Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 171 (2007) 922–937

Artificial Intelligence

www.elsevier.com/locate/artint

Argument based machine learning

Martin Možina [*], Jure Žabkar, Ivan Bratko

Faculty of Computer and Information Science, University of Ljubljana, Slovenia

Received 13 November 2006; received in revised form 3 April 2007; accepted 16 April 2007

Available online 29 April 2007

Abstract

We present a novel approach to machine learning, called ABML (argumentation based ML). This approach combines machine learning from examples with concepts from the field of argumentation. The idea is to provide expert's arguments, or reasons, for some of the learning examples. We require that the theory induced from the examples explains the examples in terms of the given reasons. 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. In this paper we realize the idea of ABML as rule learning. We implement ABCN2, an argument-based extension of the CN2 rule learning algorithm, conduct experiments and analyze its performance in comparison with the original CN2 algorithm.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Machine learning; Learning through arguments; Background knowledge; Knowledge intensive learning; Argumentation

1. Introduction

A fundamental problem of machine learning is dealing with large spaces of possible hypotheses. Usually, this problem is addressed either by biasing learning towards simpler hypotheses, i.e. applying Occam's razor, or by using experts' domain knowledge for constraining search. Domingos [9] argues that the Occam's razor approach is valid only if we honestly believe that the target phenomenon is simple, and therefore prefers the use of domain knowledge. Existing machine learning paradigms typically enable, at least in some limited form, the use of *general* domain knowledge, that is knowledge that holds for the whole domain. The problem with this is the difficulty that experts face when they try to articulate their global domain knowledge. In this paper, we present a novel approach for constraining search, that allows the use of "local" expert's knowledge. That is knowledge relating to specific situations, perhaps only valid for chosen learning examples rather than for the whole domain. In this approach, the domain expert explains some of the learning examples, and these examples are then, together with other examples, used in the learning algorithm. Explanation of examples is in terms of arguments for and against, therefore we call such examples argumented examples. The learning from such examples is called argument based machine learning (ABML), introduced by Bratko and Možina in [2].

[*] Corresponding author.
E-mail address: martin.mozina@fri.uni-lj.si (M. Možina).

ABML combines machine learning with some ideas from the field of argumentation [21]. Defeasible argumentation is a branch of artificial intelligence that analyzes processes of reasoning where arguments for and against a certain claim are produced and evaluated. A typical example of such reasoning is a law dispute at court, where plaintiff and defendant give arguments for their opposing claims, and at the end of the process the party with better arguments wins the case.

In this paper we propose a realization of ABML as an algorithm that learns if-then rules from argumented examples. In Section 2 we describe the motivation and basic ideas for argument based machine learning. In Section 3 we develop an actual implementation of argument based machine learning, i.e. argument based CN2 (ABCN2 for short) as an extension of the well-known rule learning algorithm of Clark and Niblett [7]. In Section 4, we demonstrate and analyze the advantages of ABCN2 over the original CN2 on several data sets and conduct, in Section 5, an experiment that shows the effect of unreliable, or noisy arguments on the quality of induced model.

This paper is a substantial extension of our earlier conference paper on the idea of AB enhancement of CN2 [16]. The extensions include the experiments with ABCN2 (except for the illustration with the zoo domain), extreme value correction of probability estimate in ABCN2 (an essential improvement applicable also in CN2), and an experimental study of the effect of errors in arguments on the success of learning.

It should be noted that argument based ML is not limited to CN2, nor to rule learning. Various standard ML techniques can be extended in a similar way to their argument-based variants, for example argument-based Naive Bayes, or support vector machines (SVM), or inductive logic programming (abbreviated AB-ILP). However, rule learning is the most natural framework for developing ABML techniques. In rule learning, both induced theories and arguments are represented in the same language, which is desirable. Also, rule learning has the desirable property of comprehensibility of induced theories, and the influence of user's arguments is explicitly visible in induced rules. In contrast to this, applying the ABML approach for example to SVM would not exhibit these two desirable properties. On the other hand, we would expect that the use of arguments would result in improvement in classification accuracy also in SVM, as well as in other ML methods.

2. Argument based machine learning: Problem statement and motivation

In this section we will formulate the learning problem for ABML, and contrast it to the usual ML problem. We will illustrate with examples the main ideas of how ABML works, and describe the motivation for this approach.

