Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Martin Možina

Argumentirano strojno učenje

DOKTORSKA DISERTACIJA

Ljubljana, 2009

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Martin Možina

Argumentirano strojno učenje

DOKTORSKA DISERTACIJA

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2009

University of Ljubljana
Faculty of Computer and Information Science

Martin Možina

Argument Based Machine Learning

DOCTORAL DISSERTATION

Supervisor: Acad. Prof. Dr. Ivan Bratko

Ljubljana, 2009

Povzetek

V pričujočem delu opišemo argumentirano strojno učenje (*angl.* Argument-Based Machine Learning, s kratico ABML), ki združuje strojno učenje s tehnikami argumentacije. Argumenti so orodje, s katerim domenski strokovnjaki razlagajo relacije med atributi in razredom za posamezne učne primere. Primeri razloženi z argumenti se imenujejo argumentirani učni primeri. Cilj učenja v ABML je poiskati tako hipotezo v prostoru vseh možnih hipotez, ki uporabi dane argumente za razlago razreda argumentiranih učnih primerov. S tem argumenti omejujejo prostor sprejemljivih hipotez, kar zmanjša možnost prevelikega prilagajanja podatkom in hkrati vodijo učni algoritem k hipotezam, ki so razumljivejše.

Ena glavnih razlik med ABML in ostalimi metodami, ki omogočajo uporabo predznanja, je v načinu izvabljanja znanja s strani domenskih strokovnjakov. Problem večine obstoječih metod za strojno učenje je, da zahtevajo splošno domensko znanje. Pridobivanje takega predznanja predstavlja problem, imenovan Feigenbaumovo ozko grlo, saj domenski strokovnjaki pogosto niso sposobni dobro izraziti njihovega splošnega znanja. Izkaže se, da jim je veliko lažje razlagati znanje s pomočjo argumentov na konkretnih primerih. V okviru tega smo razvili algoritem za odkrivanje kritičnih primerov; to so primeri, ki jih trenutno naučena hipoteza ne zna dobro napovedovati. V procesu učenja se tako strokovnjakom pokažejo samo ti primeri, saj bi bilo nemogoče pričakovati, da nam bodo razložili prav vse učne primere. S tem še vedno dobimo vse relevatno znanje, ostalo znanje pa metoda odkrije iz učnih podatkov.

Na podlagi osnovih principov argumentiranega učenja smo razvili metodo ABCN2, razširjeno različico znanega algoritma za učenje pravil CN2. ABCN2 se nauči kar se da točne množice pravil in pri tem zagotavlja, da bo med pravili, ki pokrijejo argumentirane učne primere, vsaj eno tako, ki vsebuje med svojimi pogoji razloge iz argumenta. ABCN2 smo še dodatno razširili z novo metodo EVC (*angl.* extreme value correction) za korigiranje ocenjene točnosti pravil glede na število pregledanih pravil v postopku učenja. Ta popravek se izkaže za pomembnega v argumentiranem učenju, saj se prostor preiskovanja razlikuje od pravila do pravila glede na dolžino ustreznih argumentov. Poleg tega se EVC izkaže za koristno orodje tudi pri učenju navadnih pravil (brez argumentov), saj so napovedne točnosti pravil z uporabo EVC statistično boljše od pravil brez te korekcije.

Delo zaključimo z eksperimentalno primerjavo ABCN2 in CN2 ter nekaterimi ostalimi metodami strojnega učenja. ABCN2 se je izkazala za boljšo metodo od

CN2 glede na točnost in razumljivost pravil v prav vseh izvedenih poskusih, vendar zaradi razmeroma majhnega števila poskusov tega nismo mogli statistično dokazati. Ker je izvajanje primerjave na dovolj velikem vzorcu domen praktično neizvedljivo, saj pri ABCN2 vedno potrebujemo sodelovanje z domenskim strokovnjakom, kar je časovno potratno, smo se odločili za analizo vpliva naključnih argumentov na uspešnost učenja z ABCN2. Izkaže se, da naključni atributi ne poslabšajo točnosti naučenih modelov in iz tega sklepamo, da argumenti ne morejo poslabšati točnosti modela, če so le boljši od naključnih.

Ključne besede

- umetna inteligenca, strojno učenje, argumentacija
- učenje pravil, učenje s predznanjem, domensko znanje
- argumentirano strojno učenje, ABML, CN2, ABCN2, korekcija ekstremnih vrednosti

Abstract

The Thesis presents a novel approach to machine learning, called ABML (argument based machine learning). This approach combines machine learning from examples with some concepts from the field of defeasible argumentation, where arguments are used together with learning examples by learning methods in the induction of a hypothesis. An argument represents a relation between the class value of a particular learning example and its attributes and can be regarded as a partial explanation of this example. We require that the theory induced from the examples explains the examples in terms of the given arguments. Thus arguments constrain the combinatorial search among possible hypotheses, and also direct the search towards hypotheses that are more comprehensible in the light of expert's background knowledge. Arguments are usually provided by domain experts. One of the main differences between ABML and other knowledge-intensive learning methods is in the way the knowledge is elicited from these experts. Other methods require general domain knowledge, that is knowledge valid for the entire domain. The problem with this is the difficulty that experts face when they try to articulate their global domain knowledge. On the other hand, as arguments contain knowledge specific only to certain situations, they need to provide only local knowledge for the specific examples. Experiments with ABML and other empirical observations show that experts have significantly less problems while expressing such local knowledge. Furthermore, we define the ABML loop that iteratively selects critical learning examples, namely examples that could not be explained by the current hypothesis, which are then shown to domain experts. Using this loop, the burden that lies on experts is further reduced (only some examples need to be explained) and only relevant knowledge is obtained (difficult examples). We implemented the ABCN2 algorithm, an argument-based extension of the rule learning algorithm CN2. The basic version of ABCN2 ensures that rules classifying argumented examples will contain the reasons of the given arguments in their condition part. We furthermore improved the basic algorithm with a new method for evaluation of rules, called extreme value correction (EVC), that reduces the optimism of evaluation measures due to the large number of rules tested and evaluated during the learning process (known as the multiple comparison procedures problem). This feature is critical for ABCN2, since arguments given to different examples have different number of reasons and therefore differently constrain the space for different rules. Moreover, as shown in the dissertation, using this method in CN2 (without arguments) results in significantly more accurate models as compared to the original

CN2. We conclude this work with a set of practical evaluations and comparisons of ABCN2 to other machine learning algorithms on several data sets. The results favour ABCN2 in all experiments, however, as each experiment requires a certain amount of time due to involvement of domain experts, the number of experiments is not large enough to allow a valid statistical test. Therefore, we explored the capability of ABCN2 to deal with erroneous arguments, and showed in the dissertation that using false arguments will not decrease the quality of the induced model. Hence, ABCN2 can not perform worse than CN2, but it can perform better given the quality of arguments is high enough.

Keywords

- artificial intelligence, machine learning, argumentation
- rule induction, knowledge-intensive learning, background knowledge, domain knowledge
- argument based machine learning, ABML, CN2, ABCN2, extreme value correction

IZJAVA O AVTORSTVU
doktorske disertacije

Spodaj podpisani *Martin Možina*,
z vpisno številko *63970103*,

sem avtor doktorske disertacije z naslovom

Argumentirano strojno učenje
(angl. Argument Based Machine Learning)

S svojim podpisom zagotavljam, da:

- sem doktorsko disertacijo izdelal samostojno pod vodstvom mentorja *akad. prof. dr. Ivana Bratka*
- so elektronska oblika doktorske disertacije, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko doktorske disertacije
- in soglašam z javno objavo elektronske oblike doktorske disertacije v zbirki "Dela FRI".

V Ljubljani, dne 30. novembra 2009 Podpis avtorja:

Acknowledgements

First and foremost, my gratitude goes to Ivan Bratko, my supervisor, who gave me the opportunity to work in a relaxing, yet very productive environment. He showed me how to carry out my research at high standards and how to successfully publish my work. I was often amazed at his ability to see the big picture, even when I could not, and at his power to generate new ideas, when I had already hit the wall. I appreciate the time and effort offered by the members of my reading committee: Blaž Zupan, Sašo Džeroski, and Floriana Esposito.

I would like to thank to all members of the Artificial Intelligence Laboratory for providing a great working environment. I particularly appreciate help from Jure Žabkar, who helped a lot during the development of the ABCN2 algorithm and he also introduced me to Jerneja Videčnik, who kindly provided data and her expertise for the infections-in-the-elderly-population domain, one of the earliest experiments with ABCN2 . I sincerely thank the co-authors Janez Demšar and Blaž Zupan for helping me to get my first paper published at the ECML'04 conference, which was a very nice experience for me at that time. I also thank them for their work on Orange, a library of machine learning methods that tremendously simplified prototyping of new algorithms. Furthermore, I must thank Saša Sadikov, Matej Guid, and Jana Krivec, all experts in chess, who are developing the chess tutoring application where many machine learning problems were identified and best solved by argument-based machine learning. Following are the rest of the lab members who were always ready to listen and provide their comments (if they could only understand me): Tadej Janež, Aljaž Košmerlj, Damjan Kužnar, Blaž Strle, Tomaž Curk, Gregor Leban, Minca Mramor, Lan Umek, Marko Toplak, Miha Štajdohar, Lan Žagar and Jure Žbontar, and some past members: Peter Juvan, Daniel Vladušič, Dorian Šuc and Aleks Jakulin.

My research work in the past years was supported by two European projects, ASPIC and X-Media. I thank John Fox for coordinating ASPIC, the project that inspired most of the work in this Thesis. Trevor Bench-Capon provided the data for Welfare Benefit Approval, which was the first successful experiment with ABML. I also sincerely thank Henry Prakken, Martin Caminada, Sanjay Modgil, Matt South, and Pancho Tolchinsky for answering so many questions I had about argumentation. After ASPIC, I joined the X-Media project, lead by Fabio Ciravegna, which was conceptually a very different project. However, with Claudio Giuliano we were able to identify a way to combine learning from text with learning from data through the use of ABML. Some principles of ABML were also used in the Fiat application for noise reduction in cars. This problem was defined by Marina Giordanino and the Fiat aerodynamics department.

I should not forget to thank the ladies from the Dean's office and the Student office who helped me a lot with all sorts of administrative issues: Dragica Furlan, Milica Vidič, Maja Kerkez, Mira Škrlj, Lucija Završnik and Jasna Bevk.

This dissertation is dedicated to my parents Janez and Alenka, my brother Miha, my wife Janja and my two beautiful daughters Tjaša and Katja.

Martin Možina

Contents

Chapter 1

Introduction

Machine learning is concerned with the development of algorithms that enable computer programs to learn and improve from experience [Mit97]. The most common type of machine learning (ML) is learning from labeled examples, called also supervised inductive learning. Each example is described by a set of descriptive attributes (inputs), and a class variable (output). The task is therefore to formulate a hypothesis in some formal language that can predict outputs of examples given inputs. This new hypothesis can be used to predict the outcome of new cases, where the true values are unknown. Some learning problems tackled with inductive learning are:

- Given examples of weather situations, learn to forecast weather in the future;
- Given examples of past patients, learn to diagnose new patients;
- Given examples of chess positions, determine the relative quality of particular pieces (e.g. the goodness of a bishop from strategical point of view).

Learning examples represent the past experience; cases where the outcome is already known. The attributes describing them are usually some natural properties, which we hope will suffice to make good predictions. For example, a weather situation can be described by wind direction, temperature, and humidity, where the class could be the weather situation on the following day. We hope that running a learning algorithm on such data will provide a hypothesis giving good prediction for new cases. However, it may also happen that the learning mechanism will fail to find a hypothesis that would predict well. This can happen either because (1) the set of attributes is not comprehensive, (2) the relation between classes and attributes is a complex function and therefore hard to learn, (3) the language representing the hypotheses is inappropriate for the learning problem, (4) or the method overfits on the given training data.

Whenever learning fails to produce acceptable results, the burden of improving lies on the domain experts. If the descriptions of examples are not sufficient to explain the outputs, they need to expand the descriptions by adding additional attributes. If the language expressivity is insufficient, an alternative formal language needs to be used. When the target hypothesis is very complex and hard to find, the machine learning algorithm needs guidance to be able to find this hypothesis. Domain experts can provide their prior knowledge about the target hypothesis (*e.g.* parts of the correct hypothesis) and this knowledge can then be used to guide the learning mechanism towards those hypotheses that seem more promising to a domain expert. The problem with this approach is the difficulty that experts face when they try to articulate their global domain knowledge, known also as the knowledge acquisition problem [Fei84].

In this Thesis, we propose an alternative approach to knowledge elicitation of background knowledge. Empirical observations have shown that humans are better at providing specific explanations than providing generic knowledge of the problem. Therefore, we ask experts to explain the class of a single example with arguments for and against, where an argument can be seen as a conclusion and a set of reasons supporting this conclusion. In this sense, a learning example can be seen as a question to experts. Possible questions in the above domains are:

- Why was it raining on this particular day, while on the day before it was sunny and warm?
- Why did the infection kill this patient, given that her body temperature was normal, etc.?
- Why is the black bishop bad in a given chess position?

Although experts might be unable to provide a general theory for any of the mentioned learning problems, they do not seem to have any problems to at least provisionally answer these questions. It should be noted that an argument given by an expert can not be regarded as a logical rule, since the relation given could be valid only for the chosen learning example rather than for the whole domain.

Argument Based Machine Learning (ABML), described in this Thesis, is an extension of classical machine learning that allows the use of *local* expert's knowledge in the form of arguments. An ABML method learns from *learning examples* (as in ML) and from *arguments* given to some of the learning examples. The resulting hypothesis must correctly predict the outcomes of examples (as in ML) by using provided arguments. For example, an argument to a specific day in the weather domain could be: “It was raining because of the low air pressure on the previous day.” Then, the resulting hypothesis must mention the reasons of this argument (low air pressure) while explaining the rain on that particular day. In other words, the induced hypothesis should contain, in some way, the positive relation between low pressure and raining.

We refer to the learning examples explained with arguments as *argumented examples*. Ideally, the complete learning data set would be argumented, however, despite arguably easier elicitation of knowledge with arguments, the work of experts would still be extensive if they needed to explain all learning examples. In the dissertation, we will describe a method for selection of *critical examples*, that is, examples that can not be correctly classified by the ABML method itself. On the basis of this method, we will define the ABML refinement loop that iteratively asks experts for explanation of the most critical example, which significantly reduces the work required by the experts, while we still obtain all relevant knowledge that could not be automatically obtained.

With ABML, it is possible to tackle all four above-mentioned problems in machine learning. To begin with, since experts are not restricted by the given descriptive attributes, they will often refer to attributes not currently present in the domain, hence suggesting these attributes should be added. Similarly, they could use specific constructs of attributes in their explanations that can not be described in the selected hypothesis language (*e.g.* the sum of attributes in propositional rule learning). Moreovoer, as experts are asked to explain only difficult learning examples and the induced hypothesis needs to be consistent with the arguments, these resulting hypothesis will contain also complex relations between output and inputs. And finally, as the arguments constrain the search space among possible hypotheses, the probability of inducing a hypothesis that overfits training data is reduced.

A final question is, whether we can trust experts’ interpretations of learning examples, especially since the experts are not always able to perform the prediction by themselves. Consider, for example, a weather specialist; they will almost always be able to provide an explanation of past weather situations, however how good is their prognosis? A striking asset of ABML is that experts do not need to be concerned whether their knowledge is absolutely correct. They can freely provide their impressions or merely express an opinion why they think this particular example has the output as given, and the ABML method should still benefit from such knowledge. In the Thesis, we shall provide evidence of this claim by demonstrating that even completely random knowledge does not hurt the accuracy of learned hypothesis.

1.1 Overview of the dissertation

The dissertation is organised in three parts and thirteen chapters.

In **Part I**, we give an introduction to machine learning and argumentation. First, we formally define classical machine learning and motivate the use of domain knowledge in learning. Then, we give a brief overview of different approaches in machine learning that can currently exploit domain knowledge. In Chapter 3, we describe some basic concepts of argumentation theory. Although these two chapters describe relevant information, they are not required to understand the rest of the dissertation. A reader can freely skip these two chapters and proceed to Chapter 4.

Part II contains a definition of ABML and describes the ABCN2 method. In Chapter 4, we start with a motivating example to illustrate the basic notions of ABML and continue with a logical formalisation of ABML. We conclude the chapter with some guidelines for building argument-based methods. The motivating example itself is enough for understanding the rest of the Thesis, therefore a reader not interested in a formal logical definition of ABML can skip most of Chapter 4. We begin Chapter 5 with a definition of argumented examples accepted by ABCN2. After, we look at the algorithm ABCN2 itself, describe the concept of AB-covering and propose some changes of the basic algorithm to improve its time efficiency. The last section of this chapter gives some details of the actual implemented product. In Chapter 6 and 7, we introduce extreme value correction for probability estimates and PILAR classification technique. They are both techniques required for efficient learning of rules from argumented examples in noisy domains. Moreover, as we will show, they also improve the quality of rule learning itself (without arguments). The last chapter of this part describes an ABML loop that iteratively selects critical examples (examples that the current hypothesis can not explain very well) that would improve the induced hypothesis the most if explained by the expert.

Part III contains chapters describing experimental evaluation of ABCN2. In Chapter 9 we begin with some basic experiments with ABCN2 to illustrate its core idea, where an argument specifies a relation between the current set of attributes and the class value. In the following chapter, we will describe some experiments where some arguments mention reasons that are not trivially represented by the current set of attributes. The next chapter (11) will demonstrate how relevant arguments can be extracted from text sources and how useful is this approach for ABML. In the last chapter of this part, we will cope with the problem of erroneous arguments and whether they can hurt the performance of ABCN2.

Chapter 13 concludes the dissertation and summarises the main findings and provides some pointers for further work.

1.2 Contributions of the dissertation

The main contributions of the dissertation are:

- Definition of general ABML principle within the Dung's argumentation framework and a set of guidelines for extending a machine learning algorithms into its argument-based version.
- Definition of an argumented example and constraints that arguments present for the induced hypothesis.
- Development and implementation of ABCN2, a tool for learning classification rules from argumented examples.
- Development of extreme value correction for probability estimates (EVC) that can remove optimism in evaluation of rules, which is due to extensive search for the best rule. We demonstrated that this method is useful for both classical rule learning and argument-based rule learning.
- A new approach for classification from rules named PILAR. It uses EVC probabilities in classification rather than just class distributions. Moreover, the approach can combine any method with rule learning, and can therefore be seen as an argument-based extension of any machine learning algorithm.
- An algorithm for construction of arguments from free text.
- Experimental evaluation of ABCN2 on several domains.

All above mentioned methods are implemented within the Orange data mining suite [DZ04], and are publicly available at `www.ailab.si/martin/abml`.

The originality and proprietary of mentioned contributions can be proven by a list of relevant publications. The ABCN2 algorithm, argumented examples, and some ABML basics were published in [MvB07], and the initial ABML idea was published in [BM04]. The extreme value correction method was published in [MDvB06], and the PILAR algorithm in [MB08]. The algorithm for construction of arguments from text that can be used in ABML was published in [MGB09]. Most of these publication contained parts of evaluation described in this Thesis, however, there are also some published works (see [MvBC^{+}06; MGK^{+}08; vMVB06]) that focused mostly on application of ABML.

Part I

Fundamental Principles and Related Work

Chapter 2

Expert Knowledge in Machine Learning

Most of the machine learning algorithms only have very limited capability to exploit domain expert knowledge (also called prior knowledge or background knowledge) that might be provided along the raw data. They implicitly assume that a machine learning expert will transform the domain description (e.g. extend the feature space) according to the given expert knowledge, which should facilitate learning. A small minority of algorithms, however, can directly exploit given prior knowledge by constraining the space of hypotheses. Argument Based Machine Learning is also a paradigm that follows the latter philosophy. In this chapter, we will formalise machine learning and lay out some convincing arguments why, on their own, machine learning procedures by itself may not succeed without the use of additional knowledge provided from experts. Moreover, these formalisations will help us later to understand the difference between the argument-based approach and the classical machine learning.

2.1 Machine learning

This Thesis is concerned with supervised learning from examples, sometimes referred to as inductive learning (or inductive generalisation), that learns the function between inputs and outputs of provided examples. Inductive learning can be also reckoned as a special type of programming, in which the programmer provides examples of inputs and outputs, and the machine learning algorithm returns a subroutine that computes outputs of examples given inputs. Inductive learning is commonly used to solve many real world problems, especially in areas where gathering measurements is relatively easy, like engineering or medicine, but where explicit relations between inputs and outputs are unknown or too complex.

An example in inductive learning is a pair (x, y), where x is a description of the example (e.g. a set of facts in first-order logic) and y is the class value (or a set of classes) of the example. It is assumed that y depends on x, i.e. $y = f(x)$, where function f is unknown. The task of machine learning is thus to find an approximation of f from a number of given learning examples. In other words, the problem of learning from examples is usually stated as:

1. Given a set of examples
2. Find a hypothesis (approximation of f) that is consistent with the examples

A hypothesis is consistent with given learning examples, if it agrees with all the data, namely, correctly predicts class value for all learning examples. Due to noise or to prevent overfitting, a full consistency is rarely required, but machine learning algorithms rather try to learn as accurate hypotheses as possible.

We will formally state the above description of learning in logical terms. This formalisation is a variant of the one described by Russel and Norvig [RN03]. Let the learning examples and hypothesis be logical sentences:

- D_e is the conjunction of example e descriptions,
- C_e is the example e's classification, and
- H is the hypothesis.

Then, the learning algorithm must find such a hypothesis H that satisfies the following formula:

$$\forall e, H \wedge D_e \vdash C_e \tag{2.1}$$

Our definition contrasts the Russel and Norvig's in two details. We used the concept of logical derivation $\vdash$ instead of logical entailment $\models$, which will help us later to define argument based machine learning. We also assumed that each example has only one classification. The inputs to machine learning algorithm are descriptions D_e and classifications C_e for all learning examples, and the algorithm is supposed to automatically construct such a hypothesis H that would make the above formula true; classification C_e of an example e should be logically derived (explained) from its descriptions D_e and the hypothesis H. From now on we shall refer to the formula 2.1 as the *derivation constraint*.

2.2 Why use domain knowledge?

A fundamental problem of inductive learning is to select a hypothesis that will generalise well, that is, it will be consistent with all examples, even with yet unseen examples. Let us assume that the space of possible hypotheses is huge and we can expect several hypotheses to be fully consistent with learning data. The critical question is thus, which of those hypotheses will generalise well. In the literature of machine learning [Mit97], we can find three frequently used approaches for selecting the most promising hypothesis: preferring simpler hypotheses, combining several hypotheses (ensemble methods), and constraining hypotheses space with domain expert knowledge. In the remainder of this section, we shall explain when and why should we exercise the third solution.

The first solution addresses the problem of selecting the best hypothesis by biasing learning towards simpler hypotheses, i.e. applying Occam's razor. Occam's razor, a principle attributed to the 14th-century English logician William of Ockham, is commonly understood by machine learning researchers as [Dom99]:

> Given two models with the same training-set error, the simpler one should be preferred because it is likely to have lower generalisation error.

The *training-set error* is the error rate of the hypothesis on learning examples, and the *generalisation error* is the error on testing (yet unseen) examples. There are several paradigms in machine learning that directly or indirectly implement this principle; in Bayesian learning, prior probabilities are often used to penalise complex hypotheses[Mit97], in minimum description length (MDL) principle [Ris78], simpler hypotheses usually have shorter description codes, and also in different approaches to pruning simple hypotheses are always preferred.

Indeed, it was shown that all approaches following the Occam's razor principle usually learn more accurate hypotheses, therefore, one could easily incorrectly conclude that more complex hypotheses will have a higher generalisation error than the simpler ones - complex hypotheses are prone to overfitting. However, Jensen and Cohen [JC00] showed that overfitting does not depend on complexity itself, but on the number of compared hypotheses in the learning procedure, since the probability of finding a hypothesis that fits the learning examples by chance increases with the number of compared hypotheses. It is therefore not the simplicity that makes Occam's razor work, but the fact that it effectively reduces the search space, and consequently reduces overfitting of the method. For example, if we would select a small number of complex candidate hypotheses in advance (e.g. a few decision trees with 100 nodes), and one of them would be consistent with learning data, we could confidently believe that it will generalise well, since it was selected from only a few hypotheses. An extensive study of Occam's razor and its problems can be found in [Dom99].

Although preferring simple hypotheses does often improve the accuracy of learned hypotheses, it will prevent us to find the best hypothesis in domains where the target concept is complex. This is one of the reasons for shift of research in machine learning to more statistical methods, e.g. support vector machines (SVM) [Vap95] and ensemble methods like boosting [FS97], bagging [Bre96] and random forests [Bre01]. SVM method starts by blowing up the original feature space into high degree polinomials or even monomials, and learns a hypothesis in the new feature space, which makes the new hypothesis inevitably complex. The overfitting in SVM is prevented by a technique called regularisation [Tib96], which constraints the values of hypothesis' parameters, enforcing the classifier to use many "complex" features in classification. The ensemble methods follow a similar strategy of that of SVM; they induce several classifiers (usually decision trees) from data by either varying the training set or other factors, which are then combined by some voting strategy. Although a single classifier in the ensemble can be complex (like new features in SVM), ensemble methods prevent overfitting by averaging several complex classifiers, where each of them has only a marginal influence on the final classification. These methods were shown to consistently achieve better hypotheses than their single hypothesis counterparts, because they reduce the following three types of errors occurring in machine learning [Die02]:

Statistical error occurs when the space of hypotheses is large given data and several hypotheses are consistent with data; in such situations averaging over all consistent hypotheses will reduce the risk of finding one that explains learning examples only by chance.

Representational error is a result of a inappropriate hypothesis representation space, which does not contain the best hypothesis; averaging over several hypotheses extends the space of hypotheses and can approximate better than a single hypothesis.

Computational error occurs when an algorithm uses a heuristic to search through a space of hypotheses and cannot guarantee to find the best hypothesis (stops in local minima); a weighted sum should reduce risk of finishing in a very bad local minimum.

Approaches like ensemble learning or SVM have shown to provide best results in terms of accuracy, but their common weakness is inability to explain their classifications; to domain experts these methods are like black boxes. In many cases, learning examples are provided by the domain experts that wish to understand better the relation between inputs and outputs [Kon93], especially if machine learning is used as a knowledge acquisition tool [Fei03].

The third alternative is to ask domain experts to provide their prior domain knowledge about the concept, which can be also used to reduce the hypotheses space. In general, domain knowledge could be any knowledge given about the learning domain that is not explicitly stated with learning examples. The use of domain knowledge in machine learning has two expected benefits:

1. induced hypothesis will be more comprehensible to domain experts, and
2. the generalisation error will be lower (accuracy will increase).

With respect to the benefit 1, prior knowledge leads to hypotheses that are consistent with expert's prior knowledge - hypotheses will explain given examples in similar terms to those used by the expert. In machine learning, there is a widespread agreement that hypotheses consistent with prior knowledge are easier for experts to comprehend. Pazzani [Paz91; PMS97] experimentally showed that people will much easier understand a new concept, if the concept is consistent with their knowledge. A more elaborate study of understanding new concepts can be found within the psychology community [Ros95; MA94], particularly in cognitive learning, which is somehow related to machine learning. They showed that when we learn about new presented materials, we always start off with our prior knowledge and try to merge them together. If the new concepts are not consistent with our prior knowledge, the new knowledge will be likely distorted or even rejected.

The reasons for the better accuracy of hypotheses are two-fold. First, prior knowledge will reduce the search space of candidate hypotheses, thus reduce the number of candidate hypotheses, which will decrease the probability to find a hypothesis that explains learning examples purely by luck. Note that this is equivalent to decreasing statistical error as defined above. Secondly, the prior knowledge will constrain methods to search subspaces that more likely to contain the correct hypothesis, which will decrease computational error. In literature, we can find some controlled experiments of utility of prior knowledge. Pazzani [PK92] showed that prior knowledge improves the accuracy of induced hypotheses, even if the knowledge is not perfect. The results of the recent challenge "Agnostic Learning vs. Prior Knowledge"*, where competitors tried to learn hypotheses with and without prior knowledge on the same data, also supports our claims. There, the hypotheses learned with prior knowledge outperformed those without prior knowledge on most of the domains. Last but not least, we will show in the last part of this Thesis that prior knowledge in form of arguments cannot hinder learning, even if it is wrong, but can significantly improve accuracy, when it is right.

2.3 An overview of using knowledge in learning

In this section, we will present an overview of machine learning methods that learn from learning examples and from domain knowledge. The list of references is by no means exhaustive, since the number of works related to this subject is too large, but we will try to mention and compare some well-known approaches. We will start with techniques that are traditionally related to domain knowledge, like EBG and ILP, and continue with some others less known to have this ability. The whole idea is to provide the reader with an impression what types of domain knowledge are there and in what way are currently used in machine learning.

Explanation-based generalisation (EBG) [MKKC86] is probably the technique that relies the most on the provided prior knowledge. EBG uses prior knowledge to explain individual learning examples, and the "hypothesis" is then a logical generalisation of these explanations. Note that EBG is not inductive learning; it assumes perfect and complete knowledge of the domain, which makes it not applicable in domains where complete knowledge is unavailable.

In other cases, when domain knowledge is only partially provided and cannot completely explain learning examples by itself, we need to induce a hypothesis using prior knowledge and learning examples. This problem was largely studied in the

*url: `http://www.agnostic.inf.ethz.ch/`

field of inductive logic programming (ILP) [LD94]. All inductive logic programs, for example HYPER [Bra01] and FOIL [QCJ95a], can accept background knowledge in the form of logic sentences that facilitate learning by simplifying the representation of target concepts. Pazzani and Kibler [PK92] developed and evaluated the FOCL system, an extension of FOIL, that introduces additional types of background knowledge, like constraints on predicates' arguments and the possibility to include initial rules. In the GRENDEL system [Coh94], user can constrain the form of learned hypothesis by specifying the language of rule antecedents with a grammar. A nice overview of using prior knowledge in ILP is presented in the work by Nedellec et. al. [NRA$^+$96].

In the research on non-ILP induction methods, we can find a quite common application of structured prior knowledge in learning structured models, such as Bayesian networks, when there is not enough data to reliable construct such models [MKGT06]. Another principle was develpoed by Núnez [MN91], who extended the ID3 [Qui86; Mit97] algorithm for induction of decision trees to accept domain knowledge in the form of ISA hierarchy and the measurement cost associated with each attribute, which resulted in more logical and understandable decision trees for the domain experts. Clark [CM93] developed a system for learning qualitative rules consistent with a qualitative model provided by experts. Similarly, in regression problems, Šuc et. al. [vVB04; vB03] developed the Q^2 approach that learns numerical models which respect qualitative constraints. The LAGRAMGE [TD97; TD01a; TD01b] equation discovery system allows a user to define declarative bias of induced equations (similar to ideas presented in [NRA$^+$96]) with a context-free grammar. Futhermore, Bohanec and Zupan [BZ04] developed a method for functional decomposition that can be guided by expert given background knowledge.

Suprisingly, prior knowledge has been also successfully applied to non-symbolic methods like SVM [FPM$^+$; SSSV98; SD05] and neural networks [TS94]. One possibility in SVMs is to use prior knowledge to guide the construction of an appropriate kernel for the problem at hand [FPM$^+$; SSSV98]. Sun and DeJong [SD05], on the other hand, propose a combination of EBG and SVM, where EBG is used to suggest which of the features should be used in the inner product evaluation (different examples can have different sets of relevant features). In knowledge-based artificial neural networks [TS94], prior knowledge is first translated into a neural network, which is afterwards refined using learning examples. Alternative approaches, like TANGENTPROP [SVLD92] and EBNN [TM93], use prior knowledge to alter the error criterion minimised by the optimisation algorithm, so that the network fits well the prior knowledge and the learning examples.

Prior knowledge has found its use also in unsupervised machine learning methods. Srikant et. al. [SVA97] propose the use of boolean constraints on presence and absence of specific items in learning association rules, for example, the user might be interested only in association rules that contain a specific item. Pei and others [PHL01] argue that these constraints are too limited, they extended the language of possible constraints defined in [SVA97] and applied it to frequent itemsets mining. In clustering, prior knowledge is usually provided at the instance level, where user can select pairs of instances that should be in the same cluster and pairs that must not be in the same cluster [WCRS01; LTJ04]. It was shown that clustering becomes more robust with the use of prior knowledge.

Combining machine learning and expert knowledge provided best results also in knowledge acquisition tasks [WWZ99]. Most of the applications in the literature combine machine learning and the experts' knowledge in one of the following ways: (a) experts validate induced models after machine learning was applied, (b) experts provide constraints on induced models, and (c) the system enables iterative improvements of the model, where experts and machine learning algorithm improve the model in turns. An example of the latter approach is described in [BS91].

Lately, domain knowledge has been mostly used in complex domains with large feature spaces. For example, in text mining, Maedche and Staab [MS00] developed an algorithm to discover non-taxonomic conceptual relations from text, while taxonomy of concepts is used as prior knowledge. In his Thesis, Budiu [Bud01] stresses the importance of domain knowledge for interpretation of meanings of words in sentence processing. WordNet [Fel98], a lexical database containing semantic relations among words, can be used to deal with the variability of natural language by constructing alternative lexical variants, and has proven to be a great addition to text processing, see for instance [SG05]. Another example of a complex domain is mining patterns in sequences. Garofalakis and others [GRS02] have increased the speed of searching patterns by constraining types of patterns with regular expressions. Almeida and Torgo [dAT01] used domain knowledge to extract features from financial time series, which lead to increase in accuracy in time series prediction.

There are certainly many other methods that use knowledge in machine learning, but we will stop the overview here. We have reached the point, where we can say with certainty the common property of these approaches: most of them require the knowledge from expert to be on the domain level, which differs from our argument-based approach, where knowledge is on the level of single examples[†]. The theoretical advantages of our approach will be explained in Chapters 3 and 4.

2.4 Formal definition of learning with prior knowledge

The problem of learning from examples and given prior knowledge, called also knowledge-based inductive learning[RN03], can be defined as:

1. Given examples and prior knowledge B.
2. Find a hypothesis that is consistent with the examples and prior knowledge B.

With B, the derivation constraint described in formula 2.1 extends to:

$$\forall e, B \wedge H \wedge D_e \vdash C_e \tag{2.2}$$

In this setting, prior knowledge B and hypothesis H are used together to explain all classifications of examples from their descriptions. A knowledge-based inductive learning method should find such hypothesis H that is consistent with this constraint.

We mentioned explanation-based learning as a special kind of learning that "learns" the complete hypothesis from given complete prior knowledge. The hypothesis, therefore, logically follows from prior knowledge:

$$B \vdash H$$

Since H is a generalisation of explanations of all learning examples, there is no need to keep prior knowledge B in the explanation constraint (although is not wrong). The full definition of explanation-based learning is thus:

$$\begin{aligned} &\forall e, H \wedge D_e \vdash C_e \\ &B \vdash H \end{aligned} \tag{2.3}$$

[†]We found only two clustering applications [WCRS01; LTJ04] that exploit knowledge based on pairs of instances, rather on the whole domain.

Chapter 3

Introduction to Argumentation

We use argumentation daily. An argument is a tool that enables expression and elucidation of opinions, which we use in conversation with others or while reasoning internally. Roughly, an argument consists of a claim and a set of reasons defending the claim. This structure facilitates understanding other's opinions or enables identification of fallacies in their reasoning. People usually argue in turns, by providing arguments and counter-arguments to initial arguments, and the arguer with the last unchallenged argument is then the winner of the argumentation. Dung [Dun95] summarised argumentation succinctly by an old saying "The one who has the last word laughs best."

The study of formal argumentation started among critical thinking and practical reasoning philosophers [Ric97; Wal06]. Critical thinking is concerned with argument identification and its evaluation by identifying the weak or missing points in the argument. Practical reasoning in argumentation is a type of decision making, in which the arguments are used to determine the best course of action in practical situations, where the knowledge of the world is not complete. We should also mention that a lot of inspiration for research in argumentation came from the domain of law [BC91a], where the argument is the basic tool that lawyers use in trials, and the combination of all arguments leads to the final decision.

Probably the most important philosophical work for the development of argumentation is the one of Toulmin [Tou58]. He showed that classical logical reasoning can not capture all aspects of argumentative reasoning, as one is almost never able to possess complete relevant information about the problem, therefore it is impossible to be sure about all exceptional cases. His work is most known for his definition of the structure of an abstract argument: an *argument* has a *conclusion* that is inferred from available *data*, a *warrant* that allows you to jump to conclusion, and a possible *rebuttal*, which is a new argument by itself that disagrees with the original argument.

These research works provided the basics for the foundation of computational argumentation theory. Argumentation has now been a part of Artificial Intelligence for the last twenty years, especially in fields like planning, decision making, dialogue, natural language processing, and multi-agent systems [RN04]. Argumentation is a type of reasoning where arguments for and against are constructed and evaluated to derive a conclusion. This approach enables reasoning with inconsistent information, which has made argumentation particularly useful to deal with knowledge presentation, knowledge elicitation and reasoning within expert systems [CRL00]. Knowledge is represented by a set of rules stored in a knowledge base used by an argumentation reasoner to construct arguments and reach conclusions. Whenever additional knowledge is introduced to the knowledge base, there is no need to change old knowledge, as the reasoner will be able to infer by itself the arguments that can be used (accepted arguments) for a particular case and those that can be not (defeated arguments). We could say that inconsistency in the knowledge base is not corrected, but explored in the argumentation process, which should enrich the explanation power of the expert system. Knowledge elicitation, a major bottleneck in knowledge engineering, is greatly simplified in argumentation-based expert systems, since:

- it enables the knowledge engineer to focus at one example at a time - domain experts are asked to explain given example with arguments and these arguments are added to the knowledge base, while an expert or a knowledge engineer does not need to be concerned if the new arguments contradict those in the knowledge base already.

- the disagreements between domain experts do not pose a problem; all provided arguments (for and against) can be imported in the knowledge base and it is left to the reasoner to select which of them are acceptable.

In the remainder of this section, we will try to explain the basic notions in argumentation theory. There exist several scientific papers formalising argumentation in different ways, however reviewing them all or even one of them in detail is far beyond the scope of this Thesis. We will rather explore the basics of argumenta-

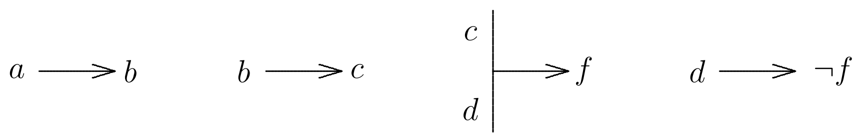


Figure 3.1: Visualisation of inference rules

tion theory relevant for the understanding of the rest of the Thesis: the structure of an argument, reasoning with a set of arguments, and how were argumentation and machine learning combined in the past.

3.1 An argument

In common sense, an argument is usually used as a synonym for explanation, proof, justification, etc. In the context of formal argumentation reasoning, an argument is simply a formula that provides reasons to believe in a conclusion. There exist several formalisations of an argument [Pol92; Vre97; SL92], and they all start off with the same structure: an argument contains conclusion and reasons supporting the conclusion. We shall describe here a somewhat simplified definition of the argument proposed by Vreeswijk [Vre97].

Let $\mathcal{L}$ be a logical language and $\mathcal{R}$ a set of defeasible inference rules. A rule $R \in \mathcal{R}$ has the form $\phi_1, \phi_2, \ldots, \phi_n \rightarrow \phi$, where $\phi_1, \phi_2, \ldots, \phi_n$ is finite, possibly empty, sequence in $\mathcal{L}$ and ϕ is a member of $\mathcal{L}$. Note that in defeasible rules conclusions are not final, there is always a possibility that another rule will contradict this rule. For instance, let $\mathcal{L} = \{a, b, c, d, e, f\}$ and $\mathcal{R}$ the following set of rules (Figure 3.1 visualises these rules):

$$\mathcal{R} = \{a \rightarrow b; b \rightarrow c; c, d \rightarrow f; d \rightarrow \neg f\}$$

The sign $\neg$ stands for negation.

Intuitively, an argument is a result of reasoning with inference rules. It is a (defeasible) proof or, in other words, a chain of one or more rules that lead from given premises (or facts) of the example to the desired conclusion [*]. Figure 3.2 shows two arguments constructed for f and $\neg f$, given that we know a and d.

[*]We provide only an intuitive explanation of how an argument is constructed from rules. We appoint an interested reader to [Vre97] for a more formal definition of this process.

$$\begin{tikzcd} a \arrow[r] & b \arrow[r] & c \arrow[dr] & & & \\ & & & f & d \arrow[r] & \neg f \\ & & d \arrow[ur] & & & \end{tikzcd}$$

Figure 3.2: Two arguments for f and $\neg f$

A single argument is always consistent with itself. But, since the construction of an argument is a monotonic process - new knowledge cannot rule out an old argument, two different arguments may interact. There are two main types of interaction that can be distinguished:

Rebutting: arguments A and B rebut each other if their conclusions are inconsistent, e.g. if A concludes f and B concludes $\neg f$. We say that arguments attack each other.

Undercutting: an argument A undercuts argument B, if A attacks the connection between the reasons and the conclusion of the B, e.g. saying there is no warrant that data provided act as reasons for argument's conclusion. In this case, the first argument attacks the second one .

3.2 Reasoning with arguments

Argument-based knowledge bases are usually inconsistent, which results in the construction of conflicting arguments, and the conclusion from these arguments can not be trivially achieved. The core of any argumentation framework is to evaluate the acceptability of different arguments. An argument can be either accepted, defeated, or provisionally accepted. The final conclusion is thus drawn from accepted and provisionally accepted arguments only.

The basic and most cited definition of an argumentation framework was provided by Dung [Dun95]:

Definition 3.2.1 (Argumentation framework)**.** *An argumentation system AF is a pair $\langle \mathcal{X}, \mathcal{A} \rangle$ in which $\mathcal{X}$ is a set of arguments and $\mathcal{A} \subseteq \mathrm{X} \times \mathrm{X}$ is the attack relation. We say that an argument x attacks an argument y iff $(x, y) \in \mathcal{A}$.*

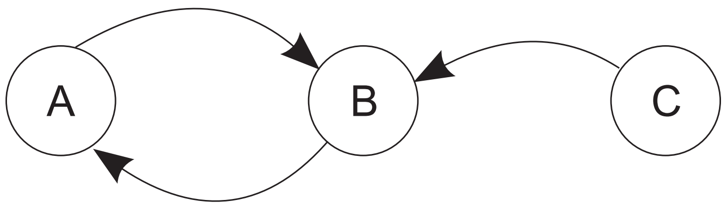


Figure 3.3: Three conflicting arguments

As mentioned, one of the tasks of argumentation framework is to determine which of the constructed arguments will be kept for inferring conclusions. We shall describe a set of semantics proposed by Dung [Dun95] for argument selection.

