performance of fourteen multi-instance learners on both synthetic and real-world datasets, in terms of accuracy and AUC-ROC. We found several examples of multi-instance learners that outperformed other learners on the bag level, but performed worse on the instance level.

Due to the fact that a bag consists of several instances, multi-instance learning algorithms usually have a high computational complexity. Our third contribution is therefore to investigate if we can predict in advance which algorithms would perform best for a given problem domain, so that we can avoid running the underperforming algorithms. As this task can be interpreted as 'learning about learning', it is known as meta-learning. The dataset from which we aim to learn this information consists of a set of statistical and information-theoretic meta-properties, together with a number of landmarkers (Pfahringer et al. 2000). These are a set of simple algorithms with a reduced running time in comparison to the original algorithms. The idea is that if the inductive bias of a landmarker is similar to that of the original algorithm, their predictive performance will be correlated. We found that the predictive power of the meta-features depended strongly on the domain. The small size of the training set did not allow us to gain more insights. This is a general problem with meta-learning, since each observation is a 'costly' run of an algorithm on a dataset.

7.1.3 Bayesian network structure learning

Our final contribution is a greedy hill-climbing algorithm to learn the structure of a Bayesian network. This algorithm assumes a frequentist setup: If we would repeat the experiment, we would almost certainly draw another sample from the population leading to slightly different results. This sample uncertainty is simulated by incorporating bootstrapping into the structure learning process. It is known that bootstrapping is a resource intensive technique, so we improve the computational efficiency of the algorithm by using a Bayesian version of the bootstrap, and incorporating a racing algorithm. The proposed structure learning algorithm is a practical example of how a frequentist problem can be approached from a Bayesian viewpoint. It is worth noting that this final contribution is purely theoretical, and experiments are still needed to confirm the viability of the approach.

7.1.4 Recommendations

To conclude, we present a set of recommendations that follow from the research in this thesis:

- Always be clear on whether you are estimating the error of a model, or that of learner. Estimating model error with k-fold cross-validation requires a stable learning problem combined with a large number of folds; Estimating the learner error is best done with ten-times ten-fold repeated cross-validation. On average over different problem domains, this parameter setting minimizes the sample variance of the performance estimate.
- Define the goal of the research precisely, and use the metric most suitable for this goal. In the case of multi-instance learning specifically, do not use bag-level accuracy as a proxy for instance-level accuracy.

7.2 Future work

In the following, we discuss some ideas for future research related to the topics presented in this dissertation.

7.2.1 Translating computational learning theory results about cross-validation into practice

Recent work in the area of statistical learning theory clearly demonstrates a relationship between the increased variance of a learner error estimate obtained with k-fold cross-validation in comparison with the true error, and the stability of the learning problem (Elisseeff, Pontil, et al. 2003; Kearns and Ron 1999; Kumar et al. 2013). In general, the aim of these papers is the derivation of a bound on the probability that the cross-validation error differs by more than ϵ from the true error. For this they rely on existing or newly proposed formal definitions of algorithmic stability.

What is still missing, however, is a translation of this theory to techniques that can determine the stability of a learning problem in practice. Being able to distinguish cases where traditional statistical inference techniques are applicable, from the problematic cases, would have a large impact on empirical machine learning research. A starting point would be to apply these theoretical stability criteria on learner-data combinations frequently used in experimental machine

learning research. A hurdle — perhaps the reason such study is still missing — is that most theoretically proposed stability measures are computationally expensive. Ideally, a practically usable, general stability criterion should be developed. This could then lead to calibrated confidence intervals and hypothesis tests that take into account the distinct characteristics of a learning problem.

7.2.2 A declarative experimentation system

A final direction for future work is the design of a declarative query language and inference system that can be used to analyze data generated from empirical research. Preliminary work in this direction was presented in:

> G. Vanwinckelen, V. Tragante Do O, D. Fierens, and H. Blockeel (2014). "Instance-level accuracy versus bag-level accuracy in multi-instance learning". In: *Data Mining and Knowledge Discovery*

Empirical testing of a hypothesis currently demands a lot from a scientist in terms of setting up experiments, analyzing the results, and ensuring all this is done correctly. In all steps of this process the scientist needs to be mindful of applying the proper statistical methodology in order to be able to draw meaningful conclusions at the end. This is not only a labor-intensive process, but also error-prone (Button et al. 2013; McIntyre and McKitrick 2005; Simmons et al. 2011). As statistics provides us with a plethora of data analysis and inference methods, it is practically impossible for a scientist to have full knowledge of all existing methods and the statistical assumptions behind them. What is more, nowadays statistical analyses are being increasingly performed by non-statisticians, making the problem even more vital (Leek 2013).

An obvious solution for avoiding incorrect statistical analysis is to take the statistical part of the experimentation process out of the hands of the user. Involving a statistical expert in a research or industry project, however, can be a costly affair. A more efficient solution would be if the researcher could formulate their hypothesis, and the remainder of the experimentation process would be automated. This solution is reminiscent of the declarative programming paradigm, where one does not specify *how* to compute something, but rather *what* to compute, leaving the actual procedure as a decision for the compiler. Declarative languages have been successful in several areas. The quintessential example is the domain specific language SQL for querying relational databases. In the field of logic programming, Prolog is a well known example. But declarative languages have also been designed for solving problems in artificial intelligence. Notable examples are Problog for probabilistic logic programming (De Raedt and Kimmig 2015), MiningZinc for constraint processing (Guns

et al. 2013), BiQL for graph querying (Dries et al. 2012), and SCCQL for constraint-based clustering (Adam et al. 2013).

The three main design principles of such a system should be as follows:

Simple First, someone who is not an expert in statistics should be able to formulate a query in the language and get meaningful results. We only expect high-level knowledge from a user of concepts such as for instance probability distributions, their expected value and variance, and confidence intervals. We do not for instance expect the user to know all the different methods to compute a confidence interval, and the situations in which they are applicable.

Population based Second, the system is population based: When we perform an experiment in machine learning, we do so with the data that is available. However, most of the time we are not really interested in the results for that particular set of data. Instead, we are interested in the results for a certain data population from which the dataset is a sample. Consider the computation of the accuracy of a decision tree on a test set. It is likely that the accuracy on that specific dataset is not our main interest. Instead, we see it as an estimate of the accuracy of the decision tree on the population from which the test set is a random sample. The declarative system should be able to express estimator uncertainty and also be able to reason with it.