Usually the problem of learning from examples is stated as:

- Given examples
- Find a theory that is consistent with the examples

We will now extend this problem statement to that of ABML. In ABML, some of the learning examples are augmented by arguments, or reasons, or (partial) justifications according to principles of argumentation [21].

In this paper we consider ABML in the framework of attribute-value learning. Each example is specified by an attribute-value vector and the class to which the example belongs. In this setting, arguments are used to explain (or argue) why a certain learning example is in the class as given (arguments for the class are called *positive arguments*) or why it should not be (arguments against are called *negative arguments*). Examples that are accompanied with arguments are called *argumented examples*.

To illustrate the idea of argumented 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 three attributes: PaysRegularly (with possible values "yes" and "no"), Rich (possible values "yes" and "no") and HairColor ("black", "blond", ...). The class is CreditApproved (with possible values "yes" and "no"). Let there be three learning examples as shown in Table 1.

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

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

This rule looks good because it is short and it correctly covers two of the three 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.

Table 1
Learning examples for credit approval

Name	PaysRegularly	Rich	HairColor	CreditApproved
Mrs. Brown	no	yes	blond	yes
Mr. Grey	no	no	grey	no
Miss White	yes	no	blond	yes

With arguments, the learning problem statement changes to:

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

To illustrate what it means to "explain the examples using given arguments", consider again the data in Table 1 and assume that an expert gave an argument for Mrs. Brown: Mrs. Brown received credit because she is rich. Now consider again the rule above that all blond people receive credit. This rule correctly classifies Mrs. Brown, but it does not explain this classification in terms of the given argument for Mrs. Brown. The rule does not even mention Mrs. Brown's property in the argument, namely that she is rich. Therefore we say that this rule does not "AB-cover" Mrs. Brown, and an ABML algorithm has to induce another rule to this effect. Therefore an argument based rule learning algorithm would induce:

IF $Rich = yes$ THEN $CreditApproved = yes$

This rule explains Mrs. Brown example using the given argument: credit was approved because $Rich = yes$. We say that this rule AB-covers Mrs. Brown.

The main motivation for using arguments in learning lies in two expected advantages:

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

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. Furthermore, a reduced number of possible hypotheses also decreases the chances that the method will overfit. 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.

Argumented examples are a means for the domain expert to introduce his (partial) domain knowledge to the learning system prior to learning. The practical advantage of providing prior knowledge in the form of arguments about *individual* examples, in comparison with providing more general domain knowledge, is in that it is easier to explain concrete, specific examples than giving general theories. As arguments are reasons for a specific example, the expert does not have to ensure that the arguments are true for the whole domain. It suffices that they are true in the context of the given argumented example. Consider a possible expert's argument for disapproving credit to Mr. Grey: Mr. Grey did not receive credit because he does not pay regularly. This is a very reasonable argument, however it does not hold for the whole domain. Mrs. Brown, for instance, did receive credit although she does not pay regularly. An argument-based learning algorithm will then automatically generalize the context of an argumented example to the extent that preserves consistency with other learning examples.

3. Implementation

In this paper we introduce an algorithm, called ABCN2, for induction of rules in argument-based framework for machine learning. ABCN2, which stands for argument based CN2, is an extension of the well-known CN2 rule induction algorithm of Clark and Niblett [7]. We will start this section with a formal definition of argumented examples that are accepted by ABCN2, and continue with a detailed description of differences between the CN2 algorithm and its argument based counterpart ABCN2. At the end of this section we will describe a method that selects examples from data, which, if argumented, would most likely affect the learning of ABCN2.

It should be noted that "CN2" inside our implementation of ABCN2 stands for a state-of-the-art version of the original Clark and Niblett's CN2 algorithm, in which various improvements were added over the years. Details of these improvements are described in [17], which also gives experimental results comparing the improved CN2 with several other major ML methods. In these experiments, this improved CN2 (without the use of arguments) performed overall significantly better in terms of accuracy and ROC curves than the original CN2 and C4.5, and comparably to SVM and Naive Bayes.

3.1. 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 because *Reasons*

or

C 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 $r_1, \ldots, r_n$,

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

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* $X \geqslant x_i$*)* means that the value of attribute X of example being greater than (greater or equal to) x_i is the reason for class value.
- $X < x_i$ *(or* $X \leqslant x_i$*)* the opposite to $X > x_i$ $(X \geqslant x_i)$.
- $X >$ *(or* $X \geqslant$*)* "X is high"; similar to $X > x_i$ $(X \geqslant x_i)$, 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.
- $X <$ *(or* $X \leqslant$*)*; "X is low", the opposite of $X >$ $(X \geqslant)$.