Definition 3.2.2 (Conflict-free, Attack, Defence)**.** *Let $\mathcal{R} \subseteq \mathcal{X}$ and $\mathcal{S} \subseteq \mathcal{X}$.*

- *A set $\mathcal{R}$ is conflict free iff there exist no x, y in $\mathcal{R}$ such that x attacks y.*
- *$r \in \mathcal{R}$ is attacked by $\mathcal{S}$ if there is some $s \in \mathcal{S}$ such that s attacks r.*
- *$\mathcal{R}$ defends an argument y iff for each argument $x \in \mathcal{X}$, if x attacks y, then there exists $r \in \mathcal{R}$ such that r attacks x.*

Definition 3.2.3 (Acceptability semantics)**.** *Let $\mathcal{R} \subseteq \mathcal{X}$ and $\mathcal{R}$ is a conflict free set of arguments.*

- *$\mathcal{R}$ is admissible if every argument in $\mathcal{R}$ is defended by $\mathcal{R}$.*
- *$\mathcal{R}$ is a preferred extension if it is the maximal (w. r. t. set-inclusion) admissible set.*
- *$\mathcal{R}$ is a complete extension if it is admissible and each argument which is defended by $\mathcal{R}$ is in $\mathcal{R}$.*
- *$\mathcal{R}$ is a grounded extension if it is the minimal (w. r. t. set-inclusion) complete extension.*
- *$\mathcal{R}$ is a stable extension if it is a preferred extension that attacks all arguments in $\mathcal{X} \setminus \mathcal{R}$.*

Let us illustrate these concepts on an example of argumentation framework with three arguments A, B, and C, as shown in Figure 3.3. A and B attack each other (e.g. rebut each other) and C attacks argument B (e.g. C undercuts B). A conflict free set of arguments is any set, where arguments do not attack each other, e.g. $\{A, C\}$. The set $\{A, C\}$ defends argument A, since it attacks B, which attacks A. The sets $\{A\}$, $\{C\}$, and $\{A, C\}$ are admissible, as they all defend themselves. On the other hand, the set $\{B\}$ is not admissible, because it does not defend itself against the attack of C. Intuitively, admissible sets are sets of arguments that can defend themselves, or in other words, the set is self-sufficient with respect to defense. Note that the empty set of arguments $\emptyset$ is also admissible. The only preferred extension of this argumentation framework is therefore $\{A, C\}$ (the largest admissible set), and likewise, $\{A, C\}$ is the only complete, grounded, and stable extension.

3.3 Argumentation and machine learning

The idea of combining ML and argumentation is not completely new. However, there have only been a few attempts in this direction. Most of them focused on the use of machine learning to build arguments that can be later used in the argumentation process, most notably in the law domain [AR03; BA03]. Gomez and Chesñevar suggested in their report [GC04a] several ideas of combining machine learning methods and argumentation. Moreover, these two authors also developed an approach where they used argumentation as a method to improve performance of a neural network [GC04b]. Their method is applied after the actual learning is already finished. Clark [Cla88] proposed the use of arguments to constrain generalization. However, he used arguments as a special kind of background knowledge that applied to the whole domain, whereas in this Thesis arguments apply to individual examples.

Part II

Argument Based Machine Learning and the ABCN2 Algorithm

Chapter 4

Argument Based Machine Learning

In Chapter 2, we showed the difference between various approaches to selecting a good hypothesis that will perform well on unseen examples. One of the possible solutions was to constrain search with given prior knowledge - the knowledge about the learning domain. The critical problem of this approach is the difficulty that experts face when they try to articulate their global domain knowledge. Argumentation, which was introduced in the previous chapter, is an approach that allows experts elicit their knowledge in a more natural way, by allowing the use of their "local" knowledge of specific situations, perhaps only valid for these situations. Therefore, we can expect that the combination of argumentation and machine learning is the one that should bring the most benefits.

In this chapter, we will lay out the core idea of Argument Based Machine Learning (ABML), a combination of machine learning and argumentation. We commence with an illustrating example that should give a quick and intuitive explanation of what ABML is. Based on the given example, we enumerate and explain several expected motivations for learning from arguments. In the third section, ABML is formally defined, and in fourth some general guidelines for implementing ABML methods are given. The principles described in this chapter are then used to develop an actual rule learning algorithm, which is described in the following chapter.

Table 4.1: Learning examples for credit approval

Name	RegularJob	Rich	AccountStatus	HairColor	CreditApproved
Mr. Bond	no	yes	Negative	Blond	yes
Mr. Grey	no	no	Positive	Grey	no
Miss White	yes	no	Positive	Blond	yes
Miss Silver	yes	yes	Positive	Blond	yes
Mrs. Brown	yes	no	Negative	Brown	no

4.1 An illustrating example

We will present here a simple example of ABML in the framework of attribute-value learning. Each example will be specified by an attribute-value vector and the class to which the example belongs. The problem of classic machine learning was already defined in chapter 2:

- Given examples;
- Find a hypothesis that is consistent with the examples.

Consider a simple learning problem: learning about credit approval. Each example is a customer's credit application together with the manager's decision about credit approval. Each customer has a name and four attributes: *RegularJob* (with possible values *yes* and *no*), *Rich* (possible values *yes* and *no*), *AccountStatus* (possible values *positive* and *negative*) and *HairColor* (*black*, *blond*, ...). The class is *CreditApproved* (with possible values *yes* and *no*). Let there be five learning examples as shown in Table 4.1.

A typical rule learning algorithm will induce the following rule from this data:

$$\text{IF } HairColor = blond \text{ THEN } CreditApproved = yes$$
$$\text{ELSE } CreditApproved = no$$

This rule looks good because it is short and it correctly covers all given examples. On the other hand, the rule may not make much sense to a financial expert. We will now look at how this may change when arguments are introduced.

In ABML, each learning example can be explained by a set of positive and negative arguments. A positive argument is used to explain (or argue) why a certain learning example is in the class as given, while a negative provides reasons why it should not be. Examples that are accompanied with arguments will from now on be called *argumented examples*. With arguments, the learning problem changes to:

- Given examples + supporting arguments for some of the examples;
- Find a theory that explains the examples using given arguments.

To illustrate the idea of argumented examples and how an ABML method learns from them, assume that an expert gave an argument for Miss White: "Miss White received credit because she has a regular job". Now consider again the rule above that all blond people receive credit. This rule correctly classifies Miss White, but it does not mention the reasons of the argument given, namely that she has a regular job. Therefore, an argument based rule learning algorithm should induce something like:

$$\begin{aligned}&\text{IF } RegularJob = yes \text{ AND } AccountStatus = Positive\\&\text{THEN } CreditApproved = yes\end{aligned}$$

This rule correctly classifies Miss White example using the given argument. As it will be shown in the following sections, using given arguments in the explanations of argumented examples is the only constraint for an ABML method. For example, we are not concerned how Mr. Bond example is explained, since it is not argumented, however, the explanation of Miss White needs to consider the attached argument.

4.2 Motivation

We shall now repeat ourselves from Chapters 2 and 3 and define a common motivation from using arguments in learning, which lies in three expected advantages; the first two are related to learning from data and prior knowledge in general:

1. Reasons (arguments) impose constraints over the space of possible hypotheses, thus reducing overfitting and guiding algorithms to better hypotheses.
2. An induced theory should make more sense to an expert as it has to be consistent with the given arguments.

The third advantage distinguishes argument based prior knowledge from other types of prior knowledge:

3. An argument focuses on a single learning example only, which allows the experts to elicit their specific example-based knowledge. This reduces the knowledge acquisition bottleneck that experts face when providing "classical" general domain knowledge.

Regarding advantage 1, by using arguments, the computational complexity associated with search in the hypothesis space can be reduced considerably, and enable faster and more efficient induction of theories. As thoroughly explained in chapter 2, the reduced number of possible hypotheses decreases chances that the best hypothesis found is not the true best one, but only an artifact of luck. Moreover, if the learning algorithm heuristically searches the hypotheses space, a reasonable reduction of space that still contains the best hypothesis will only decrease the probability that the algorithm stops in a local maxima, instead in the global one.

Regarding advantage 2, there are many possible hypotheses that, from the perspective of a machine learning method, explain the given examples sufficiently well. But some of those hypotheses can be incomprehensible to experts. Using arguments should lead to hypotheses that explain given examples in similar terms to those used by the expert, and correspond to the actual justifications.

The third (3) advantage was already greatly explained in the Argumentation Chapter 3. Argumentation has shown to be useful for knowledge elicitation, as it enables the knowledge engineer to focus at one case at a time. Similarly, in ABML, the experts need to provide knowledge relating to the specific learning example only, which could be valid only for this chosen example rather for the whole domain. As we will show later, the domain expert needs to explain only some of the learning examples.

4.3 Formal definition of argument based machine learning

In Chapter 2, we formulated the machine learning problem as a constraint satisfaction problem. The problem was stated as: given descriptions D_e and classifications C_e of each learning example e, find a hypothesis H that satisfies the constraint:

$$\forall e, H \wedge D_e \vdash C_e \tag{4.1}$$

In ABML, a learning example is annotated by a set of arguments. In the most general case, e.g. if different domain experts would argue about this example, there will be also conflicts between these arguments. As described in section 3.2, a set of arguments with corresponding attacks between these arguments represent an argumentation framework.

Definition 4.3.1 (Argumented Example)**.** *An argumented example is a learning example annotated by an argumentation framework:*

- *D_e is a conjunction of example e descriptions,*
- *C_e is the example e's classification, and*
- *AF_e is the argumentation framework appended to the learning example e.*

An argument in AF_e can either support classification C_e (a positive argument) or it can support the negated value of classification $\neg C_e$ (a negative argument).

Definition 4.3.2 (Positive Argument, Negative Argument)**.** *Let R be a conjunction of reasons.*

- *A positive argument specifies reasons in favour of classification (using word because): C_e because R*
- *A negative argument specifies reasons against the given classification (using word despite): C_e despite R*

A reason can be any basic property specified in the example's descriptions. For example, having regular job was used as a reason in the argument for Miss White, which was the only reason in that case, although a typical argument uses more than one reason. The positive argument was:

Miss White received credit *because* she has a regular job.

The conclusion of this argument is "Miss White received credit", which is the actual class of this example, and the reason is "she has a regular job". An example of a negative argument could be:

"Miss White received credit *despite* she is not rich"

Using the notation of arguments described in the previous chapter, a positive argument would be written as $R \rightarrow C_e$, meaning that class C_e can be defeasibly inferred from given reasons. Similarly, the reasons of a negative argument imply the opposite class; $R \rightarrow \neg C_e$. As the conclusions of positive and negative arguments are exactly the opposite, positive and negative arguments attack each other.

Definition 4.3.3 (Attacks between arguments)**.** *There are three different types of attack between a positive argument A_p and a negative argument A_n; they can mutually attack each other, or only A_p attacks A_n, or only A_n attacks A_p. Two positive or two negative arguments can not be conflicting. Let A_p be $R_p \to C_e$ and A_n be $R_n \to \neg C_e$.*

- *A_p attacks A_n and A_n does not attack A_p if $R_p \models R_n$, i.e. negative reasons can be logically inferred from positive reasons.*
- *A_n attacks A_p and A_p does not attack A_n if $R_n \models R_p$.*
- *A_n and A_p mutually attack each other if $R_n = R_p$ or $\neg(R_p \models R_n) \wedge \neg(R_n \models R_p)$*

In the first two types of attacks one argument undercuts the other, while in the third type arguments rebut each other. The set of positive and negative arguments together with attacks among them present an argumentation framework attached to a learning example.

As we mentioned within the motivating example, the learning problem using arguments changes to:

- Given examples + supporting arguments for some of the examples
- Find a theory that explains the examples using given arguments

In logical terms, a theory explains an example using given arguments, if the reasons of given arguments are mentioned during the derivation of the example. The definition of argument-based machine learning therefore needs to use an additional constraint in the derivation process:

$$\forall e, H \wedge D_e \vdash_{AF_e} C_e \tag{4.2}$$

To help us define the argument-based derivation $\vdash_{AF}$, we need to introduce a new function $\mathcal{R}$ over a set of arguments $\mathcal{S}$:

$$\mathcal{R}(\mathcal{S}) = \{r | \exists a \in \mathcal{S}; r \in Reasons(a)\} \tag{4.3}$$

The function $\mathcal{R}(\mathcal{S})$ returns the set of all reasons used in arguments in $\mathcal{S}$.

Definition 4.3.4 (Argument-based derivation)**.** *Let AF be an argumentation framework, let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ be all nonempty admissible sets of positive arguments*

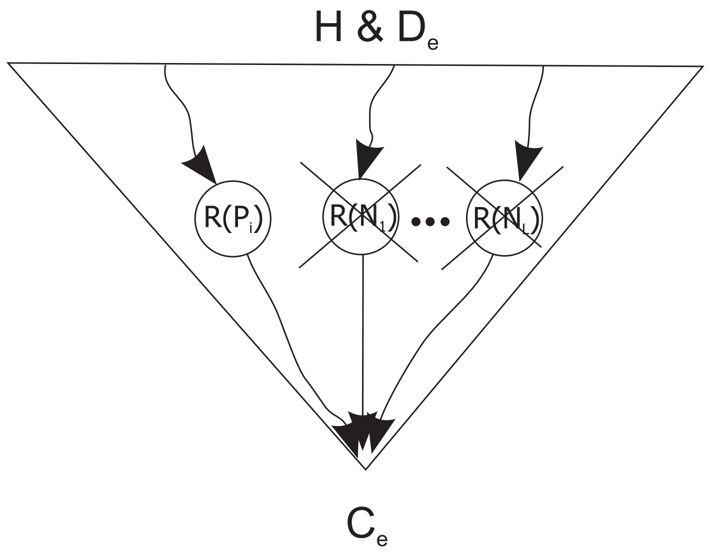


Figure 4.1: An illustration of a proof tree of an argumented example. The proof mentions one positive admissible set, and does not mention any of the negative admissible sets.

and $\mathcal{N} = \{\mathcal{N}_1, \dots, \mathcal{N}_l\}$ all nonempty admissible sets of negative arguments in AF. Then, B is argument-based derived from A with respect to AF, written as $A \vdash_{AF} B$, if:

- *$A \vdash B$, and*
- *each possible derivation (in a given deduction system) of B given A **mentions at least one positive** admissible set $\mathcal{P}_i$, and*
- *each possible derivation of B given A **does not mention any of the negative** admissible sets.*

A derivation mentions a set of arguments $\mathcal{S}$ if all reasons $\mathcal{R}(\mathcal{S})$ are mentioned within the derivation process.

Therefore, in the case of ABML, all possible proofs of classifications C_e from the induced hypothesis H and descriptions D_e should mention at least one of the admissible sets of positive arguments and none of the admissible sets with negative arguments. Remember that an admissible set is a set of not defeated (they can defend themselves) and not conflicting (no attacks between arguments) arguments. Figure 4.1 illustrates a proof tree that considers constraints by arguments.

Example 4.3.5. *Consider again Miss White from our illustrating example. Her credit approval was explained by a single positive argument p: "Miss White received credit because she has a regular job". Therefore, the only admissible set of arguments contains p only; $\mathcal{P} = \{\{p\}\}$. Since there are no negative arguments, there are no negative admissible sets. In this case, the argument-constrained derivation of Miss White classification should mention reason of p, namely "she has a regular job".*

Example 4.3.6. *Let us expand the attribute space of our illustrating learning problem 4.1 with a OwnsHouse feature (whether the person owns a house or not) and assume that Miss White owns a house. Then, an expert could give the following four arguments (2 positive and 2 negative) to Miss White:*

p_1; *Miss White received credit* because *she has a regular job.*

p_2; *Miss White received credit* because *she owns a house.*

n_1; *Miss White received credit* despite *she is not rich.*

n_2; *Miss White received credit* despite *she is blond.*

There are three positive admissible sets $\{\{p_1\}, \{p_2\}, \{p_1, p_2\}\}$ and three negative admissible sets $\{\{n_1\}, \{n_2\}, \{n_1, n_2\}\}$. Therefore, the proof of Miss White example should contain reasons of at least one admissible sets (RegularJob=yes, OwnsHouse=yes, or both) and should not contain any reasons of negative examples (Rich=no, HairColor=blond).

Example 4.3.7. *Let us further expand the feature space with CreditAmount (amount of money asked for) and HouseValue (value of applicant's house on the real-estate market). Miss White asked for a credit amount of* 200 *thousand EUR (CreditAmount=200), while her house could be sold for* 150 *thousand EUR (HouseValue = 150). The domain expert could add a new negative argument n_3: "Miss White received credit* despite *that credit amount is higher than the value of her house". The reasons of this argument imply that she owns a house (ownsHouse=yes), therefore this argument attacks p_2, while p_2 does not attack it back (owning a house does not imply its price). Figure 4.2 visualises attacks between the five arguments. In this argumentation framework argument p_2 is defeated, since it is not present in any admissible set. The Miss White example is argument-based derived only if it mentions the reasons of p_1.*

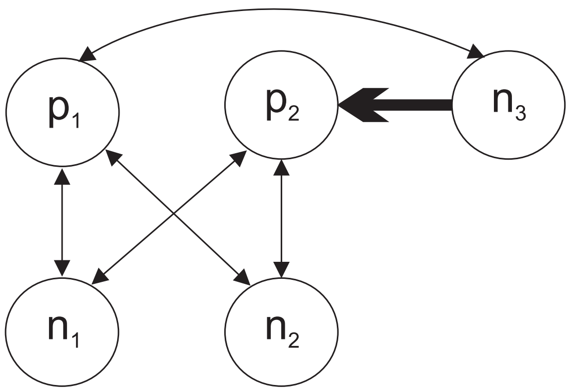


Figure 4.2: Visualisation of 5 arguments for Miss White. Arrows stand for attacks between arguments. The argument n_3 attacks p_2, however p_2 does not attack n_3, since reasons of n_3 imply reason of p_2 (the thick arrow).

Example 4.3.8. *Suppose we would additionally know that Miss White possesses another apartment worth* 100 *thousand EUR. A domain expert could then undercut argument* n_3 *by saying: "Miss White received credit because the value of all her property exceeds the credit amount." Then, the third negative argument* n_3 *would get defeated, while the* p_1 *itself would be admissible again.*

4.4 Comparison of classical and argument-based prior knowledge

The definitions of ABML and classical learning with prior knowledge can be used to distinguish both types of domain prior knowledge. Let us repeat both definitions; we stated the learning from classical prior knowledge as:

$$\forall e, B \wedge H \wedge D_e \vdash C_e \tag{4.4}$$

and argument based-knowledge as:

$$\forall e, H \wedge D_e \vdash_{AF_e} C_e \tag{4.5}$$

According to this two formulae, the main difference between the approaches is in knowledge application. The classical prior knowledge B has the same role as the induced hypothesis, it is used in the derivation of all learning examples. On the other hand, argument-based knowledge has to be mentioned within the derivation of argumented examples only. It is desirable that arguments would be mentioned also in the explanations of other examples, to use experts knowledge as often as possible, however this is not required by the method. Naturally, argument-based approach and classical prior knowledge could be used together:

$$\forall e, B \wedge H \wedge D_e \vdash_{AF_e} C_e \tag{4.6}$$

Argument-based approach to machine learning can be easily misunderstood as a special kind of explanation-based learning. Let us recall the definition of explanation-based learning:

$$\begin{aligned} &\forall e, H \wedge D_e \vdash C_e \\ &B \vdash H \end{aligned} \tag{4.7}$$

Perhaps it is not best visible from the formulae, but approaches do share some similarities. In explanation-based learning, prior knowledge is first used to explain learning examples. It is assumed that search for these proofs or explanations is time-consuming, so it might make sense to generalise these “local” proofs and store them for faster future processing of similar examples. The generalisation in the EBG sense is simply replacing facts in specific proofs with variables that do not change the validity of the proof. In ABML, arguments represent a sort of partial “local” proofs, and an ABML method also generalises these proofs. However, the proofs in ABML are not complete and the generalisation step requires inductive reasoning, while in explanation-based learning it does not. Therefore, the main difference between both approaches is that the hypothesis in explanation-based learning can be logically inferred from prior knowledge - there is no inductive learning whatsoever, while in ABML the hypothesis can can not be inferred from B.

4.5 Guidelines for building argument based machine learning methods

The first task in the construction of an ABML method is to select the base method, which will be later extended to learn from argumented examples. Here we need to consider that arguments lead to hypotheses that are more comprehensible to domain experts. This can only be demonstrated if the hypothesis language allows human explanations, e.g. decision trees or logistic regression. We believe that the best candidates for the base method in ABML are symbolic methods, where hypotheses are encoded with a set of human readable symbols. However, there is no true obstacle not to use arguments also in other methods, e.g. instance-based methods like kernel or k-nearest-neighbour (KNN) methods, but we will not be able to easily spot the increased understandability in hypotheses achieved with the help of arguments.

After we selected a machine learning method, we need to answer the following questions:

1. What is the language for representing arguments?

2. What do we mean by "reasons should be mentioned in the derivation of examples" for the selected learning method? What actually is argument-based derivation for the selected method?

3. How can we effectively guide learning towards hypotheses consistent with arguments?

4. How is statistical error considered in argumented and non-argumented examples?

The formal language for reasons in arguments should be the same as the language defining the hypothesis space. For example, if the hypothesis is a logical program, then the reasons should be logical sentences. This enables direct reflection of arguments in the induced hypothesis - the reasons of arguments construct part of the hypothesis. It greatly helps a domain expert to position his or her knowledge and observe the new, automatically obtained, knowledge in the hypothesis. The second (2) question is related to the reasoning with induced hypothesis and the third (3) is related to the search space and how can it be effectively constrained. The fourth (4) question is not crucial for the method to work (in some methods its solution is practically impossible), but it can increase the quality of induction. The arguments differently constrain different parts of the hypotheses space. In some cases, where arguments are very specific, the induction algorithm will not have much generalisation to do, and there we can expect low to no overfitting. However, the hypothesis space related to the non-argumented examples can be less constrained, whereas we can expect to deal with a higher amount of overfitting.

In the case of the ABCN2 algorithm, which is going to be described in the following chapters, we shall thoroughly study and provide answers to all four questions.

But first, we will look at two short examples to show ideas for transforming two machine learning methods, inductive logic programming and logistic regression, to their argument-based counterparts, which are meant to provide the reader with a more general understanding of these questions. The answer to the fourth question is relatively hard and not necessary for understanding the basic principles, hence it will be omitted in these two examples.

4.5.1 Argument based inductive logic programming

Inductive Logic Programming (ILP) learns hypotheses from background knowledge and descriptions of examples provided in first-order logic sentences. Each example is classified either as positive or negative. The problem is to find such a hypothesis that (together with background knowledge) proves all positive examples and no negative ones.

According to our guideline, we will code the reasons of arguments in first-order logic. As usually, an argument presents a link between example's descriptions and its class value. However, in ILP a reason does not need to be explicitly part of description of examples, but can mention a reason that can be inferred from descriptions and background knowledge. For example, we could have the following rule in background knowledge (using the syntax of Prolog):

```
union_member(X) :- % X is a member of a labor union
  regular_job(X),
  not rich(X).
```

The argument for Miss White could then change to "Miss White received credit because she is a member of a labor union." Another type of arguments in ILP come from the fact that ILP is particularly suitable for problems where learning examples are related to each other (e.g. graphs). This means that the reasons of arguments can also contain relations between examples or even contain the descriptions of some other related example. To illustrate this specific type of reasons, consider that we could add *married* relation to our credit example, where $married(person1, person2)$ means that $person1$ and $person2$ are married. After adding a fact to the domain:

```
married(mr_Bond, miss_White).
```

the expert is entitled to add the following arguments to the Miss White example: "Miss White received credit because she is married to Mr. Bond." or "Miss White received credit because Mr. Bond is rich.”

Since the usual problem statement of ILP is very similar to our definition of general machine learning, there should be no problems with the understanding of argument-based derivation concept. A hypothesis in AB-ILP is consistent with the given arguments if the proof tree of the learning example contains reasons of the given arguments. In our Miss White example, we require that fact $married(mr\_Bond, miss\_White)$ or fact $rich(mr\_Bond)$ is mentioned in the procedure, respectively for the last two example arguments.

The last question is how to efficiently search through a set of hypotheses consistent with arguments. Using a standard ILP algorithm and simply disregarding inconsistent hypotheses would work in theory, however it could be very slow, especially if an argument would mention facts that are “far” away from the argumented example. For a possible improvement, we could consider facts as nodes in graph, while relations and rules would connect these nodes. Then, an algorithm for finding the shortest path, e.g. Dijkstra’s algorithm, could be used to connect the facts mentioned in the reasons of an argument with the facts in the description of the example. These algorithms are relatively fast. The complete path from reasons to facts of example would then be used to form the basic structure of the induced hypothesis, while ILP would only need to induce additional required conditions.

4.5.2 Argument based logistic regression

Logistic regression is a linear method for classification. It is suited for domains with continuous domain descriptors (attributes) $\mathbf{X} = \{x_1, \ldots, x_n\}$ and a binary class y. Given class and attributes, logistic regression maximises log-likelihood by optimising parameters in the formula:

$$\begin{aligned} p(y|\mathbf{X}) &= \frac{1}{1+e^{-f(\mathbf{X})}} \\ f(\mathbf{X}) &= a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n \end{aligned} \tag{4.8}$$

This formula is thus used to calculate the probability of class y given description $\mathbf{X}$ of an example.

The usual interpretation of logistic regression formula is in terms of attribute changes, for example: if the coefficient a_i is positive, then the probability of class y for a particular example would increase if the value of x_i of this example would increase, while all other attribute values would stay the same. The reasons of an argument in argument-based logistic regression should thus also be given in a qualitative sense. Consider an example with the actual values:

$$(x_1 = 0.5, x_2 = 0.8, \ldots, x_i = 6, \ldots, x_n = -4.3), y = 1$$

A possible argument for this example could be: “The value of y equals 1 because the value of x_i is high”. This argument can be interpreted then as: the higher is the value of x_i, the larger is the probability $P(y = 1)$.

The argument can be easily converted into a constraint on the induction of logistic models. In the example above, the argument suggests that the value of the a_i parameter is likely to be positive. One way to assure such constraint is to use non-symmetric bias in bayesian logistic regression [GLM07] that would prefer either positive or negative values of a parameter. Alternatively, we could use any form of strictly constrained logistic regression [VR02], where the values of the parameters can be limited to be either positive or negative.

Using constraints in logistic regression will force the change of parameters, which will reflect on the classification of all examples. Such constraints are not according to the idea of argument based learning; the positiveness of a_i parameter is required only in classification of this particular example and not for the whole learning domain. However, this approach is still acceptable, as long as this change does not worsen the prediction on other learning examples. Otherwise, a solution would be a method that somehow divides the example space and learns a logistic model in each of the subspaces. An example of a such method is Logistic Model Tree [LHF03].

Chapter 5

Argument Based Rule Learning (ABCN2)

In this chapter, we will describe argument based CN2 (ABCN2 for short), a method for learning classification rules from argumented examples. ABCN2 is an extension of the well-known rule learning algorithm of Clark and Niblett ([CB91]). The work described here bases on our earlier papers on the idea of AB-enhancement of CN2 ([MvB06; MvB07].

We will begin with a definition of argumented examples accepted by ABCN2. After, we shall look at the algorithm ABCN2 itself, describe the concept of AB-covering and propose some changes of the basic algorithm to improve its time efficiency. The last section of this chapter gives some details of the actual implemented product.

There is one critical question though that we need to answer before we move to the description of the method: Why have we chosen rule learning, and why particularly CN2, as the basis of our argument-based method? The question is perfectly valid, as there are various standard ML techniques that could be extended into an argument-based variant. One of the reasons is the structure of a single argument; human-based arguments often have a form similar to propositional rules, where the reason of an argument is a conjunction of premises (see chapter 3). Therefore, following our guideline that learned knowledge should be represented the same as existing knowledge, we believe that rule learning is the most appropriate machine learning approach to be used with argumentation. CN2 was selected since it is the most commonly used and well undestood algorithm for induction of unordered rules, where by “unordered” we mean that classification from rules considers all applicable rules for the example and not just the first that triggers. We also extended CN2 in many aspects (evaluation of rules, covering strategy, classification from rule), which brings CN2 on a similar level, in terms of accuracy, with other currently popular machine learning methods.

5.1 Argumented examples

A learning example E in the usual form accepted by CN2 is a pair (A, C), where A is an attribute-value vector, and C is a class value. An attribute can be either discrete (finite unordered set of values) or continuous. In addition to such examples, ABCN2 also accepts argumented examples. An argumented example AE is a triple of the form:

$$AE = (A, C, Arguments)$$

As usual, A is an attribute-value vector and C is a class value. $Arguments$ is a set of arguments $Arg_1, \ldots, Arg_n$, where an argument Arg_i has one of the following forms:

$$C \text{ because } Reasons$$

or

$$C \text{ despite } Reasons$$

The former specifies a *positive* argument (speaks for the given class value), while the latter specifies a *negative* argument (speaks against the class value). $Reasons$ is a conjunction of reasons $r_1, \ldots, r_n$,

$$Reasons = r_1 \wedge r_2 \wedge \ldots \wedge r_n$$

where each of the reasons r_i can be in one of five possible forms. In the explanation of these forms below we assume that r_i is a part of a positive argument; for negative arguments, the explanations are exactly the opposite. The five forms of reasons are:

- $X = x_i$ means that value x_i of attribute X is the reason why example is in the class as given. This is the only allowed form for discrete attributes.
- $X > x_i$ $(or X >= x_i)$ means that the value of attribute X of example being greater than (greater or equal to) x_i is the reason for class value.

- $X < x_i$ $(orX <= x_i)$ the opposite to $X > x_i$ $(X >= x_i)$.
- $X >$ $(orX >=)$ “X is high”; similar to $X > x_i$ $(X >= x_i)$, just that we do not know the threshold value and it has to be found by ABCN2 automatically. Such an argument says that the value of X of the example is high enough for the example to be in the class as given.
- $X <$ $(orX <=)$; “X is low”, the opposite of $X >$ $(X >=)$.

We shall now revisit again our key example Miss White from Table 4.1. The positive argument said: “Miss White received the credit because she has a regular job”, while one of the negative arguments was: “Miss White received the credit despite she is not rich”. The Miss White example would in our syntax be written as:

$$((RegularJob = yes, Rich = no, AccountStatus = positive, HairColor = blond),$$
$$CreditApproved = yes,$$
$$\{CreditApproved = yes \text{ because } RegularJob = yes,$$
$$CreditApproved = yes \text{ despite } Rich = no\}).$$

Arguments given to examples additionally constrain rules *covering* this example. Remember that in CN2, rules have the form:

$$\text{IF } Complex \text{ THEN } Class$$

where $Complex$ is the conjunction of simple conditions, called *selectors*. Usually, a selector specifies the value of an attribute, for example $HairColor = blond$ or a threshold on an attribute value, for example $Salary > 5000$. A rule for our credit approval domain can be:

$$\text{IF } rich = no \text{ AND } HairColor = blond$$
$$\text{THEN } CreditApproved = yes$$

The condition part of the rule is satisfied by the attribute values of Miss White example, so we say that this rule *covers* this example.

A rule R is *consistent* with an argument ”C because $Reasons$” (or ”C despite $Reasons$”), if for all reasons r_i of $Reasons$ it is true that:

1. If the reason r_i is in one of forms: "$X = x_i$" or "$X > x_i$" or "$X < x_i$" ("$X >= x_i$" or "$X <= x_i$"), then exactly the same selector needs to be present in the complex of the rule R.

2. If the reason r_i has the form "$X >$" (or "$X <$", "$X <=$", "$X >=$"), then the complex of the rule R needs to contain a selector "$X > x_i$" (or "$X < x_i$", "$X >= x_i$", "$X <= x_i$"). The threshold value x_i does not matter for consistency.

Having the form of arguments defined, we need to implement the concept of argument-constrained derivation in argument based rule learning. To this end, we will refine the definition of *covering* relation. In the standard definition [CN89], a rule covers an example if the condition part of the rule is true for this example. In argument based rule learning, this definition is modified to: A rule R *AB-covers* an argumented example E if all of the points below hold:

1. All conditions in R are true for E (same as in CN2),

2. R is consistent with at least one positive argument of E, and

3. R is not consistent with any of negative arguments of E.

The AB-covers relation corresponds to the argument-constrained derivation defined within the general ABML framework. The complete inference from a propositional rule to an example has only one step: from conditions of the rule to the facts of the example. If arguments have to be mentioned in the explanation of the example, then the reasons of arguments need to be a part of the complex. Since arguments in rule learning can not undercut other arguments (we can assume without loss that undercutted arguments are removed prior to learning), each argument itself is admissible, which allows the use of arguments instead of admissible sets in points 2 and 3.

As an illustration of the differences between AB-covering and the usual definition of covering, consider again the Miss White example with the argument that she received credit because she has a regular job and despite she is not rich. Now consider four rules:

R1: IF $HairColor = blond$ THEN $CreditApproved = yes$

R2: IF $Rich = no$ AND $HairColor = blond$
THEN $CreditApproved = yes$

R3: IF $Rich = no$ AND $RegularJob = yes$
THEN $CreditApproved = yes$

R4: IF $HairColor = blond$ AND $RegularJob = yes$
THEN $CreditApproved = yes$

All four rules cover the Miss White example and have 100% accuracy on the data set from Table 4.1. However, Rule 1 does not AB-cover the example, because it is not consistent with the positive argument. For the same reason, rule 2 does not AB-cover the Miss White example, but this rule fails also because it is consistent with the negative argument ($Rich = no$). Rule 3 also fails due to the negative argument, although it is consistent with the positive argument. The last example AB-covers the Miss White example.

5.2 Argument based CN2 algorithm

The CN2 algorithm [CN89; CB91] consists of a covering algorithm and a search procedure that finds individual rules by performing beam search. The covering algorithm induces a list of rules that cover all the examples in the learning set. Roughly, the covering algorithm starts by finding a rule, then it removes from the set of learning examples those examples that are covered by this rule, and adds the rule to the set of rules. This process is repeated until all the examples are removed.

There are two versions of CN2: one induces ordered list of rules, and the other unordered list of rules. Our algorithm is based on the second version of CN2. In this case, the covering algorithm consists of two procedures, CN2unordered and CN2ForOneClass. The first procedure iteratively calls the second for all the classes in the domain, while the second induces rules only for the class given. When removing covered examples, only examples of this class are removed [CB91]. Essentially, CN2ForOneClass is a covering algorithm that covers the examples of the given class.

5.2.1 ABCN2: covering algorithm

Remember that in ABML an induced hypothesis must explain argumented examples using given arguments. We showed that a single rule explains an argumented example, if the rule AB-covers the example. A hypothesis induced by CN2 is a set of rules, and there are several different techniques to classify an example from a set of if-then rules. They all first select the rules that apply for the classifying example, and

Algorithm 5.1 Pseudo code of the original CN2ForOneClass procedure.

```
Procedure CN2ForOneClass(Examples ES, Class T)

Let RULE_LIST be an empty list.
while ES is not empty do
   Let BEST_RULE be Find_best_rule(ES,T)
   Add BEST_RULE to RULE_LIST.
   Remove from ES examples covered by BEST_RULE.
end while
return RULE_LIST
```

Algorithm 5.2 Covering algorithm of ABCN2 algorithm that learns rules from examples ES for given class T.

```
Procedure ABCN2ForOneClass(Examples ES, Class T)

Let RULE_LIST be an empty list.
Let AES be the set of examples in class T that have arguments; $AES \subseteq ES$
Determine threshold in vague arguments (type $X >$, etc.)
Evaluate arguments (as if they were rules) of examples in AES and sort examples
in AES according to quality of their best argument.
while AES is not empty do
   Let AE1 be the first example in AES.
   Let BEST_RULE be ABFind_best_rule(ES,AE1,T)
   Add BEST_RULE to RULE_LIST.
   Remove from AES examples AB-covered by BEST_RULE.
end while
for all RULE in RULE_LIST do
   Remove from ES examples AB-covered by RULE.
end for
Add rules obtained with CN2ForOneClass(ES,T) to RULE_LIST
return RULE_LIST
```

then combine these rules to classify the example. Therefore, to satisfy the ABML requirement, there needs to be at least one rule in the set of induced rules that AB-covers this example. This can be achieved relatively simply, by merely replacing the covering relation in original CN2 with AB-covering.

Replacing the "covers" relation in CN2 with "AB-covers" in ABCN2 ensures that both argumented and non-argumented examples are AB-covered. However, in addition to simply AB-covering all the examples, we would also prefer explaining as many as possible non-argumented examples by arguments given for the argumented examples. Therefore, we propose a change in the covering algorithm, where CN2ForOneClass is changed into ABCN2ForOneClass (see Algorithm 5.1 for the original algorithm and in Algorithm 5.2 the new one). The procedure starts by creating an empty list of rules, and makes a separate set AES of argumented examples only. Then it looks for "unfinished" arguments - arguments that have some of the reasons "vaguely" specified ($X >$ and $X <$) and finds the best splits for these reasons. Splits are initially set simply to the attribute value of the argumented example; we then iteratively generalise them to achieve the highest quality. Arguments in the examples AES are then evaluated by the rule evaluation function[*] as if the arguments were rules of the form

IF reasons of argument THEN claim of argument

The examples in AES are then sorted according to the "goodness" of their best arguments.

In the while loop, the procedure induces a rule, using ABFind_Best_rule, to cover the first argumented example. ABFind_Best_rule is a modified beam search procedure that accepts examples, an argumented example and a target class, where the resulting rule is guaranteed to AB-cover the given argumented example. This rule is added to the rule set, and the procedure removes from AES argumented examples AB-covered by this rule. The removal of all positive examples is not necessary, as each of the argumented examples differently constrains the search and thus prevents ABCN2 from inducing the same rule again. When all argumented examples are covered, all positive examples AB-covered by rules are removed, and the remaining rules are learned using classical CN2ForOneClass.

5.2.2 ABCN2: search procedure.

Algorithm 5.3 shows the argument-based search procedure. The procedure takes a set of examples to learn from, an argumented example that needs to be AB-covered by the induced rule, and the target class. In Algorithm 5.3 the underlined parts emphasize the differences between the original search procedure in CN2 and the AB-search procedure:

Initial value of set STAR is the set of positive arguments of example E. A rule induced from an argumented example must AB-cover this example, therefore it

[*]The original CN2 algorithm uses Laplacian formula of succession for evaluation of rules. We will propose an alternative evaluation in the Chapter 6.

Algorithm 5.3 Algorithm that finds the best rule that AB-covers the argumented example E. The "quality" of a complex is evaluated by a user-defined evaluation function.

```
Procedure ABFind_Best_Rule(Examples ES, Example E, Class T)

Let the set STAR contain reasons of positive arguments of E.
Evaluate complexes in STAR (using quality function).
Let $BEST\_CPX$ be the best complex from STAR.
Let $SELECTORS$ be the set of all possible selectors that are TRUE for E
Let $ARG\_REASONS$ be the set of all reasons in positive arguments of E (union
of reasons).
while STAR is not empty do
   {Specialize all complexes in STAR as follows}
   Let $NEWSTAR$ be the set
      $\{x \wedge y \,\|x \in STAR, y \in SELECTORS\}$
   Remove from $NEWSTAR$ all complexes that are consistent
   with any of negative arguments of E.
   for every complex $C_i$ in NEWSTAR do
      if $C_i$ is statistically significant(ES,T) and
         quality($C_i$) $>$ quality(BEST_CPX) then
          replace the current value of $BEST\_CPX$ by $C_i$
      end if
   end for
   Let STAR be best $N$ complexes from NEWSTAR; $N$ is a user-defined size of
   STAR (usually N=5).
   Let ABNEWSTAR be such subset of NEWSTAR,
      where complexes in ABNEWSTAR contain only
      conditions from $ARG\_REASONS$.
   Let ABSTAR be best N complexes from ABNEWSTAR.
   Let STAR be STAR merged with ABSTAR.
end while
return rule: "IF $BEST\_CPX$ THEN T.''
```

will have to contain the reasons of at least one of positive arguments. The easiest way to ensure this is to start learning from them.

Specialize with selectors that are satisfied by argumented example. This ensures the coverage of the seed example by the induced rule.

Remove all complexes that are consistent with negative arguments. Again, rules must AB-cover argumented example, therefore can not be consistent with any of the negative arguments.

Let ABSTAR be best N complexes from ABNEWSTAR. Rules that contain only conditions that are present in positive arguments are likely to be consistent with domain knowledge. Thence, these rules are deemed promising, even if at the moment they are not among first N best rules according to the equality measure.

5.2.3 Time complexity and optimisation

CN2

The complexity of learning a single rule in original CN2 depends on the data set properties and learner's settings ([CN89]). Let:

a be the number of all attributes in the given data set,

e number of examples,

s size of star (as set in CN2), and

L maximum allowed length of a rule.

The evaluation of a single rule has time complexity $O(e)$, since it has to sweep through all examples to determine which of the examples are covered by the rule and which are not. In each specialisation step of CN2, we need to evaluate $O(s \cdot a)$ rules, therefore the complete specialisation takes $O(e \cdot s \cdot a)$. Following specialisation, CN2 sorts rules by their quality to select the best N in STAR, which has time complexity $O((s \cdot a)\log(s \cdot a))$. Finally, as learning a single rule contains L specialisations, the learning of a complete rule has time complexity $O(L \cdot s \cdot a(e + \log(s \cdot a)))$.

ABCN2

Learning rules from argumented data is on one hand slower, due to additional checks in AB-cover relation, and faster, as rule learner starts specialising a given argument and not the empty complex. To estimate the time complexity, we need to define the following new values:

l_p is the average length of positive arguments,

n_p is the average number of positive arguments given to examples,

l_n is the average length of negative arguments, and

n_n is the average number of negative argument given to examples.

The time complexity of sorting elements in the beam stays the same $O((s \cdot a)\log(s \cdot a))$. The main difference is in the cover relation, viz., to pick out examples that are covered by a rule. First, a check is needed if the new attribute value is true for each example, which takes $O(e)$. Then, if the current length of rule is l, we need $O(e \cdot n_p \cdot l_p \cdot l)$ operations to find out whether the rule is consistent with at least one positive argument, and similarly, $O(e \cdot n_n \cdot l_n \cdot l)$ for negative arguments. Together, evaluation of a single specialisation requires $O(e \cdot (l \cdot (n_p \cdot l_p + n_n \cdot l_n)))$. Since learning starts with the positive argument, number of specialisations is reduced to $L - l_p$, therefore the learning of a complete rule takes:

$$\begin{aligned} & O\left((L-l_p) \cdot s \cdot a\left(e \cdot \left(\sum_{i=l_p}^{i=L} i \cdot (n_p \cdot l_p + n_n \cdot l_n)\right) + \log(s \cdot a)\right)\right) \\ = \; & O\left((L-l_p) \cdot s \cdot a\left(e \cdot \left(\frac{L(L+1) - l_p(l_p+1)}{2} \cdot (n_p l_p + n_n l_n)\right) + \log(s \cdot a)\right)\right) \end{aligned}$$

Apparently, the term in the middle $(n_p \cdot l_p + n_n \cdot l_n)$ is the main factor increasing time complexity of learning rules from arguments. Any increase in argument's length and number of arguments given to examples (positive or negative) will result in an increase of this factor, and probably also in the increase of overall time complexity. Only the length of positive arguments has an uncertain polarity of influence, for it also effects the complexity positively by reducing the number of specialisations to $L - l_p$.

