Learning to Explain with ABML

Martin Možina, Matej Guid, Jana Krivec, Aleksander Sadikov, and Ivan Bratko

Faculty of Computer and Information Science, University of Ljubljana, Slovenia,
martin.mozina@fri.uni-lj.si

Abstract. The main advantage of machine learning algorithms that learn simple symbolic models is in their capability to trivially provide justifications for their decisions. However, there is no guarantee that these justifications will be understood by experts and other users. Induced models are often strange to the domain experts as they understand the problem in a different way. We suggest the use of argument based machine learning (ABML) to deal with this problem. This approach combines machine learning with explanations provided by domain experts. An ABML method is required to learn a model that correctly predicts learning examples and is consistent with the provided explanations. The present paper describes an application of ABML to learning a complex chess concept of an attack on the castled king. The explanation power of the learned model for this concept is especially important, as it will be used in a chess tutoring application.

Keywords: machine learning, argument based machine learning, learning to explain, chess

1 Introduction

The common weakness of state-of-the-art machine learning algorithms achieving the best results in terms of accuracy (e.g. ensemble methods or SVMs) is their inability to explain their predictions. Domain experts and users alike perceive these methods and their predictions as black boxes. However, machine learning is often used to get a better understanding of the relation between inputs and outputs, especially so if machine learning is used as a knowledge acquisition tool [3]. In such cases, it is usually preferable to use methods like decision trees, rule-based models, or linear models that produce hypotheses that are mostly easily understood and interpreted.

Yet, even easily understood models are not always enough. The induced correlations can often seem illogical or simply strange to the domain experts as they would explain the same case using different terms. Pazzani [7] showed experimentally that people will grasp a new concept more easily if the concept is consistent with their knowledge. More elaborate studies of understanding new concepts were produced by the cognitive learning community [10]. They showed that when we learn about new presented materials, we always start off with our prior knowledge and try to merge them. If the new concepts are inconsistent with our prior knowledge, the new knowledge will likely be distorted or even rejected.

A recently developed approach called Argument Based Machine Learning (ABML) [6] is an extension of classical machine learning (ML) that allows the use of experts' explanations provided in the form of arguments for particular learning examples. An ABML method is required to learn a model that correctly predicts learning examples and is consistent with the provided explanations. Therefore, the explanations direct the search towards models that are more comprehensible in the light of experts' background knowledge. The benefit is twofold: (a) new knowledge is assimilated more easily, and (b) the accuracy of the model is usually increased.

We are interested in building systems for automated commentary and intelligent tutoring. The main characteristic of such systems is that their knowledge has to facilitate good explanation. In the present paper we present part of our work targeted at creating a program for automated commentary of chess games. The current chess software is terribly lacking in this respect; the chess-playing programs have long ago surpassed the human world-champion level of play, yet, they are completely unsuitable for providing commentary [11]. This is to be expected as they were designed for the former, and their language, so to speak, is not intended for the latter. The situation is in many ways reminiscent of the difference between SVMs and decision trees or rules as described in the first paragraph.

The paper is organized as follows. We first briefly introduce the ABML paradigm, and then we present its application for learning a difficult concept that will be featured in automated commentary. Afterwards, the results of both machine learning and ABML are presented and compared to one another, giving special consideration to the ability of both paradigms to give explanations for their decisions. Last section contains conclusions and specifies some necessary future work.

2 Argument Based Machine Learning

Argument Based Machine Learning (ABML) [6] is machine learning extended with some concepts from argumentation. In this section, we give a short summary of ABML taken from [6].

Argumentation is a branch of artificial intelligence that analyzes reasoning where arguments for and against a certain claim are produced and evaluated [8]. 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.

Arguments are used in ABML to enhance learning examples. Each argument is attached to a single learning example only, while one example can have several arguments. There are two types of arguments; positive arguments are used to explain (or argue) why a certain learning example is in the class as given, and negative arguments are used to explain why it should not be in the class as given.

We used only positive arguments in this work, as negatives were not required. Examples with attached arguments are called *argumented examples*.