For example, the expert's argument for approving credit to Mrs. Brown (see Table 1) can be: Mrs. Brown received credit because she is rich. A negative argument can be: Mrs. Brown received credit despite her not paying regularly. The Brown example would in our syntax be written as:

$((PaysRegularly = no, Rich = yes, HairColor = blond),$

$CreditApproved = yes,$

$\{CreditApproved = yes$ because $Rich = yes,$

$CreditApproved = yes$ despite $PaysRegularly = no\}).$

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

IF *Complex* THEN *Class*

where *Complex* is the conjunction of simple conditions, called *selectors*. For the purpose of this paper, a selector simply 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:

IF $PaysRegularly = no$ AND $HairColor = blond$ THEN $CreditApproved = yes$

The condition part of the rule is satisfied by the attribute values of Mrs. Brown 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 the forms: "$X = x_i$" or "$X > x_i$" or "$X < x_i$" ("$X \geqslant x_i$" or "$X \leqslant x_i$"), 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 "$X <$", "$X \leqslant$", "$X \geqslant$"), then the complex of the rule R needs to contain a selector "$X > x_i$" (or "$X < x_i$", "$X \geqslant x_i$", "$X \leqslant x_i$"). The threshold value x_i does not matter for consistency.

With the use of arguments in examples, the definition of a rule *covering* an example needs to be refined. In the standard definition, 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:

(1) All the 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 the negative arguments of E.

As an illustration of the differences between AB-covering and the usual definition of covering, consider again the Brown example with the argument that she received credit because she is rich and despite her not paying regularly. Now consider four rules:

R1: IF $HairColor = blond$ THEN $CreditApproved = yes$
R2: IF $PaysRegularly = no$ AND $HairColor = blond$ THEN $CreditApproved = yes$
R3: IF $PaysRegularly = no$ AND $Rich = yes$ THEN $CreditApproved = yes$
R4: IF $HairColor = blond$ AND $Rich = yes$ THEN $CreditApproved = yes$

All four rules cover the Brown example and have 100% accuracy on the data set from Table 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 Brown example, but this rule fails also because it is consistent with the negative argument ($PaysRegularly = no$). Rule 3 also fails due to the negative argument, although it is consistent with the positive argument. The last example AB-covers the Brown example.

3.2. ABCN2 algorithm

The CN2 algorithm [6,7] 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 in this paper 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 CN2ForOneClass 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 [6]. Essentially, CN2ForOneClass is a covering algorithm that covers the examples of the given class.

```
Procedure ABCN2ForOneClass(Examples ES, Class T)
Let RULE_LIST be an empty list.
Let AES be the set of examples that have arguments; $AES \subseteq ES$.
Find splits for arguments where threshold not specified (type $X >$, etc.).
Evaluate arguments (as if they were rules) of examples in AES and sort examples in AES according to the evaluations 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 RULELIST.
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.
```

Algorithm 1. Covering algorithm of ABCN2 algorithm that learns rules from examples ES for given class T.

3.2.1. ABCN2: Covering algorithm

The first requirement for ABML is that an induced hypothesis explains the argumented examples using given arguments. In rule learning, this means that for each argumented example, there needs to be at least one rule in the set of induced rules that AB-covers this example. This is achieved simply by replacing covering 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 1). 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. Arguments in the examples AES are then evaluated by the rule evaluation function[1] as if the arguments were rules of the form "IF argument THEN class". 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.

3.2.2. ABCN2: Search procedure

Algorithm 2 shows AB 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 a target class. In Algorithm 2 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 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.

[1] Rule evaluation function used is explained later in the paper.

```
Procedure ABFind_Best_Rule(Examples ES, Example E, Class T)
Let the set STAR contain positive arguments of E (written as complexes).
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
  {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.
```

Algorithm 2. Algorithm that finds best rule that AB-covers argumented example E. The "quality" of a complex is evaluated by user-defined evaluation function.

Specialize with selectors that are satisfied by argumented example. By this we ensure the coverage of the seed example by the induced rule.

Remove all complexes that are consistent with negative arguments. Again, best rule 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.