Speed-ups of ABCN2

The current algorithm ensures that at least one rule will AB-cover each learning example. The same would be achieved, though, by enforcing a rule to AB-cover only the seed example in procedure *ABCN2ForOneClass* (see Algorithm5.2) and use classical covering instead. In this way, we do not need to check whether a rule is consistent with positive arguments, as positive arguments are used in root of the rule. Moreover, consistency with negative arguments has to be checked only for the seed example and not for others. With the mentioned change, the time complexity of the algorithm decreases to:

$$O\left((L-l_p) \cdot s \cdot a\left(e + \frac{L(L+1) - l_p(l_p+1)}{2} \cdot n_n \cdot l_n + \log(s \cdot a)\right)\right) \qquad (5.1)$$

The complexity of the algorithm is now significantly reduced. To begin with, the length or number of arguments do not increase time anymore, which favours therefore to have as many positive arguments as possible. Moreover, the term $n_n \cdot l_n$ related to negative arguments is not multiplied by the number of examples e anymore that also improves the time efficiency a lot. It is also important that this improvement does not change algorithms 5.2 and 5.3, since it only affects the evaluation of rules, particularly determination which examples are covered by rule and which not. We will use this speed-up in all our experiments.

5.2.4 Implementation

The ABCN2 algorithm is implemented within the Orange-toolkit [DZ04]. It is included within the standard distribution of Orange `http://www.ailab.si/orange/` along with all the necessary documentation.

Chapter 6

Extensions of ABCN2

The main issue discussed in this chapter is evaluation of rules in ABCN2. We will show that the multiple-comparison problem described by [JC00] makes evaluation of rules unavoidably optimistic. The problem is not so acute in standard CN2, since all rules are similarly optimistic. However, in the case of ABML, particularly in the case of ABCN2, rules learned from argumented examples are typically selected from less candidates than rules induced by a standard rule learning algorithm, and thus the quality of a rule learned from an argumented example is relatively under-estimated when compared to a rule learned from standard CN2.

We shall describe a new method Extreme Value Correction based on Extreme Value Distributions that takes multiple-comparisons into account. We will begin with a description of a general algorithm and continue with a specific algorithm suited especially for rule learning. We conclude the chapter with a new strategy for rule covering, which is necessary to enable efficient application of Extreme Value Correction in ABCN2.

6.1 Extreme value correction

Extreme Value Correction (EVC) is a new method presented in this Thesis that corrects the optimistic evaluations of constructed hypotheses in machine learning algorithms. Most of these algorithms evaluate several hypotheses during learning and choose the one with the best score. Since these estimates are based on the training data from which the hypotheses themselves were constructed, they are inevitably optimistic. a decade ago Jensen and Cohen [JC00] did an extensive study of this pathology. They spotted three main characteristics of learning algorithms contributing to the optimism of evaluations: attribute selection error, overfitting and oversearching. EVC is a mechanism that aims at solving all three problems.

Overfitting is usually seen as constructing overly complicated and detailed hypotheses in order to better fit the data. The problem is traditionally dealt with by various restrictions on the model language and the search procedure, or by posterior simplification of the constructed models.

Overconfidence is related to overfitting; discovered hypotheses, mostly complicated, tend to be assigned an exaggerated, overly optimistic probabilities of being true or another related statistics. The reason lies in the way the machine learning algorithms operate. In the standard use of statistics, the hypotheses are made in advance and then tested on independent data sets, which gives unbiased estimates of their true statistical properties. Machine learning constructs hypotheses from the data and during the induction process implicitly tests them on this same data, which makes the estimates unreliable.

As an example, assume that $h_1, h_2, \ldots$ are all possible hypotheses producible by a certain learning algorithm, and let $q_1, q_2, \ldots$ be their corresponding qualities. The hypotheses can be, for instance, classification rules, and the qualities can be the probability of the predicted classes or the χ^2 statistics computed on a 2×2 table of the hypothesis' predictions and the true classes. The task of the learning algorithm is to (a) compute the qualities of all hypotheses and (b) to select the best hypothesis h_{max}, such that $\forall i : q_{max} \geq q_i$.

In all practical cases the qualities are estimated on a data sample, therefore $\widehat{q_i} = q_i + \epsilon_i$. It is reasonable to assume that $\widehat{q_i}$ is an unbiased estimate of q_i, so $E(\epsilon_i)$ over different possible samples equals zero. The problem occurs at point (b), where the algorithm picks up a single "optimal" hypothesis, selected not by its true q_i but by $q_i + \epsilon_i$. In common conditions in machine learning – the data sample is small while the number of competing hypotheses is huge – the error terms can easily overwhelm the differences in qualities, so the algorithm chooses the "luckiest" hypothesis instead of the one with the highest true quality.

Hence, while it is possible to get unbiased estimates of the quality of individual hypotheses, the machine learning algorithm would most often choose one of those for which the estimate highly exceeds the true value. In this context, ϵ_i measures the *optimism* of the assessment.

For illustration, observe the correlation between the true and the estimated class

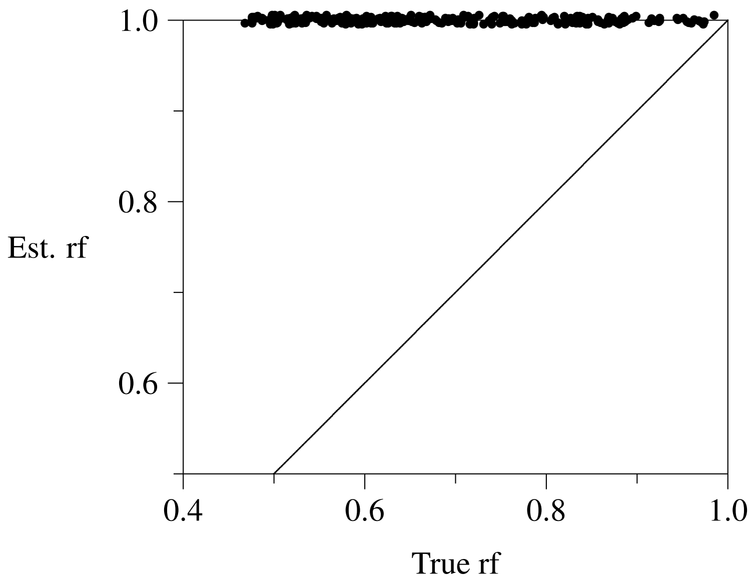


Figure 6.1: Comparison of the estimated relative frequency of the best rule and its true relative frequency. Each dot represents one experiment.

probabilities on a set of artificial data sets with controlled class probabilities for each possible rule. We prepared 300 data sets with ten binary attributes. Five attributes in each data set were unrelated with the class. For the other five, we prescribed a (random, taken from uniform distribution) class probability for each combination of their values. We then generated 2^{10} examples for each data set, one for each combination of attribute values, and assigned the classes randomly according to the prescribed probabilities for the combination of informative attributes. Note that the actual class proportions in the data set do not necessarily match the defined probabilities for a particular combination of attribute values.

For each data set the algorithm searched for the rule with the highest estimated target class probability $\widehat{q_{max}}$ and was always able to find one in which it equaled 1.0 (Fig. 6.1). The true probabilities of the target class are however uniformly distributed between 0.5 and 1.0. The induction algorithm is blind with regard to selecting the best hypothesis as well as with regard to realistically estimating its probability.

The focus of this section is on the problem of estimation, that is, correcting the overly optimistic estimate of the hypothesis chosen by the learning algorithm. We will show how to, in principle, correct the estimates in the border cases with no optimism and extreme optimism, and the realistic case. To turn the theory into a useful method for our algorithm ABCN2, we developed an algorithm for correcting optimism in rule learning.

6.1.1 Related work

The problem of overfitting, which is known in statistics for a long time, has become more important with more research groups working on more data sets for more problems every day, and even acute with the development of data mining and machine learning techniques which are designed to fit and, to any extent we allow them to, overfit the data (see, for instance [Ioa05]).

The common remedy for the problem is to use a separate data set to validate the findings. The drawback of this procedure is that it reduces the training data set and is also unsuitable for comparing the competing hypotheses during the induction process. Besides, a single validation is again prone to random effects, while cross-validation estimates the successfulness of the learning algorithm and not the quality of a particular model.

Quinlan and Cameron-Jones [QCJ95b] showed that extensive searching often produces less accurate results. A similar result was found by comparing complexity of models and their accuracy, as complex models are usually obtained by more searching. There were several approaches to balancing the complexity and accuracy. Principles like MDL, regularization, imposing prior as in Bayesian learning all try to penalize over-complicated models.

In the context of hypothesis testing, Bonferroni adjustment ([Hol79]) can sometimes be used to reduce the computed significance of hypotheses. This is however of little use in machine learning where millions and billion of hypotheses would make any finding insignificant. The Bonferroni correction also assumes mutual independence of hypotheses, which is highly violated in machine learning and makes the correction overly conservative. Since some hypotheses are considered only implicitly, it is difficult to compute the true number of tested hypotheses. Finally, this method corrects the probability of *finding a hypothesis with such a score if the hypothesis is actually random*, which is not what we are usually interested in, that is, the *probability that the hypothesis is not random*. Similar criticism also applies to other significance correction procedures like those by [Hol79] and [Hoc88].

The basic supposition of Jensen and Cohen [JC00] is similar to ours, that is, the learning algorithms measure $\widehat{q_i}$ and treat it as if it was an unbiased estimate of the true q_i, which becomes a problem when the hypothesis h_i is not a randomly chosen hypothesis but the one with the highest $\widehat{q_{max}}$, where the optimism term ϵ_i might have had a greater role than the quality of the hypothesis itself. As solutions to the problem they list using the already mentioned Bonferroni correction, new data sample, cross validation and randomization.

Our approach is based on the extreme value theory ([FT28; Col01]). Similarly to the central limit theorem which states that the sample averages of random variables with finite variance are distributed approximately normally, the extremal types theorem states that all distributions of maximal values of data samples can be approximated by one of three distributions. For instance, for a normally distributed variable X, the value of $X_{max} = \max(X_1, X_2, X_3, \dots, X_n)$ is distributed according to Gumbel's distribution ([Col01]). In machine learning, the distributions with such shapes have already been experimentally found (but not identified as such) by [JC00].

6.1.2 The general principle of extreme value correction

Many machine learning algorithms adopt in some way the following learning scheme ([JC00]):

1. Generate n candidate hypotheses $h_1, \dots, h_n$.
2. Evaluate them with evaluation function q on a given data sample S; $\widehat{q_i} = q(h_i, S)$.
3. Return the best hypothesis h_{max} according to q and S; $\widehat{q_{max}} = \max(\widehat{q_1}, \dots, \widehat{q_n})$.

The evaluation $\widehat{q_i}$ of hypothesis h_i is an estimation of the true q_i computed on a sample of examples. In statistical terms, $\widehat{q_i}$ is instantiation of a random variable Q_i whose value depends on the data sample. We can assume that $\widehat{q_i}$ is an unbiased estimate of the true q_i, e.g. $\widehat{q_i} = q_i + \epsilon_i$ where $E(\epsilon_i) = 0$.

While this holds for a randomly chosen hypothesis, machine learning algorithms select the hypothesis with the highest $\widehat{q_i}$. The large $\widehat{q_i}$ can be either due to a high true quality q_i or due to luck, ϵ_i. Let h_{max} be the chosen hypothesis and let $\widehat{q_{max}}$ be its quality. Different samples can generally yield different hypotheses, and $\widehat{q_{max}}$ is another random variable from distribution Q_{max} defined by maxima over all possible random samples, e.g. $P(x > x_0)$ equals the proportion of samples for which there exists a hypothesis with $q_i > x_0$.

Generally, $\widehat{q_{max}}$ is a positively biased estimate of the true quality of h_{max}. First, $\widehat{q_{max}}$ is unbiased if there is only a single hypotheses being tested (as it is common in classical statistics) or when there exist a hypothesis h_j which is so much better than the others that it wins in (most) samples. In this case $\widehat{q_{max}}$ (almost) equals $\widehat{q_j}$ which we assumed to be an unbiased estimate of q_j.

Now, let there be two hypotheses h_j and h_k with equal qualities $q_j = q_k$, while the qualities of other hypothesis are much lower, $\forall i, i \neq j \neq k : q_j - q_i \gg \epsilon_j - \epsilon_i$. The learning algorithm would then choose either h_j or h_k, hence for a particular sample $\widehat{q_{max}} = \max(\widehat{q_j}, \widehat{q_k}) = q_j + \max(\epsilon_j, \epsilon_k)$. It is obvious (for a formal proof see [JC00]) that $E(\max(\epsilon_j, \epsilon_k)) \geq E(\epsilon_j)$ which in our case means that $E(\max(\epsilon_j, \epsilon_k)) \geq 0$. Further on, the optimism increases with the number of competing hypotheses, while increasing the number of inferior hypotheses does not affect the distribution of q_{max} and the related optimism.

In this section we describe a general procedure for correcting the optimism of q_{max}. We will illustrate it using an artificial data set with 1000 binary attributes and 100 examples in two classes C_1 and C_2, 50 examples in each. In each experiment we assign to each attribute X_i the true probability $P(X_i = x_1|C_1) = P(X_i = x_2|C_2)$. Attributes are generated randomly according to this probabilities and independently from each other. The task is to identify the best attribute according to the χ^2 statistics for its relation with the class value, and then estimate its true χ^2 on the entire population.

Bounds for extreme value correction

As shown above, the estimates are not optimistic if there exists a hypothesis h_i which is significantly better than all others. This case requires no correction since $E(\widehat{q_{max}}) = E(\widehat{q_i}) = q_i$.

Let us illustrate this claim with our simple experiment. We constructed 10 data sets for each probability $maxP$ between 0.5 and 0.9 with step 0.03. Probability $maxP$ was set as conditional probability $P(X_i = x_1|C_1)$ of the first attribute given class value. The probabilities for the rest 999 attributes are set to 0.5. Figure 6.2 shows a graph comparing estimated averaged $\widehat{\chi^2}$ of h_{max}'s and the theoretical χ^2 on a sample of this size. If the conditional probability of the first attribute is close to 0.5, the best evaluation still suffers from high optimism, since alternative hypotheses are near. However, as the probability increases, the optimism diminishes and the estimated value $\widehat{\chi^2}$ becomes a good approximation of the theoretical value.

The largest optimism is manifested in the opposite case, when all hypotheses have equal quality. In this case optimism equals $E(\max(\epsilon_1, \epsilon_1, \dots \epsilon_N))$. A correction assuming this scenario would have to impose the highest reduction in best hypothesis'

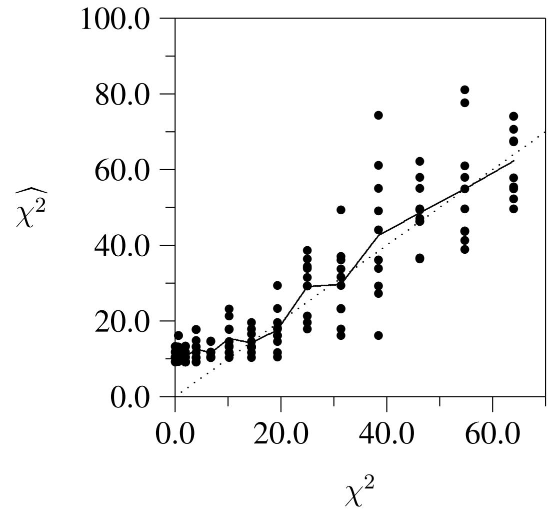


Figure 6.2: Original, uncorrected values of χ^2 versus the true values. One attribute (out of 999) stands out w.r.t. its correlation with class value, others are uncorrelated with class. Each dot corresponds to the winning hypothesis from one trial. The line shows the average for each χ^2 and the dotted line shows the optimal relation $\widehat{\chi^2} == \chi^2$.

quality.

Let $\widehat{q_i}$ be the quality of (supposedly) the best hypothesis h_i (that is, q_{max} for this sample) found by a machine learning algorithm, and let Q_i be the unknown distribution of $\widehat{q_i}$ over different samples. Q_i depends upon the true q_i, which we want to assess.

The methods starts by measuring the average quality of the best hypothesis induced from data with randomized class values. By randomizing the data we assure that all hypotheses have the same, known quality q.

1. Permute classes of the training examples to remove correlations between hypotheses and the class.

2. Find and evaluate the best hypothesis. Store the evaluation.

3. Compute the average π_p of the stored evaluations.

4. Repeat steps 1-3 until the standard error of the average is small enough.

Let Q_p be the random variable representing the quality of the null hypothesis, that is, of the hypotheses explored by the learning algorithm on our randomized data. If

	C_1	C_2
$X_i = x_1$	35	15
$X_i = x_1$	15	35

(a) Expected contingency for $P(X_i = x_1|C_1) = 0.7$

	C_1	C_2
$X_i = x_1$	44	6
$X_i = x_1$	6	44

(b) Contingency table of the best scored attribute.

	C_1	C_2
$X_i = x_1$	34.3	15.7
$X_i = x_1$	15.7	34.3

(c) Expected contingency of the best scored attribute.

Table 6.1: Contingency tables for the pessimistic correction

the hypotheses are evaluated by the χ^2 statistics, Q_p comes from the corresponding χ^2 distribution.

$P(Q_p > \pi_p)$ is then the probability of getting hypotheses better than π_p by chance assuming that the qualities are distributed according to Q_p. Similarly, $P(Q_i > \widehat{q_{max}})$ is the probability of getting $\widehat{q_{max}}$ by chance assuming that all qualities q_i are instantiations of the same random variable Q_i (which is the starting assumption for the pessimistic correction). Since the number of candidate hypotheses is equal in both cases, we assume that both probabilities are equal:

$$P(Q_p > \pi_p) = P(Q_i > \widehat{q_{max}}) \tag{6.1}$$

The distribution Q_i is the only unknown in the above equation; we know the shape of its distribution (e.g. χ^2) and we know that it depends on the true q_i. The q_i thus has a value which results in such a Q_i that $P(Q_i > \widehat{q_{max}})$ equals the already known $P(Q_p > \pi_p)$.

We will illustrate the method on a data set like the above, except that all conditional probabilities will equal $P(X_i = x_1|C_1) = P(X_i = x_2|C_2) = 0.7$. We will use χ^2 to measure the quality of hypotheses. The expected 2×2 contingency table of all attributes is shown in Table 6.1a.

The best attribute in our generated data set has the contingency table from Table 6.1b. Its $\widehat{q_{max}}$ equals

$$\widehat{q_{max}} = \frac{(44-25)^2}{25} + \frac{(6-25)^2}{25} + \frac{(44-25)^2}{25} + \frac{(6-25)^2}{25} = 56 \tag{6.2}$$

Now we estimate the average quality π_p on randomized data. After 200 repetitions of randomization, the estimated average for our domain is 11.28. Since

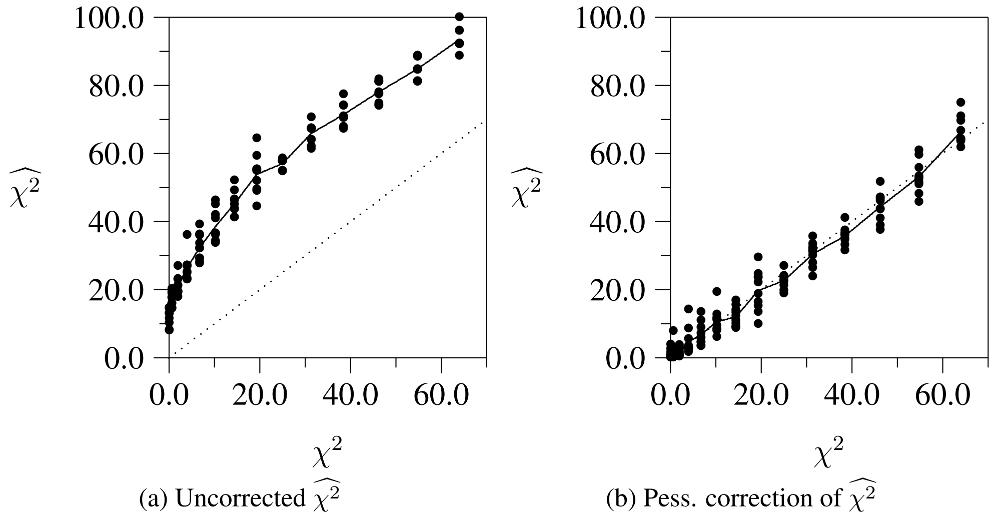


Figure 6.3: Original and corrected estimates of χ^2. All hypotheses have the same true quality.

$Q_p \sim \chi^2(1)$, the probability $P(Q_p > \pi_p)$ equals 0.00078. Having $P(Q_p > \pi_p)$ and $\widehat{q_{max}}$ it only remains to discover the distribution of random variable Q_i such that $P(Q_i > 56) = 0.00078$. Random variable Q_i is defined by the true contingency table of the attribute. The question with our quality measure is thus: what is the expected contingency table, that the chances of obtaining data as in Table 6.1b are 0.00078? Although the correct values can be in principle computed analytically, it is more practical to find it using simple bisection making use of the fact that we only have one degree of freedom, hence we only fit one variable while the others depend on it.

Table 6.1c shows expected frequencies for which the χ^2 gives the correct expected value:

$$\begin{aligned}\widehat{q_{max}} &= \frac{(44-34.3)^2}{34.3} + \frac{(6-15.7)^2}{15.7} + \\ &+ \frac{(44-34.3)^2}{34.3} + \frac{(6-15.7)^2}{15.7} = 11.2 \end{aligned} \tag{6.3}$$

Now we use these frequencies to compute the corrected value of $\widehat{q_{max}}$, $\overline{q_{max}}$:

$$\begin{aligned}\overline{q_{max}} &= \frac{(34.3-25)^2}{25} + \frac{(15.7-25)^2}{25} + \\ &+ \frac{(34.3-25)^2}{25} + \frac{(15.7-25)^2}{25} = 13.8 \end{aligned} \tag{6.4}$$

$$\begin{tikzcd} \widehat{q_{max}} & & \overline{q_{max}} \\ \text{EVD}_p \arrow[dr] & & Q_p \\ & P_a \arrow[ur] & \end{tikzcd}$$

Figure 6.4: Outline of the proposed procedure for general correction of estimates

We repeated this experiment for different settings of $P(X_i = x_1|C_1)$ with ten different random data sets for each. Figure 6.3 shows the original and corrected estimates. The corrected estimates fit the diagonal line almost perfectly.

Extreme value correction.

Extreme value correction (EVC) is the general method for correction of quality estimates. It is similar to the correction of the largest optimism, however without the assumption that all hypotheses have the same quality. Its idea is depicted in Fig. 6.4. The learning algorithm finds the best hypothesis h_{max} with quality $\widehat{q_{max}}$ which is an optimistic estimate of the h_{max}'s true quality q_{max}. If the hypotheses were induced from random data, $\widehat{q_{max}}$ would be from a distribution of extreme values which we shall denote by EVD$_p$. If we knew EVD$_p$, we could compute the significance of $\widehat{q_{max}}$, that is, the probability P_a that the learning algorithm would find a hypothesis with $\widehat{q_{max}}$ even if all hypotheses were actually random.

On the other side, unbiased estimates of random (or null) hypotheses are distributed by a known distribution depending on the used quality measure, say χ^2. Knowing the unbiased estimate $\overline{q_{max}}$ of the quality of h_{max}, we would again be able to compute its significance, which would equal the significance P_a computed by $\widehat{q_{max}}$ and EVD$_p$.

Our procedure will first estimate EVD$_p$, compute P_a, and then find $\overline{q_{max}}$ which gives the same P_a. The computed $\overline{q_{max}}$ is the unbiased estimate which we are looking for. Intuitively, this method transforms the extreme value distribution of the best hypothesis' quality in a given learning problem into its single version distribution, namely the distribution of the quality of the best hypothesis selected without search assuming we can select it with the same rate of success with respect to the probability P_a.

Fisher and Tippett [FT28] have shown that extreme values, $\max X_1, X_2, \ldots X_n$ for X's coming from any distribution can be approximated by one of three extreme value distributions, which can be generally formulated as

$$F(x;\mu,\beta,\xi) = \exp\left\{ - \left[1 + \xi \left(\frac{x-\mu}{\beta} \right) \right]^{-1/\xi} \right\} \tag{6.5}$$

The three parameters describe the distribution's *location* (μ), *scale* (σ), and *shape* (ξ).

Let EVD_p be a general EVD for the case where all hypotheses are unrelated to class. We can fit its parameters with the following randomization procedure.

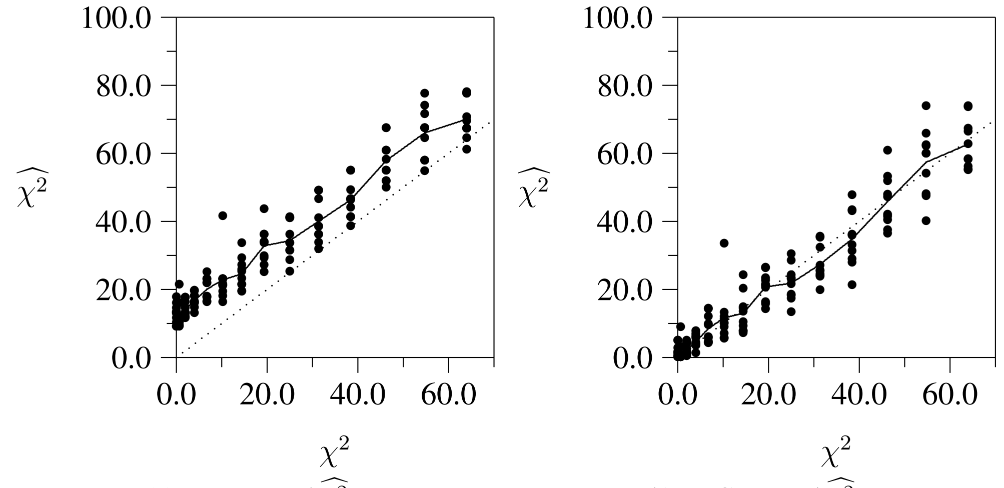
1. Permute classes of the training examples to remove correlations between hypotheses and the class.
2. Find and evaluate best hypothesis. Store best evaluation.
3. Fit parameters of EVD_p on all best evaluations computed by now.
4. Repeat steps 1-3 until expected errors of parameters are small enough.

The significance $P_a = P(EVD_p > \widehat{q_{max}})$ can be computed from the cumulative distribution of EVD. The concrete method for calculation of the "unbiased" estimate $\overline{q_{max}}$ which gives the same P_a then depends upon the chosen measure of quality.

The procedure is based on the assumption that all q_i for random hypotheses come from the same distribution. In the next section we will show an example for which this condition is not fulfilled, but also show a neat technical workaround.

Here, it is crucial to understand the semantics of extreme value correction. The value $\overline{q_{max}}$ corresponds to a value that gives the correct P_a when tested with a standard statistical test without the use of EVD. However, is it really unbiased? It is definitely not in all possible cases, since in both boundary cases we also get unbiased estimates, if the assumption on distribution of hypotheses' quality is satisfied. Yet, the assumed distributions for bounds are rather extreme, as for this technique we conjecture that the assumed distribution of true qualities is somewhere in between. Our experiments suggests that the distribution of the true qualities should be approximately normal, however by now, we were unable to construct a formal proof of our claim, which still falls under the category of future work.

To experimentally validate the EVC method, we constructed 10 data sets for each $maxP$ between 0.5 and 0.9 with step 0.03. The probability $P(X_i = x_1|C_1)$ for each attribute was randomly drawn from normal distribution $N(0.5, (maxP - 0.5)/3.2)$. This way 99.9% of attributes have either $P(X_i = x_1|C_1)$ or $P(X_i = x_1|C_2)$ between



(a) Uncorrected $\widehat{\chi^2}$ (b) EVCorrected $\widehat{\chi^2}$

Figure 6.5: Original and corrected estimates of χ^2. True qualities are distributed normally.

0.5 and $maxP$. The results of the extreme value correction are shown in Figure 6.5. Again we observe that the original estimates are optimistic while the adjusted values match the true values very well.

6.2 Extreme value correction in rule learning

Most rule learning algorithms, as well as CN2, induce models by iteratively searching for the best rule and removing the examples covered by it [FF05]. Rules are usually sought by a beam search, which gradually adds conditions to the rule with aim to decrease the number of covered "negative" examples, while at the same time losing as few "positive" examples as possible. The search is guided by two measures, one which evaluates the partial rules and the other which selects between the final rule candidates; here we will use the common approach where the same measure is used for both purposes.

A good rule should give accurate class predictions, or, in other words, have a high probability of the positive class among all examples (not only learning examples) covered by the rule. Hence a reasonable choice for the measure of rule's quality is the relative frequency of the predicted class:

$$\widehat{q_i} = \frac{\widehat{s_i}}{\widehat{n_i}} \tag{6.6}$$

where n_i is the number of learning examples covered by the rule r_i, and s_i is the number of positive examples among them.

However, this is an optimistic estimate of the true relative frequency q_i, as we have already shown theoretically as well as experimentally (Fig. 6.1). We will assume that the estimate of the number of examples that the rule covers is unbiased, $\widehat{n_i} = \overline{n_i}$ or $E(\widehat{n_i}) = n_i$, and correct the problem by finding an unbiased estimate of s_i, $\overline{s_i}$. This is possible in correction of relative frequency, as it is enough to correct only one value to remove optimism (alternative would be to correct value $\widehat{n_i}$).

Machine learning algorithms often use the m-estimate ([Ces90]) to shift the probabilities toward the prior distributions,

$$Q_i(m) = \frac{s_i + m \times p_a}{n_i + m} \tag{6.7}$$

where p_a is the prior probability and m is a parameter of the method. Fürnkranz [FF05] showed that the m-estimate presents a trade off between precision (relative frequency) and linear cost metrics, for instance, weighted relative accuracy [LFZ99; TFL00]. Different values of the parameter m can be used to approximate many common evaluation functions. For instance, when $m = 0$, m-estimate equals the relative frequency, and when $m = 2$ and $p_a = 0.5$, it equals the Laplace formula for probability, which is used in evaluation of unordered rules in CN2 ([CB91]).

To put the m-estimate to a test, we again induced a single rule for each of the 300 data sets from the introduction, this time using the m-estimate with different values of m (0, 2, 10, 20, 50, 100). With increasing values of m, the method is still optimistic for rules with lower true probability, but pessimistic for rules with higher true probability (Fig. 6.6)*. It seems that m-estimate lowers the estimated quality by the same amount for all rules, which can not adjust the estimates to lie closer to the ideal diagonal line representing the perfect correlation.

Although the m-estimate with a suitably tuned m can considerably decrease the error of the estimated probabilities, this effect seems to come from reducing the optimism by pushing the predicted probabilities towards the average, while the correlation between the true and the estimated probability remains rather poor. Thus, m-estimate and the many other similar techniques are not a satisfactory solution to the problem of overfitting, inaccurate rule quality estimates and optimistic probability predictions.

Non-linear metrics, like χ^2 or Foil's information gain, are an alternative to the linear metrics [FF05]. However, they do not seem to be solving the problem with

*We obtained similar results in experiments with other ways of constructing artificial data sets.

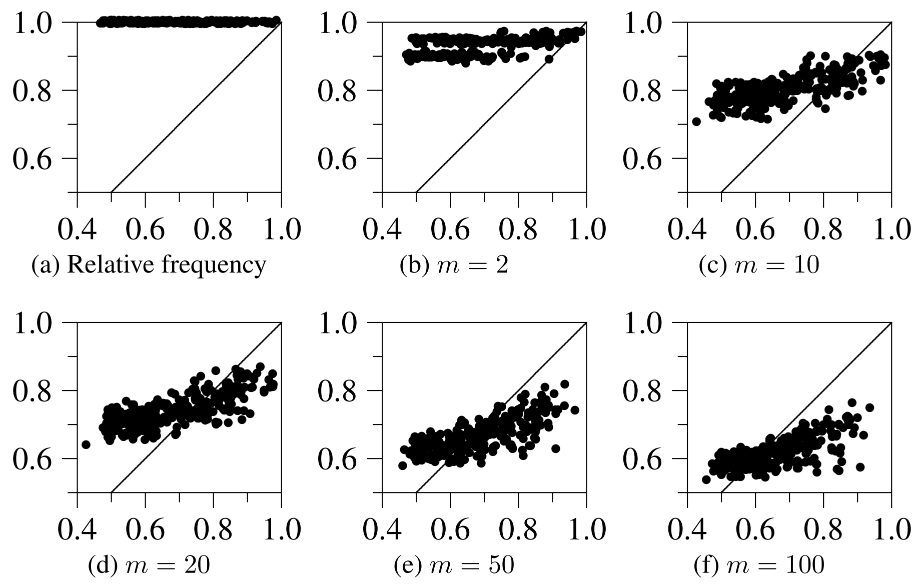


Figure 6.6: Relation between the estimated class probability $\widehat{q_{max}}$ (y-axis) and true (x-axis) class probability q_{max} for the best rules constructed from artificial data sets.

$$\begin{tikzcd}
\widehat{s_i} \arrow[d] & & \overline{s_i} \\
\widehat{LRS_i} \arrow[dr, "\text{EVD}_p"'] & & \overline{LRS_i} \arrow[u] \\
& P_a(r_i) \arrow[ur, "\chi^2"'] &
\end{tikzcd}$$

Figure 6.7: An outline of the proposed procedure

optimistic estimates. For instance, Foil's information gain is almost linear, therefore not much improvement can be expected there, while we have already demonstrated the optimism of χ^2 in the previous section. In the following, we will present an extreme value correction procedure tailored for rules that can be used with any rule evaluation measure. The work is based on our earlier published work [MDvB06], but put into a more general context as described in the previous section.

6.2.1 EVC algorithm for relative frequencies.

The outline of the proposed procedure is illustrated in Fig. 6.7. It differs slightly from the general algorithm described in section 6.1.2, as it does not directly correct the evaluation s_i/n_i of a rule, but another well related measure of quality, LRS_i. This is needed since the use of extreme value distributions requires that the values of random variables come from fixed distributions [Col01]. Likelihood ratio statistics LRS_i (definitions follows below) is distributed according to $\chi^2(1)$ and, since it disregards the number of covered positive and negative examples, fulfils this criteria, while s_i/n_i, which comes from $\beta(s_i, n_i - s_i)$ and is different for each rule, does not.

Step 1: From $\widehat{s_i}$ to $\widehat{LRS_i}$.

Let s be the number of positive examples covered by some rule and let s^c be the number of positive examples not covered by the rule. Similarly let n be the number of all covered examples and n^c be the number of examples that are not covered by the rule. LRS for 2×2 tables derived by [Dun93] as:

$$\mathrm{LRS} = 2\left[s \log \frac{s}{e_s} + (n-s)\log\frac{n-s}{e_{n-s}} + s^c \log\frac{s^c}{e_{s^c}} + (n^c - s^c)\log\frac{n^c - s^c}{e_{n^c-s^c}}\right] \tag{6.8}$$

where e_x is the expected value of x. For instance, e_s is computed as $n\frac{s+s^c}{n+n^c}$, when computed on a randomly chosen rule. Note that a similar formula for LRS, without the last two terms, was used in the original CN2 papers [CN89; CB91] for computing significance of rules. However, as that formula is approximately correct only if n is small enough when compared to n^c, we prefer to use the formula 6.8. In our case, we compute the $\widehat{LRS_i}$ by applying it to the estimates $\widehat{s_i}$, $\widehat{s_i^c}$, $\widehat{n_i}$, and $\widehat{n_i^c}$.

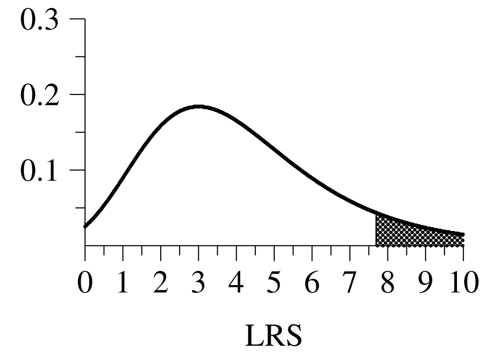
Example. We have a data set with 20 examples where the prior probability of the positive class is 0.5. Learning from that data, the rule search algorithm found a rule r_i with two conditions which covers 10 examples with 8 of them belonging to the positive class. According to Formula 6.8, its LRS is 7.7.

Step 2: From $\widehat{LRS_i}$ to $P_a(r_i)$.

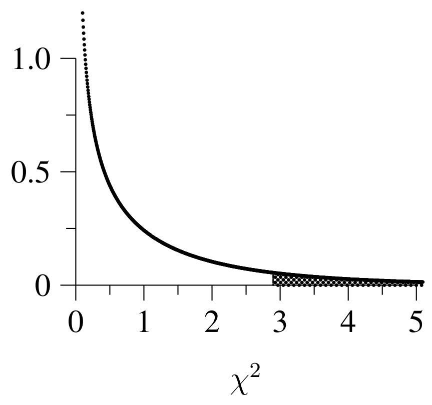
Since LRS is distributed according to $\chi^2(1)$, its extreme value distribution can be approximated in a simpler form called the Fisher-Tippett distribution:

$$F(x; \mu, \beta) = e^{-e^{-(x-\mu)/\beta}} \tag{6.9}$$

Parameters μ and β depend upon the number of rules covered by the search (which does not necessarily equal the number of *explicitly* evaluated rules), which



(a) Fisher-Tippett extreme value distribution ($\mu = 3$, $\beta = 2$)



(b) χ^2 with 1 degree of freedom.

Figure 6.8: The FT-EVD and χ^2 probability density functions

1. Let $L = 1$ (L is the maximum rule length).
2. Permute values of class in the data.
3. Learn a rule on this data (using LRS as evaluation measure), where the maximum length of rule is L.[†]
4. Record the LRS of the rule learned.
5. Repeat steps 2-4 to collect a large enough (say 100) sample of LRSs
6. Estimate parameters $\mu(L)$ and $\beta(L)$ of the Fisher-Tippett distribution (see appendix 6.4 for some tricks on decreasing the errors of $\mu(L)$ and $\beta(L)$).
7. If $\mu(L) > \mu(L-1)$, then $L = L + 1$ and return to step 2.

Figure 6.9: The algorithm for computing parameters of the Fisher-Tippett distributions

in turn depends upon the rule length and the data set and, of course, the search algorithm. Due to their independence of the actual rule, we can compute values $\mu(L)$ and $\beta(L)$ for different rule lengths before we begin learning, using the algorithm shown in Fig. 6.9.

During learning we use the cumulative Fisher-Tippett distribution function to estimate $P_a(r_i)$ for each candidate rule using the pre-computed parameters.

Example (continued). Say that the algorithm from Fig. 6.9 found $\mu(2) = 3$ and $\beta(2) = 2$ (remember that rule r has two conditions). The curve with such parameters is depicted in Fig. 6.8, so the probability $P_a(r_i)$ for the rule from our example corresponds to the shaded area right of LRS=7.7. $P_a(r)$ equals approximately 0.09.

Step 3: From $P_a(r_i)$ to $\overline{LRS_i}$.

To compute $\overline{LRS_i}$ we need to do the opposite from the last step. Looking at the $\chi^2(1)$ distribution (Fig. 6.8), we need to find such a value of $\overline{LRS_i}$ that the area under the curve to the right of it will equal the computed $P_a(r_i)$. In other words, the shaded areas under the curves in Fig. 6.8 should be the same.

Example (continued). The corresponding $\overline{\mathrm{LRS}_i}$ for our examples as read from Fig. 6.8 is 2.9. Note that this is much less than $LRS = 7.7$, which we computed directly from the data and which would essentially be used by an unmodified rule induction algorithm.

Step 4: From $\overline{LRS_i}$ to $\overline{s_i}$.

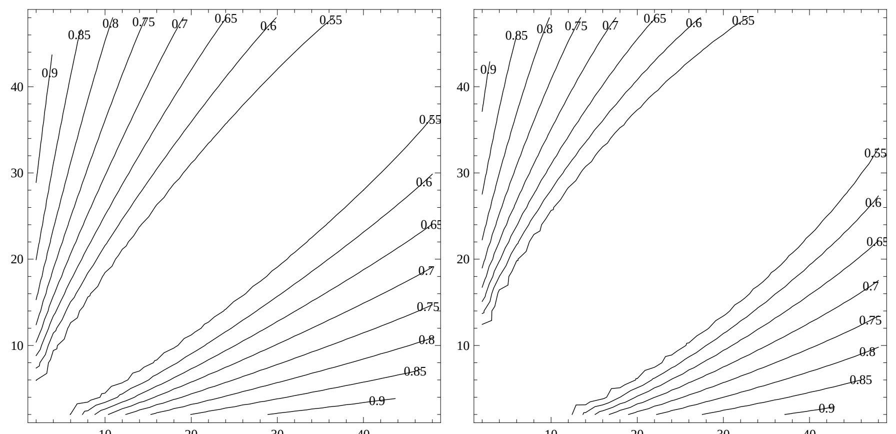
The remaining task is trivial: compute $\overline{s_i}$ from the formula for $\overline{LRS_i}$ using an arbitrary root finding algorithm. In our task we are correcting probability estimates based on relative frequencies, so we shall compute them by dividing the corrected $\overline{s_i}$ by $\widehat{n_i}$.

Example (conclusion). We used Brent's method [Atk89] to find that $\overline{LRS_i} = 2.9$ corresponds to $\overline{s_i} = 6.95$. The rule covers ten examples, so the corresponding class probability is $6.95/10 = 0.695$. Note that this estimate is quite smaller than the uncorrected 0.8.

6.2.2 Extreme value corrected relative frequency in PN space

PN-space, introduced by Fürnkranz [FF05], is a visualization of rule evaluation metrics and their behavior at different coverages and ratios between positive and negative examples. The isometrics in a such diagram connect different combinations of covered positive and negative examples that are given the same quality by the selected measure.

[†]Note that using LRS at a given rule length will always order rules the same as would $\overline{LRS}$. However, as we will be using $\overline{s_i}/\widehat{n_i}$ in the actual learning phase, in order to correctly estimate parameters of Fisher-Tippett distribution, measures $\overline{s}/\widehat{n}$ and $\overline{LRS_i}$ should be well correlated.