Declarative Finally, the system is declarative: The language allows to formulate *what* should be computed and not *how* it should be computed. The advantage is that users are shielded from having to make low-level decisions. When presented with a declarative query, the system should be able to automatically deduce the most appropriate method for a given task, possibly aided by background information. Typical decisions in statistical inference problems are about the sampling method, the minimum sample size, and the method to compute a confidence interval.

A starting point for the syntax of the language could be SQL, the prototypical declarative language for relational databases. Conceptually, the difference with SQL is that the latter only assumes the existence of finite datasets. While it contains keywords for requesting the computation of descriptive statistics such as an average or a standard deviation, these have a different interpretation than intended in our system, because they assume the data is complete. There is no concept of a — possibly infinite — population and the sample variance that originates from only having a sample available from that population.

Consider for instance a query requesting an interval estimate of the mean height of a population of male students:

```
ESTIMATE mean(Height)
FROM Student
WHERE gender='male'
ENSURING CONF(P) > 0.8
```

The query consists of four main parts. The first component specifies the population parameter that we want to estimate and is indicated by the keyword 'ESTIMATE'. In the example, the estimated statistic is the expected value of a continuous random variable, but other types of statistics can also be requested, such as for instance the probability that a variable takes on a certain value, or its variance. This part of the query corresponds with the 'SELECT' part of an SQL query.

The second query component corresponds with the 'FROM' part of an SQL query, in that it defines the sources of the data that will be used in the other two parts of the query. In fact, we use the same keyword, but it should be interpreted differently: 'Student' is the entire population under study, from which we have available a sample, for instance in the form of a simple database table.

Third, the 'WHERE' clause is optional, and specifies only to use the observations that fulfill the specified condition(s).

The final 'ENSURING' clause is also optional. It allows to specify the statistical accuracy of the population parameter estimate. In the example a constraint is put on the minimum coverage level of the confidence interval for the variable 'Height', namely 80%. Alternatively, this could also be a constraint on the maximum variance of the estimate.

Inference engine

The declarative system could rely on a Bayesian network to represent the population and perform inference. A Bayesian network, or belief network, provides a concise description of a joint probability distribution of a set of random variables E, by factoring it as a product of local conditional probability distributions. The reasons for choosing this representation are threefold: First, a mature probabilistic reasoning framework already exists for these models. Second, since Bayesian networks are generative models, meaning that the joint probability of all the variables in the data population $P(X, Y)$ is modeled, all

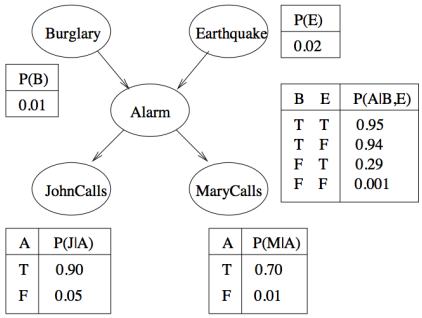


Figure 7.1: Example of a Bayesian network ('Burglary')

aspects of a dataset can be queried. Finally, Bayesian network allow for an intuitive interpretation of the existence of an edge between two variables: In the case of causal modeling, an edge between two variables A and B can be interpreted as 'A causes B'. This interpretation makes it easy for a researcher to construct a Bayesian network based on their expertise in a certain domain. Moreover, discovering cause-effect relationships is often the goal of scientific experimentation, which is indeed the application of the declarative system.

Figure 7.1 shows an example of a Bayesian network with five variables (Friedman Goldszmidt). The network consists of five discrete variables, each with a local CPT. Taking into account the conditional dependencies in the network, the joint probability distribution can compactly be written as:

$$P(B, E, A, J, M) = P(J|A)P(M|A)P(A|B, E)P(B)P(E)$$

Using the syntax of the example in the previous section, an example of a query about the burglary example is as follows:

```
ESTIMATE P(JohnCalls=1)
FROM Burglary
WHERE Alarm=1
ENSURING CONF(P) > 0.95
```

In this query, the user asks for an estimate of the probability that John calls when the alarm goes off. The answer should be presented in the form of a confidence interval, of which the confidence level is at least 95%.

7.2.3 Causal Bayesian networks

As mentioned in the previous section, one of the applications of Bayesian networks is *causal* modeling. We explain this concept in more detail by means of an example from Spirtes (2010). Imagine that an insurance company wants to include *the probability of having a heart attack in the next five years* in their charging rate, and they believe the habit of drinking red wine influences that probability. Sex and bmi are also known to have an influence on heart attacks, so the population under study only consists of males with bmi 25. The insurance company is not interested in changing the behavior of their clients, they only want to be able to predict the probability of someone having a heart attack. It may be that customers who drink red wine are also from a better socio-economic background, and that this is the real cause of them having a lower chance of a heart attack, but this is of no interest to the company. The task in this setting is predictive modeling, and it is the task that was considered in this dissertation.

This contrasts with the typical purpose of scientific research, namely discovering causal relationships: Consider for instance a study by the health department in which the goal is to determine the influence of drinking red wine on the probability of having a heart attack. The goal of the department is to identify recommendations for a healthier lifestyle so they are interested in the causal connection between red wine and heart attacks. If red wine drinkers have a lower probability of having a heart attack because of socio-economic reasons then a recommendation will have no effect, because it would only change the population of red wine drinkers. To determine a causal connection between drinking red wine and hear attacks, *manipulated* probabilities are needed. That is, the researcher has to investigate the probability of having a heart attack among people who have been assigned to drink red wine, and not among those who chose it by themselves.

We imagine that a declarative inference system would be most useful in this last setting, so the focus should be on causal modeling. A starting point would be to study the 'do-calculus' introduced by (Pearl 2003) to calculate with manipulated probabilities. Another interesting direction is active learning. In this context the work of (Tong and Koller 2001) is interesting, in which it is investigated how to select the most informative experiments for determining the structure of a causal Bayesian network.