3.3. Rule evaluation and extreme value correction

An evaluation function is used to estimate the quality of a rule. The quality of a rule is a user-defined measure to estimate how well the rule will eventually work in classification. Generally, this measure should reflect the accuracy of the rule when classifying new examples. In the original CN2-unordered implementation [6], Laplace's rule of succession is used to estimate the probability of correct classification of new cases (that is, not learning examples) by the rule. Laplace's rule of succession states that if there are p successes from a total of n trials, the probability of success in $(n + 1)$st trial is $\frac{p+1}{n+2}$. Other implementations may use other formulas for probability estimation. For example, Džeroski et al. [10] use Cestnik's m-estimate [4].

The problem with Laplace's formula and other probability estimates is that rule learning algorithms choose the best hypothesis among many candidate hypotheses. Therefore the example set covered by the best looking rule is not really a random sample, which these formulas assume. Rather, this example set is the best among those that belong to many competing rules. This gives rise to optimistic estimates. This problem, also known as multiple-comparison problem in induction algorithms [13], is even worse in the case of ABCN2; rules, learned from argumented examples, were typically selected from less hypotheses than rules induced by standard CN2 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.

In [14] we developed a method called EVC (extreme value correction) that accounts for multiple comparisons and corrects otherwise optimistic evaluation measure. The "optimistic" measure—corrected with EVC—used in that paper was m-estimate [4] with *m* set to 2. We use the same technique for the evaluation of rules in ABCN2. All the experiments with ABCN2 in this paper were done with this method of rule evaluation.

3.4. Probabilistic covering and removing strategy

When EVC is applied, one must first estimate parameters of correction model (based on extreme value distribution). These parameters depend also on the size of the learning data set, and therefore we should re-estimate these parameters each time when a rule is induced and the covered examples are removed, because the learning set becomes smaller. Unfortunately, estimation of the parameters is time consuming, which would make the use of EVC with the standard CN2 covering procedure very inefficient. To evade the problem we propose the probabilistic covering strategy where the number of learning examples remains constant during the complete induction process.

Let *example.prob* (name *prob* is used as EVC m-estimate returns class probability) be the quality of the best rule covering *example* (where "quality" is EVC m-estimate of the probability of the class the rule predicts). If there are no rules covering *example* then *example.prob* equals the prior probability of the example's class. When a new *rule* is learned (let *rule.quality* be the quality of *rule*), the removing procedure updates all probabilities of covered examples as $example.prob = maximum(example.prob, rule.quality)$. We say that, when an example with *example.prob* becomes covered by a new *rule*, where *rule.quality* is higher than *example.prob*, then *rule improves* the probability of this example; or shorter: *rule* improves example. We call this probabilistic removal strategy.

There is still one problem that remains to be solved. The execution of procedure *ABFind_Best_Rule*, as defined in Algorithm 2, is not affected by such a removing strategy and would learn the same rule all over again. Thus, the following changes are necessary:

Selection of best complex (BEST_CPX) *BEST_CPX* can be updated only if the current complex improves at least one example. This must be added to the condition in the algorithm.

Selection of best rules from STAR In the original CN2 algorithm the best N rules are selected according to *rule.quality*. This heuristic fails in our case, as we seek a rule that has high quality *and* will improve at least one example. A better measure for selecting rules would be $rule.quality * P\_improve$, where *P_improve* is the probability that final (specialized) rule will improve at least half of currently covered examples by *rule*. With this measure, the algorithm will select rules with high quality and with high probability of improving some of the examples.

The problem, though, is to estimate *P_improve*. Naturally, computing it accurately is impossible, as we do not know which rules can still be specialized. We need to make a rough estimate, and we do it as follows. Let *bestrule* be the best possible rule that can (theoretically) be obtained from *rule*, and let *aveg_prob* be the average of *example.prob* over all covered examples by this rule. Then, if *aveg_prob* is between *rule.quality* and *bestrule.quality*, *P_improve* is computed as:

$$P\_improve = \frac{bestrule.quality - aveg\_prob}{bestrule.quality - rule.quality}.$$

This formula assumes that all the qualities between current rule and the best rule are equally probable. If *aveg_prob* is higher than *bestrule.quality*, then *P_improve* is 0.0, and if *aveg_prob* is lower than *rule.quality*, then *P_improve* is 1.0.