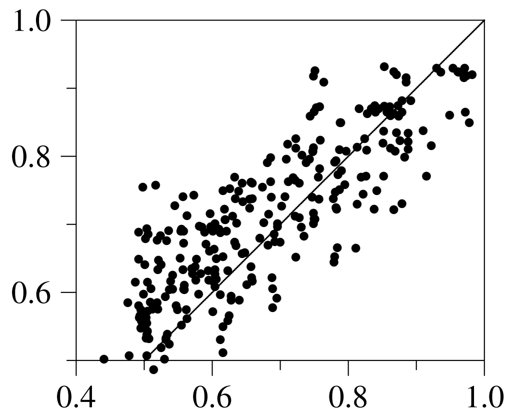


(a) $\mu = 3, \beta = 2$ (b) $\mu = 10, \beta = 2$

Figure 6.10: PN-space (ordinate is P, abscissa is N) for EVC with different values of parameters in the Fisher-Tippett distribution. Labels on isometrics correspond to corrected relative frequencies. Upper left and lower right parts are symmetric since they correspond the cases in which one or another class contains the majority of the examples covered by the rule.

Figure 6.10 shows isometrics for EVC using two different extreme value distributions. In both cases we have 50 positive and 50 negative examples. In the left diagram we used Fisher-Tippett with location parameter $\mu = 3$ and in the right diagram μ was set to 10. Higher location parameter is usually used when the algorithm compares a larger number of candidate hypotheses, therefore we can look at the latter metric also as one for rules with more conditions (where search was deeper), while the former (on the left) as one for rules with fewer conditions.

Both diagrams contain a large central space where the qualities of rules are less than 0.55. These rules have high probability of being found by chance, hence their qualities are penalized the most. Due to the higher location parameter of EVD in the right diagram, its central space is larger. This is correct since the probability to find an equal ratio of positives and negatives increases by extending the search. Furthermore, the diagrams also nicely show that rules of different lengths and with the same covered class distribution get a different evaluation. Therefore, longer rules are penalized more, as their expected optimism is higher due to a wider search.



Average quality: 0.68

Spearman correlation: 0.83

Mean squared error: 0.007

Figure 6.11: Relation between the corrected (y-axis) and the true (x-axis) class probability.

6.2.3 Experiments

We have tested the algorithm on artificial data described in introduction and on a selection of data sets from the UCI repository [AN07]. In all experiments we used CN2 [CB91; CN89] with a beam width set to 5. The algorithm was implemented as a component for the rule based learner in machine learning system Orange [DZ04].

Artificial data set

The results of using the corrected measure on the artificial data are shown in Fig. 6.11. The estimated class probabilities are nicely strewn close to the diagonal axis, which is a clear improvement in comparison with the results from Fig. 6.6.

UCI data sets

There are assumptions behind extreme value correction, which are impossible to verify on the real data. To test the practical usefulness of our correction, we observed its behavior on a set of UCI data sets [AN07]. Each data set was split evenly onto learn and test sets. For learning we then generated ten bootstrap samples from the learn set.

We ran the algorithm on the bootstrap samples and then used the examples from the test set to count the number of positive and the number of all examples covered by each induced rule. We took this ratio to be the true positive class probability for the rule (although it is, as a matter of fact, still only an estimate, it is at least an unbiased

Data set	rel	m=2	m=10	m=20	m=50	m=100	EVC
adult	0.43	0.13	0.08	0.06	0.06	0.06	**0.04**
australian	0.41	0.12	**0.05**	**0.05**	**0.05**	0.08	**0.05**
balance	0.25	0.12	0.11	0.11	0.08	0.07	**0.05**
breast (lju)	0.39	0.14	0.09	0.09	0.08	**0.06**	**0.06**
breast (wsc)	0.15	0.08	0.13	0.18	0.26	0.30	**0.05**
car	0.07	0.06	0.05	0.04	0.03	**0.02**	0.04
credit	0.41	0.11	0.07	0.07	**0.06**	0.07	**0.06**
german	0.42	0.13	0.06	0.06	0.05	0.05	**0.04**
hayes-roth	0.26	0.10	0.16	0.21	0.26	0.29	**0.08**
hepatitis	0.35	0.12	**0.05**	0.06	0.09	0.09	0.07
ionosphere	0.27	0.05	0.06	0.09	0.13	0.13	**0.03**
iris	0.20	0.07	0.09	0.12	0.17	0.22	**0.04**
lymphography	0.28	0.10	0.17	0.21	0.22	0.24	**0.05**
monks-1	0.07	0.07	0.16	0.13	0.15	0.20	**0.06**
monks-2	0.40	0.13	0.10	0.11	0.07	0.08	**0.05**
monks-3	0.32	0.09	0.08	0.11	0.14	0.13	**0.03**
mushroom	**0.00**	0.01	0.08	0.13	0.18	0.25	0.01
pima	0.48	0.15	0.05	**0.04**	**0.04**	**0.04**	0.05
SAHeart	0.46	0.19	0.08	0.07	**0.05**	0.07	0.07
shuttle	0.26	0.13	0.18	0.14	0.17	0.19	**0.11**
tic-tac-toe	0.19	0.03	0.07	0.14	0.24	0.30	**0.01**
titanic	**0.01**	0.02	0.04	0.04	0.04	0.03	0.02
voting	0.28	0.08	0.10	0.12	0.11	0.10	**0.04**
wine	0.09	0.07	0.14	0.20	0.24	0.31	**0.05**
zoo	0.16	0.09	0.22	0.31	0.42	0.47	**0.04**

Table 6.2: Mean squared errors of estimates by relative frequencies (rel.)m, m-estimate and our method (EVC)

one, since it is computed from the test data).

Results in Table 6.2 show that we succeeded in improving the probability estimates: the probability estimates by our method are far more accurate than those by any m in the m-estimate measure. The average rank of EVC is 1.6, while ranks of other estimates range from 3.84 (m=2) to 4.88 (relative frequency). The differences are highly significant (the Friedman test gives $p < 0.001$), and the Bonferroni-Dunn test at $\alpha = 0.05$ shows that EVC is significantly better than any method we compared it with.

This would, however, be easily achieved and surpassed by a method returning a single rule covering all examples and which would estimate the probability with the prior class probability. To test that our gains are not due to oversimplification we also computed the average AUC over the ten bootstrap samples. To make predictions from lists of rules, we used a simple classifier that takes the first rule that triggers for each class (we get one rule for each class), and normalize the class probabilities of these rules to sum up to 1. Although there exist better classifiers from a set of rules, we believe that using them would not considerably change the ranking of examples and the related AUC. Table 6.3 shows that the performance of our method in terms of AUC is comparable to that of the other methods. The differences here are less significant (the Friedman test gives $p = 0.64$), and the Bonferroni-Dunn test does not recognize the EVC as being significantly better than the other methods.

6.2.4 Extreme value correction in argument based rule learning

In the induction of an argument-based rule, the algorithm searches a smaller rule space when compared to learning standard rules. The larger impact on search reduction is due to the reasons of positive arguments that comprise the condition part of the rule. The part of the rule stemming from an argument is not a result of search, but provided from expert, and therefore is not the culprit for optimism in rules. The EVC should thus correct only the optimism "produced" by the other part of the condition. The negative arguments, on the other hand, have only a negligible effect in reducing the size of the search space and do not need to be considered by EVC.

The only required change to the original EVC algorithm is to learn the parameters μ and β for all possible initial size of arguments. The Algorithm is shown in Figure 6.12. The result is a set of parameters μ and β for each possible length of a rule and each possible length of the argument in the rule. When a rule in ABCN2 is evaluated, use the corresponding parameters given rule length L and argument length A.

6.2.5 When extreme value correction should be used?

The EVC method commits to the following three main assumptions:

- *Number of candidate hypotheses should be high enough.* Otherwise the maximum values will not converge to the extreme value distribution, and the related probability P_a could not be correctly estimated. Number of hypotheses is pos-

Data set	rel	m=2	m=10	m=20	m=50	m=100	EVC
adult	0.74	0.76	0.76	0.77	0.77	0.78	**0.84**
australian	0.85	0.87	0.88	0.88	0.88	0.88	**0.91**
balance	**0.82**	0.81	0.81	**0.82**	**0.82**	0.81	**0.82**
breast (lju)	0.60	**0.62**	0.60	0.58	0.60	0.60	**0.62**
breast (wsc)	0.97	0.97	0.97	0.97	0.96	0.96	**0.98**
car	0.84	0.84	0.85	0.86	0.89	**0.90**	**0.90**
credit	0.82	0.88	0.88	0.88	0.87	0.88	**0.91**
german	0.69	0.68	0.69	0.68	0.69	0.69	**0.73**
hayes-roth	0.88	0.89	0.87	0.86	0.87	0.86	**0.90**
hepatitis	**0.77**	0.76	0.76	0.73	0.73	0.71	**0.77**
ionosphere	0.90	0.91	0.89	0.89	0.89	0.91	**0.92**
iris	**0.97**	0.95	0.95	0.95	0.95	0.95	0.95
lymphography	0.78	0.81	0.83	**0.85**	0.84	0.83	0.81
monks-1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
monks-2	0.66	**0.67**	0.66	0.66	0.64	0.65	0.64
monks-3	0.97	0.98	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
mushroom	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
pima	0.68	0.72	0.72	0.72	0.72	0.73	**0.76**
SAHeart	0.59	0.62	0.63	0.63	**0.65**	**0.65**	**0.65**
shuttle	**0.99**	0.98	0.98	**0.99**	**0.99**	**0.99**	0.98
tic-tac-toe	0.96	**1.00**	**1.00**	**1.00**	0.99	0.94	**1.00**
titanic	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**
voting	0.95	0.95	0.96	0.96	**0.97**	**0.97**	0.96
wine	0.97	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.94
zoo	**1.00**	0.99	0.99	0.99	0.99	0.99	0.99

Table 6.3: AUC of classification rules constructed using relative frequencies (rel.)m, m-estimate and our method (EVC)

itively related to the number of attributes, independence between attributes, number of examples, and complexity of hypotheses space.

- *Distributions of hypotheses' qualities should be approximately normal.* Especially problematic are cases with only a few good hypotheses, while others are much worse. In such a case, EVC will still find the best hypothesis, but EVC will penalise it more than needed.

- *The true evaluation of the best hypothesis is close to 1.0.* The problem of extreme situations, where the best evaluation is close to the maximum possible

```
1. Let $L = 1$ ($L$ is the maximum rule length).
2. Permute values of class in the data.
3. Learn a rule on this data (using $LRS$ as evaluation measure), where the maximum length of rule is $L$.‡
4. Let $A = 0$ be the size of an argument (number of reasons in an argument).
5. While $A < L$ do:
6. Compute $LRS_A$ of the rule learned, where prior distribution is not original prior distribution, but the class distribution after $A$ conditions.
7. End While
8. Repeat steps 2-7 to collect a large enough (say 100) sample of $LRS_A$s
9. Estimate parameters $\mu(L, A)$ and $\beta(L, A)$ for each $A$ of the Fisher-Tippett distribution (see appendix 6.4 for some tricks on decreasing the errors of $\mu(L, A)$ and $\beta(L, A)$).
10. If $\mu(L, 0) > \mu(L - 1, 0)$, then $L = L + 1$ and return to step 2.
```

Figure 6.12: The algorithm for computing parameters of the Fisher-Tippett distributions in argument-based rule learning

value, is that evaluations can not be optimistic. However, as EVC assumes optimism, the corrected evaluation of such hypotheses will be, again, pessimistic.

In theory, all three assumptions should be considered before applying EVC, however in practice, due to the qualitative nature of assumptions, this is virtually impossible. We suggest, therefore, a pragmatic approach; to try different evaluation techniques with cross-validation, select the one with highest score and continue with the argument-based process.

6.3 Probabilistic covering

The parameters of EVD need to be computed for the whole domain before learning starts. One of the domain properties that affect these parameters is also the size of the learning data set. As the classical covering strategy removes all positively covered example ([CB91]), we would need to re-estimate these parameters each time when a rule is induced and the covered examples are removed. The alternative could be weighted covering algorithm proposed by [LFTW04], where each example's weight in learning is inversely related to the number of rules covering it. However, such strategy changes the class distribution after each learned rule, which also affects the parameters of EVD. Unfortunately, estimation of the parameters is time consuming, and re-estimating parameters each time a rule is learned would make the use of EVC with the standard CN2 covering procedure or with weighted covering algorithm very inefficient. To evade the problem we propose a new strategy called *probabilistic covering strategy* where the number of learning examples and their class distribution remains constant during the complete induction process.

Let $x.prob$ be the quality of the best rule covering x. If there are no rules covering x then $x.prob$ equals the prior probability of the example's class. When a new rule R is learned, the removing procedure updates all probabilities of covered examples as $x.prob = maximum(x.prob, q(R))$, where $q(R)$ is as usual the estimated class probability in rule. We say that, when an example x becomes covered by a new rule, where $q(R) > x.prob$, then rule R *improves* the probability of this example; or shorter: rule R improves example x. We call this *probabilistic covering*.

Furthermore, we have to enforce certain changes to the procedure of learning a single rule to prevent learning the same rule all over again, as probabilistic covering only records how well an example is explained and does not affect the way in which a rule is learned. We propose the following changes:

Selection of best rule A new rule can be learned only if it improves at least one example. This must be added as a condition in the algorithm.

Selection of N most promising rules (star) In the original CN2 algorithm best N rules are selected according to $q(R)$. This heuristic fails in our case, as we seek for a rule that has high quality *and* will improve at least one example. Let Z be a variable with normal distribution approximating the binomial distribution of a rule R. Its parameters are:

$$\begin{aligned}\mu &= |X_R| \times r(Q) \\ \sigma^2 &= |X_R| \times r(Q)(1 - r(Q))\end{aligned}$$

The probability $1 - P(Z < x.prob)$ is the probability that a randomly selected value with given average and variance will be higher than $x.prob$, which we take as a measure of hardness to improve the example x with a specialisation of rule R. Furthermore, if the probability is interpreted as the probability of improving example x, then the formula

$$EI(R) = \sum_{x_i \in X_R} [1.0 - P(Z < x.prob)] \tag{6.10}$$

corresponds to the expected number of improved examples from a rule. This is the formula used in ABCN2 to select the N most promising rules.

Probabilistic rule covering is evaluated in the next chapter together with the PILAR classification technique.

6.4 Computing parameters of extreme-value distribution

Section 6.2.1 describes an algorithm for computing extreme distributions of rules learned from random data which involves calculating the parameters of extreme value distribution for a vector of maxima of evaluations of rules distributed by χ^2 with 1 degree of freedom. The limiting distribution of all χ^2 distributions is Fisher-Tippet ([FT28; Gum54; GL54]).

The cumulative distribution function of Fisher-Tippet distribution is

$$P(x < x_0) = e^{-e^{\frac{\mu - x_0}{\beta}}} \tag{6.11}$$

where μ and β are parameters of the distribution. Distribution's mean, median, and variance are

$$\text{mean} = \mu + \beta\gamma, \quad \text{median} = \mu - \beta * \ln\ln 2, \quad \text{var} = \pi^2\beta^2/6 \tag{6.12}$$

where γ is Euler-Mascheroni constant 0.57721. The natural way to compute the parameters μ and β from the sample would be to first estimate the variance from the data and use it to compute β, followed by the estimation of μ from the sample's mean or median. However, the error of estimation of variance and mean propagates to estimations of parameters μ and β, where variance is a bigger problem than mean, as it is used for estimation of both parameters.

Gupta [Gup60] showed that for p independent and identically distributed values taken from χ^2 with one degree of freedom, where p is large, the following properties hold for their maxima M:

$$E(M) = 2\ln p - \ln\ln p - \ln\pi + 2\gamma \tag{6.13}$$

$$m(M) = 2\ln p - \ln\ln p - \ln\pi - 2\ln\ln 2 \tag{6.14}$$

$$\sigma(M) = \sqrt{2/3\pi^2} \tag{6.15}$$

Since $\sigma(M)$ is independent of the number of values p (although it has to be large), combining 6.12 and 6.15 gives $\beta = 2$. On the other hand, when p is not large, β will be less than 2, as the variance of χ^2 is less than the variance of its corresponding extreme value distribution. Particularly, in the extreme case with $p = 1$, the Fisher-Tippet would try to approximate the χ^2 distribution, where β would be 1.1. Therefore, to compute β, we need to first estimate the variance of extreme values, compute β, and trim it to values between 1.1 an 2.

Afterwards β was estimated, we can proceed to estimation of the remaining parameter μ. In our algorithm we computed the median from the vector of maximum values, so μ equals the median plus $\beta\ln\ln 2$.

Chapter 7

Classification from Rules and Combining ABCN2 with Other Methods

The main advantage of rule learning methods is in their power to spot and explain local regularities. Each rule concisely explains the correlation between the class and a set of attributes on the subspace covered by the rule. Their locality, however, raises problems when used in classification, as in some cases clashes occur and sometimes there are no rules that would cover the example to be classified. In the former case we need to apply any of the available conflict resolution strategies, while in the latter the default class is the most obvious choice.

In this chapter, we shall first give a short review of different approaches used for classification from rules and then describe a novel one called PILAR (Probabilistic Improvement of Learning Algorithms with Rules). PILAR is an algorithm for resolving rule-conflicts and also for combining rule learning with any other machine learning method. As shown at the end of the Chapter, it is possible with PILAR implicitly make any machine learning method capable of learning with arguments.

PILAR receives as input a base method and a set of unordered rules. By comparing probability predictions of the base method with class probabilities in given rules, the algorithm then observes whether in any of the patterns described with rules, there is a significant difference in probabilistic prediction of the base method and the corresponding rule. In such a situation, the prediction of the base method is accordingly corrected.

The main idea of the algorithm is to represent given rules as constraints. These constraints are used in an optimisation algorithm that makes the base method consistent with these constraints. The algorithm is experimentally evaluated with two base methods; majority classifier (as classification from rules only) and logistic regression (linear probabilistic classifier that can not account for interactions between attributes).

7.1 Related work

Lindgren [Lin04] described several approaches to combining conflicting rules. They can be divided into simple approaches, e.g. the original CN2 classification and naive Bayesian classification and in more complicated meta-learning techniques that learn on the subset of examples covered by conflicting rules in classification. Lindgren showed that the latter techniques do achieve better classification accuracies, however they are inappropriate for our needs - we need the classifications to be explainable in order to reflect arguments in rules.

Another family of methods for combining conflicting rules falls into the category of ensemble methods. Normally, one could think that any kind of ensemble learning is far from interpretable, however in the case of rule learning the learned model is simply a linear sum of rules' contributions. Eventually, the induced model is not much different from a linear regression model, whereas rules act as variables. The first such system was SLIPPER [CS99] that combined the idea of boosting [FS97] with the RIPPER [Coh95] algorithm. Variations of the ensemble learning principle can be later found in several methods for rule aggregation [WI00; RK06; FP08; DKS08]. We also decided that our algorithm PILAR should induce a linear classifier from rules. We will give a detailed motivation for our decision later in the Chapter.

7.2 The PILAR algorithm

The problem described above can be formalized as follows.

Inputs:

- A set of learning examples $\mathbf{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where one example is as usual a pair of attribute-values and a class value. Let the domain of class variable be values $c_1, \ldots, c_m$.

- A global classification model M. The expression $M_j(x_i)$ returns the probability of class c_j for example x_i.
- A set of probabilistic unordered rules $\mathbf{R} = R_1, \ldots, R_k$. Each rule R_i has a condition part defining the subspace covered by the rule, a class predicted by the rule, and the probability of predicted class given conditions.

Output:

A corrected classification model M', based on M and consistent with the probabilistic predictions of rules **R**.

7.2.1 Log-linear sum of unordered rules

Let x be an example we would like to classify. We define the log-linear model for correction of classification model M in two-class domains (c_0, c_1) as a weighted sum (**W** is a real-valued vector):

$$\begin{aligned} f(x) &= \ln \frac{M_1(x)}{1 - M_1(x)} + \mathbf{W} \cdot \mathbf{R}(x) = \\ &= \ln \frac{M_1(x)}{1 - M_1(x)} + w_0 + w_1 \times R_1(x) + \ldots + w_k \times R_k(x), \end{aligned} \tag{7.1}$$

where $M_1(x)$ is the probability given by the base method for class c_1 and weights $w_1, \ldots, w_k$ given to rules are all nonnegative reals. The term $R_i(x)$ is defined as:

$$R_i(x) = \begin{cases} 0, & \text{if conditions of } R_i \text{ are false for example } x; \\ 1, & \text{if } R_i \text{ predicts class } c_1; \\ -1, & \text{if } R_i \text{ predicts class } c_0. \end{cases} \tag{7.2}$$

The predicted probability of class c_1 for example x is computed from $f(x)$ through the logit link function:

$$\widehat{P}(c_1|x) = M'_1(x) = \frac{1}{1 + e^{-f(x)}} \tag{7.3}$$

Although this particular model can only be used in domains with two classes, it can be extended to its multi-class version, in the same way as basic logistic regression is extended into a multinomial logistic regression [HL00]. We mentioned that all the weights should be nonnegative. Negative weights are implausible, since the rule's effect on $P(c_1|x)$ would be inconsistent with the rule's class in conclusion. For example, a rule predicting class c_1 should not be allowed to decrease the probability $P(c_1|x)$.

The motivation for using log-linear classification from rules comes from its following four features:

1. Natural translation from the weighted sum to probabilistic prediction. Regardless of the value of $f(x)$, the predicted probability $P(c_1|x)$ will always be between 0 and 1.

2. Handling correlations between rules. If two rules are dependent (or even the same in the most extreme case), the method for fitting weights can assign a lower weight to one of these two rules.

3. Handling conflicting rules. These rules will have differently signed contributions in the log-linear sum, hence the probability predicted to examples where both rules trigger will be somewhere in between the probabilities predicted to examples where only one or the other rule triggers.

4. Log-linear model enables simple and understandable explanations of classifications by the means of a nomogram [Har01; MDKZ04]. The nomograms were successfully used for several methods that can employ a log-linear model (e.g. logistic regression, SVM, etc.). We show a visualisation of a model obtained from rules in Chapter 7.3.4.

7.2.2 Rules as constraints

We will first describe the transformation of a set of rules into a set of constraints that should be satisfied by the corrected classification model M'. Remember that a probabilistic IF-THEN rule R has the following structure:

$$\text{IF } Complex \text{ THEN } \widehat{P}_R(c_j)$$

The condition term $Complex$ determines the examples that are covered by this rule, while the conclusion predicts class c_j and an estimation $\widehat{P}_R(c_j)$ of its conditional probability $P(c_j|Complex)$. It will soon be evident that the estimation $\widehat{P}_R(c_j)$ needs to be *unbiased*, namely, it gives an estimation of the true conditional probability $P(c_j|Complex)$, if averaged over all possible subsets drawn from the population where $Complex$ is true. Therefore, we shall use the EVC method to compute $\widehat{P}_R(c_j)$, which was shown to decrease bias due to extensive searching (see Chapter 6).

Algorithm 7.4 The PILAR algorithm for finding weights **W** in improved model M' that satisfy the constraint in Formula 7.9.

```
*Procedure LCR(M, Rules)*

**Let W** be the weights associated to rules. All weights are set to 0.
**Let** $M'$ be the log-linear model using weights **W** as shown in Equations 7.3 and 7.1
**Let** $s$ be 2.
**while** $s > 0.001$ **do**
  **Let** s = s / 2
  **Let** $changed = True$
  **while** changed **do**
    **Let** $changed = False$
    Find rule $R_i$ with maximum value: $\widehat{P}_{R_i}(c_j) - \frac{M'_j(X_{R_i})}{|X_{R_i}|}$.
    **Let NW = W**
    **NW**$[i]$ = **NW**$[i] + s$
    **Decrease** weights in NW, where **NW**$[j] > 0$ and $\widehat{P}_{R_j}(c_j) < \frac{M'_j(X_{R_j})}{|X_{R_j}|}$.
    **if NW**$[i] >$ **W**$[i]$ **then**
      changed = True
      **W = NW**
    **end if**
  **end while**
**end while**
**Return** $M'$
```

The log-linear model returns an estimation of class probability $\widehat{P}(c_j|x_i)$ for a specific example x_i, and a rule R gives an estimation of class probability $\widehat{P}_R(c_j)$ for a set of examples (where condition is true). First, we will define a relation between $\widehat{P}_R(c_j)$ and probability $\widehat{P}(c_j|x_i)$, and then use this relation as the leading idea of the PILAR algorithm.

Let $C_j(x_i)$ be a random variable with value 1 if the example x_i has class value c_j, and 0 otherwise. The expected value of $C_j(x_i)$ is

$$E(C_j(x_i)) = 1 \times P(c_j|x_i) + 0 \times (1 - P(c_j|x_i)) = P(c_j|x_i) \tag{7.4}$$

Note that the probability $P(c_j|x_i)$ is the true class probability of the particular example. Although we know the actual class value of this learning example, it is not necessary that this probability is 1, since there can be several examples in the whole population with equal values of attributes bearing different class values. We will denote examples covered by rule R with X_R. As the expected number of covered positive examples (again, it can be different from the actual number) in X_R is the expected sum of all $C_j(x_i)$ from X_R, and,

$$E\left(\sum_{x_i \in X_R} C_j(x_i)\right) = \sum_{x_i \in X_R} E(C_j(x_i)) = \sum_{x_i \in X_R} P(c_j|x_i) \tag{7.5}$$

the expected number of examples from class c_j in examples X_R is the sum of all probabilities for class c_j in examples from X_R. Let $M'_j(X_R)$ be the sum of probability estimations given by our log-linear model over examples X_R:

$$M'_j(X_R) = \sum_{x_i \in X_R} \widehat{P}(c_j|x_i) \tag{7.6}$$

Then, $M'_j(X_R)$ is an estimation of $\sum_{x_i \in X_R} P(c_j|x_i)$. The true conditional probability of class c_j of rule R is:

$$P_R(c_j) = \frac{\sum_{x_i \in X_R} P(c_j|x_i)}{|X_R|} \tag{7.7}$$

The $\frac{M'_j(X_R)}{|X_R|}$ is thus an estimation of $P_R(c_j)$, therefore the following equality could be used as a constraint on the learned model:

$$\frac{M'_j(X_R)}{|X_R|} = \widehat{P}_R(c_j) \tag{7.8}$$

We say that a rule is consistent with the prediction of a global model when the above constraint holds. Ideally, we would like to assign such weights to the rules in the log-linear model that would make all rules consistent. However, in real situations, this is rarely satisfiable. For example, estimation of class probability in a rule sometimes depends on rule properties (e.g. rule length), and in such situations we could encounter two different rules covering the same set of examples X_R, while having different estimations of class probabilities $\widehat{P}_R(c_j)$. Hence, for the sake of applicability, we need to relax the "ideal" constraint.

The next constraint is based on the following idea: if $\frac{M'_j(X_R)}{|X_R|}$ is higher than $\widehat{P}_R(c_j)$ and the rule's weight is 0, while other rules are consistent with the model, then rule R is redundant with respect to other rules and is not needed in the model. Therefore, we would like to have a model that uses only a subset of rules in classification, and disregard redundant ones. This can be formulated as constraint:

$$\forall R_i : \left(\frac{M'_j(X_{R_i})}{|X_{R_i}|} = \widehat{P}_{R_i}(c_j) \wedge w_i \geq 0\right) \vee \left(\frac{M'_j(X_{R_i})}{|X_{R_i}|} > \widehat{P}_{R_i}(c_j) \wedge w_i = 0\right) \tag{7.9}$$

where c_j is the class predicted by the rule R_i. This constraint is always satisfiable. The algorithm that computes such weights is shown as Algorithm 7.4.

7.3 Evaluation

PILAR is an algorithm for improving a learning method on attribute subspaces indicated by classification rules. In the extreme case, where the base method is the majority classifier, PILAR acts as an approach for classification from rules. This will also be the subject of our initial evaluation, where PILAR will be compared to some other strategies for resolving conflicting rules. Later, we will use PILAR to improve logistic regression and compare its results with the basic logistic regression, and finally we will show a visualisation of a model produced by PILAR with a nomogram.

7.3.1 Linear models vs. non-linear models

The first experiment serves as a motivation for using linear models. A comparison between PILAR (learns a linear model) and several other techniques will show that linear models perform better than the selected non-linear. Currently there are several non-linear approaches available for classification from unordered rules. They could be classified in two classes; some of them are simple [CB91] and enable classification "by hand", but they usually perform worse with respect to accuracy than some more sophisticated methods [Lin04]. As already mentioned, the latter methods are not appropriate for our purpose, because they are less understandable, therefore we shall here compare PILAR only to the former methods.

We used three different classification methods in our experiments:

CN2 classical CN2 [CN89] classification that sums class distributions of all covering rules,

BAY a classifier that combines rules with naive-Bayesian formula,

PIL PILAR,

and two different evaluation functions:

m m-estimate (m=2) formula, as used in by Džeroski in CN2 [DCP93], and

EVC EV-correction formula.

Along with the m-estimate formula, we will also set the significance threshold to 0.05, whereas in the EVC case, as the evaluation itself takes care of overfitting, this threshold will remain at 1.0.

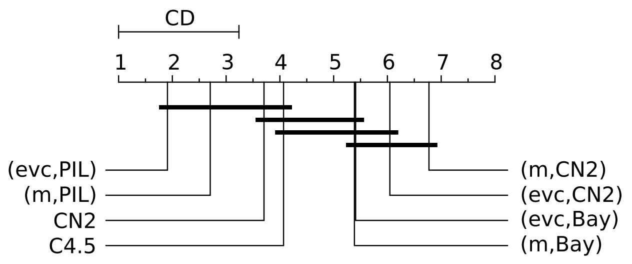


Figure 7.1: Comparisson of Brier scores's of different combinations of evaluation metric and classification strategy.

Together they call for six possible combinations, each described by a pair (evaluation method, classification method). For example, (m,CN2) is a rule learning algorithm, where m-estimate is used for rule evaluation and CN2-type of classification of rules. Furthermore, we also added the classical CN2 algorithm [CB91] for unordered rules and the C4.5 algorithm [Qui93] into comparison as the baseline algorithms in classical rule induction.

As we are focusing on probabilistic IF-THEN rules, Brier score [Bri50] will be used as the measure for comparison. The results of experiments on several UCI domains [AN07] are shown in Table 7.1.

We limit ourselves on binary-class problems only. To compare these classifiers, we follow Demšar [Dem06], and make the Friedman test, which gives 73.04. This value is well over the threshold for rejecting the null hypothesis, therefore the classifiers do not perform equally. Next, we calculate the critical difference diagram (see 7.1) according to the Nemeneyi statistics. As you can see, the statistical comparison splits methods in two parts. The left one, the better one, are methods using PILAR, while on the right part methods uses either CN2 classification method or Bayesian classification method. The original CN2 and C4.5 algorithms are in the middle. Interestingly, the original CN2 performs better than the (m,CN2), because the probabilistic covering strategy increases the correlation between rules - as the examples are not really removed after covered, which is problematic in the case of CN2 classification technique. On the other hand, PILAR deals efficiently with correlations between rules.

Table 7.1: Brier scores of CN2(with m) and CN2(with EVC) using different classifiers on several UCI data sets. Bold values mark the best method(s) for given data set.

Data set	(m,CN2)	(m,BAY)	(m,PIL)	(EVC,CN2)	(EVC,BAY)	(EVC,PIL)	CN2	C4.5
adult	0.347	0.412	0.307	0.335	0.516	**0.224**	0.287	0.261
auto-mpg	0.260	0.193	**0.125**	0.238	0.202	0.139	0.167	0.168
breast-cancer	0.388	0.541	0.420	0.379	0.457	**0.360**	0.509	0.382
crx	0.416	0.253	0.239	0.416	0.275	**0.195**	0.276	0.234
galaxy	0.268	0.115	**0.063**	0.248	0.137	0.063	0.084	0.107
german	0.412	0.507	0.451	0.405	0.518	**0.337**	0.391	0.422
heart-disease	0.436	0.383	0.320	0.415	0.344	**0.252**	0.352	0.404
housing	0.362	0.254	0.213	0.355	0.277	**0.186**	0.250	0.281
imports-85	0.240	0.159	0.153	0.222	0.147	**0.143**	0.158	0.245
kr-vs-kp	0.461	0.110	**0.011**	0.457	0.184	0.021	0.029	0.011
monks-1	0.375	0.003	0.001	0.293	0.017	0.001	**0.000**	0.030
monks-2	0.443	0.522	**0.153**	0.498	0.459	0.377	0.460	0.494
monks-3	0.308	0.025	0.024	0.227	0.027	0.027	0.067	**0.022**
prostate	0.433	0.514	0.491	0.418	0.458	**0.337**	0.443	0.453
SAheart	0.433	0.635	0.591	0.424	0.585	**0.378**	0.533	0.460
servo	0.176	0.110	**0.071**	0.154	0.118	0.102	0.095	0.111
shuttle-landing-control	0.266	0.062	**0.023**	0.153	0.098	0.045	0.024	0.033
sick	0.111	0.046	0.029	0.110	0.059	0.028	0.042	**0.022**
tic-tac-toe	0.407	0.096	**0.001**	0.399	0.081	0.013	0.014	0.221
titanic	0.377	0.368	**0.312**	0.377	0.368	0.313	0.313	0.321
voting	0.170	0.100	0.061	0.144	0.132	0.061	0.087	**0.058**
wdbc	0.240	0.100	**0.058**	0.235	0.108	0.058	0.102	0.111
Avg. rank	6.773	5.386	2.705	6.045	5.409	1.909	3.705	4.068

7.3.2 PILAR vs other linear models

Recently, using linear combination for classification from rules has become quite popular, which is manifested by a large number of publications [CS99; WI00; RK06; FP08; DKS08]. As in PILAR, these approaches assign a weight to each of the rules. By fitting weights to rules, the methods try to optimise a selected criterion (e.g. log-likelihood), which is usually extended with a regularisation term to prevent overfitting. For example, in the RuleFit system [FP08], the criterion is Brier score and the regularisation term is the "lasso" penalty. Roughly, the optimisation formula is:

$$\arg\min_{\mathbf{W}} \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda \cdot \sum_{i=1}^{m} |w_i| \tag{7.10}$$

L stands for squared error between the true class value y_i and its predicted probability $\hat{y}_i$. The method tries to minimise the formula, the sum of Brier score on learning data and the sum of absolute values of all weights given to rules. Parameter λ balances between the accuracy term and the penalising term.

It would be difficult to conduct a fair comparison of all five mentioned methods and PILAR, as implementation of those classification methods is tightly coupled with the selected rule learner, whereas we would prefer to compare those methods on the same rule set. Only such comparison would give an answer to the question: what is the best way to combine rules. Instead, we implemented a simple algorithm that contains most of the properties of the methods mentioned above. The algorithm receives a set of rules, and then (1) creates a dataset with the same number of examples as the learning set, (2) removes all attributes, (3) for each rule adds an attribute, and (4) for each new example-attribute pair assigns value 1, if corresponding rule covers the example, and 0 otherwise. The class values of examples stay the same. Afterwards, (6) logistic regression is used to fit a log-linear model on this data and to obtain weights for the rules. We used Bayesian logistic regression [GLM07] with Gaussian prior (the regularisation term), where the penalising parameter was optimised with an internal cross-validation procedure.

We compared the (EVC,PIL) method with the (EVC,LR), which takes the same rules as (EVC,PIL), however uses logistic regression (as described above) instead of PILAR. The results of comparison between the methods are shown in Table 7.2. At a first glance, there is no real difference between the approaches and also the Wilcoxon t-test [Wil45] does not reject the null hypothesis that the methods perform equally (p=0.37).

Table 7.2: Brier scores of (EVC,PIL) and (EVC,LR) methods. Bold values mark the best method(s) for given data set.

Data set	(EVC,PIL)	(EVC,LR)
adult	**0.224**	0.244
auto-mpg	**0.139**	0.152
breast-cancer	**0.360**	0.372
crx	0.195	**0.188**
galaxy	0.063	**0.058**
german	**0.337**	0.353
heart-disease	**0.252**	0.277
housing	0.186	**0.179**
imports-85	0.143	**0.131**
kr-vs-kp	0.021	**0.013**
monks-1	0.001	**0.000**
monks-2	0.377	**0.177**
monks-3	0.027	**0.023**
prostate	**0.337**	0.372
SAheart	**0.378**	0.405
servo	**0.102**	0.110
shuttle-landing-control	0.045	**0.014**
sick	0.028	**0.026**
tic-tac-toe	0.013	**0.001**
titanic	**0.313**	0.313
voting	**0.061**	0.064
wdbc	0.058	**0.048**

In terms of accuracy, both methods perform similarly and therefore we expect that also all above mentioned methods would perform similarly, if given the same rules. The only significant difference is on the monks-2 data set. Logistic regression performs better there, because it is not constrained by the sign of weights, while PILAR is not allowed to assign negative weights. In the case of monks-2, it is important to allow negative weights. Overall, the difference in accuracies on other data sets is only minimal. However, the main problem of classification with logistic regression is that the user has to select the regularisation parameter or this parameter has to be optimised with the internal cross-validation procedure. In our experiments, the (EVC,LR) needed significantly more time to fit weights when compared to (EVC,PIL). With respect to accuracy, both methods perform similarly on most data sets.

7.3.3 Improving machine learning methods: logistic regression

In this experiment, we test the ability of PILAR to improve another general method. We used stepwise logistic regression [HL00] as the base method. We point out two interesting questions:

1. Is it possible to improve probabilistic predictions of logistic regression with rules?
2. Can PILAR decrease the quality of logistic regression?

One would expect that the answer to the first question is yes. Logistic regression is basically a weighted sum of attributes, and can not by itself successfully exploit interactions between attributes. This can be a critical weakness in certain domains. On the other hand, we could also assume that increased complexity (with the use of rules) will sometimes decrease the quality of the base method, as complexity of learning method is related to overfitting.

The results of experiment (Brier scores) of logistic regression and corrected logistic regression are shown in Table 7.3. These results suggest that the answer to the first question is yes, and to the second is no. The corrected logistic regression is statistically better than normal logistic regression (Wilcoxon T-test gives $p < 0.001$), moreover, the accuracy of corrected logistic regression is only slightly lower on some data sets (Brier never increased for more than 1%), otherwise the Brier score was better - in some cases substantially better. Naturally, it is vital to regard these results as defeasible, since the results were obtained only on some domains, yet the results are still very promising.

7.3.4 Visualisation of PILAR model with a nomogram

We selected the Titanic domain for demonstration of a model explanation. Each example is described with three attributes: *sex*, *age*, and *status*, and classified whether the passenger has survived or not. The rule learner induced all together 21 rules.

The nomogram is shown in Figure 7.3.4. Only twelve rules were included in the model, others had weight set to zero. Each horizontal line corresponds to one rule. The conditions of the rule are described as text at the left. The value yes on the line denotes the rule contribution if it triggers for the example, otherwise its contribution is zero. In the first line, we have a rule that always triggers, therefore its contribution (approx. 0.25) is always added to an example. The next rule is

Table 7.3: Brier scores of logistic regression, PILAR with logistic regression, and PILAR on rules only on several UCI data sets. The latter method serves merely as a a reference. Bold values mark the best method(s) for given data set. PILAR with logistic regression is significantly better than logistic regression ($p < 0.001$).

Data set	LR	(PIL,EVC)+LR	(PIL,EVC)
adult-sample	0.244	**0.217**	0.224
auto-mpg	0.149	0.158	**0.139**
breast-cancer	0.427	0.426	**0.360**
crx	0.209	**0.193**	0.195
galaxy	0.067	**0.051**	0.063
german	0.326	**0.324**	0.337
heart-disease	**0.251**	0.251	0.252
housing	0.184	**0.175**	0.186
imports-85	0.128	**0.124**	0.143
monks-1	0.362	**0.001**	0.001
monks-2	0.463	0.421	**0.377**
monks-3	0.038	0.029	**0.027**
prostate	0.266	**0.259**	0.337
SAheart	**0.364**	0.370	0.378
servo	**0.079**	0.085	0.102
shuttle-landing-control	**0.023**	0.024	0.045
sick	0.092	0.032	**0.028**
tic-tac-toe	0.314	0.024	**0.013**
titanic	0.327	0.319	**0.313**
voting	0.070	0.073	**0.061**
wdbc	**0.049**	0.057	0.058

IF $sex = male$ THEN $survived = no$, which has contribution almost -1, etc. When classifying a new example, one needs to cumulate all contributions of relevant rules and then read the probability prediction at the bottom ruler.

The nomogram can be also used as a tool for explanation. Three rules for $survived = no$ are particularly interesting:

1. IF $sex = male$ THEN $survived = no$
2. IF $sex = male$ AND $age = adult$ THEN $survived = no$
3. IF $sex = male$ AND $age = adult$ AND $status = second$ THEN $survived = no$

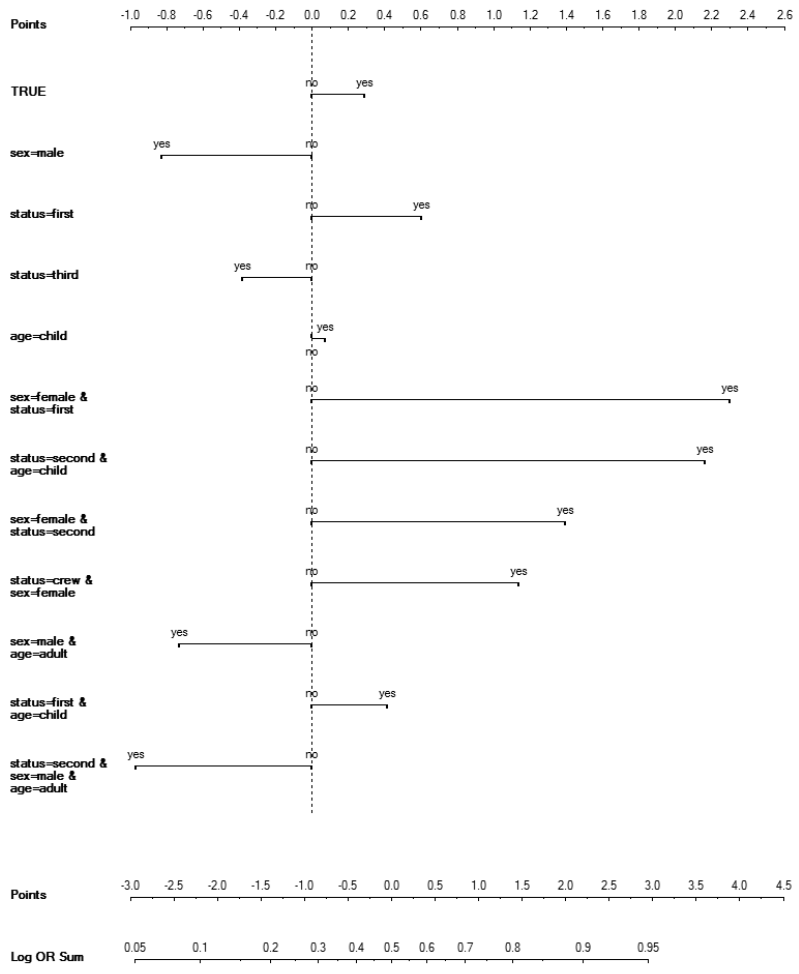


Figure 7.2: A nomogram visualising weighted rule's model for Titanic domain.