7.2.4 Transfer learning for Bayesian networks

One of the key assumptions in this dissertation is the existence of a population from which we only have a sample available. In many studies, data collection is a time-consuming and expensive task, leading to small sample sizes. This may be an issue for the practical applicability of the declarative experimentation system, where we represent the dependencies in the data in the form of a Bayesian network. When insufficient data is available, the structure of such a network is difficult to learn.

Transfer learning aims to overcome this problem by exploiting the common characteristics between a problem domain with ample data and one with little data. Two different tasks can be distinguished in this area: First, the performance of a predictive model in a data-deficient domain can be improved by including data from a related domain in the learning process. Second, the information exchange is in both directions and models are learned on two domains simultaneously. An interesting research direction is the development of a transfer learning algorithm for learning the structure of a Bayesian network.

An interesting structure learning algorithm suitable for this purpose was recently proposed by Schmidt et al. (2007): Here, a hill-climbing algorithm searches the space of possible network structures, where the set of potential parents and children of a node is pruned by means of L1-regression. Perhaps it is possible to adapt this algorithm for transfer learning as follows: Park and Casella (2008) showed that L1-regularization can also be seen as a Bayesian method, where a Laplace prior with mean zero is placed on the coefficients of the regression model. We could use the Bayesian lasso to learn the neighborhoods of the nodes in a problem domain with ample data. Next, in the data-poor domain, instead of instantiating the Bayesian lasso with priors centered at zero, we center the priors at the coefficients learned in the other domain. It is also worth investigating if this idea can be extended to the task of learning on two domains simultaneously. For instance by making use of the Bayesian group lasso (Raman et al. 2009).

7.2.5 Building a knowledge base for statistical inference

When evaluating a supervised learning algorithm, the experiment always consists of the following components:

- The learning algorithm
- The population

- The performance measure
- The sampling method
- The proper statistical inference method.

When designing a declarative experimentation system that can handle this task automatically, the learning algorithm, population and performance measure have to be provided by the user as they depend on his or her interest. Optionally, the system could present results for a set of default performance measures. The selection can depend on information submitted by the user about the problem setting. The optimal sampling method and statistical inference method could be automatically determined based on the other elements of the experiment. Criteria that we imagine would have an influence on this decision are for instance the size of the dataset, the number of features, and the learning algorithm. Such criteria could be implemented in the form of an expert system consisting of a set of 'if-then-else' rules. However, as was already discussed in this dissertation, for certain tasks the optimal approach is still an open problem. In fact, new insights and best practices for statistical inference are published almost everyday. It is therefore impossible for a researcher who is not specialized in the topic to always keep up with the latest developments. A solution would be to automatically extract this knowledge from the scientific literature and store the learned facts in a database that can be accessed by the expert system. Such a *knowledge base construction* system would rely heavily on natural language processing techniques. Examples of related existing systems are the DeepDive and the Never Ending Language Learning project (Carlson et al. 2010; C. Zhang 2015). We recognize this is a very challenging task, as the system will find a multitude of conflicting facts derived in different experimental settings. Moreover, the experimental settings associated with the rules and facts in the knowledge base will never correspond exactly with the experimental setting under investigation.

Appendix A

Variance of repeated cross-validation

In this appendix we apply the theoretical framework developed by Bengio and Grandvalet (2004) for estimating the variance of cross-validation to repeated cross-validation. We describe a preliminary proof that in some cases repeating cross-validation error estimates indeed have reduced sample variance in comparison to a single cross-validation. The variance reduction is largest for twofold cross-validation.

A.1 Variance of cross-validation

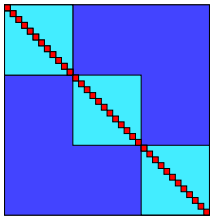


Figure A.1: The structure of the covariance matrix of a threefold cross-validation estimator on a sample of size n. The matrix is $n \times n$. The size of the lightblue blocks equals $m \times m$ ($\frac{n}{k} = m$).

In this section we concisely repeat the derivation of a formula for the variance of the cross-validation error estimator as given in (Bengio and Grandvalet 2004). It is important to realize that we assume here that k-fold cross-validation is used to estimate the error of a *learning algorithm* L, applied on samples of size n drawn from a population P.

We saw in Chapter 3 that this error estimate $\hat{\varepsilon}_{cv}$ equals:

$$\hat{\varepsilon}_{cv}(L,T) = \frac{1}{k}\sum_{i=1}^{k}\hat{\varepsilon}_i(L(T \setminus T_i), T_i) = \frac{1}{n}\sum_{i=1}^{n} e_i(L(T \setminus T_i), T_i)$$

In this formula, $e_i(L(T \setminus T_i), T_i)$ is an individual cross-validation error of the model learned on the $k-1$ folds $\{T \setminus T_i\}$ on an instance from fold T_i.

Consider the cross-validation error vector $\mathbf{e} = (e_1, e_2, ..., e_n)$. By definition, the variance of the cross-validation error estimate $\hat{\varepsilon}_{cv}$ can be written as:

$$\mathrm{var}(\hat{\varepsilon}_{cv}) = \frac{1}{n^2}\sum_{i,j} = cov(e_i, e_j). \tag{A.1}$$

Bengio and Grandvalet (2004) showed that the structure of the covariance matrix of $\mathbf{e} = (e_1, e_2, ..., e_n)$ is as follows:

Theorem 1.

- *All diagonal elements are identical:* $\forall i, var(e_i) = \sigma^2$
- *All off-diagonal elements of the* k $m \times m$ *diagonal blocks are identical:* $\forall (i,j) \in T_k$ *and* $j \neq i, cov(e_i, e_j) = \omega$, *with* $m = n/k$
- *All the remaining entries are identical:* $\forall i \in T_l$ *and* $j \in T_k$ *and* $l \neq k, cov(e_i, e_j) = \gamma$

Figure A.1 gives an example of this structure for $k = 3$.

- The variance σ^2 is the average (taken over training sets) variance of errors for 'true' test examples when learner L is fed with training sets of size $m(k-1)$.
- The within-block covariance ω would also apply to 'true' test examples; it arises from the dependence of test errors stemming from the common training set.