Arguments are usually provided by domain experts, who find it natural to articulate their knowledge in this manner. While it is generally accepted that giving domain knowledge usually poses a problem, in ABML they need to focus on one specific case only at a time and provide knowledge that seems relevant for this case and does not have to be valid for the whole domain. The idea can be easily illustrated with the task of commenting chess games. It would be hard to talk about chess moves in general to decide precisely when they are good or bad. However, if an expert is asked to comment on a particular move in a given position, he or she will be able to offer an explanation and provide relevant elements of this position. Naturally, in a new position the same argument could be incorrect.

An ABML method is required to induce a theory that uses given arguments to explain the examples. 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. If an ABML method is used on normal examples only (without arguments), then it should act the same as a normal machine learning method.

In this paper, we will use a modified version of the ABCN2 [6] method (the modification is described in the following section), which is an argument based extension of the well known method CN2 [2]. ABCN2 learns a set of unordered probabilistic rules from argumented examples. In ABCN2, the theory (a set of rules) is said to explain the examples using given arguments, when there exists at least one rule for each argumented example that contains at least one positive argument in the condition part. This definition is a bit simplified, since it omits the use of negative arguments, as they are not relevant for this paper[1].

2.1 Interactions between expert and ABML

In ABML, experts are asked to provide their prior knowledge in the form of arguments for the learning examples rather than the general domain knowledge. However, asking experts to give arguments to 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 or changes the class of the example; the explanation is encoded in arguments and attached to the learning example.
4. Return to step 1.

[1] Due to space limitations, we will only roughly describe some of the mechanisms of ABML (see [6] or/and its website *www.ailab.si/martin/abml* for precise details).

To finalise 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?

Identifying critical examples The main property of critical examples is that the current hypothesis can not explain them 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. We use a 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.

Are expert's arguments good or should they be improved? Here we describe in details the third (3) step of the above algorithm, where the expert is asked to explain the critical example. First, the expert considers the class of the example whether it is incorrectly labelled. In such a case, the class value of the example is changed. Otherwise, the expert explains the example with arguments. Using expert's arguments, ABML will sometimes be able to explain the critical example, while sometimes this will still not be entirely possible. In such cases, we need additional information from 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?" The answer can be either "I don't know" (the expert is unable to explain the example) or a set of arguments $A_1, \ldots, A_k$ all confirming the example's class value can be given. If the system gets the answer "don't know", it will stop this procedure and try to find another critical example.

Step 2: Adding arguments to 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 is simply added to the 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 as new attributes before the argument can be added to the example.

Step 3: Discovering counter examples. Counter examples are used to spot if arguments suffice to successfully explain the critical example or not. If ABML fails to explain the example, then the counter examples will show where the problem is. Here, ABML is first used to induce a hypothesis H_1 using previous learning data only and H_2 using learning data together with 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 expert needs to revise the 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 critical example in one class and why counter example in the other?" The answer is added to the initial argument.

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

3 Case Study: Experimental Setup

Our goal was to formalize the concept of *an attack on the castled king* for the purposes of generating automatic commentary. That is, the computer should be able to reliably judge whether and why a particular side in a chess game aims at attacking the opponent's king that has already castled.

The concept of attack on the castled king can be considered as a positional concept of high complexity. There is some general agreement in the chess literature and among chess players about the intuition behind this concept. However, small details may prevail in an assessment whether particular side has chances to generate pressure against the opponent's king, and many factors can potentially influence such an assessment (see Fig. 1). Moreover, chess masters usually also rely on *dynamic* factors in chess positions when estimating chances of attacking successfully, while our concept should be *static* in its nature for its ultimate purpose of annotating chess games (see [4] for details). Our formalized model, however, does not aim at correctly estimating chances of success when attacking, but merely to estimate whether the attack is possible at all.