3.5. Classification from rules

Learned rules can also be used to predict the class of a new example. In the case where only one rule triggers for this example, classification is simply the class of the rule. In cases where several rules trigger, we need to resolve clashes between opposing rules. In standard CN2 these conflicts are resolved by summing the distributions of covered examples of all the rules to find the most probable class. However, such classification works only in cases where covering rules are sufficiently independent, which turns out to be false in many cases. Moreover, such a classification technique does consider only the distribution of covered examples, without considering the rule's quality, that becomes a problem when classifying from rules that were induced from arguments. The patterns (rules) learned from argumented examples usually cannot be learned by classical CN2, sometimes because there are not enough learning examples. Hence, rules from arguments might have relatively small coverage and might become relatively unimportant in such a classification technique. On the other hand, we showed in the previous section how to compute the quality of a rule that accounts for the number of tried hypotheses. In order to make rules from argumented examples competitive, we used in our experiments a simple classification technique based on the quality of rules: we take the best rule (having highest quality) for each class covering the given example, and classify the example in the class predicted by the best rule. Note that this classification can also be used to give probabilistic class predictions by normalizing the qualities of the best rules so that they sum to 1.

3.6. Identifying critical examples

Giving arguments to all examples is not likely to be feasible in practice because it would require too much effort by a domain expert who provides arguments. Can we help the expert by suggesting which examples to explain by arguments? We developed a method to automatically find "problematic" examples, that is examples where arguments would be likely to have a significant effect on learning. So the "commentator" of examples (expert) is asked to concentrate on these "problematic" cases. This idea is realized by the following iterative procedure:

(1) *Induce a theory from plain examples, without arguments* (*using classical CN2*).
(2) *Find a critical example that would be useful to be argumented*. This step involves a search for the most problematic example (e.g. outlier) in the learning set. For this task we propose a k-fold cross-validation repeated n times (e.g. $n = 4$, $k = 5$), so that each example is tested n times. Test examples are then evaluated according to a chosen criterion (e.g. the number of misclassifications of the example, average squared error of probability estimation, etc.) In our tests we used the number of misclassifications, thus the example that was misclassified in most of the cases is chosen as the example that needs to be argumented. If there are several such examples, then the algorithm picks one at random.
(3) *If a problematic example was not found* (*in step* 2)*, then stop the iteration*.
(4) *An expert gives arguments to the selected example*. Two things can happen here: in the preferred case the expert finds argumenting this example easy; in the undesired case, the expert finds this to be hard. The second case may be due to different reasons (deciding attributes are not available, the example is indeed an outlier, or argumenting may be hard in the chosen representation of arguments). Each of these cases can be mended in different ways,which still needs to be explored. In the extreme, this example can simply be discarded from the learning set.
(5) *Induce rules on the learning set using ABCN2 and new argumented example*.
(6) *Return to step* 2.

4. Experiments with ABCN2

In this section we will present three experiments with ABCN2 on "ZOO", "Credit Screening", and "South African Heart disease" UCI domains [18] to illustrate how ABCN2 works in practice. Then we describe two larger applications of ABCN2: an application to a legal domain and an application to a medical domain. In the experiments we used an implementation of ABCN2 within the Orange-toolkit [8].

4.1. ZOO data set

ZOO data set [18] 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.

The set was split into a learning set (70%) and a test set (30%). Without arguments (classical CN2), the following rules were induced:

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

Considering learning data alone, the induced rules fit perfectly. However, classification accuracy on the test set is only slightly over 90%. Now, according to our method of involving argumented examples (Section 3.6), 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 fishes and the second from birds, as both, fishes 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 learning 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.

4.2. 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 learning set (70%) and test set (30%). CN2 induced the following three rules for not receiving credit.

```
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
```

These three rules 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". After 5 iterations, when remaining problematic examples could not be argumented any more, the following rules were induced:

```
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
```

While performing this experiment we noticed several interesting things:

- *Efficient reconstruction of expert's prior knowledge*. The first five and the last rule in the above set correspond to complete background knowledge given by experts. This was achieved by asking our "expert" to explain only five examples, which indicates the effectiveness of our method for detecting problematic examples. This also indicates how effective the interactive use of ABCN2 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 critical cases would, in our example, be sufficient to completely formalize the expert's prior intuitions.
- *Consistent rules with domain theory*. Six of seven rules are consistent with the given prior knowledge. Pazzani [19,20] argued that induced models are more comprehensible if they are consistent with prior knowledge, even if this makes them more complex. Accordingly, the latter set of rules should be more comprehensible to an expert.
- *Improved classification accuracy*. The accuracy 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.
- *Better than domain theory itself*. Due to 6th rule, induced rule set classified correctly two additional examples in the test set.