- The between-blocks covariance γ is due to the dependence of training sets (which share $n(k-2)/k$ examples) and the fact that test block T_k appears in all the training sets D_l for $l \neq k$.

If we combine this with formula A.1, the variance of the cross-validation error estimate can be written as a linear combination of the second moments σ, ω, and γ:

$$\text{var}(\epsilon_{cv}) = \frac{1}{n}\sigma^2 + \frac{m-1}{n}\omega + \frac{n-m}{n}\gamma.$$

For leave-one-out cross-validation $m = 1$ so:

$$\text{var}(\epsilon_{loo}) = \frac{1}{n}\sigma^2 + \frac{n-1}{n}\gamma.$$

Notice that when n goes to infinity, the variance does not go to zero, but to γ.

A.2 Variance of repeated cross-validation

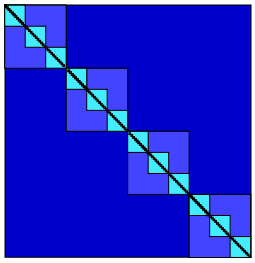


Figure A.2: The structure of the covariance matrix of the 4 times repeated 3-fold cross-validation estimator.

We now apply the same reasoning for deriving a formula for the variance of R times repeated $k$fold cross-validation. In this case, the error vector has nR elements: $\mathbf{e} = (e_{1,\pi_1}, ..., e_{n,\pi_1}, ..., e_{1,\pi_R}, ..., e_{n,\pi_R})$, with π_i denoting the i-th repetition.

The covariance matrix of $\mathbf{e}$ has a similar structure as the structure of the covariance matrix described in Theorem 1. There will be at most four distinct covariance values: σ^2, ω and γ as defined for cross-validation, for errors that are computed on the same partition. We assume that all partitionings are interchangeable, so the blocks on the diagonal of the covariance matrix are simply copies of the covariance matrix of one-time cross-validation. Additionally,

we define the covariance between the errors from two different repetitions: $\alpha = cov(e_i|\pi_l, e_j|\pi_k)$. As an example, the structure of the covariance matrix of four times repeated threefold cross-validation is shown in Figure A.2.

The variance of repeated cross-validation can then be written as:

$$\text{var}(\epsilon_{rcv}) = \frac{\sigma^2}{nR} + \frac{m-1}{nR}\omega + \frac{n-m}{nR}\gamma + \frac{R-1}{R}\alpha$$

Let us try to determine α: We know that repeating leave-one-out cross-validation does not reduce the error variance, as each repetition gives exactly the same result. Therefore, the following equality should hold:

$$\lim_{R\to\infty} \text{var}(\epsilon_{rcv}) = \alpha = var(\epsilon_{loo}) = \frac{1}{n}\sigma^2 + \frac{n-1}{n}\gamma$$

However, because γ depends on the amount of overlap of the training sets, the fact that the formulas have the same structure does not mean both variances are equal for the same n.

Using the above result, we can write the variance of the repeated cross-validation estimator as:

$$\epsilon_{rcv} = \frac{\sigma^2}{n} + \frac{m-1}{nR}\omega + (\frac{n-1}{n} + \frac{1-m}{nR})\gamma$$

We can now answer the question if repeating cross-validation reduces the variance in comparison to a single cross-validation:

$$\text{var}(\epsilon_{cv})\text{-var}(\epsilon_{rcv}) = \frac{m-1}{nR}(R(\omega-\gamma) - (\omega+\gamma)) \tag{A.2}$$

$$\text{var}(\epsilon_{cv})\text{-var}(\lim_{R\to\infty}\epsilon_{rcv}) = \frac{m-1}{n}(\omega-\gamma) \tag{A.3}$$

From Bengio and Grandvalet (2004) we know that:

$$0 \leq \sigma^2$$

$$0 \leq \omega \leq \sigma^2$$

$$-\frac{m}{n-m}\sigma^2 \leq \gamma \leq \sigma^2.$$

When averaging over all possible partitionings, the maximum variance reduction occurs when $\omega = \sigma^2$, and $\gamma = -\frac{m}{n-m}\sigma^2$. This means maximum correlation between the errors computed for the same model, and a negative correlation

between errors on different folds. Upon further inspection, the maximum reduction in variance is attained when $k = 2$, as in that case $\gamma = -\sigma^2$ and $\omega = \sigma^2$.

The formula also shows, however, that it is possible that repeating cross-validation has larger variance than single cross-validation. This happens when $\gamma > \omega$. Since γ is the covariance between the errors due to overlapping training sets, we would expect this to happen most often when the number of folds is large and tends towards n.

Bibliography

Adam, A., H. Blockeel, S. Govers, and A. Aertsen (2013). "SCCQL: A constraint-based clustering system", *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Springer, pp. 681–684.

Aha, D. (1990). "Incremental constructive induction: An instance-based approach", *Proceedings of the 7th International Conference on Machine Learning*. Morgan Kaufmann, pp. 117–121.

Aha, D., D. Kibler, and M. Albert (1991). "Instance-based learning algorithms", *Machine learning* 6.1, pp. 37–66.

Alpaydm, E. (1999). "Combined 5×2cv F test for comparing supervised classification learning algorithms", *Neural Computation* 11.8, pp. 1885–1892.

Amores, J. (2013). "Multiple instance classification: Review, taxonomy and comparative study", *Artificial Intelligence* 201, pp. 81–105.

Andrews, S., I. Tsochantaridis, and T. Hofmann (2003). "Support vector machines for multiple-instance learning", *Advances in Neural Information Processing Systems 15 (NIPS)*. MIT Press, pp. 577–584.

Auer, P., P. Long, and A. Srinivasan (1998). "Approximating hyper-rectangles: learning and pseudo-random sets", *Journal of Computer and System Sciences* 57.3, pp. 376–388.

Auer, P. and R. Ortner (2004). "A boosting approach to multiple instance learning", *Proceedings of the 15th European Conference on Machine Learning*. Vol. 3201. Lecture Notes in Computer Science. Springer, pp. 63–74.

Bayes, M. and M. Price (1763). "An Essay towards solving a Problem in the Doctrine of Chances", *Philosophical Transactions of the Royal Society of London 53*, pp. 370–418.