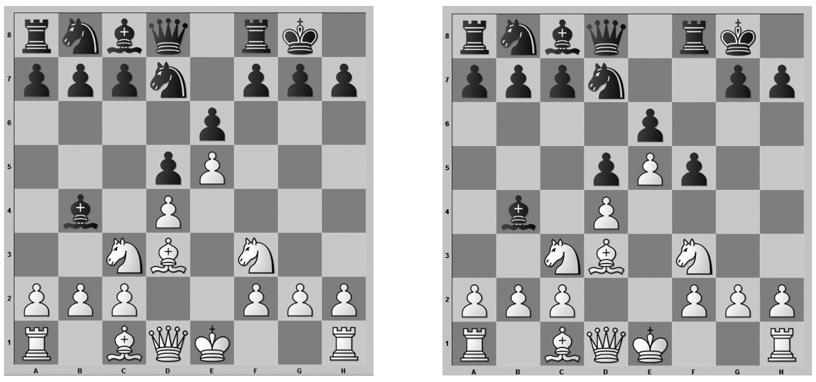

Fig. 1. In the position in the left-hand diagram White-to-move gains a strong attack by applying the classic bishop sacrifice: 1.Bd3xh7+ Kg8xh7 2.Nf3-g5+. However, by only the f7 pawn being on f5 square as shown in the right-hand diagram, White does not have any pressure against the opponent's king at all.

3.1 Data set description

The positions in the data set were gathered from the Chess Informant database (see *http://www.sahovski.com*). Our chess experts (woman grandmaster Jana Krivec and FIDE master Matej Guid) selected 577 middlegame positions from real chess games where the black player has castled on the king side and white still has a queen on the board. They classified each position either as *attack* (white has chances for an attack) or *no attack* (white cannot or is not planning to attack the black king). Of the 577 positions, 350 were labelled as *attack* and 228 as *no attack*. We then randomly selected 277 positions for learning and 278 for testing; stratification was used to preserve class distributions in both subsets.

Initially, the chess experts carefully selected twelve attributes they considered useful for identifying an attack on the king in a chess position. Each of these attributes corresponds to a well-known chess position feature that is regularly considered by chess players. Some examples of such attributes are: *pawn_attack* (number of white pawns on files f,g, and h that progressed to fourth row or higher), *pawn_defense* (number of black pawns protecting the black king), *close_combat_pieces* (number of white pieces in the area around the black king), etc. As will become evident later, this set of descriptive attributes was significantly enlarged during the interaction between the expert and the computer.

3.2 Problem definition

Our goal is to learn a set of rules for classifying a position with respect to whether the white player is attacking the black king or not. Rules will be evaluated according to the following criteria:

Classification accuracy: the usual measure in machine learning for evaluating goodness of a learned model.

Precision and recall: besides classification accuracy, we aim at high precision, since in our tutoring application, the cost of misclassifying an example without attack as *attack* is much higher as misclassifying an example with attack as *no attack*.

Explanation power of rules: it is not the accuracy of classification that is most important to meet the requirements stated in the introduction to our application: it is the quality, in terms of interpretability, of the rules generated that is crucial, as we aim at providing good commentary.

Since our main intention is to use these rules to explain why there is an attack in a position, we decided to learn only rules for the class *attack*. However, to avoid low precision with such rules, we modified the ABCN2 algorithm to induce only rules with pure class distributions, namely, a rule can cover only positions with an attack and should not cover any positions without an attack. Furthermore, the procedure for selecting critical examples in the ABML refinement loop had to be changed to return only positions with an attack. If a negative position (without attack) became critical — which is possible, as we use internal cross-validation — the critical example became a positive position covered by the rule that misclassified the negative example.

4 Case study: experiments and results

The first set of rules was generated from examples without any arguments. ABCN2 altogether induced 12 rules achieving 84% classification accuracy, 87% precision, and 86% recall. The high accuracy, precision and recall imply that the initial set of attributes was relatively well-defined. Afterwards, we began with the ABML refinement loop. In the remainder of this section we will present a single iteration of the loop to illustrate the interaction between the experts and the algorithm and then, at the end, provide results of the final model.

Figure 2 shows one of the critical examples, automatically selected by our algorithm.