All three rules have contribution around -1. According to these rules, being a male was tough when the accident happened. However, being a male and also an adult was even worse. Note that being an adult by itself was not necessarily bad, otherwise such rule would be in the model. Furthermore, an adult male from the second class stood the least chances for survival.

7.4 PILAR + any method = any ABML method

PILAR can be used as a simple bridge between argument based knowledge and any machine learning method. We start with learning rules from argumented examples in Algorithm 5.2, but then skip the last part where rules are learned with the standard *CN2ForOneClass*. Afterwards, we can choose any machine learning algorithm to learn a model and use PILAR to combine this model with rules. The resulting model is based on the chosen learning algorithm corrected with the rules learned from argumented examples.

Although this simple algorithm can be seen as an argument-based generalisation of all machine learning algorithms, we believe that it can not replace actual extensions of these algorithms. These rules learned from arguments can only slightly correct the base model in some local spots, whereas an AB-extension of the method would actually learn a completely different model.

7.5 Discussion

PILAR is foremost a method for classification from rules. It combines rules with a weighted linear sum, where weights are automatically learned from learning data. Additionally, PILAR also allows improving probabilistic prediction of other probabilistic machine learning methods. Rules are used as constraints on the final model, requiring that its probabilistic predictions are on average similar to the probabilistic estimates given by rules.

We studied the usefulness of our method as a rule classification technique and as a correction method for logistic regression. In both cases, PILAR proved to be a promising method and statistically outperformed competing methods. Given our experimental findings, we believe that it would improve (or at least not worsen) any general classification method. Furthermore, PILAR also preserves the splendid explanation property of rules, e.g. their weights could be visualised in a nomogram.

There already exist some approaches that combine linear classifiers (e.g. logistic regression) with non-linear (decision trees) [LHF03]. While these approaches are conceptually different from ours, it would be still interesting to explore, if it is, on average, better to correct a linear classifier with rules or induce one model that contains both ideas.

Chapter 8

ABML Refinement Loop: Selection of Critical Examples

In ABML experts are asked to provide their prior knowledge in the form of arguments for specific learning examples rather than the general domain knowledge. However, asking experts to give arguments for the whole learning set is not likely to be feasible, because it would require too much time and effort. The following loop describes the skeleton of the procedure that picks out critical examples - examples that ABML can not explain without some help:

1. Learn a hypothesis with ABML using given data.
2. Find the most critical example and present it to the expert. If a critical example can not be found, stop the procedure.
3. Expert explains the example; the explanation is encoded in arguments and attached to the learning example.
4. Return to step 1.

To finalize the procedure we need to contemplate the following two questions:

- How do we select critical examples ?
- How can we achieve to get all necessary information for the chosen example?

8.1 Identifying critical examples

The main property of critical examples is that the current hypothesis can not explain them very well, or in other words, it fails to predict their class. Since ABCN2 gives probabilistic class prediction, we define the most critical example as the example with the highest probabilistic error. The probabilistic error can be measured in several ways. One possibility is to learn a hypothesis on the whole set and test on the same set, although, if the method is prone to overfitting, not many critical examples are to be found. Therefore, we propose the k-fold cross-validation repeated n times (e.g. $n = 4, k = 10$), so that each example is tested n times. The most critical example is thus the one with highest average probabilistic error.

The selection of the most mispredicted example as the one being shown to the expert has a possible deficiency. It is not always true that this example will benefit the most to the quality of the next learned hypothesis. For instance, the example could be an outlier, where the probability of encountering a similar example is small. Therefore, the argument would improve classification of this example only and none other.

This suggests a selection of a critical example that would, after being argumented, increase the most the quality (e.g. accuracy) of the learned hypothesis. Having a procedure able to do exactly this, we could expect a smaller number of iterations required to reach the best possible hypothesis, hence, reducing the overall time of expert's involvement in the process. A possible approach would be to apply a clustering algorithm[JMF99] on all critical examples and choose an example from a large cluster. An argument for this example is likely to help also other examples in the same cluster. Another approach could be to try a mechanism for outlier detection[HA04]. If an example is an outlier, there is not much chance that the expert will be able to provide good arguments, and more, such arguments are unlikely to help other critical examples. Note that a similar problem has been raised several times in active learning, see for instance [MN98; NS04]. However, by now we were not yet able to develop a credible algorithm for this task, therefore the question is still open for future work.

8.2 Are expert's arguments good or should they be improved?

Here we describe in detail the third (3) step of the above algorithm, where the expert is asked to explain the critical example. The provided arguments can be complete - they perfectly explain the example - or incomplete. In the former case, the expert provided all relevant information and the process can move to the next step. If arguments are incomplete, then ABML will sometimes be able to explain the example using arguments and inducing the incomplete part, while sometimes this will still not be entirely possible. In such cases, we need additional information from the expert. The whole procedure for one-step knowledge acquisition is described with the next 5 steps:

Step 1: Explaining critical example. In this step the expert is asked the following question: "Why is this example in the class as given?" Her answer can be either "I don't know" (she is unable to explain it) or she can articulate a set of arguments $A_1, \ldots, A_k$ all confirming the example's class value. If the system gets the first answer, it will stop this procedure and try to find another critical example.

Step 2: Adding arguments to the example. Arguments A_i are given in natural language and need to be translated into domain description language (attributes). Each argument supports its claim with a number of reasons. When a reason is simply an attribute value of the example, then the argument can be directly added to example. On the other hand, if reasons mention other concepts, not currently present in the domain, these concepts need to be included in the domain before the argument can be added to the example.

Step 3: Discovering counter examples. Counter examples are used to spot if arguments are enough to successfully explain the critical example or not. If arguments are incomplete and ABML fails to improve them, then the counter examples will show where the problem is. Here, ABML is first used to induce a hypothesis H_1 using data without new arguments and H_2 using data together with the new arguments. A counter example is defined as: it has a different class value from the critical example, its probabilistic error increases in H_2 with respect to H_1, and H_2 mentions arguments (given to the critical example) while explaining the counter example.

Step 4: Improving arguments. The experts needs to revise her initial arguments with respect to the counter example. This step is similar to steps 1 and 2 with one essential difference; the expert is now asked ”Why is the critical example in one class and the counter example in the other?” The answer is added to the initial argument.

Step 5: Return to step 3 if counter example found.

8.3 Similarity and differences with active learning

Active Learning is a type of machine learning where the learning method has certain control over the selection of learning examples. It is assumed that all possible examples are initially unclassified, and the learning method gradually selects examples, which get classified by an expert, and are then used in learning. The approach is especially useful in domains where labeling (assigning class values) to an arbitrary example takes a lot of time or/and is expensive. A good active learning method is one that selects as few learning examples as possible and reaches acceptable accuracy.

The standard loop in active learning consists of the following steps:

1. Select randomly k initial learning examples from the set of unclassified examples and obtain their class values. These examples represent the current set of classified examples E.

2. Learn a classifier C given E.

3. Select the most valuable example e (that would improve C the most if added to E) from the remaining unclassified examples .

4. Obtain class value of e, add it to the set of examples E, and return to step 2.

Different methods in active learning use different strategies to select examples in step 3.

The ABML refinement loop described in the previous sections appears similar to the loop of active learning. In each turn they both add information to the learning set of examples; in ABML, additional arguments for an example are provided, while in active learning a new example is classified and appended to the learning set. However, although they are similar, they are also very different. In ABML, we assume that all classes of examples are known, but the method can not explain some of them, therefore needs additional knowledge to explain them. On the other hand, active learning does not know the class values and searches for an example that would improve the learned hypothesis the most. This difference also necessarily defines different strategies that the approaches adopt. In the argument based machine learning loop, we try to find an example that poses the greatest problems for the learning method, while in active learning we select an unclassified example where the current classifier is the least certain how to classify. Common to both approaches is the desire to find an example with large area of effect - adding information to that example would increase the accuracy of model the most. We shall quickly review some basic approaches to active learning (a more comprehensive review can be found in [Set09]) and show how and if their ideas are suitable to be used within the ABML loop. As we shall see, all methods have limited (if any) usability for the ABML loop.

Uncertainty sampling

In uncertainty sampling, the method selects the example for which the current classifier is the least certain [LA94], since it assumes that this and similar examples are therefore not represented enough in the learning set. For example, if we have a probabilistic classifier and a two-class problem, then the method would select the unclassified example where the prediction was closest to 0.5.

This approach is intuitive in active learning, however its usefulness in argument machine learning is unclear. Asking an expert to provide arguments for an example where the method is the least certain would improve the quality of hypothesis, as the example is still problematic. Nevertheless, such an example is less critical than the one with the highest probabilistic error. Moreover, there seems to be no mechanism to select an example with largest area of effect. Hence, we expect that selection of critical examples following this strategy would require more steps in ABML.

Query by committee

Here, we first learn a set of classifiers using different machine learning methods. These classifiers are then tested on unclassified examples. The method selects the example where these classifiers disagreed the most [SOS92].

This is a similar method to uncertainty sampling. It has the same deficiencies for ABML loop: example where all classifiers are wrong would be a better selection method and there are no guidelines how to select an example with largest area of effect.

Version space reduction

This principle is described in [CAL94] and it works only for binary classes. The idea is to learn two models from all examples, the most general, and the most specific consistent classifier in the given version space [Mit97]. Consistency is defined as usual as 100% classification accuracy on learning data. The method then selects an example from the unclassified set, where these two models disagree.

This approach could be also applied in the ABML loop. The example that would be differently classified by the most general and the most specific hypothesis (example would be before removed from learning set), would be shown to an expert for explanation. However, since version spaces assume noise-free domains, its applicability to real domains is questionable.

Future error reduction

The idea of this method, sometimes referred to also as "estimated error reduction", is to select an example that will the most reduce the prediction error of the classifier [RM01]. The selection procedure builds several models from the current learning set extended by one unclassified example, which is labeled into one of the possible class values. The selected example is the one that would, on average over all assigned classes, minimise the expected error on all testing examples.

This principle would in ABML be translated to: select such an example that would, if argumented, the most reduce the error of the method. This sounds nice, especially since it would implicitly avoid selecting outliers as critical examples. However, while learning models for all possible classes for all unclassified examples is possible, learning over all possible arguments is not, since the number of all possible arguments is usually very large.

Density-weighted methods

This approach [SC08] was designed to directly deal with the problem of outliers. The main idea is that informative instances should not only be those which are uncertain, but also those which are "representative" of the input distribution (i.e., inhabit dense regions of the input space). Such methods still suffer from the same dilemma (knowing class values vs. not knowing class values), however, the combination of the method for finding the most critical example together with the method for finding a representative example seems very promising.

Part III

Experiments and Evaluation

Chapter 9

Introductory experiments

In this last part of the Thesis, we will describe several applications and experiments where ABCN2 was involved. These experiments will be used to demonstrate the functionality of ABCN2 and to compare the results of learning with or without arguments. They will also illustrate the argument based machine learning loop that is switching between learning and querying domain experts. Furthermore, these experiments shall also be used as an evaluation of the ABCN2 method. A problem with evaluation is that a statistical comparison of ABCN2 and other (non-abml) methods is virtually impossible. As each experiment involves asking domain experts questions about the domain, which requires some of their time, these experiments can relatively take a lot of time (they can not be run automatically). For this reason, we can do only a smaller number of experiments and in each of them qualitatively answer some of the following questions:

- Does learning with arguments provide favorable hypotheses in comparison with learning without arguments? When should we prefer one or the other? This will be measured in terms of accuracy and comprehensibility of learned hypotheses.
- How effective is the method for selection of critical examples, that is, how well does this mechanism guide the expert's attention to important cases and missing information?
- How difficult is it for the experts to provide their prior knowledge in the form of arguments?

- Can we efficiently reconstruct an existing knowledge base? Is it possible to use an existing knowledge base as a source of arguments, where the result of learning is then an updated knowledge base?

- What if the explanation of examples does not mention the attributes describing the example? We will see that arguments can also suggest new attributes.

- Can relevant arguments be extracted automatically from relevant literature, rather than provided by an expert?

- What if arguments provided are not perfectly correct? What if the arguments are not correct at all - how to deal with noise in experts' knowledge?

We split the evaluation to four chapters. The present one will describe four experiments with ABCN2 to illustrate the core idea of ABCN2: learning to classify animals, an experiment using the "South African Heart disease" UCI domain [AN07], an application to a legal domain, and an application to prognosis of bacterial infections in geriatric population. All four experiments are basic by nature. They do not require any special handling with arguments, where arguments are provided by an expert and are simply conjunctions of available attributes. In the following chapter, we will describe some experiments where arguments do mention only available attributes, but also others yet not present in the domain or a combination of current attributes that can not be easily expressed with a conjunction of reasons. The next chapter will demonstrate how relevant arguments can be extracted from text sources and how helpful are they. In the last chapter of this part, we will cope with the problem of erroneous arguments and how prone is ABCN2 to these errors.

9.1 Animal classification

The ZOO data set [AN07] contains descriptions of 101 animals (instances) with 17 attributes: *hair, feathers, eggs, milk, predator, toothed, domestic, backbone, fins, legs, tail, catsize, airborne, aquatic, breathes, venomous, and type*, which is the class attribute. *Type* has seven possible values: *mammal, bird, reptile, fish, amphibian, insect, and other*. The main advantage of this data set is that, just using an encyclopedia, we can provide good arguments to automatically selected critical examples. We expect that using arguments will help to improve the comprehensibility and classification accuracy of induced rules.

Figure 9.1: Rules induced by CN2 for the ZOO data set.

IF milk=yes THEN type=Mammal
IF fins=yes AND breathes=no THEN type=Fish
IF feathers=yes THEN type=Bird
IF legs=6 AND breathes=yes THEN type=Insect
IF legs=4 AND hair=no AND predator=yes THEN type=Amphibian
IF legs=0 AND venomous=yes AND catsize=no AND toothed=yes THEN type=Reptile
IF toothed=no AND legs=4 AND hair=no THEN type=Reptile

The set was split into a learning set (70%) and test set (30%). Figure 9.1 shows the set of induced rules without using any arguments.

Considering learning data alone, the induced rules fit perfectly. However, classification accuracy on the test set (which is what truly matters) is only slightly over 90%. Now, according to our method of involving argumented examples (chapter 8), we have to find the most problematic example using the learning set only. The internal (on learning set only) cross-validation procedure found that the most frequently misclassified example was a reptile, the tortoise. Notice that the rules covering reptiles split reptiles in two subgroups; in the first group are legless, poisonous, small, and toothed reptiles (snakes) and in the other are toothless, with four legs, and hairless reptiles (turtles). A problem with these two rules is that there also exist four-legged reptiles with teeth (crocodile, tuatara, etc. - tuatara was misclassified in the test set). A good argument for tortoise to be a reptile is that it has the backbone and it lays eggs (tortoise is a reptile because backbone=yes AND eggs=yes). Using that argument for tortoise in ABCN2, the two rules for reptiles were replaced by the following two rules:

- IF **backbone=yes** AND **eggs=yes** AND aquatic=no AND feathers=no THEN type=Reptile
- IF eggs=no AND breathes=no THEN type=Reptile

Naturally, our argument is not a perfect general rule for recognizing reptiles, so it was extended by ABCN2 with attributes *aquatic* and *feathers*. The first attribute separates reptiles from fish and the second from birds, as both, fish and birds, have backbone and lay eggs. It is interesting that our argument did not only affect the rule that was learned from this argument, but also the other rule for reptiles.

The next problematic example found was a sea snake (a reptile). A sea snake is an air-breathing snake that lives underwater. According to encyclopedia, a sea snake should be correctly classified with the above rule, however we noticed that, in the data, sea snake is characterized as a non-breathing reptile and that it does not lay eggs. It is obviously a mistake in data and is also the reason for the induction of the second rule. It is interesting to note how the interactive use of our AB learning method also helps to identify errors in data.

The next problematic example found in the learn set was a newt. The argument was provided that the newt is an amphibian because it has backbone, lays eggs and is related to water. This resulted in the rule:

IF **backbone=yes** AND **aquatic=yes** AND **eggs=yes** AND legs=4
THEN type=Amphibian

After adding these arguments to the two examples, ABCN2 induced a perfect set of rules for the ZOO data set. There were no misclassified examples in the test set.

This example clearly illustrates the effectiveness of our method for choosing critical examples (both identified examples were really critical) and shows how arguments can be used to learn concepts that are otherwise very hard to learn. The example also nicely illustrates that it is much easier to give arguments only to an individual example (e.g why tortoise is a reptile), than it would be to articulate general rules that correctly classify the animals. Moreover, the rules learned from arguments are consistent with prior knowledge and the resulting hypothesis achieved higher classification accuracy than the initial (learning without arguments) one.

9.2 Welfare benefit approval

In this section, we shall describe an experiment in a domain of law published in one our earlier papers [MvBC$^+$06]. At that time, we had not yet developed EVC and PILAR (Chapters 6 and 7), therefore the ABCN2 in this section uses the original CN2's classification and rule evaluation.

The problem of many areas of law - especially administrative law - is that many cases are routinely decided, which often involve the exercise of some discretion, or involve some degree of operationalisation of the concepts used in the legislation: for example aged over 65 rather than elderly [BC91b]. We would generally wish to assume that such discretion or operationalisation is consistent so that like cases are decided in a like manner, that some kind of rule is being followed.

We would like to analyse if there is a way of deciding what the rule being followed is from an automated consideration of the data. Such a question has relevance to a number of interesting and important issues:

- If there is well defined legislation which defines what the rule should be we may wish to ensure that the rule is being followed;
- If the domain is a discretionary one, we may wish to discover the rule itself;
- Similarly, we may wish to discover the way in which the stated conditions have been operationalised;
- Some people have argued that the rule followed in practice is different from the rule which exists in theory (or which might be elicited from experts). For example Edwards [Edv95] suggested that some areas of law exhibited a systematic bias. Such conjectures could be informed and justified were we able to discover the ”real” rule from a database recording the actual practice.

If therefore we had a reliable technique to extract rules explaining the data in a field of law, it would have many interesting uses.

One problem with many experiments to explore the efficacy of techniques designed to extract knowledge from data is that they use data for which the relationships present are not known at the outset. As a result, what has been discovered, and what has been missed, cannot be established definitively. Often accuracy of classification is taken to validate the knowledge extracted, but this test is, as will be discussed below, rather one-sided. In the work reported here we will use a specially constructed data set, the properties of which are known, and which is thus able to serve as a measurable test of the technique.

The data we use has been used in several previous AI and Law experiments, reported in [BC93; BCC00; JG03]. The use of this same data set allows for comparison between what can be derived using the various techniques.

9.2.1 The data set

As previously mentioned the data set used in these experiments is that first used in [BC93]. The data concerns a fictional welfare benefit. The benefit is payable if six conditions are satisfied. These conditions were chosen to represent different kinds of condition that are found in the legal domain, so that we can see whether the different form of conditions affects their discoverability.

The notional benefit was a fictional welfare benefit paid to pensioners to defray expenses for visiting a spouse in hospital. The conditions were:

1. The person should be of pensionable age (60 for a woman, 65 for a man);
2. The person should have paid contributions in four out of the last five relevant contribution years;
3. The person should be a spouse of the patient;
4. The person should not be absent from the UK;
5. The person should have capital resources not amounting to more than 3,000;
6. If the relative is an in-patient the hospital should be within a certain distance: if an out-patient, beyond that distance.

These conditions represent a range of typical condition types: 3 and 4 are Boolean necessary conditions, one which should be true and one false; 5 is a threshold on a continuous variable representing a necessary condition, and 2 relates five Boolean variables, only four of which need be true. 1 and 6 relate the relevance of one variable to the value of another: in 1 sex is relevant only for ages between 60 and 65, and in 6 the effect of the distance variable depends on the Boolean saying whether the patient is an in-patient or an out-patient. We can see these six conditions either as explicit conditions or as ways of making operational concepts such as elderly, sufficient contribution record, close relative, presence in the UK, insufficient capital resources, and attributable expenses respectively.

A possible criticism of this experiment could be based on the fact that our experimental data was artificial rather than real world. However, in machine learning experimentation with artificial data is quite common and acceptable in testing machine learning methods. It has the advantage over the use of real-world data in that the experiment is better controlled and the success of learning is easier to assess. Moreover, the underlying rules used to generate the synthetic data were not told beforehand to those conducting experiments with ABCN2, but were known only to the data owner (Trevor Bench-Capon).

For this experiment a data set of 2400 records was used: 1200 satisfying all of the conditions, and equal numbers of the remainder being designed to fail. For records designed to fail one of the conditions, satisfaction or otherwise of the remaining conditions was decided randomly for each condition separately. There are thus twelve attributes relevant to the decision: *age*, *sex*, the five contribution conditions (called *cont5*, *cont4*, *cont3*, *cont2* and *cont1*), *spouse*, *absent*, *capital*, *distance* and *inpatient*. In addition to these attributes each record contains fifty two irrelevant attributes, half of which are continuous and half Boolean. An ideal set of rules would be:

1. IF age $<$ 60 THEN qualified = no;
2. IF age $<$ 65 and sex = m THEN qualified = no;
3. IF any two of cont5, cont4, cont3, cont2 and cont1 = n
   THEN qualified = no;
4. IF spouse=no THEN qualified=no;
5. IF absent=yes THEN qualified=no;
6. IF capital$>$3000 THEN qualified=no;
7. IF inpatient=yes AND distance$>$ 750 THEN qualified=no;
8. IF inpatient=no AND distance$\leq$ 750 THEN qualified=no;

Probably we should expect (3) to be expressed as ten separate rules containing each pair of the contribution factors, which would be a total of sixteen rules to describe the problem fully.

9.2.2 Experiment with ABCN2

To evaluate the results produced by ABCN2, we split the original data (2400 records) into a learning set containing 70% of the cases, and a test set (the remaining 30%) used to assess the accuracy of the generated rules on new cases. The random selection was stratified to pertain the class distribution.

The first set of rules was generated from examples without any arguments. So the resulting rules were as if generated with CN2. These rules were:

1. IF capital $>$ 2900 THEN qualified = no;
2. IF age $\leq$ 59 THEN qualified = no;
3. IF absent $=$ yes THEN qualified = no;

4. IF spouse = no THEN qualified = no;

5. IF cont4 =no AND cont2 = no THEN qualified=no;

6. IF age $>$ 89 THEN qualified = no;

Of these (1) - (5) are correct (or very close). Nine contributions rules and the two distance rules are missing and rule (6) is wrong. There is thus considerable scope for improvement. None the less a high degree of accuracy is achieved by these six rules: 99% for both the learning and the test sets. Accuracy of classification is not, however, of prime interest: the motivations given at the beginning of section make it clear that it is the interpretability of the rules discovered that is of primary interest.

After inducing these rules with CN2, our plan was to give arguments to some of the examples in the learning set and using ABCN2 to induce better rules. In the case of our legal data, the procedure for finding the most critical example found a misclassified example that failed on the contributions condition. The argument given for this example was that *cont5*, *cont4* and *cont1* are all false. When this argumented example, together with all the other (non-argumented) examples was given to ABCN2, two additional contribution rules were induced:

IF $cont5 = no$ AND $cont4 = no$ AND $cont1 = no$
THEN $qualified = no$;

IF $cont2 = no$ AND $cont3 = no$ THEN $qualified = no$;

and the accuracy increased slightly.

In a third learning iteration, an argument was added to an additional misclassified case in which distance was too great for an inpatient. This time the erroneous rule (6) disappeared and was replaced by a rule relating to inpatiency and distance (although with an approximate threshold) and another contributions rule:

IF $inpatient = yes$ AND $distance > 735$ THEN $qualified = no$;

IF $cont1 = no$ AND $cont5 = no$ THEN $qualified = no$;

Iterations four, five and six added three more arguments based on failure of the contribution conditions, which resulted in a different contributions rule. In iteration seven, the argument was given that distance was too small for outpatient. This produced the rule

IF $inpatient = no$ AND $distance \leq 735$ THEN $qualified = no$;

and rearranged the contribution rules somewhat.

At this point there were no critical examples found any more, and so no further argumentation with our iterative procedure was possible. The final accuracy on the test set was 99.8%. This means that one out of 720 test cases was misclassified, and all the rest were classified correctly. The final set of rules were:

1. IF $capital > 2900$ THEN $qualified = no$;
2. IF $age \leq 59$ THEN $qualified = no$;
3. IF $absent = yes$ THEN $qualified = no$;
4. IF $spouse = no$ THEN $qualified = no$;
5. IF $cont4 = no$ AND $cont2 = no$ THEN $qualified = no$;
6. IF $inpatient = yes$ AND $distance > 735.0$ THEN $qualified = no$;
7. IF $inpatient = no$ AND $distance \leq 735$ THEN $qualified = no$;
8. IF $cont3 = no$ AND $cont2 = no$ THEN $qualified = no$;
9. IF $cont5 = no$ AND $cont3 = no$ AND $cont1 = no$
   THEN $qualified = no$;
10. IF $cont4 = no$ AND $cont3 = no$ AND $cont1 = no$
    THEN $qualified = no$;
11. IF $cont5 = no$ AND $cont4 = no$ AND $cont1 = no$
    THEN $qualified = no$;

(1) - (5) are all good rules and remain from the first pass. (6) and (7) have the right format, but the threshold is slightly inaccurate. The remaining four rules approximate the ten ideal contribution rules. The total number of argumented examples after the seven iterations was seven. The one misclassified example was (omitting the irrelevant attribute values): (($age = 84$, $sex = male$, $cont1 = no$, $cont2 = yes$, $cont3 = no$, $cont4 = yes$, $cont5 = yes$, $spouse = yes$, $absent = no$, $capital = 130$, $distance = 1320$, $inpatient = no$), $qualified = no$). This example is misclassified because the particular combination of contribution conditions is not covered by the approximate contribution rules induced.

Learning from artificial data sets is usually considered easier than learning from real world data sets. One reason for this is that artificial data are typically noise-free whereas real world data typically contain noise. So to cope with real world data, a learning method has to be able to deal with noise. To analyse the ABCN2's ability to deal with noise in data, we artificially introduced random noise of varying severity in our learning data as described below. Intuitively we expected that background knowledge in the form of arguments should improve the method's resistance to noise in comparison with CN2. This expectation was confirmed by the experiments.

The experimental procedure for this experiment was as follows. First, we split the data set to learning set (70%) and test set (30%). Then we added random noise into the learning set, induced rules with both CN2 and ABCN2, and measured the accuracy of both sets of rules on the (noise-free) test set. To study how the severity of noise affects the success of learning, we repeated the experiment for various rates of noise. The chosen rates were: 0%, 2%, 5%, 10%, 20%, and 40%. A noise rate p means that with probability p the class value of each learning example is replaced by a random value drawn from $\{$yes, no$\}$ with distribution (0.5, 0.5). Each time CN2 and ABCN2 were run with correspondingly "noised" examples plus the same noise-free argumented examples defined above. This whole procedure was repeated 10 times in order to obtain confidence intervals of the estimated average classification accuracy for both CN2 and ABCN2 for each noise rate.

Figure **??** and Table 9.1 show the results of CN2 and ABCN2. Both methods were run with the default settings of their parameters. These standard values are: m=2 in m-estimate, α=0.001(likelihood ratio statistics threshold), and minimal coverage of a rule is set to 2 (that is, a rule to be acceptable has to cover at least two examples).

This figure shows that, as expected, ABCN2 clearly outperforms CN2 when rate of noise in data increases. The difference in average accuracies between ABCN2 and CN2 jumped from 0.3% at 0% noise to 3.3% at 20% and to 1.7% at 40% noise. ABCN2 outperformed CN2 on 0%, 20%, and 40% noise with high statistical significance (t-test,p<0.001; see Table 9.1). ABCN2 was also the better method for other noise rates but with lower significances.

The ability to handle a large amount of noise is very important in the kind of routine legal decision making we are addressing. Errors rates are very high: Groothuis and Svenson[GS00] report experiments which suggest that 20% may be a low estimate of incorrectly decided cases. The problem is internationally widespread. The

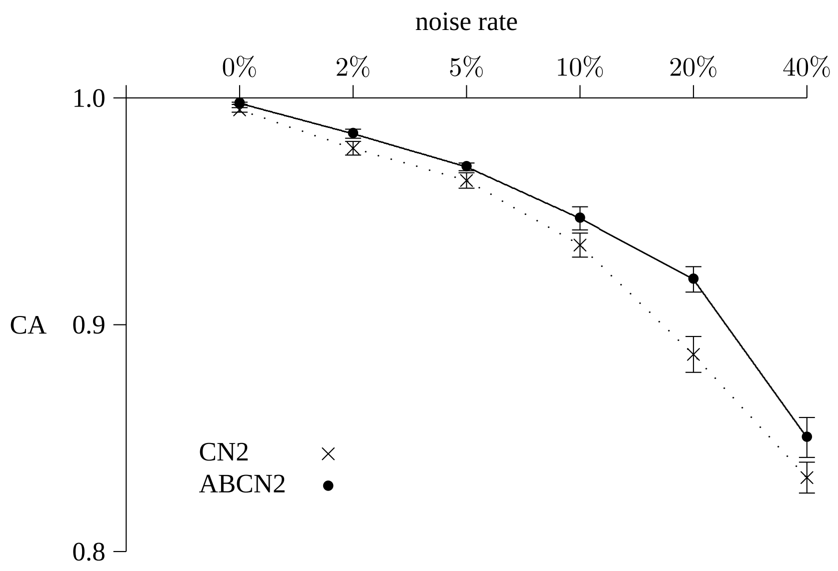


Figure 9.2: CN2 and ABCN2 compared on learning data sets with different proportions of noise in class variable.

noise	CN2	ABCN2	Sig.
0%	0.9947±0.0010	0.9976±0.0005	<0.001
2%	0.9778±0.0030	0.9842±0.0020	0.002
5%	0.9636±0.0034	0.9696±0.0017	0.005
10%	0.9351±0.0053	0.9469±0.0051	0.037
20%	0.8869±0.0079	0.9200±0.0056	<0.001
40%	0.8326±0.0068	0.8503±0.0088	0.001

Table 9.1: Results of CN2 and ABCN2 on noisy data. The first column shows noise rates, second and third contain average accuracy and standard error of accuracy estimate for CN2 and ABCN2 respectively, and the last column gives significances of differences between averages computed with pairwise t-test.

US National Bureau of Economic Research reports of a particular benefit[*]:

> "The multistage process for determining eligibility for Social Security Disability Insurance (DI) benefits has come under scrutiny for the length of time the process can take - 1153 days to move through the entire appeals process, according to a recent Social Security Administration (SSA) analysis - and for inconsistencies that suggest a potentially high rate of errors. One inconsistency is the high reversal rate during the appeals process - for example, administrative law judges, who represent the second level of appeal, award benefits in 59% of cases. Another inconsistency is the variation in the award rates across states - from a high of 65% in New Hampshire to a low of 31% in Texas in 2000 - and over time - from a high of 52% in 1998 to a low of 29% in 1982."

Again an official UK Publication produced by the Committee of Public Accounts[†]:

> "Finds that the complexity of the benefits system remains a major problem and is a key factor affecting performance. Skills of decision-makers need to be enhanced through better training and wider experience. Too few decisions are right first time, with a error rate of 50% for Disability Living Allowance. There are also regional differences in decision making practices that may lead to payments to people who are not eligible for benefits."

This makes it clear that robustness in the face of large amounts of noise is essential if a learning techniques is to be applied to data from this domain.

9.2.3 Discussion

In this experiment, we found that ABCN2 performed a little better than CN2 in terms of accuracy - which is very well indeed. However, the quality, in terms of interpretability, of the rules generated showed a crucial advantage of ABCN2. The argument based machine learning loop enabled adding domain knowledge in the form of the justification of particular decisions, and this was able to give a great improvement

[*]From Web Page: *http://www.nber.org/aginghealth/winter04/w10219.html*

[†]Getting it right: Improving Decision-Making and Appeals in Social Security Benefits. Committee of Public Accounts. London: TSO, 2004 (House of Commons papers, session 2003/04; HC406)

to the quality of the rules. Additionally, we explored the robustness of ABCN2 in the face of erroneous decisions. It was found in the experiment that ABCN2 is more robust against noise than its non-argument original CN2. This is important because many of the administrative law applications to which we would wish to apply the technique exhibit quite large error rates [GS00].

9.3 Infections in elderly population

In this section, we will describe an application of ABCN2 to a medical domain [vMVB06]. The particular problem is to build a model from data which would help the physician, at the first examination of an older patient with an infection, to decide the severity of the infection and consequently, whether the patient should be admited to the hospital or could be treated as an outpatient.

The elderly population is a unique one and that is also true from the medical perspective. Compared to younger population, people over 65 years of age usually react to a disease in a different way. Many symptoms may not even be present or they are masked by others which makes it a very difficult task for a medical doctor to diagnose a condition, to decide a proper treatment or to estimate the patient's risk of death. From a wider perspective, the proportion of elderly in the population is growing rapidly and so are the costs for medical treatment, which presents an emerging economic problem.

9.3.1 Data

The data for our study was gathered at the Clinic for Infectious Diseases in Ljubljana, from June 1st, 2004 to June 1st, 2005. The physicians included only the patients over 65 years of age with CRP value over 60 mg/l, which indicated a bacterial etiology of the infection. The patients were observed 30 days from the first examination or until death caused by the infection. The data includes 40 clinical and laboratorial parameters (attributes) acquired at the first examination for each of 298 patients (examples). The infections are distinguished with respect to the site where bacteria is found or on the clinical basis (respiratory, urinary tract, soft tissues, other). The continuous attributes were categorized by the physician. The distribution of the class attribute *death* (whether death has occurred or not) is the following:

- 34 examples (11,4%) for 'death = yes'

- 263 examples (88,6%) for ’death = no’

9.3.2 Arguments

Attribute	**Value**	
GENDER	Z	**Positive arguments**
AGE_YEARS	92	
AGE	C	DEATH=YES because RESPIRATORY_RATE_D=“>= 16”
NURSING_HOME_RESIDENT	NO	DEATH=YES because SATURATION_D=“<= 90”
COMMORBIDITY	0	DEATH=YES because BLOOD_PRESSURE_D=”<= 100”
DIABETES	NO	DEATH=YES because TEMPERATURE_D=“> 37.9”
HEART	NO	DEATH=YES because LEUKOCYTES_D=“>= 12”
KIDNEY	NO	DEATH=YES because CREATININE_D=“>= 100”
LIVER	NO	DEATH=YES because BLOOD_UREA_D=“>= 13”
LUNG	NO	DEATH=YES because NA_D=“> 147”
IMMUNITY	NO	DEATH=YES because AGE_YEARS is high
CENTRAL_NERVE_SYSTEM	NO	DEATH=YES because WEAKNESS=YES
MOBILITY	YES	DEATH=YES because CONSCIOUSNESS=DISSORIENTED
CONTINENCE	YES	
BEDSORE	NO	**Negative arguments**
CATHETER	NO	
IMPLANT	NO	DEATH=YES despite MOBILITY=YES
VOMITING	NO	DEATH=YES despite CONTINENCE=YES
DIABLOODPRESSUREHEA	NO	DEATH=YES despite TROMBOCYTES_D=“>= 100”
WEAKNESS	YES	DEATH=YES despite HEART_RATE_D=“< 100”
CONSCIOUSNESS	DISSORIENTED	DEATH=YES despite RODS_D=“< 10”
TROMBOCYTES_D	>= 100	DEATH=YES despite CRP_D=“< 150”
TEMPERATURE_D	> 37.9	DEATH=YES despite COMMORBIDITY=0
RESPIRATORY_RATE_D	>= 16	
SATURATION_D	<= 90	
HEART_RATE_D	< 100	
BLOOD_PRESSURE_D	<= 100	
LEUKOCYTES_D	>= 12	
RODS_D	< 10	
CRP_D	< 150	
CREATININE_D	>= 100	
BLOOD_UREA_D	>= 13	
GLU_D	< 15	
NA_D	> 147	
INFECTION_TYPE	RESPIRATORY	
DEATH (class value)	YES	

Table 9.2: A sample argumented example from the infections database.

The physician, who was treating the patients, provided positive and negative arguments to 32 examples, where all argumented examples were from class *death = yes*, namely she gave the reasons she believed caused death for each selected patient. A sample argumented example is shown in Table 9.2. All arguments provided mention only one attribute value as a reason, since the expert believed that influences are independent; one bad value of an attribute will always be bad, no matter what the other values of attributes are. The critical examples were selected manually by the expert prior to learning, and not by the method itself.

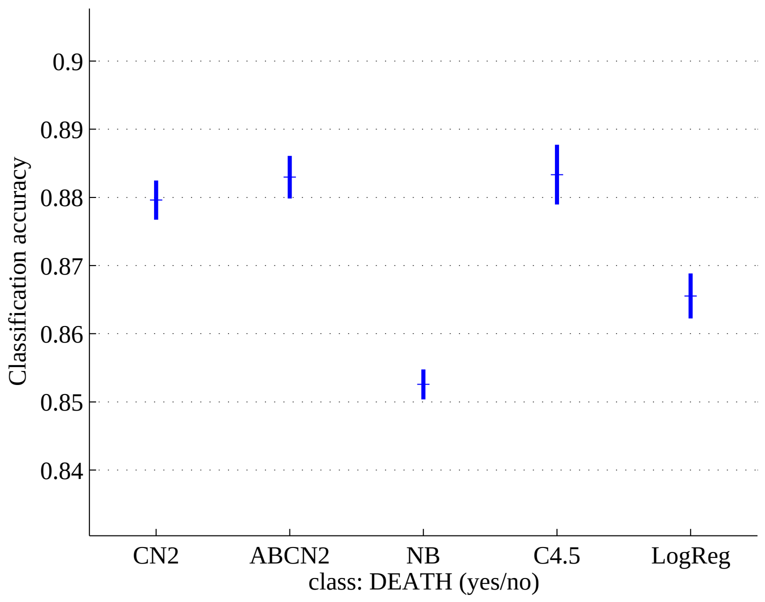


Figure 9.3: Mean values and standard errors of classification accuracy across tested methods.

One could, at this point, ask an interesting question about these arguments: whether they would, if used as rules, describe the domain sufficiently well. We built a simple classifier from the given arguments and tested it on the same data set; for each case, we counted the number of applicable arguments for class *death* = *yes* and compared this number to the number of arguments for class *death* = *no*. The accuracy of a such classifier is only slightly above 40%, therefore there is still a large space available for machine learning to improve. Since the default accuracy in this domain is 88.6% it indicates that the knowledge which is hidden in the arguments is far from perfect. However, please note that this experiment is not used to validate the expert knowledge. To do that, at least the arguments to examples from the opposite class should be given as well. Our intention is merely to show that the knowledge given by the arguments is neither perfect nor complete though it can still help to improve learning.

9.3.3 Experiments

Learning and testing was performed by 10-fold cross validation which was carried out 10 times with different random splits of examples into folds. We were able to use cross-validation in this experiment, since the arguments were provided prior to

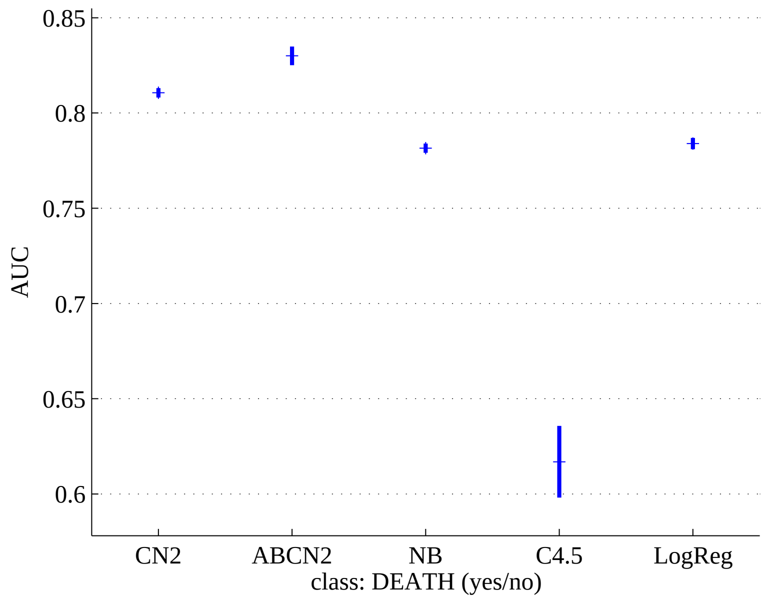


Figure 9.4: Mean values and standard errors of AUC across tested methods.

learning. We compared the algorithms ABCN2, CN2, Naïve Bayes (NB), decision tree (C4.5) and logistic regression (LogR). Algorithms are compared with regard to classification accuracy, area under ROC (AUC) and Brier score. The results are shown in Fig. 9.3- 9.5.

Observing classification accuracy, that is the percentage of correct classifications, we can see that CN2, ABCN2 and C4.5 achieve similar results while NB and LogR perform significantly worse (Fig. 9.3). Although classification accuracy is important it should be accompanied by other estimates, especially because the majority classifier itself is quite accurate in this domain due to imbalance between the two classes. Therefore, we also measure AUC and Brier score. Figures 9.4 and 9.5 show that, according to AUC and Brier score, ABCN2 significantly outperforms all other methods. It is important to note that for imbalanced domains, as our domain, AUC and Brier score are more relevant measures of success than accuracy.