Beleites, C. and R. Salzer (2008). "Assessing and improving the stability of chemometric models in small sample size situations", *Analytical and Bioanalytical Chemistry* 390.5, pp. 1261–1271.

Bengio, Y. and Y. Grandvalet (2004). "No Unbiased Estimator of the Variance of K-Fold Cross-Validation", *Journal of Machine Learning Research* 5, pp. 1089–1105.

Ben, H. (2012). *Air Quality Prediction Hackathon*. `http://blog.kaggle.com/2012/05/01/chucking-everything-into-a-random-forest`. Accessed: 2014-07-30.

Bjerring, L. and E. Frank (2011). "Beyond Trees: Adopting MITI to Learn Rules and Ensemble Classifiers for Multi-Instance Data", *Proceedings of the 24th Australian Joint Conference on Artificial Intelligence*. Perth, Australia. Springer, pp. 41–50.

Blockeel, H., D. Page, and A. Srinivasan (2005). "Multi-instance tree learning", *Proceedings of the 22d International Conference on Machine learning*. ACM Press, pp. 57–64.

Blum, A. and A. Kalai (1998). "A note on learning from multiple-instance examples", *Machine Learning* 30.1, pp. 23–29.

Borra, S. and A. D. Ciaccio (2010). "Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods", *Computational Statistics & Data Analysis* 54.12, pp. 2976–2989.

Bouckaert, R. (2004). "Estimating Replicability of Classifier Learning Experiments", *Proceedings of the International Conference on Machine Learning (ICML)*.

Bouckaert, R. R. (2003). "Choosing Between Two Learning Algorithms Based on Calibrated Tests", *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 51–58.

Bousquet, O. and A. Elisseeff (2002). "Stability and generalization", *Journal of Machine Learning Research* 2, pp. 499–526.

Breiman, L. (1996). "Bagging predictors", *Machine learning* 24.2, pp. 123–140.

Burman, P. (1989). "A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods", *Biometrika* 76, pp. 503–514.

Burnham, K. P. and D. R. Anderson (2003). *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media. Chap. 4.

– (2004). "Multimodel inference: understanding AIC and BIC in model selection", *Sociological methods & research* 33.2, pp. 261–304.

Button, K. S., J. P. Ioannidis, C. Mokrysz, B. A. Nosek, J. Flint, E. S. Robinson, and M. R. Munafò (2013). "Power failure: why small sample size undermines the reliability of neuroscience", *Nature Reviews Neuroscience* 14.5, pp. 365–376.

Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell (2010). "Toward an Architecture for Never-Ending Language Learning.", *AAAI Conference on Artificial Intelligence*. Vol. 5, p. 3.

Castillo, C., M. El-Haddad, J. Pfeffer, and M. Stempeck (2014). "Characterizing the Life Cycle of Online News Stories Using Social Media Reactions", *Proceedings of the 17th Conference on Computer Supported Cooperative Work*. ACM, pp. 211–223.

Cessie, S. le and J. van Houwelingen (1992). "Ridge Estimators in Logistic Regression", *Applied Statistics* 41.1, pp. 191–201.

Chapelle, O., V. Vapnik, O. Bousquet, and S. Mukherjee (2002). "Choosing multiple parameters for support vector machines", *Machine Learning* 46.1, pp. 131–159.

Chen, Y. and J. Wang (2004). "Image categorization by learning and reasoning with regions", *Journal of Machine Learning Research* 5, pp. 913–939.

Chickering, D. M. (1996). "Learning Bayesian networks is NP-complete", *Learning from data: Artificial intelligence and statistics V* 112, pp. 121–130.

Clyde, M. and H. Lee (2001). "Bagging and the Bayesian bootstrap", *Artificial Intelligence and Statistics*. Morgan Kaufman Publishers.

Cranor, L. and B. LaMacchia (1998). "Spam!", *Communications of the ACM* 41.8, pp. 74–83.

Dash, D. and G. F. Cooper (2004). "Model averaging for prediction with discrete Bayesian networks", *Journal of Machine Learning Research* 5, pp. 1177–1203.

Demsar, J. (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research* 7, pp. 1–30.

De Raedt, L. and A. Kimmig (2015). "Probabilistic (logic) programming concepts", *Machine Learning* 100.1, pp. 5–47.

Devroye, L. P., T. Wagner, et al. (1980). "Distribution-free consistency results in nonparametric discrimination and regression function estimation", *The Annals of Statistics* 8.2, pp. 231–239.

Dietterich, T., R. Lathrop, and T. Lozano-Pérez (1997). "Solving the multiple instance problem with axis-parallel rectangles", *Artificial Intelligence* 89.1-2, pp. 31–71.

Dietterich, T. G. (1998). "Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms", *Neural Computation* 10.7, pp. 1895–1923.

Dong, L. (2006). "A comparison of multi-instance learning algorithms". MA thesis. University of Waikato.

Dooly, D., Q. Zhang, S. Goldman, and R. Amar (2003). "Multiple Instance Learning of Real Valued Data", *Journal of Machine Learning Research* 3, pp. 651–678. ISSN: 1532-4435.

Doran, G. and S. Ray (2014). "A theoretical and empirical analysis of support vector machine methods for multiple-instance classification", *Machine Learning* 97.1-2, pp. 79–102. ISSN: 0885-6125.

Dries, A., S. Nijssen, and L. De Raedt (2012). "BiQL: a query language for analyzing information networks", *Bisociative Knowledge Discovery*. Springer, pp. 147–165.

Eaton, D. and K. Murphy (2012). "Bayesian structure learning using dynamic programming and MCMC", *arXiv preprint arXiv:1206.5247*.

Efron, B. (1983). "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation". English, *Journal of the American Statistical Association* 78.382, pp. 316–331. ISSN: 01621459.

Efron, B. and R. Tibshirani (June 1997). "Improvements on Cross-Validation: The .632+ Bootstrap Method", *Journal of the American Statistical Association* 92.438, pp. 548–560.

Efron, B. and R. J. Tibshirani (1994). *An introduction to the bootstrap*. CRC press.

Elidan, G. (2011). "Bagged structure learning of bayesian network", *International Conference on Artificial Intelligence and Statistics*, pp. 251–259.