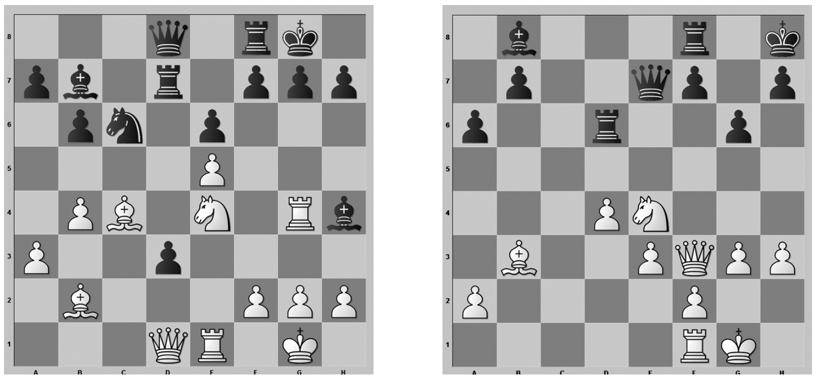

Fig. 2. Computer: *"Why is it possible for White to attack the opponent's king?"* (left) *But why is it NOT possible for White to attack the opponent's king here?* (right)

Current rules at the time failed to classify the example on the left-hand diagram as *attack*, as it was previously judged by the experts. The following question was given to the experts: *"Why is it possible for White to attack the opponent's king?"* The experts used their domain knowledge to give the following arguments: *"White rook is placed on a half-open file in front of the castled position of the black king and there are several white pieces joining the attack."* Based on the experts' explanation, the following argument was attached to this critical position:

- "*open_file* is *true*" (white queen or white rook are placed on an open or half-open file *f*, *g* or *h*),
- "*attacking_pieces* is high" (number of white pieces that attack one of the following squares: *f6*, *g6*, *h6*, *f7*, *g7*, *h7*, *g8* or *h8*).

Note that both attributes were already included into the domain at the time. The following rule covering the critical example was produced:

```
IF open_file is true AND attacking_pieces > 1
   AND defends_around_king <=10
THEN ATTACK = true;
```

The method automatically found additional restrictions to improve the experts' explanation. The attribute *defends_around_king* expresses the number of squares in the square f5-h5-h7-f7 defended by black pieces including black pawns.

Afterwards, the algorithm found a counter-example to this argument shown on the right-hand diagram in Figure 2. Now the question to the experts was: *But why is it NOT possible for White to attack the opponent's king here?* The experts were asked to compare this position to the position in the critical example. They gave the following answer: *"White queen indeed is placed on a half-open file in front of the castled position of the black king, but here it just does not seem so strong. In general, having a white queen or a white rook placed on an open or half-open file g or h is stronger than on an open or half-open file f."*

Previously, the attribute *open_file* indicated an open file f, g, or h with a strong piece on it. During this step, it was changed so that only open or half-open files g or h counted.

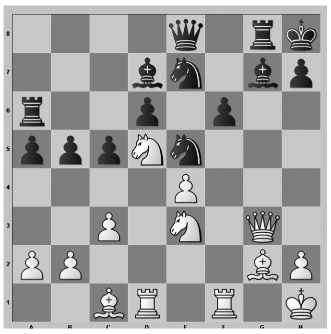

Fig. 3. Computer: *"Why is it NOT possible for White to attack the opponent's king in this position, comparing to the one on the left-hand diagram in Fig. 2?"* Again, the experts have been asked to give a description based on their general knowledge in the presented domain. Their answer was: *"In this position, more black pieces participate in the defence of the king."*

Then, the position shown in Fig. 3 became the new counter-example. The experts again explained the difference between the critical and the new counter-example: *"In this position, more black pieces participate in the defence of the king."*. A new attribute, *defends_by_pieces_around_king*, was therefore programmed and included into the domain. It is the number of squares in the square f5-h5-h7-f7 defended by black pieces, not counting black pawns. Based on the experts explanation, the previous argument was now extended by "*defends_by_pieces_around_king* is low." With the new attribute and the extended argument the following rule was obtained:

```
IF $open\_file$ is $true$ AND $attacking\_pieces > 1$
    AND $defends\_by\_pieces\_around\_king <= 13$
THEN $ATTACK = true$;
```

The algorithm could not find any counter-examples to the above argument, therefore the ABML refinement loop proceeded to the next iteration.

In total, the ABML process consisted of 38 iterations. In ten of those iterations, experts decided to change the class of the example and in others arguments were provided. Arguments altogether suggested 14 new attributes that increased the number of attributes to 26. The final model contained 16 rules and the improvement of the model's quality is obvious: the classification accuracy increased to 91%, precision to 93% and recall to 92%.