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.

Similar to other experiments with ABCN2, this experiment also showed how given prior knowledge can be used to automatically improve the accuracy of induced models. A similar learning problem—learning about credit status—occurred within the European 6th framework project ASPIC.[2] Learning about credit status is a part of a larger B2B (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 data set consisted of 5000 companies described by 18 attributes (three of them were actually relevant for credit status). We began the experiment with the induction of rules from 2500 examples (learning set) without considering any prior knowledge. The system induced a set of 40 rules. These rules correctly classified 95% of the examples in the test set (the remaining 2500 examples). After adding arguments to two problematic examples, ABCN2 induced six rules only, and the classification accuracy of these rules on the test set was 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.

4.3. South Africa heart-disease domain

South Africa heart-disease domain (*SAHeart*) [22] 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 about this domain for specifying argumented examples. So the main difference between this experiment and the previous two is that our knowledge about the domain is superficial (might be also wrong occasionally). However, despite our simple, imperfect domain knowledge, arguments still had a positive impact on the results.

We began our experiment with plain CN2 which achieved $67.5 \pm 1.0\%$ classification accuracy estimated with 10-times cross-validation. Afterwards we used our technique to find five problematic examples:

(1) Problematic patient 1 was a 49 years old smoker. We deemed these two attributes as important, therefore our argument was: chd=yes because age>45 and tobacco>0.
(2) The argument for a second patient was: chd=yes because age>45 and family_history=present and adiposity> (adiposity is high). We used qualitative format for adiposity, as we were not sure when does adiposity become dangerous for health.
(3) The argument for a patient was: chd=yes because age>45 and spb>140 (patient is older than 45 and has systolic blood pressure higher than 140).
(4) The argument for a fourth patient: chd=yes because spb>200 (systolic blood pressure is extremely high, higher than 200).
(5) A fifth patient: chd=yes because age>45 and family_history=present and
cholesterol> (low density lipoprotein cholesterol is high). Again, we were not sure when does cholesterol become dangerous.

Learning on the same data set and using above arguments ABCN2 achieved $70.6 \pm 1.5\%$ classification accuracy, which is a statistically significant improvement. Moreover, ABCN2 scored better also on other measures of quality (AUC and Brier Score).

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.

4.4. A legal application

In many areas of law—especially administrative law—many thousands of cases are routinely decided. Often these cases involve the exercise of some discretion, or involve some degree of operationalization of the concepts used in

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

the legislation, so that the decision can be based on ascertainable facts rather than the vaguer terms in which the legislation may be couched. In [15], the question was investigated: Can the rules being followed be reconstructed by ML from past cases? In particular, ABCN2 was applied to data concerning a fictional welfare benefit. For this experiment a data set of 2400 cases was used. Each case was described in terms of 64 attributes, twelve attributes being relevant to the decision, and the remaining 52 were irrelevant attributes. The same data set had been used in previous experiments with ML in a legal domain. ABCN2 performed at least as well as the others in terms of accuracy, which was very high indeed. However, it is not the accuracy of classification that is considered to be most important to meet the requirements of ML application in law: it is the quality, in terms of interpretability, of the rules generated that is crucial. In the initial learning stage, without arguments, the performance of ABCN2 was similar to that of the others. After the inclusion of domain knowledge taken from the justification of particular decisions, we were able to give a great improvement to the quality of the rules. Additionally, the robustness of ABCN2 in the face of erroneous decisions was explored, that is noise in learning data. It was found that ABCN2 is much more robust against noise than original CN2. This is important because many of the administrative law applications to which the technique could be applied exhibit quite large error rates.

4.5. A medical application

In [23] an application of ABCN2 to a medical domain is described. The medical problem there was severe bacterial infections in geriatric population. This problem is quite significant as elderly population, people over 65 years of age, is rapidly growing as well as the costs of treating this population. The data for this study was gathered at the Clinic for Infectious Diseases in Ljubljana. 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 goal was to build a model from data which would help the physician, at the first examination of the patient, to determine the severity of the infection, and consequently, whether the patient should be admitted to hospital or could be treated as an outpatient. A further goal was to obtain an understandable model, not a black box, to see which parameters play a decisive role.