Elisseeff, A., M. Pontil, et al. (2003). "Leave-one-out error and stability of learning algorithms with applications", *NATO science series sub series iii computer and systems sciences* 190, pp. 111–130.

Feller, W. (1968). *An introduction to probability theory and its applications: volume I*. Vol. 3. John Wiley & Sons New York. Chap. 10.

Fisher, R. A. (1936). "The use of multiple measurements in taxonomic problems", *Annals of human genetics* 7.2, pp. 179–188.

Foulds, J. and E. Frank (2010). "A Review of Multi-Instance Learning Assumptions", *Knowledge Engineering Review* 25, pp. 1–25.

Frank, E. and X. Xu (2003). *Applying propositional learning algorithms to multi-instance data*. Tech. rep. University of Waikato.

Freund, Y. and R. Schapire (1995). "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting", *Proceedings of the 2d European Conference on Computational Learning Theory*. Springer-Verlag, pp. 23–37.

Friedman, N., M. Goldszmidt, and A. Wyner (1999). "Data analysis with Bayesian networks: A bootstrap approach", *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 196–205.

Friedman, N. and D. Koller (2003). "Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks", *Machine learning* 50.1-2, pp. 95–125.

Friedman, N., I. Nachman, and D. Peér (1999). "Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm", *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 206–215.

Fung, G., M. Dundar, B. Krishnapuram, and R. Rao (2007). "Multiple instance learning for computer aided diagnosis", *Advances in Neural Information Processing Systems 19 (NIPS)*. MIT Press, pp. 425–432.

Fu, Z., A. Robles-Kelly, and J. Zhou (2011). "MILIS: Multiple Instance Learning with Instance Selection", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.5, pp. 958–977. ISSN: 0162-8828.

Gärtner, T., P. Flach, A. Kowalczyk, and A. Smola (2002). "Multi-Instance Kernels", *Proceedings of the 19th International Conference on Machine Learning*. Morgan Kaufmann, pp. 179–186.

Giraud-Carrier, C. (2008). "Proceedings of the 7th international conference on machine learning and applications",

González-Brenes, J. P. and C. Matías (2011). *RTA Freeway Travel Time Prediction*. `http://blog.kaggle.com/2011/03/25/jose-p-gonzalez`. Accessed: 2014-07-30.

Grandvalet, Y. and Y. Bengio (2006). "Hypothesis testing for cross-validation", *Montreal Universite de Montreal, Operationnelle DdIeR* 1285.

Greiner, R. (1996). "PALO: A probabilistic hill-climbing algorithm", *Artificial Intelligence* 84.1-2, pp. 177–208.

Guns, T., A. Dries, G. Tack, S. Nijssen, and L. De Raedt (2013). "MiningZinc: A Modeling Language for Constraint-Based Mining.", *IJCAI*. Vol. 13, pp. 1365–1372.

Hanczar, B. and E. R. Dougherty (2010). "On the comparison of classifiers for microarray data", *Current Bioinformatics* 5, pp. 29–39.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning*. 2nd ed. Springer.

Hothorn, T., K. Hornik, and A. Zeileis (2006). "Unbiased recursive partitioning: A conditional inference framework", *Journal of Computational and Graphical Statistics* 15.3, pp. 651–674.

Hulten, G. and P. Domingos (2002). "Mining complex models from arbitrarily large databases in constant time", *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 525–531.

Kearns, M. and D. Ron (1999). "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation", *Neural computation* 11.6, pp. 1427–1453.

Kim, H. (2009). "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap", *Computational Statistics & Data Analysis* 53.11, pp. 3735–3745.

Kohavi, R. (1996). "Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid", *Proceedings of the 2d International Conference on Knowledge Discovery and Data Mining*. Vol. 7. AAAI Press, pp. 202–207.

Kohavi, R. (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection", *International Joint Conference on Artificial intelligence (IJCAI)*. 2. Morgan Kaufmann Publishers Inc., pp. 1137–1143.

Kotsiantis, S., D. Kanellopoulos, and P. Pintelas (2006). "Data preprocessing for supervised leaning", *International Journal of Computer Science* 1.2, pp. 111–117.

Kumar, R., D. Lokshtanov, S. Vassilvitskii, and A. Vattani (2013). "Near-optimal bounds for cross-validation via loss stability", *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 27–35.

Laplace, P. (1840). "Essai Philosophique sur les Probabilités (English translation: A Philosophical Essay on Probability, Dover, New York, 1952)", *Bachelier, Paris*.

Leek, J. (2013). *The vast majority of statistical analysis is not performed by statisticians*. `http://simplystatistics.org/2013/06/14/the-vast-majority-of-statistical-analysis-is-not-performed-by-statisticians/`. Accessed: 2013-06-24.

Lerman, K. and T. Hogg (2010). "Using a Model of Social Dynamics to Predict Popularity of News", *Proceedings of the 19th International Conference on World Wide Web*. ACM, pp. 621–630.

Lichman, M. (2013). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml`.

Liu, G., J. Wu, and Z. Zhou (2012). "Key Instance Detection in Multi-Instance Learning.", *Proceedings of the 4th Asian Conference on Machine Learning*. Ed. by S. C. H. Hoi and W. L. Buntine. Vol. 25. JMLR Proceedings. JMLR.org, pp. 253–268.

Li, Y., J. Kwok, I. Tsang, and Z. Zhou (2009). "A Convex Method for Locating Regions of Interest with Multi-instance Learning", *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*. Bled, Slovenia: Springer-Verlag, pp. 15–30. ISBN: 978-3-642-04173-0.

Long, P. and L. Tan (1998). "PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples", *Machine Learning* 30.1, pp. 7–21.

Mandel, M. and D. Ellis (2008). "Multiple-instance learning for music information retrieval", *Proceedings of the 9th International Conference on Music Information Retrieval*, pp. 577–582.

Markatou, M., H. Tian, S. Biswas, and G. Hripcsak (Dec. 2005). "Analysis of Variance of Cross-Validation Estimators of the Generalization Error", *Journal of Machine Learning Research* 6, pp. 1127–1168. ISSN: 1532-4435.

Markovitch, S. and D. Rosenstein (2002). "Feature generation using general constructor functions", *Machine Learning* 49.1, pp. 59–98.