A possible criticism of this experiment could be the high number of iterations between the expert and the algorithm. According to our previous experience with ABCN2, this number was always considerably smaller. At the moment, we are unable to explain it; finding an explanation for this phenomenon remains for the future work.

5 Comparison of Explanation Power

The previous section showed the difference in discrimination power between the stand-alone machine learning and ABML. While the increased accuracy of ABML is certainly useful, it is the reasons for this or that decision that are likely to be of even greater importance from the viewpoint of creating intelligent commentary. The present section thus attempts to evaluate the difference in explanation power between the two paradigms. The evaluation mostly relies on the opinions of chess experts, consequently being somewhat subjective.

Let us start by appraising the rules themselves. Stand-alone ML produced 12 rules. Three of those rules included at least one of the two terms that the experts marked as counter-intuitive if not downright absurd. These terms are: (1) the attacking side has *less* than three pieces on the offensive, and (2) the defending side has *more* than two pawns shielding the king. Even without expert opinion it is clear that it should be just the other way around: the more pieces the attacking side has, and the lesser pawn shield the defending side has, the higher chances of an attack. However, it is not uncommon for ML to produce such seemingly nonsensical explanations as an artefact of the data. In our case, it is likely the consequence that no learning positions where one side would have many attacking pieces and the other no pawn shield were included. Such positions are rarely, if ever, encountered in professional play as the weaker side would already resign by then.

On the other hand, ABML produced 16 rules, and none of them included any illogical terms as deemed by the experts.

The emphasis we give to such illogical terms is not unwarranted. While such terms can often be ignored when pure discriminative power is the (main) issue, they are very harmful when the model is used to explain the data, and even more so when used for tutoring. A teacher using illogical argumentation (even of a correct decision) is never a good idea.

How harmful are the three rules using illogical terms in our case? ML rules *correctly* decided that one side aims at attacking the opposing king in 131 cases, however, the harmful argumentation was present in no less than 112 of these 131 cases. Imagine your teacher using the wrong argumentation in 85% of the cases.

The difference in the number of rules between the two paradigms is also worth noting. Basically, each rule represents a specific pattern of arguing in favor of one side aiming to attack the opponent's king. Having more rules thus means having a wider range of concepts with which the system can explain the target notion. Such increased richness of the explanation language is usually a desirable property when building a commentating/tutoring system.

Figures 4 and 5 illustrate the difference in explanation power between the rules learned by the two paradigms. The second figure presents a position in which the ML rules do not detect the attack, while the first figure shows position where both ML and ABML correctly detected the attack, yet the explanation they offered differs significantly. Reason that seemed illogical to experts is written in *italic*. The chess experts examined over 30 explanations and unanimously preferred the richer and correct explanation given by the ABML.

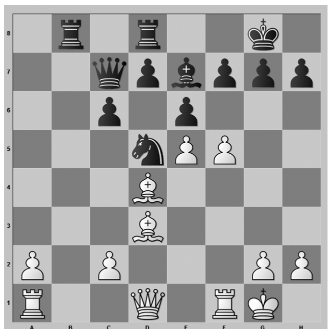

Fig. 4. ML rules: is attack, since pawn_attack>0 (number of pawns on files f,g,h and on rows 4,5,6), *pawn_defense*>2 (number of black pawns surrounding the black king), and pieces_on_king_side>0 (number of white pieces without pawns on files f,g,h). **ABML rules**: is attack, since strong_piece_attack>0 (number of strong pieces that can get to an open or semi open file on files f,g,h in 1 move), defends_pieces_around_king<6 (number of squares defended by pieces around black king), and pawns_fifth_row>0 (number of white pawns on files f,g,h on fifth row).