Using 10×10 cross validation, CN2 achieved 87.9% classification accuracy, AUC was 0.810 and Brier Score was 0.175. In this experiment we used a more sophisticated classification based on minimax principle described in [23] and [17]. The performance of CN2 was compared also to that of C4.5, Naive Bayes and Logistic Regression, while CN2 was better than all other methods except for C4.5's classification accuracy, which was 88.3%.

The argumentation was done by the physician who was treating the patients and could give the reasons she believed caused death for 32 examples of death cases (class value *death*=*yes*). The physicians do not have perfect knowledge about this domain, therefore the given arguments were very likely not completely correct (as they were in the first two experiments). Note that we did not use the method for finding critical examples here. Using ABCN2 on argumented data set significantly outperformed all other learning methods tried according to the criteria AUC and Brier Score. According to classification accuracy, it was also significantly better than CN2, Naive Bayes and Logistic Regression, and equal to C4.5.

ABCN2 achieved better results than CN2 according to all three measures by using the arguments given by the expert. The question was studied why and how ABCN2 outperformed CN2 in the comparison. As the arguments were only given 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 restrictions on 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 *thrombocytes* < 100 AND mobility $=$ no THEN death $=$ yes

was present in both rule sets. It says that if the patient has low thrombocytes and is not mobile then he will die. This rule covers 6 examples with class value *death*=*yes* and 1 with *death*=*no* in the learning set, which means that the relative frequency of *death*=*yes* is $6/7 = 0.86$. However, the evaluation function based on extreme value distributions [14] used in CN2 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 thrombocytes* < 100’. The search space in ABCN2 is smaller, which means that the probability of finding such a rule by chance is lower. So, the estimated quality of the rule is higher.

5. Can imperfect arguments be damaging?

It may well happen that the expert, when argumenting examples, will make mistakes and supply imperfect arguments (which probably did happen in both medical domains described above). We now turn to the question, how critically does ABCN2 depend 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 section 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 [18], comparing CN2 with ABCN2 using 2, 5, 10, or 20 *randomly* argumented examples. The examples that were argumented were selected randomly. Each argumented example can have up to five *random* positive arguments, and each argument can have up to five *random* reasons.

The results are shown in Table 2. 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. 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.

6. Related work

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 [1,3]. Gomez and Chesñevar suggested in their report [11] 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 [12]. Their method is applied after the actual learning is already finished. Clark [5] 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 our paper arguments apply to individual examples.

7. Conclusion

In this paper we have described a novel approach to machine learning that draws on some concepts of argumentation theory. We introduced a new type of examples, argumented examples. Arguments are used to constrain the search among possible hypotheses.

We implemented ABCN2, an argument based extension of CN2. We have shown several advantages of argument based approach to learning rules:

Table 2
Classification accuracy and AUC of ABCN2 on several UCI data sets with different number of randomly argumented examples

Dataset↓ #rand.arg.→	CA					AUC				
	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

- 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.
- 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 (ZOO, Credit data, South African heart disease domain, legal domain, and infections domain), ABCN2 achieved higher classification accuracy 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.

The number of experimental domains in our experiments with ABCN2 is relatively low in comparison with typical experimental work in machine learning, so further experiments with the method will be useful. It should be noted however that experimenting with argument-based learning methods is more demanding than the usual experimentation in machine learning using e.g. UCI data sets. ABML requires the involvement of an expert, or at least some other source of knowledge, to provide arguments and assess the results of learning from the semantic point of view.

The extreme value correction in probability estimates is a substantial improvement for rule learning. However, it comes at computational cost. The 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.

One aspect that deserves investigation is related to the appropriateness of expert-supplied arguments. One problem that has been noticed occurs in cases when the expert is overly diligent and provides very specific arguments. These may occasionally prevent ABCN2 from inducing good simpler theories. Another point that may be improved concerns our method of selecting critical examples that are suggested to the expert for explanation. The method selects the most frequently misclassified example. Although this has worked well in the experiments, we feel that the method is debatable and may possibly be improved. The present selection method does not take into account how natural the selected example will be to be explained by the expert in terms of arguments. The selected example may be an outlier which will be troublesome to explain, and resulting arguments may not be useful for learning.

Acknowledgements

This work was carried out under the auspices of the European Commission's Information Society Technologies (IST) programme, through Project ASPIC (IST-FP6-002307). It was also supported by the Slovenian research agency ARRS.