Maron, O. and T. Lozano-Pérez (1998). "A framework for multiple-instance learning", *Advances in Neural Information Processing Systems 11 (NIPS)*. MIT Press, pp. 570–576.

Maron, O. and A. Ratan (1998). "Multiple-instance learning for natural scene classification", *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, pp. 341–349.

McIntyre, S. and R. McKitrick (2005). "Hockey sticks, principal components, and spurious significance", *Geophysical Research Letters* 32.3.

Molinaro, A. M., R. Simon, and R. M. Pfeiffer (2005). "Prediction error estimation: a comparison of resampling methods", *Bioinformatics* 21.15, pp. 3301–3307.

Nadeau, C. and Y. Bengio (1999). "Inference for the Generalization Error", *Advances in Neural Information Processing Systems 12 (NIPS)*, pp. 307–313.

Neapolitan, R. E. et al. (2004). *Learning bayesian networks*. Vol. 38. Pearson Prentice Hall. Chap. Part II.

Park, T. and G. Casella (2008). "The bayesian lasso", *Journal of the American Statistical Association* 103.482, pp. 681–686.

Pearl, J. (2003). "Causality: models, reasoning and inference", *Econometric Theory* 19.675-685, p. 46.

Perlich, C., F. Provost, and J. S. Simonoff (Dec. 2003). "Tree Induction vs. Logistic Regression: A Learning-curve Analysis", *Journal of Machine Learning Research* 4, pp. 211–255.

Peterson, M. (2017). *An introduction to decision theory*. Cambridge University Press.

Pfahringer, B., H. Bensusan, and C. Giraud-Carrier (2000). "Meta-Learning by Landmarking Various Learning Algorithms", *Proceedings of the International Conference on Machine learning (ICML)*, pp. 743–750.

Platt, J. (1999). "Fast training of support vector machines using sequential minimal optimization", *Advances in Kernel Methods-Support Vector Learning* 208, pp. 185–208.

Quinlan, J. (2003). *C4.5: programs for machine learning*. Morgan Kaufmann.

Raftery, A. E., T. Gneiting, F. Balabdaoui, and M. Polakowski (2005). "Using Bayesian model averaging to calibrate forecast ensembles", *Monthly Weather Review* 133.5, pp. 1155–1174.

Raman, S., T. J. Fuchs, P. J. Wild, E. Dahl, and V. Roth (2009). "The Bayesian group-lasso for analyzing contingency tables", *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 881–888.

Ramdas, A. and A. Balsubramani (2015). "Sequential Nonparametric Testing with the Law of the Iterated Logarithm", *stat* 1050, p. 10.

Ramon, J. and L. De Raedt (2000). "Multi instance neural networks", *Proceedings of the 17th International Conference on Machine Learning, Workshop on Attribute-Value and Relational Learning*, pp. 53–60.

Ray, S. and M. Craven (2005). "Supervised versus multiple instance learning: An empirical comparison", *Proceedings of the 22d International Conference on Machine Learning*. Vol. 22. ACM Press, pp. 697–704.

Ray, S., S. Scott, and H. Blockeel (2011). "Multi-instance learning", *Encyclopedia of Machine Learning, first edition*, pp. 701–710.

Rubin, D. B. et al. (1981). "The bayesian bootstrap", *The annals of statistics* 9.1, pp. 130–134.

Schmidt, M., A. Niculescu-Mizil, K. Murphy, et al. (2007). "Learning graphical model structure using L1-regularization paths", *AAAI Conference on Artificial Intelligence*. Vol. 7, pp. 1278–1283.

Sculley, D. (2012). *Results from a semi-supervised feature learning competition*. `http://eecs.tufts.edu/~dsculley/papers/semisupervised-feature-learning-competition.pdf`. Accessed: 2014-07-30.

Settles, B., M. Craven, and S. Ray (2008). "Multiple-instance active learning", *Advances in Neural Information Processing Systems 20 (NIPS)*. MIT Press, pp. 1289–1296.

Shao, J., D. He, and Q. Yang (2008). "Multi-semantic Scene Classification Based on Region of Interest", *Proceedings of the International Conference on Computational Intelligence for Modelling Control & Automation*. Washington, DC, USA: IEEE Computer Society, pp. 732–737. ISBN: 978-0-7695-3514-2.

Simmons, J. P., L. D. Nelson, and U. Simonsohn (2011). "False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant", *Psychological science* 22.11, pp. 1359–1366.

Smith, J., J. Everhart, W. Dickson, W. Knowler, and R. Johannes (1988). "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus", *Proceedings of the Symposium on Computer Applications and Medical Care*, pp. 261–265.

Spirtes, P. (2010). "Introduction to causal inference", *Journal of Machine Learning Research* 11.May, pp. 1643–1662.

Strobl, C., A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis (2008). "Conditional Variable Importance for Random Forests", *BMC Bioinformatics* 9.307.

Szabo, G. and B. A. Huberman (Aug. 2010). "Predicting the Popularity of Online Content", *Communications of the ACM* 53.8, pp. 80–88.

Tao, Q., S. Scott, N. Vinodchandran, and T. Osugi (2004). "SVM-based generalized multiple-instance learning via approximate box counting", *Proceedings of the 21th International Conference on Machine learning*. Morgan Kaufmann.

Teyssier, M. and D. Koller (2012). "Ordering-based search: A simple and effective algorithm for learning Bayesian networks", *arXiv preprint arXiv:1207.1429*.

Tong, S. and D. Koller (2001). "Active learning for structure in Bayesian networks", *International Joint Conference on Artificial intelligence (IJCAI)*. Vol. 17. 1, pp. 863–869.

Tragante do O, V., D. Fierens, and H. Blockeel (2011). "Instance-level accuracy versus bag-level accuracy in multi-instance learning", *Proceedings of the 23d Benelux Conference on Artificial Intelligence*. URL: `https://lirias.kuleuven.be/handle/123456789/316681`.

Vanschoren, J. and H. Blockeel (2008). "Investigating classifier learning behavior with experiment databases", *Data Analysis, Machine Learning and Applications*, pp. 421–428.