9.3.4 Discussion

ABCN2 achieved better results than CN2 according to all three measures by using arguments given by expert. The question is how the induced hypotheses from both measures differ and why ABCN2 is the better method. To examine the hypotheses,

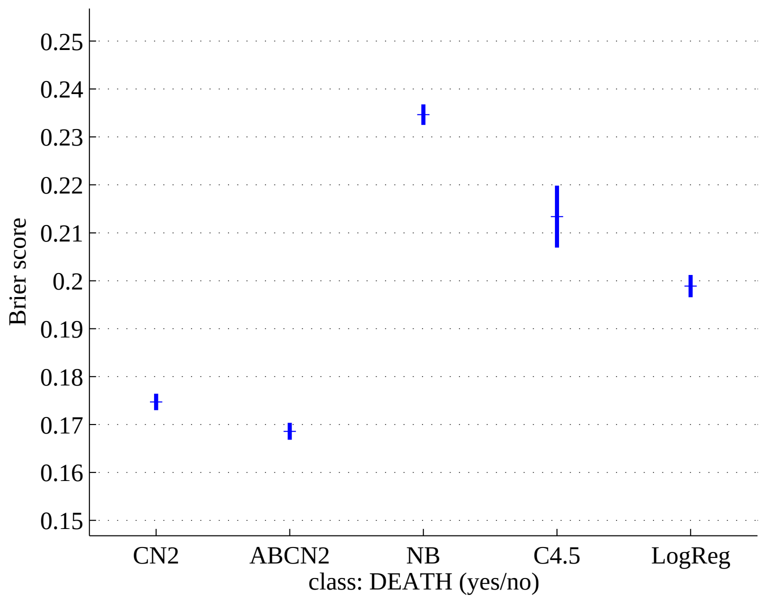


Figure 9.5: Mean values and standard errors of Brier score across tested methods.

we induced a set of rules from the whole data set with ABCN2 and CN2. As the arguments were given only to examples with class value *death=yes*, the induced rules for *death=no* were the same for both methods. Both methods induced 14 rules for class *death=yes*, however there were two important differences between these two sets of rules. First, due to the restriction of hypotheses space with arguments, about half of the rules were different. While inspecting the rules that were the same in CN2's and ABCN2's set, we noticed that the quality estimates of these rules were different. For example, the rule:

IF trombocites<100 AND mobility=no THEN death=yes

was present in both rule sets. It covers 6 examples with class value *death=yes* and 1 with *death=no*, which means that the relative frequency of *death=yes* is 6/7 = 0.86. However, extreme value correction of relative frequency estimated the probability of this class (given that the conditions are true) as 0.47, which is much less than 0.86. This happens because there is a high probability that such a rule would be found by chance. On the other hand, when learning with ABCN2, the evaluation of the same rule is 0.67. In CN2, this rule was obtained by searching the whole space unguided by expert knowledge while in ABCN2 the rule was built from the argument *'death=yes BECAUSE trombocites<100'*. The search space in ABCN2 is smaller (only one condition was added), which means that the probability of finding such a rule by chance is lower. So, the expected quality of the rule is higher.

We also asked our expert (Jerneja Videčnik) to examine the rules and compare them. Unfortunately, she could not decide which rules are more understandable to her. We believe that this occurs due to the large number of arguments with only one reason given to each example, while our restriction is that the rule must be consistent with at least one positive argument. The rule must, therefore, contain only one of the given reasons and can neglect the others.

9.4 South Africa heart-disease domain

South Africa heart-disease domain (*SAHeart*) [RdPB$^+$83] contains a sample of 462 cases (male) in heart-disease high-risk region of the Western Cape, South Africa. Learning examples are described with the following 10 variables: systolic blood pressure, cumulative tobacco, cholesterol, adiposity, family history, type-A behavior, obesity, alcohol, age, and coronary heart disease (chd) as the class attribute. The task is to explain which attributes cause coronary heart disease.

In this experiment we used our own vague knowledge, obtained in schools, from newspapers or internet sources, about this domain for explaining some of the examples. So the main difference between this experiment and previous three is that there is no real domain expert involved and the knowledge about the domain is superficial (might be also wrong). This experiment can be regarded also as a test of ABCN2 for handling wrong arguments. However, as we will see, despite our simple, imperfect domain knowledge, arguments still had a minor positive impact on the results.

We began the experiment by testing some standard machine learning algorithms: CN2 [CB91], CN2e (CN2 extended with PILAR classification, see Chapter 7 and extreme value correction, see Chapter 6), logistic regression [HL00], naive Bayesian classification [KK07], SVM [Vap95; CL01], and C4.5 [Qui93]. The methods learned on the learn set (60 % stratified sample) and tested on the other 40% of the complete sample. The Table 9.3 shows the results. Logistic regression outperforms all other methods, at least according to the probabilistic measures, while CN2e, NBC, and SVM are closely following. The terrible performance of CN2 can be attributed to almost nonexistent overfitting prevention.

Afterwards, we tried our technique to see if we can improve the accuracy of ABCN2 (CN2e+arguments) and argumented the first 10 problematic examples. Some

Table 9.3: Accuracy of some standard machine learning methods on SA Heart data set. Accuracy is measured with classification accuracy, AUC, and brier score.

Method	CA	AUC	Brier score
CN2	0.65	0.66	0.57
CN2e	0.71	0.77	0.37
LogReg	0.72	0.80	0.34
NBC	0.73	0.77	0.38
SVM	0.72	0.77	0.38
C4.5	0.73	0.72	0.40

examples of arguments were:

1. One patient was 55 years old smoker. We deemed these two attributes as important, therefore our argument was: chd=yes because age is high (age>) and tobacco>0.

2. Another patient had high low density lipoprotein cholesterol (the bad one), was aged 52, and the disease was known to happen in his family. High bad cholesterol, age, and genetics are known to be one of the prevailing factors for this disease. Therefore, the argument appended to this patient was: chd=yes because age>45 and family_history=present and cholesterol> 5.

3. Then, there was a also patient that did not have heart disease, but ABCN2 classified it wrong. It was a young patient (age 32) without any family history of this disease. The argument was: chd=no because age<40 and family_history=absent.

4. Another argument was: chd=yes because spb>200 (systolic blood pressure is extremely high, higher than 200).

With altogether 10 arguments the classification accuracy of ABCN2 increased from 71% to 75%, which is a nice improvement. The probabilistic measures, however, did not change much; AUC stayed the same, while Brier score decreased from 0.37 to 0.36 (which is good). We are strongly aware that our arguments might not be always correct, and that a professional cardiologist could construct much better arguments for problematic examples. However, the idea of this experiment was to show how such, imperfect arguments can still improve the quality of induced hypothesis.

At this point of the experiment, we wondered if it would be possible to combine high classification accuracy of ABCN2 with good probability prediction of logistic regression. We tried to use the ability of PILAR's classification (see chapter 7) to improve other methods using rules, in this case the logistic regression. Initially, we combined logistic regression with CN2e (without arguments) and got exactly the results of logistic regression - rules did not contribute at all. Alternatively, we tried to use arguments to improve logistic regression with PILAR. Starting with argumenting first 10 problematic cases did not practically change the accuracy of logistic regression. Even, after 20 arguments, the accuracy was almost still unchanged: classification accuracy increased to 0.73, AUC increases to 0.81, while the Brier score increased to 0.35. We were unable to improve logistic regression without knowledge of the domain.

However, the last part of this experiment is not necessarily only a negative result. It should be noted that, while the accuracy of the hypothesis stayed the same, its structure changed a lot to reflect the arguments that were provided to examples. This could be useful especially in medicine, where experts often prefer models that correspond to their knowledge, even if they are less accurate. For example, in one of our past applications (not related to argument based machine learning) [ADZ$^+$07], the main medical expert Noriaki Aoki clearly preferred the model induced by naive Bayesian classifier to the one induced by logistic regression, although logistic regression was significantly better, since he could better understand the former one.

Chapter 10

Arguments Implying New Attributes

An argument in rule learning was defined as a conclusion and a conjunction of reasons, where each reason mentions some property of the example. Initially, the property of the example corresponded to its attribute value, however in some of our experiments this interpretation was not sufficient. It can happen in two possible cases:

- A reason is a combination of several attributes, where combination is not a conjunction of these attributes. In these cases, we will construct a new attribute from the already existent and add it to the domain.

- A reason can mention a property that is not yet described with the current attribute set. In such cases, we need to add a new attribute to the domain and obtain values of the new attribute for all learning examples.

10.1 Japanese credit screening database

Japanese Credit Screening Database contains 125 persons applying for credit described with 10 attributes. The class attribute is whether a person did get the credit or not. This domain definition also contains imperfect prior knowledge (accuracy 83% on the examples) - it was generated by talking to individuals at a Japanese company that grants credit. This prior knowledge will be used in our experiment as a substitute for the domain expert. We will assume that this "expert" cannot give a complete definition of the target concept, but can only give arguments for certain examples. Again, data was split to a learn set (70%) and test set (30%).

Figure 10.1: Rules induced by CN2 on Credit Screening Database.

IF problem_region=yes AND monthly_payment$<=9$ THEN credit=no
IF jobless=yes AND money_bank$<=50$ AND monthly_payment>2.0 THEN credit=no
IF item=bike AND money_bank$<=10$ THEN credit=no

Figure 10.2: Rules induced by ABCN2. Five examples were argumented in the learning set.

IF problem_region=yes AND years_work$<=10$ THEN credit=no
IF jobless=yes AND sex=male THEN credit=no
IF sex=female AND jobless=yes AND enough_money=no THEN credit=no
IF age$>$59 AND years_working$<$3 THEN credit=no
IF jobless=yes AND sex=female AND married=no THEN credit=no
IF enough_money=no AND age$<=$19 THEN credit_approved=no
IF item=bike AND sex=female THEN credit=no

CN2 induced three rules shown in Figure 10.1 for not receiving credit that achieved 84% accuracy on the learning set and 76% accuracy on the test set. We proceeded with our standard procedure of finding problematic examples and getting arguments from our "expert". Figure 10.2 shows rules obtained after 5 iterations, when remaining problematic examples could not be argumented any more. In one of the arguments, a reason specified unability to pay the credit from the current assets. The argument was:

Credit=no because *money_bank* $<$ *monthly_payment* $\times$ *num_of_payments* ...

The first reason in the argument contains multiplication and a comparison between several attribute value, both types of conditions that can not be captured by a rule induced with CN2. We had to add a new attribute *enough_money* that equals *no* if the above condition is satisfied and *yes* otherwise. The accuracy of these rules on learning set is 85% and is comparable to one achieved with CN2 without arguments. However, on the test set ABCN2 achieved 89% accuracy, which is a significant improvement over CN2's 76% achieved without arguments.

This experiment indicated several points of interest regarding the reconstruction of expert's prior knowledge. Six of the seven final induced rules correspond precisely to complete background knowledge given by experts. This was achieved by asking our "expert" to explain only five examples, which indicates the effectiveness of the ABML refinement loop. This also indicates how effective ABML is as a tool for extracting expert's informal background knowledge. Imagine that we did not have experts' prior knowledge already formalized, and that we wanted to extract it from the expert. The way to do this with interactive use of ABCN2, is to generate questions for the expert by identifying critical examples. Expert's explanations in terms of arguments of the five critical cases would, in our example, be sufficient to completely formalize the expert's prior intuitions.

This experiment also nicely illustrates the difference between induced rules resulting from data only, and actual causal rules that generated the data. Both hypotheses, with and without arguments, have a similar accuracy on the learning set, but to a domain expert the first set of rules would be difficult to understand as they show unfamiliar dependencies. Moreover, the first set of rules scored significantly lower accuracy on the test set, meaning that the first hypothesis merely reflected a spurious relation in the learning set.

10.2 ZEUS credit assignment problem

A similar learning problem to the one above - learning about credit status - occurred within European the 6th framework project ASPIC[*]. Learning about credit status is a part of a larger business-to-business scenario used in ASPIC as the main large-scale demonstration application for argument-based methods. In that experiment we also showed how ABCN2 can be used to improve existing knowledge bases in argumentation based expert systems.

The main problem of the scenario is to determine whether client should be granted credit for a purchase. This is determined by the system through a set of if-then business rules. A part of them is used to determine whether the applicant credit status is "good", "bad", "average", or "unavailable". In the initial knowledge base (given by experts), the following three rules were used for classification:

Credit History = Good If all past debts were paid before or on the date due.

Credit History = Average If equal or less number of past debts were paid in less than a week late, to the ones paid on time.

[*]Argument Service Platform with Integrated Components (ASPIC), http://www.argumentation.org/

Credit History = Bad If more debts paid after the deadline than the ones paid on time.

Along to classification rules, ZEUS also provided a data set of 5000 companies described with 18 attributes (four of them actually relevant for credit history). We began the experiment with the induction of rules from 2500 examples (learning set) without considering given classification rules. The system learned a set of 40 rules. As there was a lot of data available, the method was able to correctly classify 95% of examples in test set (the remaining 2500 examples) which is a quite good result with respect to classification accuracy.

In the next step we ran the algorithm for finding problematic example. It was a company that had "bad" credit status as a class value. We looked up the values of this example and noticed that third rule from experts apply. This company did historically pay more often late than on time (attributes *past debts after date, past debts on time, past debts before time*). The argument for bad is thus:

$$\textit{Credit History} = \textit{Bad} \text{ because } \textit{past\_debts\_after\_date} > \\ > \textit{past\_debts\_on\_time} + \textit{past\_debts\_before\_time}$$

The problem with given argument that it is not consistent with the format of if-then rules learned by CN2. The condition part of a rule is a conjunction of several attribute-value pairs, whereas our condition contains comparison of three attribute values and also a sum of two of them. We solved this problem by constructing another boolean attribute that has value 1 when the above condition is true and 0 when it is not. We named it "latePayer" and the argument thus changed to:

$\textit{Credit History} = \textit{Bad}$ because $\textit{latePayer} = 1$

With the additional argument and the same 2500 learning examples, ABCN2 learned only 27 rules. Due to the argument and the new attribute the number of rules for classes "bad" and "average" dropped significantly. At the same time classification accuracy measured on test data was slightly improved to 97%.

In the next pass we repeated the search for problematic example, this time it was from class "Credit history"="good". The argument from initial knowledge base was therefore:

$\textit{Credit History} = \textit{Good}$ because past $\textit{debts\_after\_date} < 1$

We relearned rules from learn set and the two given arguments and ABCN2 induced only six rules, while classification accuracy of these rules on test set increased to 99.9%. This is a significant improvement in terms of classification accuracy, but even more spectacular is the improvement in terms of the complexity of the induced theories, from the initial 40 rules to 6 only. The improvement after the first argument is expectable, as CN2 is incapable to learning a rule of that format. However, the improvement after the second argument is surprising, as there should be no reason why a rule learner could not find this condition automatically. The problem is the myopical property of rule learning evaluation functions. These add one condition at a time, however the *debts_after_date* < 1 by itself does not increase the probability of class *good*. The condition becomes good only after adding another condition.

10.3 Construction of sophisticated chess concepts

In this case study, we demonstrate the use of argument based machine learning for knowledge acquisition of a sophisticated chess concept [MGK+08]. Knowledge acquisition is still one of the most difficult tasks of artificial intelligence [Fei84]. The problem was addressed in various ways [Boo89; Chk03; Coo94], proposing assorted cognitive techniques like interviews, observations, analogy, etc. to elicit as much knowledge from experts as possible. Nevertheless, the problem still remains largely unsolved [Fei03]. Machine learning has long ago been proposed as an alternative way of addressing this problem [FR86; For86]. While it was shown that it can be successful in building knowledge bases [LS95], the major problem with this approach is that automatically induced models rarely conform to the way an expert wants the knowledge organised and expressed. Models that are incomprehensible have less chance to be trusted by experts and users alike. In striving for better accuracy, modern trends in machine learning (e.g. support vector machines) do not seem to be doing anything to alleviate this problem. A common view is that a combination of a domain expert and machine learning would yield the best results [WWZ99]. And this is where ABML comes into place.

For the particular chess concept, we considered the elicitation of the well-known chess concept of bad bishop. This concept is used in a chess tutoring application developed by Sadikov et al. [SMG+06]. The idea is to improve the chess playing programs with an understanding of static positions. While todays chess playing programs are extremely good at playing chess, their use for commenting or tutoring is rather limited. Such programs evaluate chess positions numerically, but are then not able to explain a numerical evaluation in terms of concepts that human chess players use when they reason about the position. For example, the program may say that the current positions value is 1.70 in favor of White. Now the beginner chess player would ask "Why"? An answer, which is beyond today's chess programs, might be: "Because Black has a bad bishop."

10.3.1 Experiment

Watson [Wat99] gives the following definition of a bad bishop as traditional: a bishop that is on the same colour of squares as its own pawns is bad, since its mobility is restricted by its own pawns and it does not defend the squares in front of these pawns. Moreover, he puts forward that centralisation of these pawns is the main factor in deciding whether the bishop is bad or not.

In the experiments, the dataset for learning consisted of 200 middlegame positions from real chess games where the black player has only one bishop[†]. These bishops were then a subject of evaluation by the experts[‡]. In 78 cases, the bishops were assessed as bad. Each position had also been statically evaluated (i.e. without applying any search) by the evaluation function of the well-known open source chess program CRAFTY, and its positional feature values[§] served as attribute values for learning. We randomly selected 100 positions for learning and 100 for testing (stratification was used, preserving the proportion of positive and negative examples).

In the first iteration of the previously mentioned process, only CRAFTY's positional features were used and no arguments have been given yet. ABCN2 induced all together 4 rules achieving 72% classification accuracy on the test set. Figure 10.3 shows the first critical example, automatically selected by our algorithm.

The initial rules failed to classify this example as "not bad", as was previously judged by the experts. The following question was given to the experts: "Why is the black bishop not bad?" It turned out that the concept mentioned by the experts (see the caption in Figure 10.3) was not yet present in the domain attributes - the only CRAFTY's positional feature that could potentially describe bishop's mobility, BLACK_BISHOPS_MOBILITY, expresses the number of squares that the bishop at-

[†]The learning data set and a detailed explanation of domain's attributes can be found at: *http://www.ailab.si/matej/*.

[‡]The chess expertise was provided by woman grandmaster Jana Krivec and FIDE master Matej Guid.

[§]CRAFTY's evaluation function uses about 100 positional features.

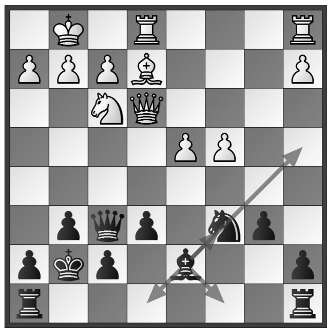


Figure 10.3: *Why is the black bishop not bad?* The experts used their domain knowledge to produce the following answer: "The black bishop is not bad, since its mobility is not seriously restricted by the pawns of both players."

tacks, but doing so takes into account all pieces (not only pawns) that block the bishop's diagonals, restricting its mobility. A new attribute, IMPROVED_BISHOP_MOBILITY, was therefore programmed and included into the domain. It is the number of squares accessible to the bishop, taking into account only own and opponent's pawn structure. Based on the experts' explanation, the argument "IMPROVED_BISHOP_MOBILITY is high" was added to this example.

Taking only the bishop's mobility into account turned out not to be enough for ABCN2 to determine the goodness of the bishop. Also, the method, which at the time only had CRAFTY's attributes and the newly included attribute at its disposal, failed to find additional restrictions to improve the experts' argument. To solve this problem, we used the idea of counter examples presented in Chapter 8. The method found a counter example shown in Figure 10.4. This example is classified as "bad", although the value of the attribute IMPROVED_BISHOP_MOBILITY is high.

The experts were now asked to *compare* the black bishops in the two examples: "Why is the black bishop in Figure 10.4 bad, and the bishop in Figure 10.3 is not?" Again, the experts have been asked to give a description based on their knowledge in the presented domain. Based on this description (given in Figure 10.4), another attribute, BAD_PAWNS, was included into the domain. This attribute evaluates pawns that are on the colour of the square of the bishop ("bad" pawns in this sense). With some help of the experts, a look-up table with predefined values for the pawns that are on the same colour of squares as the bishop was designed in order to assign weights to such pawns. According to the previously mentioned Watson's definition, centralisation of the pawns was taken into account. The argument given to the example shown in Figure 10.3 was then extended to "IMPROVED_BISHOP_MOBILITY is

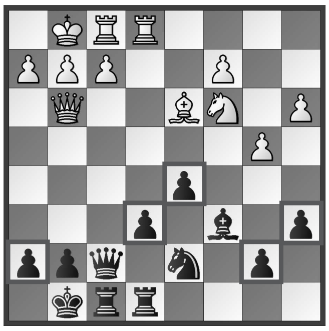


Figure 10.4: *Why is the black bishop bad, comparing to the one in Figure 10.3?* The experts' explanation was: "The important difference between the two examples is the following: in the example in Figure 10.4 there are more pawns on the same colour of squares as the black bishop, and some of these pawns occupy the central squares, which further restricts the bishop's possibilities for taking an active part in the game."

high AND BAD_PAWNS is low," and with this argument the method could not find any counter examples any more. The new rule covering the critical example is:

if IMPROVED_BISHOP_MOBILITY$\geq$4 and BAD_PAWNS$\leq$32
then BISHOP=NOT_BAD; *class distribution [0,39]*

The above rule evidently uses given argument in its condition. The method operationalised the first condition of the argument as IMPROVED_BISHOP_MOBILITY$\geq$4 ($\geq$4 stands for high here), while in the second it decided that the value of 32 is critical for attribute BAD_PAWNS to distinguish a bad and a not bad bishop. The rule covers 39 learning examples (out of 100) and all of them are from class NOT_BAD, which suggests that the rule is good indeed.

The arguments can consist of both newly included attributes and/or existing ones. During the process, after they were given another critical example selected by the method, the experts expressed the following commentary: "The bishop is not bad, since the pawns that are on the same square colour are not sufficiently blocked by opponent's pawns and pieces." Their domain knowledge was again translated into domain description language - attribute BLOCKED_BAD_PAWNS was added to the domain. As in the previous example, the method selected the position shown in Figure 10.4 as the most appropriate counter example. The "bad" black pawns in this position are also not blocked by opponent's pawns and pieces, but the bishop is regarded as bad anyway. The experts' explanation of the crucial difference between the two examples was the same as above in this case. The existing attribute BAD_PAWNS was therefore used to improve the argument to “BLOCKED_BAD_PAWNS is low AND BAD_PAWNS is low”. The method was in this case able to induce the rest of the rule:

if BLOCKED_BAD_PAWNS$\leq$3
and BAD_PAWNS$\leq$26
and IMPROVED_BISHOP_MOBILITY$>$1
then BISHOP=NOT_BAD; *class distribution [0,19]*

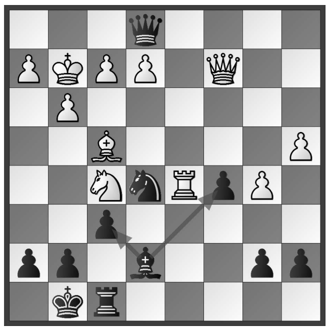


Figure 10.5: *Why is the black bishop bad?* The following commentary was given: “The black bishop is bad, since both of its diagonals are blocked by its own pawns.”

The ABML-based knowledge-elicitation process was used to induce rules to determine both good (i.e. not bad) and bad bishops. The automatically selected critical example shown in Figure 10.5 represents an example with other class value than the previous ones. The experts were in this case asked to describe why the black bishop is *bad*. Based on their answer (see Figure 10.5), another attribute was introduced into the domain: BLACK_PAWN_BLOCKS_BISHOP_DIAGONAL, which takes into account own pawns that block the bishop’s diagonals. The argument “BLACK_PAWN_BLOCKS_BISHOP_DIAGONAL is high” was added to the example, however a counter example presented in Figure 10.6 was found by the method and was shown to the experts. The question was: “Why is the bishop in Figure 10.6 not bad, and the bishop in Figure 10.5 bad?”

In this case, the experts were unable to express the crucial differences between the selected examples regarding the goodness of the bishop in a way that would enable to translate her description into domain description language. The description (see Figure 10.6), although completely relevant in the given position, is practically impossible to convert into appropriate attributes, since it would require several very

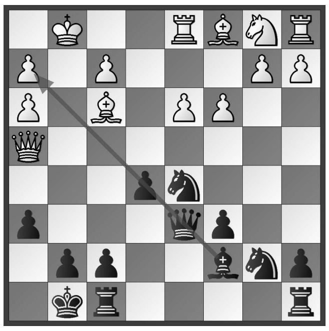


Figure 10.6: *Why is the bishop not bad, comparing to the bishop in Figure 10.5?* The experts: “The black bishop is not bad, since together with the black queen it represents potentially dangerous attacking force that might create serious threats against the opponent’s king.”

sophisticated attributes to describe the dynamic factors expressed in the experts’ commentary. In such a case (i.e. when the expert is unable to provide an argument that could be translated into domain description language), the ABML method searches for another counter example (if available). In this case, the example in Figure 10.7 was given to the experts as a counter example to the one in Figure 10.5.

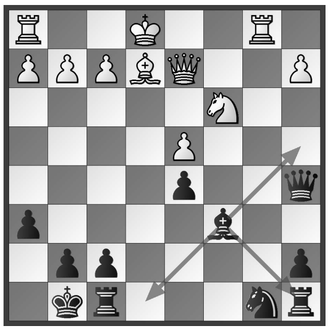


Figure 10.7: *Why is the bishop not bad, comparing to the bishop in Figure 10.5?* The experts described the difference: “The black bishop is not bad, since its mobility is not seriously restricted, taking the pawn structure into account.”

Based on the experts’ commentary (see Figure 10.7), the existing attribute IMPROVED_BISHOP_MOBILITY was used to improve the argument to:

“BLACK_PAWN_BLOCKS_BISHOP_DIAGONAL is high AND IMPROVED_BISHOP_MOBILITY is low.

The following rule, explaining this critical example, can be found in the new set of induced rules:

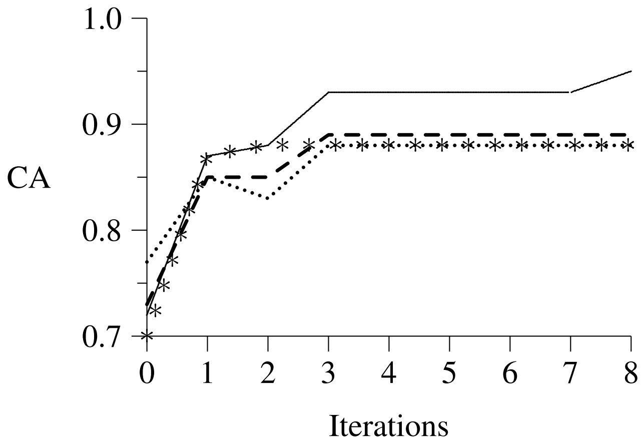


Figure 10.8: Progress of classification accuracies (CA) through iterations for ABCN2 (solid line), logistic regression (stars $*$), C4.5 (dashed line) and classic CN2 (dots).

if BLACK_PAWN_BLOCKS_BISHOP_DIAGONAL$\geq$20
and IMPROVED_BISHOP_MOBILITY$\leq$3
then BISHOP=BAD; *class distribution [18,0]*

In total, there were eight critical examples presented to the experts The final model scored 95% accuracy on the test set.

10.3.2 Discussion

The ABML-based knowledge-elicitation process presented in our case study consisted of eight (8) iterations. During the process, seven (7) arguments were attached to automatically selected critical examples, and five (5) new attributes were suggested by arguments and therefore included into the domain. This experiment also demonstrated the use of counter-examples; in some of the iterations, initially given arguments were not enough for ABCN2 to induce a good rule, therefore experts had to revise their arguments using counter-examples. After each iteration, the obtained rules were evaluated on the test dataset. The improvement of the model is evident: from the initial 72% classification accuracy (Brier score 0.39, AUC 0.80), the final 95% accuracy (Brier score 0.11, AUC 0.97) was achieved after the end of the process.

The question is, whether these improvements were mainly due to the addition of new attributes or were the arguments also just important? The Figure 10.8 shows that the arguments also mattered significantly. We compared the progressions of classification accuracies of ABCN2 with some other ("non-ABML" - using only newly added attributes) machine learning algorithms, namely logistic regression, decision trees (C4.5), and the classic CN2. The accuracies of all methods improved during the process, however ABCN2 (which also used the arguments given by the experts) outperformed all the others. The obtained results suggest that the performance of other algorithms could also be improved by adding appropriate new attributes. However, using arguments is likely to lead to even more accurate models.

The main advantage of ABML over classical machine learning is the ability to take advantage of expert's prior knowledge in the induction procedure. This leads to hypotheses comprehensible to experts, as it explains learning examples using the same arguments as the expert did. In our case study this was confirmed by chess experts. According to them, the final set of rules are more alike to their understanding of the bad bishop concept than the initial rules were. Furthermore, the final rules were also recognised to be in accordance with the traditional definition of a bad bishop.

Our domain experts clearly preferred the ABML approach to manual knowledge acquisition. The formalisation of the concept of bad bishop turned out to be beyond the practical ability of our chess experts (a master and a woman grandmaster). After trying to manual define this concept, their rules achieved 56% accuracy on the test set. They described the process as time consuming and hard, mainly because it is difficult to consider all relevant elements. ABML facilitates knowledge acquisition by fighting these problems directly. Experts do not need to consider all possibly relevant elements, but only elements relevant for a specific case, which is much easier. Moreover, by selecting only critical examples, the time of experts involvement is decreased, making the whole process much less time consuming.

Chapter 11

Automatically Extracted Arguments from Text

In ABML applications, arguments are usually provided by domain experts. In this chapter, we will demonstrate a possible way of extracting arguments from text and using them in ABML. The work was done as a part of the X-Media* project, which is an European project that addresses the issue of cross-media knowledge management in complex distributed environments.

11.1 Extracting arguments from text

The extraction of arguments from text is based on relation extraction from text. Relation extraction is an important task in natural language processing, with many practical applications such as question answering, ontology population, and information retrieval. It requires the analysis of textual documents, with the aim of recognizing particular types of relations between named entities, nominals, and pronouns. Reliably extracting relations in natural-language documents is still a laborious and unsolved problem. Traditionally, relation extraction systems have been trained to recognize relations between names of people, organizations, locations, and proteins. In the last two decades, several evaluation campaigns such as MUC [MUC], ACE [ACE], SemEval [sem] have helped to understand and properly formalize the problem, and provide comparative benchmarks.

We are interested in finding semantic relations between class values and descriptive attributes (taken from data), and using them as arguments. For example, taking

*http://www.x-media-project.org/

the ZOO domain from Chapter 8, given the class value *reptile* and the attribute *eggs* we are interested in relations such as “Most reptiles lay eggs” and “Reptiles hatch eggs.” Specifically, the relationships that exist between classes and attributes are extracted from the whole English Wikipedia,[†] an online encyclopedia written collaboratively by volunteers, that has grown to become one of the largest online repositories of encyclopedic knowledge, with millions of articles available for a large number of languages.

To extract such relations from textual documents, we have to deal with two major problems. The first concerns the lack of information on the relation type we are seeking. In relation extraction, we usually know in advance the type of the relations to be extracted, here we only know class values and attributes, namely the conclusion and reasons of a possible relation. Thus, the task is restricted to discover whether or not a relation exists between class and an attribute.

The second problem is related with the lexicalization of the class values and attribute descriptions. The names of attributes and classes should be meaningful, or, in other words, should be similar to those used in texts. Using their background knowledge, humans can naturally interpret the concepts expressed by class values and attributes, however, due to the variability of natural language, it can be very difficult to find occurrences of the lexicalizations of such concepts in the same sentence and, consequently, to determine whether or not a relation exists. To address the first problem, we do not try to find specific assertions of relations in text, but rather we exploit the simple idea that if many different sentences reference both the class value and attribute, then the class value and attribute are likely to be related. On the other hand, to deal with the variability of natural language, we generated alternative lexical variants using a WordNet [Fel98], a lexical database containing semantic relations among words. Specifically, we generated variants for all class values and attributes using the following semantic relations in WordNet: synonyms (e.g., breathe $\rightarrow$ respire) and morphological derivations (e.g., predator $\rightarrow$ predators).

As most relation extraction systems [HSG04; BM05; GLR07], we identify relations mentioned in text documents considering only those pairs that are mentioned in the same sentence. Let $c_1, \ldots, c_k$ be class values in data and $a_1, \ldots, a_n$ attributes. Then, the number of relations $\#r(c_i, a_j)$ is defined as the number of sentences across the whole English Wikipedia, where the class c_i and the attribute a_j co-occur.

We shall now define the construction of an argument given the number of relations

[†]`http://en.wikipedia.org`

between class and attribute values. An argument is a conjunction of a set of reasons, where each reason is related to a single attribute in the domain. To determine whether and attribute a_j is a possible reason for class c_i, we first evaluate whether $\#r(c_i, a_j)$ is statistically different from the expected value $E(\#r(c_i, a_j))$, namely if the number could be obtained purely by chance. A possible method for this task is the standard χ^2 test for 2×2 matrices.

When $\#r(c_i, a_j)$ is statistically different from $E(\#r(c_i, a_j))$, it can be either higher or lower. If $\#r(c_i, a_j) > E(\#r(c_i, a_j))$, then we say that a_j is a positive reason for c_i. Such a positive argument can be given to an example if it is from class c_i and the value of a_j is "positive". The positiveness of attribute values must be defined prior to learning and it intends to distinguish between values that should occur more frequently in the class-attribute relations in text than it is expected. Although, it is impossible to say which of the values will have this property, we believe that a good heuristics to select positive attribute values is to select those that ascribe the presence of a property described by a_j to the example. For instance, if an animal has the value of attribute *breathes* 1, this value states that the animal is breathing (presence of this property), and the value of attribute is positive. If the number of found relations is less than expected, i.e. $\#r(c_i, a_j) < E(\#r(c_i, a_j))$, then we can use such reason only if the example has negative value of a_j.

An argument for a certain example is thus constructed from all positive and negative reasons consistent with the values of this example. Sometimes, such an argument will be overly specific. To alleviate this problem all arguments are pruned with REP (reduced error pruning) [Fur97] principle before they are appended to the example.

11.2 Case study: animal classification

The approach will be illustrated and evaluated on the domain ZOO, the same one as used in the first section of Chapter 8. We began the experiment by learning rules with ABCN2 without any arguments extracted from text. Learning and testing on the full data set accounted for 100% classification accuracy. Induced rules were:

- IF milk=yes THEN type=mammal
- IF feathers=yes THEN type=bird
- IF eggs=yes AND fins=yes THEN type=fish
- IF aquatic=no AND legs=6 THEN type=insect

- IF feathers=no AND eggs=yes AND backbone=yes AND aquatic=no THEN type=reptile
- IF milk=no AND domestic=no AND hair=no AND tail=yes AND fins=no AND legs=0 THEN type=reptile
- IF toothed=yes AND legs=4 AND eggs=yes AND aquatic=yes THEN type=amphibian
- IF feathers=no AND hair=no AND airborne=no AND backbone=no AND predator=yes THEN type=other
- IF fins=no AND backbone=no AND legs=no THEN type=other

Afterwards, we sought through Wikipedia for relations between class values (e.g., *mammal*) and positive attribute values (e.g., *milk=yes*). Table 11.1 shows the alternative lexical variants for some attributes generated using WordNet and morphological derivations. In this search we omitted to use class *other*, since it does not represent any actual animal class. Table 11.2 shows number of all relations found for the ZOO domain. For example, we found a strong correlation between the class bird and the attribute *feather*, merely expanding the attribute with the lexical variants *feathers* and *plumage*. On the other hand, despite the fact that it is intuitive for humans to answer the question if a reptile has approximately the same size of a cat, it is almost impossible to find occurrences of the class reptile and the attribute "catsize" in the same text, due to the erroneous lexicalization of this attribute introduced for comparing animals by size. The last row of Table 11.2 show that the attribute catsize gets scores of zero for all classes.

The absolute values $\#r(c_i, a_j)$ are not strongly related to the correlation between class c_i and attribute a_j. For instance, it seems that *aquatic* is the most important feature of *amphibians*. But, is it really, as being aquatic is common for all classes? On the other hand, the text extraction tool found only 6 relations between *amphibians* and *breathing*. However, there is still a strong positive relation between them, due to a much lesser presence of the concept breathing and animal type amphibian in text when compared to other attributes and animals.

For this reason, we applied the standard χ^2 ($sig = 0.05$) test to determine whether $\#r(c_i, a_j)$ is statistically different from $E(\#r(c_i, a_j))$ (any appropriate statistical test could be applied here). Table 11.3 shows results of χ^2 test, where:

value 0 means that the relation is not significant;

Table 11.1: The alternative lexical variants for attributes generated using WordNet and morphological derivations.

attribute	alternative lexical variants
hair	hairs, fur, furs
feather	feathers, plumage
egg	eggs, spawn, spawns
milk	milking
airborne	winged
...	
domestic	domesticated, pet
catsize	-

Table 11.2: Number of positive relations $\#r(c_i, a_j)$ between animal classes and attributes found in Wikipedia.

	amphibian	reptile	insect	mammal	bird	fish
hair	1	16	70	187	106	87
feathers	0	14	17	15	894	31
eggs	34	117	339	174	894	645
milk	2	2	10	67	25	120
airborne	2	15	117	15	196	21
aquatic	81	184	271	1008	284	1072
predator	5	16	123	95	285	217
toothed	10	59	54	150	102	250
backbone	0	1	3	11	7	51
breathes	6	3	8	11	16	45
venomous	6	43	39	44	41	47
fins	4	4	3	17	21	529
legs	27	42	206	51	364	111
tail	16	26	53	57	504	279
domestic	9	40	31	81	317	166
catsize	0	0	0	0	0	0

value 1 denotes positive reasons, and

value -1 denotes negative reasons.

Table 11.3 shows which of the attributes can be used as arguments for every class. For example, in the case of amphibians, we can use attributes *aquatic*, *breathes*, and *legs* as reasons in the argument, if the values of the corresponding case are positive ("yes" or > 0 for legs). Similarly, attributes *hair*, *feathers*, *predator* can be used as reasons when attribute values are negative ("no" or $= 0$ for legs).

Table 11.3: Positive (1) and Negative (-1) reasons. Relations with value 0 are not significantly different from the expected value and cannot be used as reasons in arguments.

	amphibian	reptile	insect	mammal	bird	fish
hair	-1	0	1	1	-1	-1
feathers	-1	-1	-1	-1	1	-1
eggs	0	0	1	-1	1	-1
milk	0	-1	-1	1	-1	1
airborne	0	0	1	-1	1	-1
aquatic	1	1	-1	1	-1	1
predator	-1	-1	1	-1	1	0
toothed	0	1	-1	1	-1	1
backbone	0	0	0	0	-1	1
breathes	1	0	0	0	-1	1
venomous	0	1	1	0	-1	-1
fins	0	-1	-1	-1	-1	1
legs	1	0	1	-1	1	-1
tail	0	-1	-1	-1	1	0
domestic	0	0	-1	-1	1	-1
catsize	0	0	0	0	0	0

After augmenting all examples with arguments, the rules for *mammals*, *birds*, *fishes*, *insects*, and *other* stayed the same. The rules for reptiles changed with the use of arguments:

- IF toothed=yes AND milk=no AND fins=no AND legs=0 THEN type=reptile
- IF feathers=no AND milk=no AND fins=no AND breathes=yes AND backbone=yes AND aquatic=no
  THEN type=reptile

The rules are similar to the ones above with some differences. Specifically, the second rule in the original set mentions *domestic=no* as a condition for a reptile, although there are many reptiles used as pets (e.g., turtles, snakes, etc.). It might be easier to understand the how these rules were obtained, if we look at a particular argument given to an animal. All possible reasons for reptiles are: *feathers=no, milk=no, aquatic=yes, predator=no, toothed=yes, venomous=yes, fins=no*, and *tail=no*. When the method gives an argument to a particular example, e.g. pitviper, it uses only reasons that are true for this example. In the case of pitviper, the relevant reasons would thus be: *feathers=no, milk=no, toothed=yes, venomous=yes*, and *fins=no*. The complete argument is:

Pitviper is a *reptile*, because *feathers=no* and *milk=no* and *toothed=yes* and *venomous=yes* and *fins=no*.

Afterwards, to remove the unnecessary conditions, the reduced error pruning mechanism is applied and the resulting argument is:

Pitviper is a *reptile*, because *milk=no* and *toothed=yes* and *fins=no*.

The first rule for reptiles used this arguments as the basis and added a condition *legs=0* to finalise the induction of the rule.

The rule for amphibians changed to:

- IF legs=4 AND breathes=yes AND aquatic=yes AND hair=no THEN type=amphibian.

The original and the new rule are again very alike. In the latter attributes *toothed* and *eggs* were replaced with *breathes* and *hair*. From a point of an expert, the second rule is better, since it is not entirely true that *amphibians* do not have *teeth*. Most amphibian larvae have tiny teeth. Nevertheless, although most adult amphibians retain their teeth, teeth can be reduced in size or not present at all.

We evaluated the method with 10-times repeated 10-fold cross-validation to avoid effects of randomness of one split only. In each iteration, all examples in the learn set were argumented, then a model was built from these and evaluated on the test set. Using ABCN2 without arguments resulted in 94.51% classification accuracy, while ABCN2 with arguments scored, on average, 96.75% classification accuracy. As a comparison, some standard machine learning methods (as implemented in Orange [DZ04]) scored 90% (SVM), 92.57% (C4.5) and 92.6% (naïve Bayes). The scores differ from those in Chapter 8, since another type of evaluation was used.

Chapter 12

Can Imperfect Arguments be Damaging?

It may well happen that the expert, when argumenting examples, will make mistakes and supply imperfect arguments. We now turn to the question, how critically ABCN2 depends on the quality of arguments?

Theories learned by ABCN2 have to be consistent with the given arguments. In the case of imperfect arguments, this may be a problem, as the resulting theories will reflect these imperfections, and possibly have lower accuracy than learning without arguments at all. In this chapter, we will give an experimental evidence that, in general, this problem is unlikely to cause much damage.

We start with the expectation that accuracy of theories learned from examples and arguments given by experts is higher than accuracy of hypotheses learned from examples and random arguments. Experts, obviously, have better knowledge of the domain than simple guessing, and so the given experts' arguments will be better than random arguments. Consequently, the induced hypotheses will be, on average, more accurate. The effect will be detrimental only if the expert exploits his or her knowledge to maliciously hinder learning by intentionally giving extremely bad, false arguments. In the following we will not consider such extremely unnatural situations, but investigate whether completely uninformative, random arguments will damage the performance in comparison with no arguments at all.

We made an experiment, using 25 UCI data sets [AN07], comparing CN2 with ABCN2 using 2, 5, 10, or 20 *randomly* argumented examples. Each argumented example can have up to five *random* positive arguments, and each argument can have up to five *random* reasons.

Table 12.1: Classification accuracy and AUC of ABCN2 on several UCI data sets with different number of randomly argumented examples.