References

[1] K.D. Ashley, E.L. Rissland, Law, learning and representation, Artificial Intelligence 150 (2003) 17–58.
[2] I. Bratko, M. Možina, Argumentation and machine learning, in: Deliverable 2.1 for the ASPIC project, 2004.
[3] S. Brüninghaus, K.D. Ashley, Predicting the outcome of case-based legal arguments, in: G. Sartor (Ed.), Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL), Edinburgh, United Kingdom, June 2003, pp. 233–242.
[4] B. Cestnik, Estimating probabilities: A crucial task in machine learning, in: Proceedings of the Ninth European Conference on Artificial Intelligence, 1990, pp. 147–149.
[5] P. Clark, Representing arguments as background knowledge for constraining generalisation, in: D. Sleeman (Ed.), Third European Working Session on Learning, October 1988.
[6] P. Clark, R. Boswell, Rule induction with CN2: Some recent improvements, in: Machine Learning—Proceedings of the Fifth European Conference (EWSL-91), Berlin, 1991, pp. 151–163.
[7] P. Clark, T. Niblett, The CN2 induction algorithm, Machine Learning Journal 4 (3) (1989) 261–283.
[8] J. Demšar, 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.
[9] P. Domingos, The role of Occam's razor in knowledge discovery, Data Mining and Knowledge Discovery 3 (4) (1999) 409–425.
[10] S. Džeroski, B. Cestnik, I. Petrovski, Using the m-estimate in rule induction, CIT J. Comput. Inf. Technol. 1 (1993) 37–46.
[11] S.A. Gomez, C.I. Chesnevar, Integrating defeasible argumentation and machine learning techniques, Technical report, Universidad Nacional del Sur, 2004.
[12] S.A. Gomez, C.I. Chesnevar, Integrating defeasible argumentation with fuzzy art neural networks for pattern classification, Journal of Computer Science and Technology 4 (1) (April 2004) 45–51.
[13] D.D. Jensen, P.R. Cohen, Multiple comparisons in induction algorithms, Machine Learning 38 (3) (March 2000) 309–338.
[14] M. Možina, J. Demšar, J. Žabkar, I. Bratko, Why is rule learning optimistic and how to correct it, in: J. Fuernkranz, T. Scheffer, M. Spiliopoulou (Eds.), Proceedings of 17th European Conference on Machine Learning (ECML 2006), Berlin, Springer-Verlag, 2006, pp. 330–340.
[15] M. Možina, J. Žabkar, T. Bench-Capon, I. Bratko, Argument based machine learning applied to law, Artificial Intelligence and Law 13 (1) (2006) 53–73.
[16] M. Možina, J. Žabkar, I. Bratko, Argument based rule learning, in: Proceedings of 17th European Conference on Artificial Intelligence (ECAI 2006), Riva Del Garda, Italy, IOS Press, 2006.
[17] M. Možina, J. Žabkar, I. Bratko, D3.4: Implementation of and experiments with abml and mlba, ASPIC Deliverable D3.4, 2006.
[18] P.M. Murphy, D.W. Aha, UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/mlrepository.html, Irvine, CA: University of California, Department of Information and Computer Science, 1994.
[19] M. Pazzani, S. Mani, W.R. Shankle, Beyond concise and colorful: Learning intelligible rules, in: Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, AAAI Press, 1997, pp. 235–238.
[20] M.J. Pazzani, Influence of prior knowledge on concept acquisition: Experimental and computational results, Journal of Experimental Psychology: Learning, Memory and Cognition 17 (1991) 416–432.
[21] H. Prakken, G. Vreeswijk, Logics for defeasible argumentation, in: Handbook of Philosophical Logic, vol. 4, second ed., Kluwer Academic Publishers, Dordrecht, 2002, pp. 218–319.
[22] J. Rousseauw, J. du Plessis, A. Benade, P. Jordann, J. Kotze, P. Jooste, J. Ferreira, Coronary risk factor screening in three rural communities, South African Medical Journal 64 (1983) 430–436.
[23] J. Žabkar, M. Možina, J. Videčnik, I. Bratko, Argument based machine learning in a medical domain, in: E.P. Dunne, T.J.M. Bench-Capon (Eds.), Proceedings of Computational Models of Argument (COMMA), 2006, pp. 59–70.