Vanwinckelen, G. and H. Blockeel (2012). "On estimating model accuracy with repeated cross-validation", *Proceedings of the 21st Belgian-Dutch Conference on Machine Learning (BeneLearn)*. Belgium, May.

– (2013). "A declarative query language for statistical inference", *ECML/PKDD Workshop on Languages for Data Mining and Machine Learning*. Czech Republic, September.

– (2014a). "A meta-learning system for multi-instance classification", *ECML/PKDD Workshop on Learning from Multiple Contexts*. France, September.

– (2014b). "Look before you leap: Some insights into learner evaluation with cross-validation", *ECML/PKDD Workshop on Statistically Sound Data Mining*. France, September.

– (2014c). "Look before you leap: Some insights into learner evaluation with cross-validation (Poster)", *Intelligent Data Analysis*. Belgium, October.

Vanwinckelen, G. and W. Meert (2014). "Predicting the popularity of online articles with random forests", *ECML/PKDD Workshop on Predictive Web Analytics*. France, September.

Vanwinckelen, G., M. V. Otterlo, K. Driessens, and S. Pollin (2011). "Power control for secondary users based on distributed measurements", *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*. Germany, May.

Vanwinckelen, G., V. Tragante Do O, D. Fierens, and H. Blockeel (2014). "Instance-level accuracy versus bag-level accuracy in multi-instance learning", *Data Mining and Knowledge Discovery*.

Vanwinckelen, G., D. Verbeeck, W. Meert, and H. Blockeel (2013). "Optimal mobile connectivity using a practical coverage map", *LICT Scientific symposium on adaptivity in ICT*. Belgium, September.

Vilalta, R. and Y. Drissi (2002). "A perspective view and survey of meta-learning", *Artificial Intelligence Review* 18.2, pp. 77–95.

Von Neumann, J. and O. Morgenstern (2007). *Theory of games and economic behavior*. Princeton University Press.

Wang, J. and J. Zucker (2000). "Solving the multiple-instance problem: A lazy learning approach", *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann, pp. 1119–1126.

Wilcox, R. R. (2010). *Fundamentals of modern statistical methods: Substantially improving power and accuracy*. Springer Science & Business Media. Chap. 7.

Witten, I. H., E. Frank, M. A. Hall, and C. J. Pal (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Witten, I. and E. Frank (2005). *Data Mining: Practical machine learning tools and techniques*.

Wolpert, D. H. (Oct. 1996). "The Lack of a Priori Distinctions Between Learning Algorithms", *Neural Computation* 8.7, pp. 1341–1390.

Xu, X. (2003). "Statistical learning in multiple instance problems". MA thesis. Department of Computer Science, University of Waikato.

Xu, X. and E. Frank (2004). "Logistic regression and boosting for labeled bags of instances", *Proceedings of the Pacific Asia Conference on Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science. Springer, pp. 272–281.

Yeh, I., K. Yang, and T. Ting (2009). "Knowledge discovery on RFM model using Bernoulli sequence", *Expert Systems With Applications* 36.3P2, pp. 5866–5871.

Zhang, C. (2015). "DeepDive: a data management system for automatic knowledge base construction". PhD thesis. University of Wisconsin-Madison.

Zhang, M. (2009). "Generalized multi-instance learning: Problems, algorithms and data sets", *Proceedings of the WRI Global Congress on Intelligent Systems*. Vol. 3. Morgan Kaufmann, pp. 539–543.

Zhang, Q. and S. Goldman (2001). "EM-DD: An improved multiple-instance learning technique", *Advances in Neural Information Processing Systems 14 (NIPS)*. MIT Press, pp. 1073–1080.

Zhou, Z., X. Xue, and Y. Jiang (2005). "Locating regions of interest in CBIR with multi-instance learning techniques", *Proceedings of the 18th Australian Joint conference on Advances in Artificial Intelligence*. Springer-Verlag, pp. 92–101. ISBN: 3-540-30462-2, 978-3-540-30462-3.

Zhou, Z. and M. Zhang (2003). "Ensembles of multi-instance learners", *Proceedings of the 14th European Conference on Machine Learning*. Lecture Notes in Computer Science. Springer, pp. 492–502.

Zucker, J. and Y. Chevaleyre (2001). "Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem", *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*. Springer, pp. 204–214.

List of Publications

Journal Article

G. Vanwinckelen, V. Tragante Do O, D. Fierens, and H. Blockeel (2014). "Instance-level accuracy versus bag-level accuracy in multi-instance learning", *Data Mining and Knowledge Discovery*

Conference Papers

G. Vanwinckelen, M. V. Otterlo, K. Driessens, and S. Pollin (2011). "Power control for secondary users based on distributed measurements", *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*. Germany, May

G. Vanwinckelen and H. Blockeel (2012). "On estimating model accuracy with repeated cross-validation", *Proceedings of the 21st Belgian-Dutch Conference on Machine Learning (BeneLearn)*. Belgium, May

Workshop Papers

G. Vanwinckelen and W. Meert (2014). "Predicting the popularity of online articles with random forests", *ECML/PKDD Workshop on Predictive Web Analytics*. France, September

G. Vanwinckelen and H. Blockeel (2014a). "A meta-learning system for multi-instance classification", *ECML/PKDD Workshop on Learning from Multiple Contexts*. France, September

G. Vanwinckelen and H. Blockeel (2014b). "Look before you leap: Some insights into learner evaluation with cross-validation", *ECML/PKDD Workshop on Statistically Sound Data Mining*. France, September

G. Vanwinckelen and H. Blockeel (2014c). "Look before you leap: Some insights into learner evaluation with cross-validation (Poster)", *Intelligent Data Analysis*. Belgium, October

Abstracts and Posters

G. Vanwinckelen and H. Blockeel (2013). "A declarative query language for statistical inference", *ECML/PKDD Workshop on Languages for Data Mining and Machine Learning*. Czech Republic, September

G. Vanwinckelen, D. Verbeeck, W. Meert, and H. Blockeel (2013). "Optimal mobile connectivity using a practical coverage map", *LICT Scientific symposium on adaptivity in ICT*. Belgium, September

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
ARTIFICIAL INTELLIGENCE
Celestijnenlaan 200A box 2402
B-3001 Heverlee