	CA					AUC				
Dataset↓ #rand.arg.→	0	2	5	10	20	0	2	5	10	20
adult	0.805	0.805	0.806	0.806	0.807	0.880	0.881	0.880	0.880	0.881
australian	0.871	0.868	0.864	0.872	0.859	0.924	0.925	0.925	0.925	0.927
balance	0.833	0.833	0.832	0.830	0.830	0.820	0.823	0.816	0.820	0.819
breast (lju)	0.720	0.720	0.717	0.717	0.724	0.717	0.719	0.723	0.720	0.702
breast (wsc)	0.940	0.941	0.940	0.941	0.942	0.987	0.989	0.989	0.989	0.990
car	0.771	0.767	0.774	0.778	0.773	0.916	0.915	0.922	0.924	0.921
credit	0.858	0.858	0.861	0.864	0.861	0.910	0.911	0.916	0.911	0.917
german	0.708	0.709	0.708	0.710	0.710	0.749	0.746	0.750	0.755	0.755
hayes-roth	0.832	0.824	0.825	0.810	0.817	0.959	0.959	0.944	0.958	0.949
hepatitis	0.814	0.820	0.820	0.801	0.807	0.853	0.830	0.819	0.855	0.810
ionosphere	0.926	0.926	0.929	0.920	0.906	0.954	0.954	0.952	0.954	0.952
iris	0.927	0.927	0.927	0.933	0.927	0.979	0.979	0.979	0.981	0.981
lymphography	0.824	0.824	0.824	0.830	0.844	0.930	0.931	0.928	0.924	0.931
monks-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
monks-2	0.657	0.657	0.657	0.646	0.666	0.740	0.735	0.747	0.699	0.738
monks-3	0.989	0.989	0.989	0.989	0.989	0.991	0.991	0.988	0.990	0.989
mushroom	1.000	0.999	1.000	0.999	1.000	1.000	1.000	1.000	1.000	1.000
pima	0.751	0.751	0.749	0.745	0.749	0.835	0.836	0.835	0.833	0.832
SAHeart	0.673	0.675	0.680	0.677	0.682	0.748	0.746	0.752	0.754	0.753
shuttle	0.937	0.937	0.933	0.941	0.937	0.997	0.997	0.998	0.995	0.996
tic-tac-toe	0.993	0.994	0.993	0.993	0.994	0.998	0.999	0.998	0.998	0.998
titanic	0.787	0.787	0.785	0.786	0.787	0.741	0.741	0.742	0.742	0.742
voting	0.945	0.942	0.945	0.945	0.949	0.983	0.983	0.986	0.979	0.979
wine	0.948	0.959	0.960	0.960	0.938	0.999	0.999	0.999	0.999	0.999
zoo	0.961	0.961	0.961	0.951	0.941	0.998	0.998	0.995	0.998	0.998
Avg. rank	2.88	2.90	3.02	2.90	3.3	2.94	3.12	2.88	3.00	3.06
Wilcoxon (p) [0 vs. x]	NA	0.47	0.45	0.47	0.37	NA	0.31	0.37	0.34	0.48

The results are shown in Table 12.1. There were almost no visible differences in the performance of induced theories between learning from randomly argumented examples and no arguments at all. Also, when using Wilcoxon T-test, neither of the methods (using different number of random arguments) could be proved to be statistically worse than CN2. Finally, comparing their ranks with the Bonferroni-Dunn post-hoc test [Dem06] visualised in Figures 12.1 and 12.2 as well support our hypothesis. Therefore, using this result and the natural assumption that expert's arguments are on average better than random, we conclude that experts' arguments cannot significantly worsen the classification accuracy. ABCN2 will under any normal circumstances either outperform the original CN2, or perform similarly, but it is unlikely to be inferior to learning without arguments.

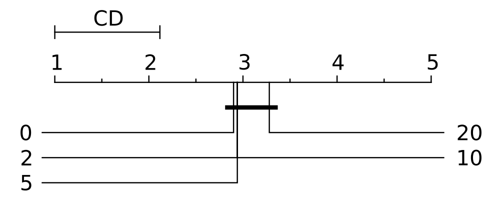


Figure 12.1: A critical difference diagram of classification accuracies of ABCN2 models with different number of random arguments.

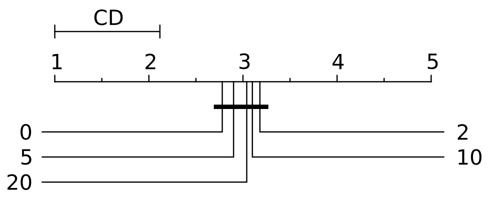


Figure 12.2: Comparisson of AUC's of ABCN2 models with different number of random arguments.

Chapter 13

Concluding Remarks and Further Work

Argument Based Machine Learning is a novel approach to machine learning that draws on some concepts of argumentation theory. Argumentation is used to facilitate efficient knowledge elicitation from experts. They need to focus on a particular learning example only and try to articulate the reasons why is this example in the class as given. Since an argument is not assumed to be completely correct knowledge, it does not matter whether the expert is confident in his or her answer or not. They could even just express their opinion or answer by using intuition about the learning example, and it should still be useful for an ABML method. Furthermore, argumentation can be also used to determine undefeated (admissible) arguments, when the arguments attack each other. This usually happens if the example is explained by a number of experts that disagree with each other. Although we provided formal grounds for such argumentation, we did not encounter such a case during experimentation, and therefore this remains subject of future work.

Since the argument knowledge is not objectively correct and is also specific to the particular example, it needs to be used in a different way than classical prior knowledge. We say that the induced hypothesis should argument-based derive each learning example, which means that the hypothesis should mention arguments of the example in the derivation of the class value of this example. The concept of deriving is, however, different for each learning principle, as demonstrated in Chapter 4 for ILP and logistic regression and in Chapter 5 for rule learning.

In Chapter 5, we described the ABCN2 algortihm, an argument-based extension of CN2, and argumented examples in rule learning. CN2 is a well-known method for learning rules, but very basic. The question is, if we would achieve even better results by using another, more sophisticated method, e.g. RIPPER [Coh95], which uses reduced error pruning to post-prune induced rules? The only definite answer would give an experiment, however, one needs to consider that CN2 as used in this Thesis is not the same CN2 as used a decade ago; the extreme value correction of probability estimates produces probability estimates that are closer to its true values and PILAR classification assigns each rule a weight of its importance - in a sense, it is a way to do what post-pruning does.

The PILAR classification technique described in Chapter 7 and extreme value correction in probability estimates (Chapter 6) are both techniques that substantially improve the quality of rule learning. However, they come at some computational cost. In extreme value correction, the required initial determination of the parameters of the extreme value distribution for the particular learning problem is typically roughly comparable to the rest of rule learning in ABCN2. In PILAR, after rules have been induced, an optimisation to find the best parameters is needed. However, we do not regard the additional time required as a critical problem for ABCN2, since a large portion of the time sink comes from interaction with experts, while time needed for learning is negligible.

The argument based refinement loop iteratively selects critical examples that should be explained by an expert. In this way, experts need to focus on difficult cases only and leave simple cases to the learning algorithm. At the moment, the algorithm always suggests the most misclassified example as the most critical one. Although this criterion is intuitive - ask what you do not know, and shows promising results in experiments, it is probably not optimal. If the domain contains outliers (examples with unusual attribute values for given class), these will be presented to experts for explanation, but experts will unlikely be able to explain them. The question is thus: is there a way to select critical examples that are not outliers?

The experiments showed several advantages of ABML:

- Expressing expert knowledge in the form of arguments for individual examples is easier for the expert than providing general theories.

- Critical examples whose arguments are expected to most improve the learning, are automatically identified by our method.

- When expert's arguments can not be used efficiently, the method provides counter examples. Experts can thus improve their initial argument by taking counter examples into account. We noticed that these examples have also improved the expert's understanding of the learning problem, as they were not aware of the possible counter examples.

- ABCN2 produces more comprehensible rules than CN2, because it uses expert-given arguments to constrain learning, thereby suppressing spurious hypotheses.

- In the experiments with a number of test data sets, ABCN2 achieved higher accuracy (classification accuracy, brier score, AUC) than classical CN2. Although this might not be true for all domains, we can expect that arguments will, in general, improve accuracy of hypotheses. We showed experimentally that, on average, imperfect, or even completely random arguments are unlikely to harm the classification accuracy of ABCN2.

- Accuracy of ABCN2 is in general increased due to two reasons: 1) arguments imply conditions that should be mentioned within rules and 2) arguments imply missing attributes for successful description of a domain.

In Chapter 11, we proposed a technique to automatically extract arguments from Wikipedia and demonstrated it on the animal classification domain. The idea is to eliminate the reliance on an expert who may not be available. However, a single experiment only raises a question whether it is applicable also to other domains. We believe that it is. The most promising are domains, where learning examples have already attached commentaries. In medicine, for instance, doctors usually provide their explanation of laboratory results. Another example of such a domain are technical experiments (e.g., efficiency of jet engines), where experts usually explain obtained results.

According to the results of experiments, we can say that ABCN2 is a successful tool. However, rule learning is only a paradigm appropriate for some of the learning problems, while other problems ought to be solved with different approaches. ABCN2 can be therefore seen as an example of transforming a classical learning technique into its argument-based counterpart. Although, the main principle of ABML is simple - using arguments in explanations of examples, we still had to solve many quite intricate problems.

There are many methods that could be extended to work with argumented examples. The most challenging issues in the contemporary research of machine learning are in solving complex domains like understanding of free text or analysis of graphs and long sequences. We believe that in such problems argument-based approach could be even more beneficial. For example, in the case of understanding a sentence, it is difficult for an expert to provide general guidelines how to interpret sentences, but explaining the meaning of a single sentence is usually not a problem. A similar thing could be said for domains that involve graph mining, like weather prediction, analysis of molecules in chemistry, image analysis, etc. A meteorologist can often explain a particular course of events, but explaining the general theory is far from possible. Since it seems that these complex problems are relational in nature (graphs are relational by definition, the meaning of sentences relies on previous sentences), the most logical next ABML method would be ILP or a variant of ILP. At the moment, these problems are deemed too complex for conventional ILP, and currently less natural solutions are applied; they first involve extraction of patterns and then learn with classical machine learning. We believe that argument-based approach would simplify the search complexity of ILP to make it usable on these domains.

Dodatek A

Razširjeni povzetek v slovenskem jeziku (Extended Abstract in Slovene Language)

Argumentirano strojno učenje

A.1 Uvod

Strojno učenje je veja raziskav umetne inteligence, ki raziskuje različne tehnike za avtomatsko (računalniško) učenje. Najpogosteje uporabljen tip strojnega učenja je induktivno učenje, oz. učenje iz primerov, kjer je vsak primer sestavljen iz množice **atributov** (opisne spremenljivke) in **razreda** (odvisna spremenljivka). Cilj strojnega učenja je odkriti **teorijo**, ki zna iz atributov izračunati vrednost razreda za vse možne primere. Dobljena teorija je zapisana v izbranem formalnem jeziku v obliki modela, ki se lahko uporabi tako za obravnavo novih, še neznanih primerov, kot tudi za boljše razumevanje obravnavanega problema. Nekaj tipičnih problemov učenja iz primerov:

- Iz podatkov o vremenu za preteklo obdobje se nauči napovedovati vreme v prihodnosti.
- Iz podatkov o uspešnosti operacije za nekaj pacientov (*npr.* vstavljanje kolčne proteze) se nauči napovedovati uspešnost operacije pri novih, še ne operiranih, pacientih.
- Iz primerov šahovskih pozicij in strokovnih komentarjev k vsaki od teh pozicij, se nauči tak model, ki omogoča avtomatsko generiranje komentarjev za poljubno šahovsko pozicijo.

V praksi imamo na voljo le nekaj učnih primerov, zato rezultatu strojnega učenja pravimo **hipoteza** in ne teorija, saj lahko preverimo njeno pravilnost samo na učnih podatkih. Hipoteza je **konsistentna** z učnimi podatki, če napove za vse učne primere pravilno vrednost razreda. Učni primeri opisujejo pretekle dogodke, pri katerih je vrednost razreda, ki ga napovedujemo, znana. Atributi običajno opisujejo neke naravne lastnosti primera in pričakujemo, da je iz teh atributov možno izpeljati vrednost razreda za vsak primer. Recimo, v vremenski domeni bi atributi lahko bili: vlažnost, veter, zračni pritisk, itd. Razred bi lahko bil količina padlega dežja na kvadratni meter naslednji dan. Vendar se v realnosti izkaže, da se metode učenja velikokrat ne uspejo naučiti dobrih hipotez zaradi različnih razlogov, npr.: (a) hipotez konsistentnih s podatki je mnogo, metoda vzame napačno, (b) nabor atributov je nepopoln, iz danih atributov ni mogoče izpeljati vrednost razreda, (c) relacija med razredom in atributi je kompleksna, katere metoda ni uspela odkriti ali (d) izbrani formalni jezik za opisovanje hipotez je neprimeren za dani učni problem.

Ena največjih težav (točka a) v strojnem učenju je poiskati "pravo" hipotezo (ali teorijo) med mnogimi v celotnem **prostoru hipotez**. Prostor hipotez določa vse dovoljene kombinacije atributov in parametrov uporabljenih v preslikovanju med atributi in razredom. Pogosto je veliko hipotez konsistentnih z učnimi podatki, vendar mnoge med njimi niso prave, ne opisujejo vseh možnih primerov v domeni, temveč se prilagajajo le učnim primerom. Ta znan pojav v strojnem učenju imenujemo preveliko prilagajanje podatkom (*angl.* overfitting) in ga navadno rešujemo na dva načina: s preferiranjem enostavnih hipotez (uporaba Occamovega rezila) ali z uporabo domenskega predznanja. Oba načina omejita prostor hipotez. Domingos [Dom99] ugotavlja, da je izbira enostavnih hipotez smiselna le, če je zelo verjetno, da je tudi prava hipoteza enostavna. V nasprotnem primeru se je uporaba domenskega predznanja skoraj vedno izkazala za bolj uspešno. V preostalih razlogih (b-d) je uporaba domenskega znanja praktično edina možnost, s katero lahko pomagamo učenju. Če so atributi nezadostni, mora domenski strokovnjak predlagati nove opisne atribute. Kadar je relacija med razredom in atributi kompleksna, lahko strokovnjak vodi algoritem učenja pri iskanju z omejitvijo prostora hipotez, ki verjetno vsebuje dobro hipotezo.

Problem večine obstoječih sistemov za strojno učenje je, da zahtevajo splošno domensko znanje, ki skorajda vedno velja in je uporabno za vse učne primere. Pridobivanje znanja je v splošnem znan problem, imenovan Feigenbaumovo ozko grlo [Fei84], saj domenski strokovnjaki pogosto niso sposobni dobro izraziti njihovega znanja. Po drugi strani se izkaže, da jim je veliko lažje razlagati znanje z uporabo konkretnih primerov. V doktorski disertaciji predlagamo nov način pridobivanja znanja. Od strokovnjakov ne zahtevamo splošnega domenskega znanja, temveč jim pokažemo le nekaj učnih primerov, za katere s svojim znanjem razložijo relacijo med atributi in razredom. Te razlage učnih primerov imenujemo **argumenti**. V tem kontekstu si lahko zamišljamo vsak učni primer kot vprašanje strokovnjaku in njihoviegovi odgovori so uporabljeni kot predznanje v učenju. Primer vprašanj:

- Zakaj je tega dne deževalo, če je bil prejšnji dan topel in sončen?
- Zakaj je pacient umrl zaradi infekcije, glede na to, da ni bilo opaziti povišane telesne temperature?
- Zakaj je lovec v dani šahovski poziciji slab?
- ...

Zavedati se moramo, da argument ni pravilo. Ker je strokovnjak podal argument v kontekstu enega samega učnega primera, je povezava med atributi in razredom v argumentu lahko specifična samo za dani učni primer in je ne moremo enostavno posploševati. Primere razširjene z argumenti kličemo **argumentirani primeri**.

Argumentirano strojno učenje (*angl.* Argument-Based Machine Learning, s kratico ABML), opisano v tej doktorski disertaciji, je razširitev strojnega učenja za učenje iz učnih podatkov in argumentov. Cilj učenja v ABML je hipoteza, ki izpelje vrednosti razreda iz atributov za vse učne primere in je hkrati konsistentna z danimi argumenti. Hipoteza je konsistentna z argumentom, če uporabi razloge iz argumenta pri izpeljavi argumentiranega primera. Za primer poglejmo potencialni odgovor strokovnjaka na vprašanje: "Zakaj je tega dne deževalo, če je bil prejšnji dan topel in sončen?" Potencialni odgovor: "Oblačnost in deževje sta se naslednji dan pojavila, ker je bil v zraku nizek pritisk." Na podlagi tega odgovora, se mora ABML metoda naučiti take hipoteze, ki bi med drugim uporabila razlog "nizek pritisk" pri razlagi tega učnega primera.

Kljub enostavnejšemu pridobivanju znanja s pomočjo argumentov, bi domenski strokovnjak še vedno moral vložiti mnogo dela, če bi si želeli imeti argumentirane prav vse učne primere. V praksi je to neizvedljivo, saj domenski strokovnjaki niso pripravljeni vložiti tako veliko časa. Za ta namen smo razvili metodo za detekcijo kritičnih učnih primerov, to so primeri, ki jih trenutno naučena hipoteza ne zna dobro napovedovati. Z razlago teh primerov strokovnjak doda tisto znanje, ki je za metodo ključno - metoda ga ni uspela sama odkriti iz podatkov. S tem novim znanjem bo metoda odkrila hipotezo s katero bo uspela napovedati kritični primer in posredno imela boljšo točnost napovedovanja. Vse skupaj potem ponavljamo (proces se imenuje ABML učni cikel), dokler metoda ne uspe več poiskati novih kritičnih primerov.

V praksi se izkaže, da lahko z ABML rešujemo vse zgoraj naštete potencialne težave pri strojnem učenju. Med razlago učnih primerov lahko domenski strokovnjak uporabi nov atribut, ki še ni vključen v domeno, in s tem sugerira vključitev tega atributa v domeno. Prav tako lahko v razlogih uporabi kompleksne relacije oz.

relacije, ki jih dan formalni jezik ne more opisati (npr. vsota atributov pri učenju pravil) in s tem predlaga izpeljani atribut, ki bi lahko pripomogel k učenju boljšega modela. In nenazadnje, argumenti omejijo prostor iskanja, kar posredno zmanjša verjetnost prevelikega prilagajanja učnim podatkom.

V naslednjem razdelku bomo najprej našteli vse prispevke znanosti pričujoče doktorske disertacije. Nadaljevali bomo z osnovno definicijo strojnega učenja in jo intuitivno opisali na enostavnem problemu učenja. Nadaljevali bomo z opisom implementiranega produkta ABCN2, ki omogoča učenje klasifikacijskih pravil iz argumentov in učnih podatkov. V četrtem razdelku bomo opisali nekaj eksperimentov in aplikacij z ABCN2 in v zadnjem razdelku povzeli in zaključili bistvene ugotovitve tega dela.

A.1.1 Prispevki znanosti

- Formalna definicija argumentiranega strojnega učenja (z uporabo Dungovega argumentacijskega okvirja) in motivacija za njegovo uporabo. Domenski strokovnjaki veliko lažje artikulirajo svoje znanje s pomočjo argumentov, kar je bistvena prednost uporabe argumentov v primerjavi z ostalimi oblikami predznanja.

- Definicija argumentiranih primerov in omejitve, ki jih podani argumenti predstavljajo za učne algoritme.

- Razvoj in implementacija algoritma ABCN2 za učenje klasifikacijskih pravil, ki se uči iz primerov in danih argumentov. ABCN2 se nauči množice pravil, ki je konsistentna z danimi učnimi podatki in argumenti.

- Razvoj metode EVC, ki glede na velikost preiskanega prostora popravi oceno kvalitete pravila. Ta metoda je pomembna za ABCN2, ker ne podcenjuje pravil, ki vsebujejo argumente v svojih pogojih. Poleg tega se izkaže, da metoda koristi tudi klasičnemu učenju pravil, saj se njihove točnosti izboljšajo.

- Novi postopek, imenovan PILAR, za klasifikacijo s pravili, ki temelji na EVC oceni. PILAR deluje bolje od trenutno znanih postopkov za klasifikacijo, saj EVC ocene bolje predstavljajo kvaliteto pravila kot porazdelitev primerov, ki jih pravilo pokrije. Metoda PILAR omogoča tudi uporabo pravil za korekcijo modelov naučenih z drugimi metodami, kar posredno omogoča uporabo argumentov v poljubni metodi strojnega učenja.

- Razvit je bil postopek za izbiro problematičnih primerov, ki bo omogočala domenskim strokovnjakom, da se osredotočijo le na nekaj ključnih primerov. Ta algoritem je nujno potreben, saj domenski strokovnjaki nimajo časa natančno razložiti prav vseh učnih primerov.
- Razvt je bil smo postopek za samodejno pridobivanje argumentov iz teksta. V našem eksperimentu smo izboljšali model za klasifikacijo živali z uporabo argumentov pridobljenih iz Wikipedijinih člankov.
- Evalvacija metode ABCN2 na več domenah. Ker domen ni dovolj za statističen dokaz prednosti argumentov, smo statistično dokazali, da naključni argumenti ne škodijo. Iz tega sklepamo, da lahko z argumenti kvečjemu izboljšamo točnost naučenega modela, ne morejo pa je poslabšati.

Vse zgoraj omenjene metode so implementirane v sistemu Orange [DZ04].

Tabela A.1: Učni primeri za problem odobritve kredita

Ime	RedniPlacnik	Premozen	StatusRacuna	BarvaLas	KreditOdobren
g. Tajkun	ne	da	negativen	plavolas	da
g. Medved	ne	ne	pozitiven	rjava	ne
gdč. Bevk	da	ne	pozitiven	plavolas	da
ga. Bogataj	da	da	pozitiven	plavolas	da
g. Računalnikar	da	ne	negativen	zelena	ne

A.2 Argumentirano strojno učenje

Običajno se definicija problema strojnega učenja glasi:

- Iz danih učnih primerov;
- Poišči hipotezo konsistentno z učnimi primeri.

Za lažje razumevanje ideje strojnega učenja si poglejmo enostaven problem učenja opisan v Tabeli A.1. Vsak primer je opisan s petimi atributi: *Ime*, *RedniPlačnik*, *Premožen*, *StatusRacuna* (status bančnega računa) in *BarvaLas* ter z razredom, ki označuje, če je bil stranki odobren bančni kredit ali ne. Cilj strojnega učenja je torej poiskati hipotezo, ki bi iz atributnega opisa stranke ugotovila, ali se stranki odobri kredit ali ne.

Algoritem za učenje pravil bi se zelo verjetno iz teh podatkov naučil naslednje pravilo:

$$\text{ČE } BarvaLas = plavolas \text{ POTEM } KreditOdobren = da$$
$$\text{DRUGAČE } KreditOdobren = ne$$

Očitno je pravilo konsistentno z danimi učnimi podatki, saj za vse učne primere napove pravilni razred. Vendar tako pravilo ne bi bilo smiselno večini bančnih uslužbencev, ker barva las le ni vedno dober indikator plačljive sposobnosti stranke.

V argumentiranem strojnem učenju (ABML) ima domenski strokovnjak možnost z argumentom razložiti razred nekaterih učnih primerov. Recimo, da si izbere primer gdč. Bevk in argumentira: "gdč. Bevk je dobila kredit, ker je redni plačnik mesečnih obrokov". Razširjene učne primere, ki poleg opisa z atributi in razredom vsebujejo še razlago z argumentom, imenujemo **argumentirani učni primeri**. V ABML argumenti vplivajo na učenje hipoteze; definicija argumentiranega strojnega učenja se tako glasi:

- Iz danih argumentiranih učnih primerov;
- Poišči hipotezo konsistentno z argumentiranimi učnimi primeri. Hipoteza je konsistentna z argumentiranim učnim primerom, če razlaga (izpeljava) razreda iz atributov vsebuje razloge omenjene v argumentu.

Zgoraj omenjeno pravilo tako ni več konsistentno s primerom gdč. Bevk, saj pri razlagi, zakaj ji je bil odobren kredit, omeni barvo las in ne rednega plačevanja obrokov, ki je bilo omenjeno v argumentu. Boljše pravilo za gdč. Bevk, ki bi se ga naučila ABML metoda, je:

ČE $RedniPlacnik = da$ IN $StatusRacuna = pozitiven$
POTEM $KreditOdobren = da$

V splošnem pričakujemo tri prednosti argumentiranega strojnega učenja:

1. Ker domenski strokovnjaki razlagajo posamezne primere, pričakujemo, da bodo lahko izrazili več relevantnega domenskega znanja.
2. Argumenti dodatno omejujejo prostor hipotez in s tem zmanjšujejo možnost prevelikega prilagajanja.
3. Naučene hipoteze bodo bolj smiselne strokovnjakom, saj pri razlagi primerov uporabljajo iste razloge kot so jih strokovnjaki sami.

Argumentirano strojno učenje je novo področje, vendar ideja kombiniranja strojnega učenja in argumentacije ni popolnoma nova. Pogosto se uporabi strojno učenje za gradnjo argumentov, ki so nato uporabljeni v argumentaciji [BA03; AR03]. Gomez and Chesnevar v svojem poročilu [GC04a] omenita nekaj idej združevanja argumentacije in strojnega učenja, kasneje pa razvijeta izboljšano metodo za učenje nevronskih mrež, ki uporablja argumentacijo [GC04b]. Clark v svojem delu [Cla88] omeni uporabnost argumentov kot predznanja, vendar se njegovi argumenti nanašajo na celotno domeno in ne na posamezne primere.

A.3 Argumentirano učenje pravil

V tem razdelku si bomo ogledali konkretno metodo argumentiranega strojnega učenja. Za osnovo bomo vzeli znani algoritem za učenje klasifikacijskih pravil CN2 [CB91] in ga razširili v metodo ABCN2, ki se uči pravil iz argumentov in učnih primerov. Najprej bomo formalizirali strukturo argumentiranih učnih primerov, s katerimi ABCN2 operira, nadaljevali z opisom osnovnega algoritma ABCN2 in na koncu predstavili nekaj izboljšav, ki so potrebne za bolj učinkovito in kvalitetno učenje pravil iz argumentiranih podatkov.[*]

A.3.1 Argumentirani učni primeri

Učni primer v strojnem učenju sestoji iz vrednosti atributov A oz. neodvisnih spremenljivk in vrednostjo razreda C (odvisne spremenjivke), ki jo hočemo napovedati. V argumentiranem strojnem učenju se učni primer razširi z dodatno razlago v obliki argumentov; tak učni primer imenujemo argumentirani učni primer AE in ga zapišemo kot trojko:

$$AE = (A, C, Arguments)$$

Element $Arguments$ je seznam *pozitivnih* in *negativnih* argumentov Arg_i, ki razlagajo (vendar ne dokazujejo) vrednost razreda. Pozitiven argument sugerira implikacijo med vrednostimi atributov in razredom:

$$C \text{ ker } r1 \wedge \ldots \wedge r_k$$

medtem ko negativni argument označuje vrednosti atributov, ki običajno negativno vplivajo na razred:

$$C \text{ čeprav } r1 \wedge \ldots \wedge r_j$$

Vsak argument vsebuje trditev, ki je kar vrednost razreda in konjunkcijo pogojev, kjer se vsak pogoj nanaša na en sam atribut. Pogoji so lahko numerični: $X = x_i$ (enakost), $X > x_i$ (vrednost je večja od izbrane mejne točke) in $X < x_i$ (vrednost je manjša od mejne točke), ali kvalitativni: vrednost atributa X je visoka oz. vrednost X je nizka.

V našem enostavnem problemu učenja odobravanja kredita se je argument glasil: "Gospodična Bevk je dobila kredit, ker redno plačuje mesečne obroke," ki ga zapišemo v naši notaciji kot:

$$KreditOdobren = da \text{ ker } RedniPlacnik = da$$

[*]Na tem mestu bi veljalo omeniti, da učenje pravil ni edina možna metoda v ABML. V doktorski disertaciji predstavimo varianto logistične regresije in ILP za učenje na argumentiranih primerih, kar bomo v tem razširjenem povzetku izpustili.

Podobno bi storili v primeru negativnega argumenta, le besedo “ker” bi morali zamenjali s “čeprav”. Na primer, argument: “Gospodična Bevk je dobila kredit, čeprav ni premožna”, bi interpretirali kot:

$$KreditOdobren = da \text{ čeprav } Premozen = ne$$

A.3.2 ABCN2 algoritem

Algoritem CN2 spada med pokrivne algoritme za učenje pravil; v vsakem ciklu se postopek nauči najboljše pravilo na nepokritih primerih, doda pravilo v seznam vseh pravil, odstrani primere, ki jih pravilo pokrije in ponavlja, dokler niso vsi primeri pokriti. Obstajata dve verziji: CN2 za učenje urejenih pravil [CN89] in CN2 za učenje neurejenih pravil [CB91]. Bistvena razlika med pravili naučenimi z enim in drugim algoritmom je v interpretaciji naučenih pravil - pri klasifikaciji z urejenimi pravili uporabimo prvo pravilo v seznamu, ki proži za testni primer, medtem ko pri neurejenih pravilih upoštevamo vsa pravila, ki prožijo, in jih kombiniramo s poljubno metodo za reševanje konfliktov med pravili. Metoda ABCN2 je razširitev slednjega algoritma.

Pri definiciji argumentiranega strojnega učenja smo zahtevali konsistenco naučene hipoteze z argumentiranimi primeri, kjer mora hipoteza omeniti razloge iz argumenta pri razlagi razreda. V učenju pravil je hipoteza množica naučenih pravil, le-ta pa pri razlagi primera omeni pogoje pravil, ki pokrivajo argumentirani primer. V CN2 pravilo pokrije primer, če so vsi pogoji v pravilu resnični za vrednosti danega učnega primera. V ABCN2 ta definicija ne zadošča, saj v argumentiranem strojnem učenju zahtevamo razlago primera s pogoji omenjenimi v argumentu. Nova definicija relacije pokritosti (oz. AB-pokritost) je:

1. Vsi pogoji v pravilu morajo biti resnični za primer (tako kot v CN2).
2. Pravilo mora v svojih pogojih omeniti vsaj enega od pozitivnih argumentov (če je primer argumentiran).
3. Pravilo v pogoju ne sme omeniti negativnih argumentov.

Kot primer si lahko pogledamo nekaj pravil, ki vsa pokrivajo našo gdč. Bevk, vendar le eno od njih AB-pokriva ta primer. Spomnimo se, argument se je glasil: ”gdč. Bevk je dobila kredit, ker redno plačuje mesečne obroke, čeprav ni premožna”. Pravila so:

R1: ČE $BarvaLas = plavolas$ POTEM $KreditOdobren = da$

R2: ČE $Premozen = ne$ IN $BarvaLas = plavolas$
POTEM $KreditOdobren = da$

R3: ČE $Premozen = ne$ IN $RedniPlacnik = da$
POTEM $KreditOdobren = da$

R4: ČE $BarvaLas = plavolas$ IN $RedniPlacnik = da$
POTEM $KreditOdobren = da$

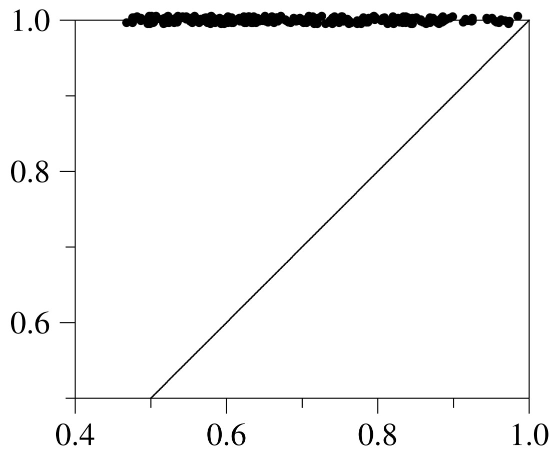
Vsa štiri pravila so konsistentna z danimi učnimi podatki in pokrivajo primer gdč. Bevk. Vendar prvo pravilo ne omeni pozitivnega argumenta, torej ne AB-pokriva primera. Zaradi istega razloga drugo pravilo ne pokrije primera, še več, drugo pravilo omeni še negativni argument, torej krši tudi tretji pogoj v definicij AB-pokritosti. Tretje pravilo prav tako ne zadostuje pogojem, saj prav tako omeni negativni argument, medtem ko četro pravilo AB-pokriva argumentirani primer.

Cilj učenja z ABCN2 je poiskati čim bolj točno množico pravil, ki AB-pokrijejo vse učne primere. Uporaba relacije AB-pokritosti v osnovnem CN2 bi teoretično že zadoščala, vendar tak algoritem ne bi bil učinkovit. V pričujočem doktorski disertaciji zato predlagamo algoritem, ki v vsakem ciklu zahteva, če še obstajajo nepokriti argumentirani primeri, da naučeno pravilo pokrije vsaj en argumentiran primer. V skladu s to zahtevo se spremeni algoritem za iskanje najboljšega pravila: ta vrne najboljše pravilo, ki AB-pokrije vsaj en argumentirani primer. Celotni ABCN2 algoritem je opisan v poglavju 5.2 (Algoritma 5.2 in 5.3).

A.3.3 Ocenjevanje kvalitete pravila v ABCN2

Za izbiro najboljšega pravila v ABCN2 se za primerjavo med pravili uporablja mera kvalitete pravila. V splošnem naj bi kvaliteta pravila odražala pričakovano točnost pravila, to je točnost na vseh možnih primerih v populaciji in ne le na učnem vzorcu. V originalnem CN2 algoritmu je mera za točnost Laplace-ova formula $\frac{p+1}{n+2}$ [CB91], kjer je p število pozitivnih pokritih primerov, n pa število vseh pokritih primerov. Kasnejše različice CN2 algoritma so uporabljale različne mere točnosti, npr. v Džeroski et al. [DCP93] so uporabili Cestnikovo m-oceno [Ces90].

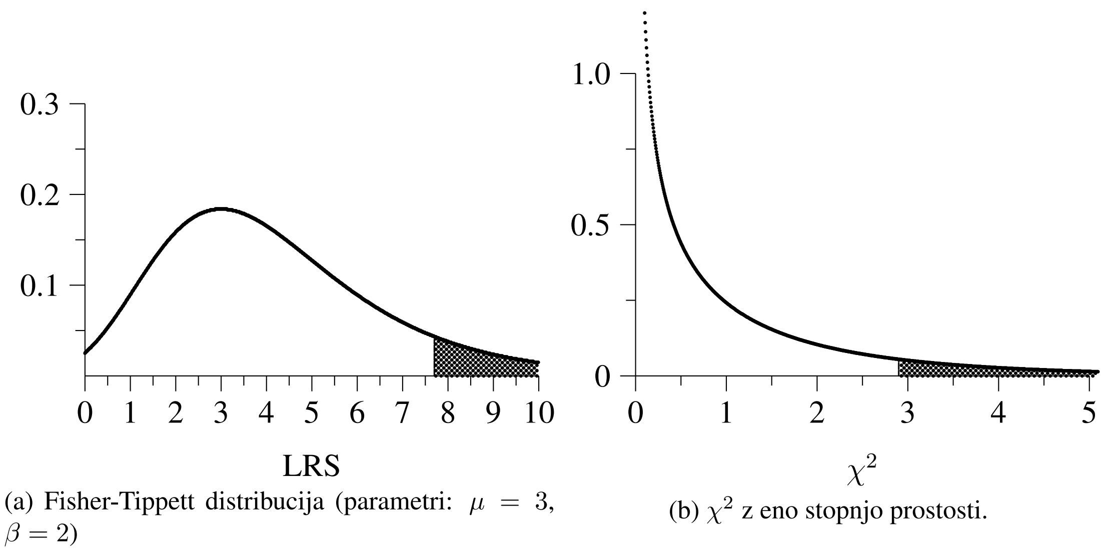
Vendar imajo Laplace-ova formula, m-ocena in vse ostale uporabljene mere skupni problem: ne upoštevajo velikosti preiskanega prostora pri učenju pravila. Namreč, v domenah z velikim številom opisnih atributov je zelo verjetno, da bo najboljše pravilo imelo relativno visoko točnost na učnih podatkih, čeprav bi bili atributi generirani popolnoma naključno, medtem ko se kaj takega v domenah z manjšim



Slika A.1: Primerjava izračunane ocene točnosti pravila z relativno frekvenco (ordinata) in pravo točnostjo pravila (absciso).

število atributov zelo verjetno ne more zgoditi. Ta problem, ki je v literaturi razmeroma znan (*angl. multiple-comparison procedures*) [JC00], se še močneje izraža v argumentiranem učenju pravil. Tu so nekatera pravila naučena z razmeroma malo preiskovanja (so konsistentna z argumenti) in so zaradi tega v primerjavi z drugimi pravili relativno podcenjena. Za ilustracijo problema si poglejmo uporabo relativne frekvence kot mere točnosti pravila. Graf A.1 vsebuje rezultate večkratnega učenja pravila na umetnih podatkih, kjer smo poznali točnost pravila na celotni množici (se kaže na abscisi) in izmerjeno točnost z relativno frekvenco (ordinata). Izkaže se, da je relativna frekvenca popolnoma neuporabna mera za točnost pravila, saj vedno lahko poiščemo pravilo s 100% točnostjo na učnih podatkih neodvisno od njene točnosti na vseh podatkih. Pravimo, da je relativna frekvenca optimistična ocena, saj pravilom priredi vrednosti, ki je višje od pravih vrednosti.

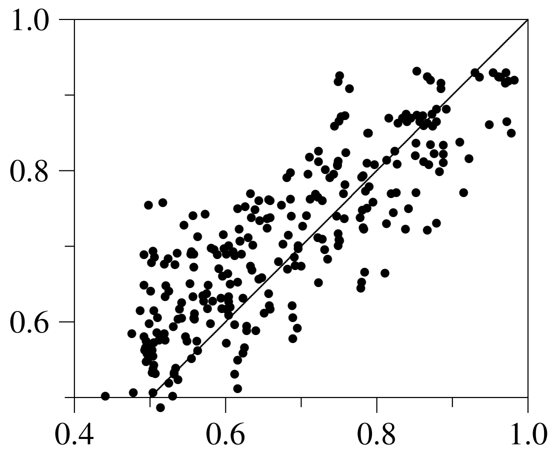
Razvili smo metodo EVC (kratica; *angl. extreme value correction*), ki popravi optimizem funkcije kvalitete pri učenju pravil. Metoda temelji na predpostavki, da za dano funkcijo kvalitete ne smemo uporabljati običajne porazdelitve (npr. beta porazdelitev pri Laplace-ovi formuli, χ porazdelitev pri χ-kvadrat metodi), temveč bi bilo bolje uporabiti ustrezno esktremno porazdelitev, saj je kvaliteta najboljšega pravila ekstremna točka med vsemi kvalitetami vseh pravil. EVC preslika ekstremno vrednost kvalitete v ustrezno vrednost na ne-ekstremni porazdelitvi in pri tem ohrani relativno verjetnostno pozicijo točke. Slika A.2 kaže primer take preslikave iz Fisher-Tippett porazdelitve (ekstremna porazdelitev za χ porazdelitev) v χ porazdelitev. Naj opozorimo, da se pri korekciji z našo metodo vrednost kvalitete vedno zmanjša, saj



(a) Fisher-Tippett distribucija (parametri: $\mu = 3$, $\beta = 2$)

(b) χ^2 z eno stopnjo prostosti.

Slika A.2: Verjetnostne distribucije



Slika A.3: Primerjava izračunane ocene kvalitete pravila s popravljeno relativno frekvenco (ordinata) in pravo kvaliteto pravila (absciso).

se odstrani efekt ekstremne vrednosti oz. efekt obširnega preiskovanja. Z EVC lahko popravimo poljubno metodo za evalvacijo pravil. Slika A.3 kaže relacijo med popravljeno relativno frekvenco in pravo točnost pravil. Popravljena relativna frekvenca mnogo bolje odraža pravo točnost, kot pa samo relativna frekvenca.

Pri uporabi EVC metode v učenju moramo natančno izračunati ustrezne koeficiente ekstremne porazdelitve, ki zavisijo od izbrane funkcije kvalitete in lastnosti učnih podatkov (npr. število primerov, število atributov). Ker so koeficienti odvisni od lastnosti učnih podatkov, ne smemo, ko se algoritem nauči eno pravilo, odstraniti primerov, ki jih to pravilo pokrije. S tem bi namreč spremenili lastnosti podatkov in naučeni koeficienti ne bi več veljali. Za rešitev tega problema smo v doktorski disertaciji razvili nov postopek imenovan "verjetnostno pokrivanje", ki ne odstranjuje pokritih primerov, vendar še vedno dobro usmerja iskanje novih pravil.

A.3.4 Algoritem PILAR: klasifikacija s pravili in popravljanje poljubne metode strojnega učenja

Osnovna prednost neurejenih pravil je njihova sposobnost opisovanja lokalnih relacij med atributi in razredom. Vendar imajo pravila tudi slabost in sicer, da je njihova uporaba za klasifikacijo novih primerov v večini primerov netrivialna, saj lahko za dani primer proži več pravil, ki so med seboj konfliktna. Lahko bi rekli, da so pravila skupek lokalnih modelov in pri napovedovanju moramo agregirati njihove prispevke.

Večina metod strojnega učenja zgradi en sam globalni model, ki ga lahko enostavno uporabimo za napovedovanje, vendar nima možnosti lokalne razlage. V disertaciji tako predlagamo algoritem PILAR, ki kombinira oba postopka. Ta algoritem združi osnovno globalno metodo s pravili z uporabo log-linearnega modela in prilagodi koeficiente modela, da lokalne napovedi globalnega modela ustrezajo lokalnim napovedim pravil. V posebnem primeru, ko je globalni model primitiven (npr. napovedovanje večinskega razreda), PILAR metoda deluje kot agregator pravil. V doktorski disertaciji primerjamo uporabo PILAR-ja kot agregatorja pravil in kot popravek logistične regresije. V obeh primerih dobimo statistično boljše rezultate.

A.4 ABML učni cikel

Argumenti v argumentiranih primerih so običajno pridobljeni s strani domenskih strokovnjakov, ki s svojim predznanjem razložijo relacijo med razredom primera in vrednostimi atributov. Zato ne moremo pričakovati, da bi strokovnjaki argumentirali vse učne primere, saj za kaj takega ne bi imeli časa. Pripraviti jim moramo izbiro učnih primerov, ki bodo z argumenti občutno vplivali na kvaliteto naučene hipoteze. Na primer, ne bi bilo smiselno argumentirati primerov, ki jih metoda že sama dobro razloži.

V ta namen smo razvili iterativni postopek za odkrivanje in argumentiranje kritičnih primerov, ki ga imenujemo ABML učni cikel. Cikel začne z "navadnimi" učnimi primeri (brez argumentov) in v vsakem ciklu doda argument k enemu učnemu primeru. Postopek, ki se ustavi, ko ne najde več novega kritičnega primera, ima štiri korake:

1. Nauči se hiptezo z ABML na danih učnih podatkih.
2. Poišči kritični primer in ga predloži domenskemu strokovnjaku za argumentacijo. Če kritičnega primera ne najdemo, se postopek ustavi.
3. Domenski strokovnjak razloži dani primer z argumenti v naravnem jeziku. Ti argumenti se ročno prevedejo v formalne argumente, ki jih sprejema ABML metoda, in dodajo h kritičnemu učnemu primeru.
4. Vrni se na korak 1.