6 Related Work

The learning algorithm our work is based on, ABCN2, combines expert explanations with a rule learning algorithm, which can be seen as a special type of integrating machine learning and knowledge acquisition. It was already shown that a combination of a domain expert and machine learning yields the best results [14]. In the field of inductive logic programming (ILP), several approaches were proposed and they all can integrate prior expert knowledge with machine learning (e.g. HYPER and FOIL). Additionally, ABML refinement loop is also an interactive algorithm; it queries experts in turns for more knowledge. In literature, several similar approaches were already proposed, e.g. [1, 9]. However, these systems require experts to directly revise the learned models or to provide new attributes that would enable further learning, whereas in our system, the experts are asked to explain *particular* learning examples. We prefer the latter

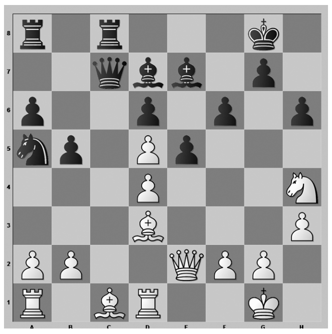

Fig. 5. ML rules: no attack. **ABML rules**: is attack, since attacking_pieces>2 (number of pieces attacking area around black king), defends_pieces_around_king<6, knight_go_on_safe_spot>0 (number of squares around king where knight can be safely positioned), and close_defending_pieces<2 (number of black pieces standing close to the black king)

principle, since empirical observations have shown that humans are better at providing specific explanations than providing generic knowledge pertaining to the problem.

According to its name, explanation-based generalization (EBG) [5] could be the closest relative to ABML. However, this technique heavily relies on the provided general prior knowledge (not explanations). First, EBG uses this prior knowledge to automatically construct explanations of individual learning examples, and the "learned" model is then a logical generalization of these explanations. Note that EBG is not inductive learning; it assumes perfect and complete knowledge of the domain, which makes it inapplicable in domains where complete knowledge is unavailable.

An interesting extension of EBG is the DISCIPLE system [13, 12] for learning rules in problem solving domains. This system uses the EBG technique if perfect domain theory is available. However, in the case of imperfect theory, the system will ask the expert for explanation of examples and use these explanations to construct general rules. Knowledge elicitation through explanations in DISCIPLE is similar to that of ABML, however the methods differ in the selection of critical examples and how expert's explanations are used in learning.

7 Conclusion

We argued a case for using a new paradigm, ABML, instead of classical machine learning for the purpose of learning high-level concepts and their explanations. One potential use of such concepts and explanations is for generating automatic commentary and/or tutoring. The paper showed how the interaction between the experts and ABML can lead to improved argumentation power of the learned models that go along with the increased accuracy of such models.

The concept chosen for the case study in the paper is a very complex one, especially so, because its notion is a bit vague. However, we believe the concept was successfully grasped by the final model, albeit requiring more work than initially thought. The amount of work required seems disproportionate in comparison with other, similar concepts that we successfully learned in the past. It is possible, though, that the notion of an attack in chess is just vague enough for the experts to be a very difficult problem to tackle. An analysis of this is an obvious avenue for the future work.

Acknowledgments. This work was partly funded by ARRS, Slovenian Research Agency.

References

1. Wray Buntine and David Stirling. Interactive induction. *Machine intelligence*, 12:121–137, 1991.
2. 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.
3. Edward A. Feigenbaum. Some challenges and grand challenges for computational intelligence. *Source Journal of the ACM*, 50(1):32–40, 2003.
4. Matej Guid, Martin Možina, Jana Krivec, Aleksander Sadikov, and Ivan Bratko. Learning positional features for annotating chess games: A case study. In *Lecture Notes in Computer Science*, volume 5131, pages 192–204, 2008.
5. Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
6. Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10/15):922–937, 2007.
7. 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.
8. Henry Prakken and Gerard Vreeswijk. *Handbook of Philosophical Logic, second edition*, volume 4, chapter Logics for Defeasible Argumentation, pages 218–319. Kluwer Academic Publishers, Dordrecht etc, 2002.
9. Debbie Richards. Ripple down rules: a technique for acquiring knowledge. pages 207–226, 2003.
10. 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.
11. 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.
12. G. Tecuci, M. Boicu, C. Boicu, D. Marcu, B. Stanescu, and M. Barbulescu. The disciple-rkf learning and reasoning agent. *Computational Intelligence*, 21(4):462–479, 2005.
13. G. Tecuci and Y. Kodratoff. Apprenticeship learning in imperfect domain theories. *Machine Learning: An Artificial Intelligence Approach*, 3:514–551, 1990.
14. 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.