Osnovna ideja koraka 2 je poiskati primer, ki ga metoda ne more uspešno razložiti, z drugimi besedami, metoda napačno napove razred primera. V najbolj enostavni izvedbi za iskanje teh primerov se naučimo modela in ga testiramo na učnih podatkih. Napačno klasificirani primeri tako postanejo kritični primeri. Problem tega pristopa je, da ne deluje pri metodah z lastnostjo prevelikega prilagajanja (*angl. overfitting*), kjer je bolje uporabiti metodo križnega preverjanja za odkrivanje kritičnih primerov.

A.5 Eksperimenti in aplikacije

Za evalvacijo metode ABCN2 bomo v tem razdelku predstavili vrsto eksperimentov in aplikacij. To je edini možni način evalvacije, saj korektna statistična primerjava ni praktično izvedljiva, ker bi potrebovali mnogo učnih domen in za vsako domenskega strokovnjaka, ki bi razlagal učne primere. Z izvedenimi eksperimenti in aplikacijami odgovorimo na nekaj ključnih vprašanj, ki se tičejo evalvacije metode:

- Ali so hipoteze naučene iz argumentiranih podatkov boljše kot hipoteze iz navadnih (neargumentiranih podatkih) s stališča točnosti in razumljivosti?
- Kako učinkovita je metoda za izbiro kritičnih primerov?
- Ali oziroma kako težko je domenski strokovnjakom podajati znanje v obliki argumentov?
- Ali lahko rekonstruiramo (in izboljšamo) dano bazo znanja, kjer se trenutno znanje v bazi uporablja kot izvor argumentov?

- Ali lahko (in kako) uporabimo argumente, ki omenjajo informacijo, ki ni zapisana v atributih?
- Ali lahko argumente avtomatsko pridobimo iz neodvisne literature (npr. internet)?
- Kako napake argumentov vplivajo na točnost naučenih hipotez?

A.5.1 Klasifikacija živali

Prvi eksperiment z ABCN2 smo izvedli na domeni za klasifikacijo živali, kjer je cilj klasificirati živali glede na njihov tip v živalskem drevesu (domena ZOO [AN07]). Domena vsebuje 101 žival, vsaka je opisana s sedemnajstimi atributi in klasificirana v enega od naslednjih razredov: *sesalec, ptič, plazilec, riba, dvoživka, insekt, ostalo*. Ta domena je zelo ustrezna za začetno evalvacijo, ker ne potrebujemo pravega domenskega strokovnjaka, ampak lahko ustrezne argumente kar poiščemo iz enciklopedije.

Domeno smo razdelili na učno množico (70%) in testno množico (30%). Najprej smo se naučili pravil brez uporabe argumentov, kjer smo z ABCN2 na testni množici dosegli točnost približno 90%. Nato smo znotraj ABML učnega cikla identificirali tri kritične primere:

1. Prvi kritični primer je bil *želva*. Želva je plazilec in za plazilce je med drugim značilno, da imajo hrbtenjačo in ležejo jajca (argument: želva je plazilec, ker hrbtenjača=da in leže_jajca=da). S pomočjo tega argumenta je ABCN2 dodal pravilo (poudarjen del pravila izhaja iz argumenta):

   CE ***hrbtenjaca=da*** IN ***leze_jajca=da*** IN
   vodna_zival=ne IN *perje=ne* POTEM tip=plazilec

2. Naslednji kritični primer je bila morska kača (plazilec). V opisnih atributih tega primera piše, da morska kača ne diha, čeprav v vseh knjigah piše, da morska kača diha s pljuči. Očitno gre pri tem primeru za napako v podatkih.

3. Zadnji kritični primer je bil močerad (dvoživka). Naš argument je: močerad je dvoživka, ker ima hrbtenjačo, leže jajca in mora živeti v vlažnem okolju za preživetje. Pravilo naučeno iz tega argumenta je bilo:

CE ***hrbtenjaca=da*** IN ***leze_jajca=da*** IN ***vodna_zival=da***
IN *stevilo_nog=4* POTEM tip=dvozivka

V eksperimentu smo torej vse skupaj dodali dva argumenta in oba sta bila nepopolna, saj je ABCN2 oba specializiral z dodatnimi pogoji. Končna točnost pravil na učni množici je bila 100%. Eksperiment nazorno prikazuje učinkovitost ABML učnega cikla, saj smo s pomočjo samo dveh argumentov uspeli povečati točnost za 10%. Poleg tega so nova pravila bolj v skladu s strokovnim znanjem - enciklopedijo in s tem že na prvi pogled izgledajo pravilneje. Izkazalo se je tudi, da je ABML učni cikel primeren za odkrivanje napak v učnih primerih.

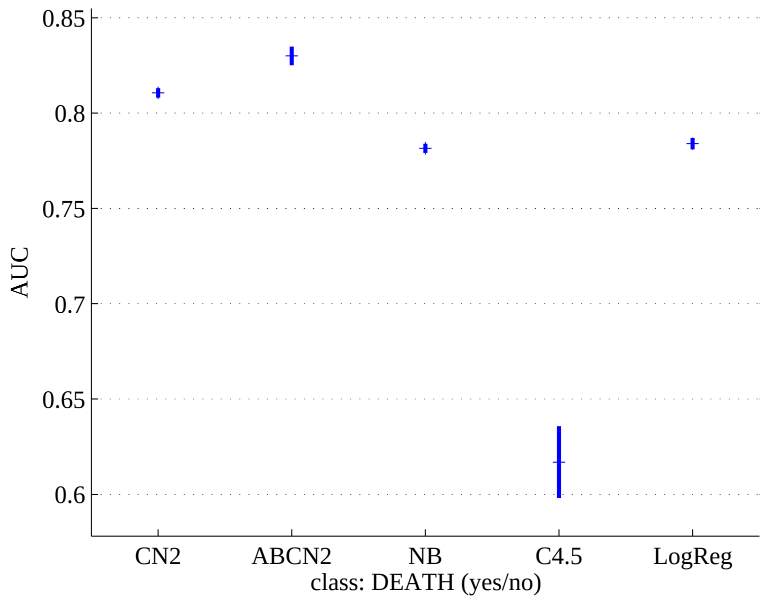
A.5.2 Odobritev socialne pomoči

V tem delu bomo povzeli rezultate eksperimenta z ABCN2 v pravni domeni: odobritev socialne pomoči za obisk družinskih članov v bolnici [MvBC+06]. Domena je sicer umetna, vendar je bila zaradi realnosti atributov in primerov večkrat citirana v pravni literaturi, ki so jo v preteklosti reševali z različnimi tehnikami strojnega učenja.

Točnost osnovne CN2 metode brez uporabe argumentov je zelo visokih 99%. Po sedmih ABML ciklih se je točnost z uporabo ABCN2 povečala na 99.8%. Ker je bila točnost učenja brez argumentov že zelo visoka, seveda ni možno dobiti mnogo boljše rezultate z argumentiranim učenjem. Glavna prednost ABCN2 v tej domeni je razumljivost naučenih pravil, saj so bila končna pravila mnogo bolj razumljiva domenskemu strokovnjaku (Trevor Bench Capon), ker so vsebovala njegove razlage primerov. Izkazalo se je tudi, da je ABCN2 mnogo bolj odporna na šum od ostalih metod, kar je ključnega pomena za tak tip aplikacij, saj odločitve administracije vsebujejo veliko napačnih odločitev.

A.5.3 Prognostika infekcije med starejšimi občani

V tej domeni smo z ABCN2 razvili model za pomoč zdravnikom pri ocenjevanju kritičnosti infekcije med starejšimi občani. Na podlagi tega modela se naj bi zdravniki odločili, ali pacient ostane v bolnici ali lahko oddide domov. Podatke smo dobili iz ljubljanske klinike za infekcijske bolezni in vključujejo vse ljudi starejše od 65 let, ki so prišli v bolnico med 1.junijem 2004 in 1.junijem 2005 zaradi okužbe. Razred označuje ali je pri pacientu v roku 30 dni po nastopu okužbe nastopila smrt.



Slika A.4: Povprečne vrednosti in standardne napake AUC.

Tu je domenski strokovnjak (zdravnik) argumentiral vse pozitivne primere (kjer je pacient preminil) in nobenega negativnega. Z uporabo argumentov je bila ABCN2 signifikantno boljša metoda (glede na klasifikacijsko točnost, AUC in Brier Score) od CN2, naivnega Bayesa in logistične regresije (glej sliko A.4 za primerjavo AUC). Kljub boljšim rezultatom učenja z argumenti in upoštevaje, da so bila naučena pravila konsistentna s strokovnim znanjem, se domenskemu strokovnjaku (dr. Jerneja Videčnik, dr. med.) ABCN2 pravila niso zdela bolj (niti ne manj) smiselna od pravil dobljenih z navadnim CN2. Ta anomalija nastane zaradi kratke dolžine argumentov. Vsi argumenti so namreč vsebovali le en razlog, na primer:

$$smrt = da \text{ ker } frekvenca\_dihanja >= 16$$

Ker tak argument omeji preiskovanje razmeroma malo (pravilo mora vsebovati le ta pogoj, ostali so poljubni) in ker so bila pravila razmeroma dolga (3-5 pogojev na pravilo), se pravila niso občutno razlikovala po razumljivosti. Vseeno pa so ti kratki argumenti še vedno omogočili bolj točno učenje.

A.5.4 Prognostika bolezni srca

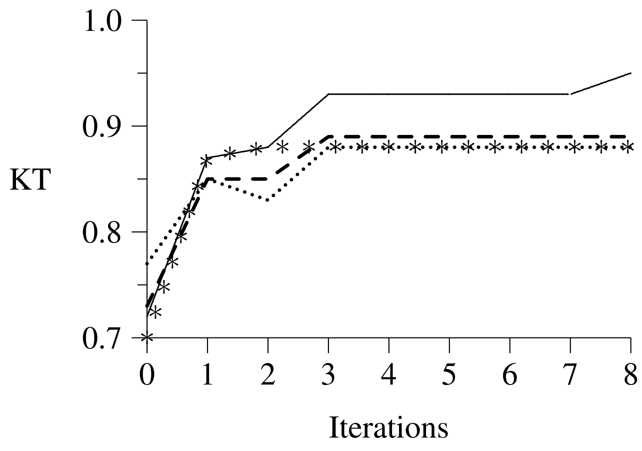
Domena za prognostiko bolezni srca (*SAHeart*) [RdPB$^+$83] je prosto dostopna in opisuje 462 moških iz Južne Afrike. Učni primeri so opisani z devetimi atributi: *krvni pritisk, kajenje, holesterol, debelost, družinska anamneza, stil življenja (type-A), kožna guba, alkohol in starost* ter razredom *chd*, ki označuje, če pacient ima srčno bolezen ali ne.

Pri tem poskusu nismo imeli na voljo domenskih strokovnjakov. Zanimalo nas je, če lahko z uporabo našega splošnega znanja (ki je lahko napačno) o srčnih boleznih pomagamo ABCN2, da se nauči bolj točne hipoteze. Ta test služi tudi kot preizkus ABCN2, kako občutljiva je metoda na napake v argumentih. Po desetih iteracijah ABML učnega cikla se je klasifikacijska točnost ABCN2 zvišala iz začetnih 71% na 75%, Brier Score se je izboljšal iz 0.37 na 0.36, le AUC se ni spremenila. Ne glede na to, da naši argumenti niso bili popolni, smo še vedno uspeli izboljšati začetni model. To je pomembna ugotovitev, saj kaže, da predznanje ne rabi biti popolnoma pravilno in je lahko še vedno uporabno. Seveda se zavedamo, da bi kardiologovi argumenti verjetno še bolj izboljšali točnost naučenega modela.

A.5.5 Odobravanje kredita (Japanese Credit Screening Database)

Japonska domena za odobravanje kredita opisuje 125 prosilcev kredita z desetimi atributi in razredom, ki označuje, če je oseba dobila kredit ali ne. Domeni je priloženo tudi suboptimalno domensko znanje (6 pravil, ki jih posamezniki v bankah uporabljajo za podeljevanje kredita), ki doseže 83% klasifikacijsko točnost na učnih primerih. V tem poskusu smo uporabili to domensko znanje kot trenutno bazo znanja, ki nam služi kot izvor argumentov. Cilj učenja z ABCN2 je bil (a) rekonstruirati to bazo znanja in jo (b) izboljšati.

Brez argumentov se je ABCN2 naučil le tri pravila, ki so dosegala 76% klasifikacijsko točnost. Po petih iteracijah ABML učnega cikla je model obsegal 7 pravil in 89% klasifikacijsko točnost. S tem smo popolnoma uspeli rekonstruirati originalno bazo znanja, saj je, od vseh sedmih naučenih pravil, šest pravil popolnoma ustrezalo originalni bazi znanja. Sedmo pravilo tako predstavlja dodatno pridobljeno znanje, ki je še izboljšalo klasifikacijsko točnost iz 83% na 89%.



Slika A.5: Izboljšanje klasifikacijske točnosti (KT) skozi 8 ABML učnih ciklov za ABCN2 (polna črta), logistično regresijo (zvezdice $*$), C4.5 (prekinjena črta) in navadni CN2 (pikčasta črta).

A.5.6 Konstrukcija kompleksnih šahovskih konceptov

Dandanašnji računalniški programi za igranje šaha so izredno dobri pri igranju šaha, vendar skorajda neuporabni za učenje šaha ali razlago šahovskih partij. Vsako pozicijo označijo s številosko vrednostjo, ki označuje agregirano situacijo pozicije. Na primer, vrednost 1.20 naj bi pomenila, da je beli v prednosti za dobrega kmeta, čeprav ni nujno, da tudi dejansko ima kmeta več; lahko ima iniciativo v napadu, boljše postavljene lovce, itd. Trenutni programi ne znajo razumljivo razložiti te vrednosti človeku, zakaj je beli v boljšem položaju. Nedavno smo razvili sistem za učenje šaha [SMG$^+$06], ki razlaga vrednosti hevristične funkcije s spremembami vrednosti posameznih atributov v evaluaciji pozicije. Izkaže se, da hevristične funkcije v glavnem uporabljajo primitivne šahovske koncepte (npr. število kmetov, mobilnosti figur, itd.), ki pa niso dovolj za razumevanje splošnih šahovskih pozicij.

Naslednji eksperiment demonstrira uporabnost ABCN2 za učenje kompleksnih šahovskih konceptov. Izbrali smo koncept slabega lovca. Vzeli smo 200 šahovskih pozicij in jih opisali z atributi, ki so tipično uporabljeni v hevrističnih funkcijah za igranje šaha. Jana Krivec (ženska velemojstrica) in Matej Guid (FIDE mojster) sta v vsaki od teh pozicij ocenila lovca kot “slabi lovec” ali “dober lovec” in pri tem uporabljala njuno šahovsko predznanje.

Za začetek sta domenska strokovnjaka Guid in Krivec poskusila z uporabo danih atributov ročno generirati pravila, ki bi razlikovala dobrega od slabega lovca. Vendar se je to izkazalo za zelo težek problem, saj so njuna pravila delovala relativno slabo, dosegla so samo 59% klasifikacijsko točnost. ABCN2 je na danih podatkih (brez argumentov) dosegla mnogo boljšo točnost 72%. Nato smo izvedli 8 ABML učnih ciklov in pri tem dodali 7 argumentov. V nekaterih kritičnih primerih Guid in Krivec nista uspela razložiti danih pozicij s samo obstoječimi atributi, zato smo v procesu dodali nekaj dodatnih atributov. Točnost končnega modela je dosegla visokih 95%. Iz slike A.5 je očitna učinkovitost algoritma za izbiro kritičnih primerov, saj prvih nekaj argumentov najbolj poveča točnost naučenih pravil. Vidi se tudi, da je dodajanje atributov, ki so bili predlagani z argumenti, mnogo prispevalo k točnosti ostalih metod. ABML učni cikel je torej dober postopek tudi za odkrivanje manjkajočih atributov. Ta eksperiment jasno kaže izboljšanje klasifikacijske točnosti z uporabo argumentov. Še bolj pa kaže, da je strokovnjakom mnogo lažje razlagati posamezne primere in te razlage uporabiti v ABML metodi (končna točnost 95%), kot generirati celoten napovedni model brez pomoči računalnika (točnost samo 59%).

A.5.7 Avtomatska konstrukcija argumentov iz teksta

V vseh ABML aplikacijah, opisanih do tega trenutka, so domenski strokovnjaki argumentirali učne primere. V Možina s sodelavci [MGB09] smo argumente ekstrahirali iz teksta kot prikaz uporabe ABML, kadar nimamo dostopa do domenskih strokovnjakov. Naš pristop predpostavlja, da je v tekstu mnogo lažje odkriti specifične relacije za posamezne primere, kot pa avtomatsko generiranje celotnih hipotez samo iz teksta.

Argumente smo konstruirali s pomočjo algoritmov za iskanje semantičnih relacij v tekstu; iskali smo semantične relacije med danimi vrednostimi atributov in razreda za posamezne primere. Recimo, da nas zanima, zakaj je želva plazilec, pri čemer iz atributnega opisa vemo, da želva leže jajca. Naš algoritem v tekstu poišče vse stavke, kjer se besedi "jajca" in "plazilec" hkrati pojavljata, npr.: "večina plazilcev leže jajca"ali "plazilci valijo jajca". Na podlagi števila takih stavkov z enostavno statistično metodo ugotovimo, če je povezava med vrednostjo atributa in razreda statistično signifikantna, in če je, to relacijo uporabimo kot del argumenta. Za boljše odkrivanje semantičnih povezav v naravnem jeziku smo uporabili WordNet [Fel98], bazo z mnogimi semantičnimi povezavami med besedami, za odkrivanje sinonimov in raznovrstnih izpeljank (npr. plazilec $\rightarrow$ plazilci).

Pristop smo testirali na domeni za klasifikacijo živali. Vsi argumenti so bili avtomatsko konstruirani iz člankov na Wikipediji.[†] Za evalvacijo smo 10-krat ponovili 10-kratno križno preverjanje in vsakič generirali argumente za prav vse učne primere. Brez argumentov je ABCN2 v povprečju dosegla 94.5% klasifikacijsko točnost, a z argumenti 96.7%, kar je statistično boljši rezultat. ABCN2 se nauči boljšega modela tudi od ostalih standardnih metod strojnega učenja (npr. naivni Bayes 92.6%, SVM 90%, C4.5 92.57%). Rezultati se v tem eksperimentu rahlo razlikujejo od prej opisanega eksperimenta na tej domeni, zaradi spremenjene tehnike evalvacije.

A.5.8 Vpliv napačnih argumentov na točnost hipoteze

Ali lahko napačni (oz. slabi) argumenti poslabšajo točnost končnega modela glede na originalni model? Ker argumenti omejujejo prostor sprejemljivih hipotez, bi lahko pričakovali, da bodo napačni argumenti omejili prostor iskanja na prostor slabih hipotez in s tem onemogočili ABCN2 poiskati dober končni model.

Z obsežnim eksperimentiranjem smo pokazali, da argumenti ne morejo poslabšati modela dokler so argumenti boljši ali enako dobri kot naključno generirani argumenti. Izkaže se namreč, da naključno generirani argumenti v povprečju ne poslabšajo točnosti končnega modela, iz česar lahko sklepamo, da tudi argumenti, ki so boljši od naključnih ne bodo poslabšali modela. Možno bi bilo s pazljivo konstruiranimi argumenti, ki so slabši od naključnih, poslabšati končni model. Vendar, ker tega od domenskih strokovnjakov, ki si želijo dobiti čim boljše modele, ne moremo pričakovati, lahko trdimo, da napačni (oz. slabi) argumenti ne poslabšajo točnosti končnega modela.

A.6 Zaključek

V disertaciji smo definirali argumentirano strojno učenje, ki združuje strojno učenje z nekaj koncepti iz argumentacijske teorije. Argumentacijska teorija je posebna veja logike, ki ne zahteva nokonfliktnosti med pravili in dejstvi ter s tem omogoča enostavno pridobivanje ekspertnega znanja in simulira človeško sklepanje. V argumentiranem strojnem učenju se domenski strokovnjak fokusira na konkretni učni primer in poskuša z argumentom razložiti relacijo med razredom in atributi.

Učni primer, razložen z argumentom, se imenuje argumentirani učni primer. V argumentiranem strojnem učenju se učimo iz argumentiranih učnih podatkov, kjer

[†]`http://en.wikipedia.org`

zahtevamo, da je naučena hipoteza konsistentna z danimi učnimi podatki in njihovimi argumenti. Na podlagi osnovih principov argumentiranega učenja smo razvili metodo ABCN2, ki je razširjenja različica znanega algoritma za učenje pravil CN2.

ABCN2 smo še dodatno razširili z metodi EVC in PILAR. Prva metoda korigira ocenjeno točnost pravil glede na število pregledanih pravil v postopku učenja. PILAR je klasifikacijska metoda, ki upošteva korekcije metode EVC in zgradi log-linearni globalni model iz naučenih pravil. Obe metodi sta se izkazali kot pomembni izboljšavi ABCN2, vendar povečujeta zahtevani čas učenja. Slednje nas ne moti, saj je sam čas učenja še vedno zanemarljivo majhen v primerjavi s časom, ki ga porabimo za sodelovanje s strokovnjakom.

Predlagani algoritem za izbiro kritičnih primerov vedno izbere primer, ki ga trenutna metoda napačno klasificira. Vprašanje za prihodnje delo je, če je to vedno smiselno. Namreč, metoda lahko napačno klasificira tudi osamelec (*angl.* outlier), ki ga strokovnjaki zelo verjetno ne bodo znali razložiti oz., če bodo našli razlago, je le-ta lahko izsiljena. Ali je možno za dani učni primer oceniti verjetnost, da bo domenski strokovnjak poznal odgovor? S tem vprašanjem se bomo ukvarjali v prihodnosti.

Z eksperimenti smo ugotovili naslednje prednosti ABML pristopa in metode ABCN2:

- Domenskim strokovnjakom je lažje argumentirati specifične primere kot podajati splošno domensko znanje.
- Kritični primeri, ki bi z argumenti najbolj vplivali na točnost modela, so avtomatsko odkriti znotraj ABML učnega cikla.
- S protiprimeri, ki spodbijajo dane argumente, omogočimo strokovnjakom izpopolnitev njihovih argumentov. V enem od eksperimentov (učenje šahovskih konceptov) se je izkazalo, da s tem celo izboljšamo znanje strokovnjakov, saj se do tega trenutka niso zavedali teh protiprimerov.
- Pravila naučena z ABCN2 so strokovnjakom bolj razumljiva kot pravila naučena samo s CN2.
- V številnih eksperimentih in aplikacijah je ABCN2 dosegel boljšo klasifikacijsko točnost (in tudi druge mere kvalitete) kot CN2.

V enem od zadnjih eksperimentov smo razvili metodo za avtomatsko odkrivanje argumentov iz teksta. Ideja tega postopka je odkriti argumente za dane primere, kadar nam domenski strokovnjak ni na voljo. Trenutno smo s tem postopkom dobili dobre rezultate na enem eksperimentu in v nadaljnjem delu planiramo ta princip aplicirati še na druge domene.

Pričujoče delo opisuje le eno implementacijo ABML koncepta; za učenje neurejenih pravil. Naše prepričanje je, da bi se ta koncept izkazal za še bolj uporabnega v kompleksnejših domenah (npr. razumevanje teksta, grafov, zaporedij), kjer je prostor hipotez običajno še veliko večji in je brez uporabe predznanja praktično nemogoče izvesti uspešno učenje. Verjetno najbolj primerna metoda za učenje v teh domenah je ILP, saj omogoča opis kompleksnih relacij (rekurzivne relacije, relacije med primeri, itd.) v naučeni hipotezi. V doktorski disertaciji smo opisali nekaj idej za razvoj metode AB-ILP (argument based ILP). V nadaljnjem delu nameravamo te ideje implementirati in jih testirati na realnih domenah.

Bibliography

[ACE] ACE – Automatic Content Extraction, http://www.nist.gov/speech/tests/ace.

[ADZ+07] Noriaki Aoki, Janez Demšar, Blaž Zupan, Martin Možina, Ernesto A. Pretto, Jun Oda, Hiroshi Tanaka, Katsuhiko Sugimoto, Toshiharu Yoshioka, and Tsuguya Fukui. Predictive model for estimating risk of crush syndrome : a data mining approach. *The Journal of trauma, injury, infection, and critical care*, 62(4):940–945, 2007.

[AN07] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[AR03] Kevin D. Ashley and Edwina L. Rissland. Law, learning and representation. *Artificial Intelligence*, 150:17–58, 2003.

[Atk89] Kendall E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley and Sons, New York, 1989.

[BA03] Stefanie Brüninghaus and Kevin D. Ashley. Predicting the outcome of case-based legal arguments. In G. Sartor, editor, *Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 233–242, Edinburgh, United Kingdom, June 2003.

[BC91a] Trevor J. M. Bench-Capon. *Knowledge-based systems and legal applications*. Academic Press Professional, Inc., San Diego, CA, USA, 1991.

[BC91b] Trevor J.M. Bench-Capon. *Knowledge Based Systems and Legal Applications*, chapter Knowledge Based Systems Applied To Law: A Framework for Discussion, pages 329–342. Academic Press., 1991.

[BC93] Trevor J.M. Bench-Capon. Neural nets and open texture. In *Fourth International Conference on AI and Law*, pages 292–297, Amsterdam, 1993. ACM Press.

[BCC00] Trevor J.M. Bench-Capon and Frans Coenen. An experiment in discovering association rules in the legal domain. In *11th International Workshop on Database and Expert Systems Applications*, pages 1056–1060, Los Alamitos, 2000. IEEE Computer Society.

[BM04] Ivan Bratko and Martin Možina. Argumentation and machine learning. In: Deliverable 2.1 for the ASPIC project, 2004.

[BM05] Razvan Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems*, Vancouver, British Columbia, 2005.

[Boo89] John H. Boose. A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition*, 1(1):3–37, 1989.

[Bra01] Ivan Bratko. *PROLOG Programming for Artificial Intelligence. Third edition*. Addison-Wesley, Harlow, England, 2001.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Bri50] Glenn W. Brier. Verification of forecasts expressed in terms of probabilities. *Mon. Wea. Rev.*, 78, 1950.

[BS91] Wray Buntine and David Stirling. Interactive induction. *Machine intelligence*, 12:121–137, 1991.

[Bud01] Raluca Budiu. *The role of background knowledge in sentence processing*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA., 2001. Available as Technical Report No. CMU-CS-01-148.

[BZ04] Marko Bohanec and Blaž Zupan. A function-decomposition method for development of hierarchical multi-attribute decision models. *Decision Support Systems*, 36:215–233, 2004.

[CAL94] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, 1994.

[CB91] Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Machine Learning - Proceeding of the Fifth Europen Conference (EWSL-91)*, pages 151–163, Berlin, 1991.

[Ces90] Bojan Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149, 1990.

[Chk03] Timothy Chklovski. *Using Analogy to Acquire Commonsense Knowledge from Human Contributors*. PhD thesis, MIT Artificial Intelligence Laboratory, 2003.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cla88] Peter Clark. Representing arguments as background knowledge for constraining generalisation. In Derek Sleeman, editor, *Third European Working Session on Learning*, October 1988.

[CM93] Peter Clark and Stan Matwin. Using qualitative models to guide inductive learning. In *Proceedings of the tenth international machine learning conference*, 1993.

[CN89] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning Journal*, 4(3):261–283, 1989.

[Coh94] William W. Cohen. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68:303–366, 1994.

[Coh95] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[Col01] Stuart Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 1 edition, december 2001.

[Coo94] Nancy J. Cooke. Varieties of knowledge elicitation techniques. *Int. J. Hum.-Comput. Stud.*, 41(6):801–849, 1994.

[CRL00] Daniela V. Carbogim, David Robertson, and John Lee. Argument-based applications to knowledge engineering. *Knowl. Eng. Rev.*, 15(2):119–149, 2000.

[CS99] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *In Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 335–342. AAAI Press, 1999.

[dAT01] Pedro de Almeida and Luís Torgo. The use of domain knowledge in feature construction for financial time series prediction. In *Progress in Artificial Intelligence*, pages 29–39, 2001.

[DCP93] Sašo Džeroski, Bojan Cestnik, and Igor Petrovski. Using the m-estimate in rule induction. *Journal of Computing and Information Technology*, 1(1):37–46, 1993.

[Dem06] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[Die02] Thomas G. Dietterich. *The Handbook of Brain Theory and Neural Networks, Second Edition*, chapter Ensemble Learning, pages 405–409. The MIT Press, 2002.

[DKS08] Krzysztof Dembczyński, Wojciech Kotlowski, and Roman Slowiński. Maximum likelihood rule ensembles. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 224–231, New York, NY, USA, 2008. ACM.

[Dom99] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[Dun93] Ted E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[Dun95] Phan M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[DZ04] Janez Demšar and B. Zupan. Orange: From experimental machine learning to interactive data mining. White Paper [http://www.ailab.si/orange], Faculty of Computer and Information Science, University of Ljubljana, 2004.

[Edv95] Lilian Edvards. Modelling law using a feminist theoretical perspective. *Law, Computers and Artificial Intelligence*, 4:95–110, 1995.

[Fei84] Edward A. Feigenbaum. Knowledge engineering: the applied side of artificial intelligence. In *Proc. of a symposium on Computer culture: the scientific, intellectual, and social impact of the computer*, pages 91–107, New York, NY, USA, 1984. New York Academy of Sciences.

[Fei03] Edward A. Feigenbaum. Some challenges and grand challenges for computational intelligence. *Source Journal of the ACM*, 50(1):32–40, 2003.

[Fel98] Christiane Fellbaum. *WordNet. An Electronic Lexical Database*. MIT Press, 1998.

[FF05] Johannes Fürnkranz and Peter A. Flach. Roc 'n' rule learning – towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, January 2005.

[For86] Richard Forsyth. Machine learning for expert systems. *Machine Learning*, 1(1):3–37, 1986.

[FP08] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *Ann. Appl. Stat.*, 2(3):916–954, 2008.

[FPM$^+$] Paolo Frasconi, Andrea Passerini, Stephen Muggleton, , and Huma Lodhi. Declarative kernels. Technical Report RT 2/2004, Dipartimento di Sistemi e Informatica, Universit'a di Firenze, 2004.

[FR86] Richard Forsyth and Roy Rada. *Machine learning: applications in expert systems and information retrieval*. Halsted Press, New York, NY, USA, 1986.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[FT28] Ronald A. Fisher and Leonard H.C. Tippett. Limiting forms of the frequency distribution of the largest and smallest member of a sample. *Proc. Camb. Phil. Soc.*, 24:180–190, 1928.

[Fur97] Johannes Furnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–171, 1997.

[GC04a] Sergio A. Gomez and Carlos I. Chesnevar. Integrating defeasible argumentation and machine learning techniques. Technical report, Universidad Nacional del Sur, 2004.

[GC04b] Sergio A. Gomez and Carlos I. Chesnevar. Integrating defeasible argumentation with fuzzy art neural networks for pattern classification. *Journal of Computer Science and Technology*, 4(1):45–51, April 2004.

[GL54] Emil J. Gumbel and Julivs Lieblein. Some applications of extreme value models. *American Statistician*, 8(5):14–17, 1954.

[GLM07] Alexander Genkin, David Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[GLR07] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Relation extraction and the influence of automatic named entity recognition. *ACM Transactions on Speech and Language Processing*, 5(1), 2007.

[GRS02] Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Mining sequential patterns with regular expression constraints. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):530–552, 2002.

[GS00] Marga M. Groothius and Jörgen S. Svensson. Expert system support and juridical quality. In *Jurix*, pages 1–10, Amsterdam, 2000. IOS Press.

[Gum54] Emil J. Gumbel. Statistical theory of extreme values and some practical applications. *National Bureau of Standards Applied Mathematics Series (US Government Printing Office)*, 33, 1954.

[Gup60] Shanti S. Gupta. Order statistics from the gamma distribution. *Technometrics*, 2:243–262, 1960.

[HA04] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126, 2004.

[Har01] Frank E. Harrell. *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer, New York, 2001.

[HL00] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression, 2nd Edition*. Wiley-Interscience, 2000.

[Hoc88] Yosef Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75:800–803, 1988.

[Hol79] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[HSG04] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discoverying relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, Barcelona, Spain, 2004. Association for Computational Linguistics.

[Ioa05] John P. A. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2(8), 2005.

[JC00] David D. Jensen and Paul R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, March 2000.

[JG03] Benjamin Johnston and Guido Govenatori. Induction of defeasible logic theories in the legal domain. In *Ninth International Conference on AI and Law*, pages 204–213, Edinburgh, 2003. ACM Press.

[JMF99] Anil K Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[KK07] Igor Kononenko and Matjaž Kukar. *Machine learning and data mining : introduction to principles and algorithms*. Chichester: Horwood Publishing, 2007.

[Kon93] Igor Kononenko. Inductive and bayesian learning in medical diagnosis. *Applied Artificial Intelligence*, 7:317–337, 1993.

[LA94] David D. Lewis and William A.Gale. A sequential algorithm for training text classifiers. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[LD94] Nada Lavrac and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

[LFTW04] Nada Lavrac, Peter Flach, Ljupčo Todorovski, and Stefan Wrobel. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5:153–188, 2004.

[LFZ99] Nada Lavrač, Peter Flach, and Blaž Zupan. Rule evaluation measures: A unifying view. In Saša Džeroski and Peter Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 174–185, Bled, Slovenia, 1999.

[LHF03] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. In *Proceedings of the 16th European Conference on Machine Learning*, 2003.

[Lin04] Tony Lindgren. Methods for rule conflict resolution. In *In Proceedings of the 15th European Conference on Machine Learning (ECML-04)*, pages 262–273, Pisa, 2004. Springer.

[LS95] Pat Langley and Herbert A. Simon. Applications of machine learning and rule induction. *Commun. ACM*, 38(11):54–64, 1995.

[LTJ04] Martin H.C. Law, Alexander Topchy, and Anil K. Jain. Clustering with soft and group constraints. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 662–670. Springer Berlin, 2004.

[MA94] Gerogy L. Murphy and Paul D. Allopenna. The locus of knowledge effects in concept learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 19:203–222, 1994.

[MB08] Martin Možina and Ivan Bratko. Rectifying predictions of classifiers by local rules. In Johannes Fuernkranz and Arno Knobbe, editors, *From local patterns to global models*, pages 128–140, 2008.

[MDKZ04] Martin Možina, Janez Demsar, Michael W. Kattan, and Blaz Zupan. Nomograms for visualization of naive bayesian classifier. In *PKDD*, pages 337–348, 2004.

[MDvB06] Martin Možina, Janez Demšar, Jure Žabkar, and Ivan Bratko. Why is rule learning optimistic and how to correct it. In Johannes Fuernkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proceedings of 17th European Conference on Machine Learning (ECML 2006)*, pages 330–340, Berlin, 2006. Springer-Verlag.

[MGB09] Martin Možina, Claudio Giuliano, and Ivan Bratko. Arguments extracted from text in argument based machine learning. In *Proceedings of 1st Asia Conference on Intelligent Information and Database Systems*, 2009.

[MGK$^+$08] Martin Možina, Matej Guid, Jana Krivec, Aleksander Sadikov, and Ivan Bratko. Fighting knowledge acquisition bottleneck with argument based machine learning. In *Proceedings of 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 234–238, Patras, Greece, 2008. IOS Press.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), October 1997.

[MKGT06] Vikash K. Mansinghka, Charles Kemp, Thomas L. Griffiths, and Joshua B. Tenenbaum. Structured priors for structure learning. In *In Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI*. AUAI Press, 2006.

[MKKC86] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[MN91] nez Marlon Nú˙ The use of background knowledge in decision tree induction. *Mach. Learn.*, 6(3):231–250, 1991.

[MN98] Andrew K. McCallum and Kamal Nigam. Employing em in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 359–367. Morgan Kaufmann, 1998.

[MS00] Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In *Proc. of ECAI'2000*, pages 321–325, 2000.

[MUC] MUC - Message Understanding Conferences, http://cs.nyu.edu/faculty/grishman/muc6.html.

[MvB06] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based rule learning. In *Proceedings of 17th European Conference on Artificial Intelligence (ECAI 2006)*, Riva Del Garda, Italy, 2006. IOS Press.

[MvB07] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10/15):922–937, 2007.

[MvBC$^+$06] Martin Možina, Jure Žabkar, Trevor Bench-Capon, , and Ivan Bratko. Argument based machine learning applied to law. *Artificial Intelligence and Law*, 2006.

[NRA$^+$96] Claire Nedelléc, Celine Rouveirol, Hilde Adé, Francesco Bergadano, and Brigid Tausend. *Advances in Inductive Logice Programming*, chapter Declarative bias in ILP, pages 82–103. IOS Press, 1996.

[NS04] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 79–86. ACM Press, 2004.

[Paz91] Michael J. Pazzani. Influence of prior knowledge on concept acquisition: Experimental and computational results. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 17:416–432, 1991.

[PHL01] Jian Pei, Jiawei Han, and Laks V.S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *17th International Conference on Data Engineering (ICDE'01)*, page 433, 2001.

[PK92] Michael Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9(1):57 – 94, 1992.

[PMS97] M. Pazzani, Subramani Mani, and Rodman W. Shankle. Beyond concise and colorful: Learning intelligible rules. In *Third International Conference on Knowledge Discovery and Data Mining*, pages 235–238, Newport Beach, CA, 1997. AAAI Press.

[Pol92] John L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57:1–42, 1992.

[QCJ95a] Ross J. Quinlan and Mike R. Cameron-Jones. Induction of logic programs: Foil and related systems. *New Generation Comput*, 13(3&4):287–312, 1995.

[QCJ95b] Ross J. Quinlan and Mike R. Cameron-Jones. Oversearching and layered search in empirical learning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1019–1024, Montreal, Canada, August 1995.

[Qui86] Ross J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Qui93] Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann,San Diego, 1993.

[RdPB+83] J. Rousseauw, J. du Plessis, A. Benade, P.Jordann, J. Kotze, P.Jooste, and J. Ferreira. Coronary risk factor screening in three rural communities. *South African Medical Journal*, 64:430–436, 1983.

[Ric97] Henry S. Richardson. *Practical Reasoning about Final Ends.* Cambridge University Press, 1997.

[Ris78] Jorma Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

[RK06] Ulrich Rückert and Stefan Kramer. A statistical approach to rule learning. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 785–792, New York, NY, USA, 2006. ACM.

[RM01] Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (Second Edition).* Prentice Hall, 2003.

[RN04] Chris Reed and Timothy J. Norman. *Argumentation Machines: New Frontiers in Argument and Computation.* Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[Ros95] Jeremy Roschelle. Learning in interactive environments: Prior knowledge and new experience. In J. H. Falk and L. D. Dierking, editors, *Public institutions for personal learning: Establishing a research agenda*, pages 37–51, 1995.

[SC08] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1069–1078. ACL Press, 2008.

[SD05] Qiang Sun and Gerald DeJong. Explanation-augmented svm: an approach to incorporating domain knowledge into svm learning. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 864–871, New York, NY, USA, 2005. ACM.

[sem] Semeval-2007, http://nlp.cs.swarthmore.edu/semeval/.

[Set09] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[SG05] Mark Stevenson and Mark A. Greenwood. A semantic approach to ie pattern induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, page 379–386, Ann Arbour, MI, 2005.

[SL92] Guillermo R. Simari and Ronald P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53:125–157, 1992.

[SMG+06] Aleksander Sadikov, Martin Možina, Matej Guid, Jana Krivec, and Ivan Bratko. Automated chess tutor. In *Proceedings of the 5th International Conference on Computers and Games*, 2006.

[SOS92] Sebastian H. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, New York, NY, USA, 1992. ACM Press.

[SSSV98] Bernhard Sch olkopf, Patrice Simard, Alexander J. Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640–646, 1998.

[SVA97] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining associations rules with item constraints. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 67–73, 1997.

[SVLD92] Patrice S. Simard, Bernard Victorri, Yann Lecun, and John Denker. Tangent-prop - a formalism for specifying selected invariances in an adaptive network. In *Advances in Neural Information Processing Systems (NIPS) 4*, San Mateo, CA, 1992.

[TD97] Ljupčo Todorovski and Sašo Džeroski. Declarative bias in equation discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[TD01a] Ljupčo Todorovski and Sašo Džeroski. Theory revision in equation discovery. In *Discovery Science*, pages 389–400, 2001.

[TD01b] Ljupčo Todorovski and Sašo Džeroski. Using domain knowledge on population dynamics modeling for equation discovery. In *European Conference on Machine Learning*, pages 478–490, 2001.

[TFL00] Ljupčo Todorovski, Peter Flach, and Nada Lavrač. Predictive performance of weighted relative accuracy. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the 4th European Conference of Principles of Data Mining and Knowledge Discovery (PKDD-00)*, pages 255–264, Lyon, France, 2000.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

[TM93] Sebastian Thrun and Tom M. Mitchell. Integrating inductive neural network learning and explanation-based learning. In *Proceeding of the 1993 International Joint Conference on Artificial Intelligence*, 1993.

[Tou58] Stephen Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.

[TS94] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70(1-2):119–165, 1994.

[Vap95] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[vB03] Dorian Šuc and Ivan Bratko. Improving numerical prediction with qualitative constraints. In *ECML 2003 - 14th European Conference on Machine Learning*, Dubrovnik, Croatia, September 2003.

[vMVB06] Jure Žabkar, Martin Možina, Jerneja Videčnik, and Ivan Bratko. Argument based machine learning in a medical domain. In Paul E. Dunne and Trevor J.M. Bench-Capon, editors, *Proceedings of Computational Models of Argument (COMMA)*, pages 59–70, 2006.

[VR02] William N. Venables and Brian D. Ripley. *Modern Applied Statistics with S. Fourth Edition*. Springer, 2002.

[Vre97] Gerard A.W. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90:225–279, 1997.

[vVB04] Dorian Šuc, Daniel Vladušič, and Ivan Bratko. Qualitatively faithful quantitative prediction. *Artificial Intelligence*, 158(2):189 – 214, 2004.

[Wal06] Doulgas Walton. *Fundamentals of Critical Argumentation*. Cambridge University Press, 2006.

[Wat99] John Watson. *Secrets of Modern Chess Strategy*. Gambit Publications, 1999.

[WCRS01] Kiri Wagsta, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning*, pages 577–584, 2001.

[WI00] Sholom M. Weiss and Nitin Indurkhya. Lightweight rule induction. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1135 – 1142. Morgan Kaufmann, 2000.

[Wil45] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[WWZ99] Geoffrey I. Webb, Jason Wells, and Zijian Zheng. An experimental evaluation of integrating machine learning with knowledge acquisition. *Mach. Learn.*, 35(1):5–23, 1999.