ABCN2e: an extension of the ABCN2 algorithm

Martin Možina

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

Abstract

In this white paper, we present some recent extensions of the argument-based rule learning algorithm ABCN2. The main differences are: a) ABCN2e learns all rules in one pass and does not need consecutive calls to "FindBestRule", b) ABCN2e uses extreme-value correction (EVC) to evaluate rules, and c) learned rules are combined into a classifier with a log-linear weighted sum.

Keywords: argument-based machine learning, rule learning, ABCN2

1. Argument-based machine learning

An argument is comprised of a series of premises that are intended to give a reason for the conclusion. Humans mostly use arguments to justify or explain their beliefs and sometimes to convince others. In artificial intelligence, argumentation is a branch that analyzes automatic reasoning from arguments - how arguments for and against a certain claim are produced and evaluated. Argument-based machine learning (ABML) is a combination of argumentation and machine learning. Arguments enhance learning examples and an ABML method learns from such examples [9].

ABML uses arguments to elicit and represent background knowledge. While asking experts to provide general background knowledge can be a difficult task, asking them to articulate their knowledge through arguments has proved to be much more efficient, because in ABML experts need to argue about one specific case and provide knowledge relevant for this case, which does not have to be valid for the whole domain [4, 7].

Arguments are used in ABML to enhance learning examples. Each argument is attached to a single learning example, 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. Examples with attached arguments are called *argumented examples*.

An ABML method needs to induce a model that is consistent with given arguments. That means that the model has to explain the examples using the same terms that experts used during argumenting these 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.

1.1. Argumented examples

A detailed description of argumented examples and the structure of arguments is given in Mozina et al. [9]. The following description is a short summary.

Let $(\mathbf{X}, y)$ denote a learning example, where $\mathbf{X}$ is a feature (or attribute) vector and y is a class value. An argumented example is a triple $(\mathbf{X}, y, \mathcal{A})$, where $\mathcal{A}$ is a set containing a) positive arguments explaining why is this example in class y and b) negative arguments explaining why this example should not be in class y. Each argument $a_i \in \mathcal{A}$ is composed of a conjunction of reasons $r_1 \wedge r_2 \wedge \cdots \wedge r_n$. Allowed formats of reasons are:

- $X = x_i$ specifies that example has class value because (or despite) X_i equals x_i,
- $X < x_i$ (or $X > x_i$) argues for (or against) class y because the value of X is less than x_i (or is larger),
- $X <$ (or $X >$) argues for (or against) class y because the value of X is low (or high).

1.2. ABCN2e

In this section, we describe a new general argument-based rule learning method that learns a set of classification rules from argumented examples.

We first need to redefine the *covering* relation. In the standard definition (see for example CN2 [2]) a rule covers an example if the condition part is true for this example. A rule R AB-covers an argumented example $(\mathbf{X}, y, \mathcal{A})$ if:

- All conditions in R are true for $\mathbf{X}$ (same as in CN2),
- R is consistent with at least one positive argument of $\mathcal{A}$.
- R is not consistent with any of the negative arguments of $\mathcal{A}$.

A rule is consistent with an argument if it contains the reasons of the argument in its condition part.

The first argument-based rule learning algorithm was ABCN2 [9], which extended the CN2 algorithm for learning unordered rules [1]. The CN2 learns rules using the separate-and-conquer principle: it starts by learning the best rule it can find for the given data, then removes data covered by this rule and repeats the procedure on the remaining data. The argument-based extension of CN2 started by learning rules that cover argumented examples and afterwards learned remaining rules. Whereas this approach worked well in many cases, ABCN2 sometimes learned rules from argumented examples that prevented induction of (more accurate) rules that would otherwise be learned. This unfortunate behavior sometimes caused a decrease in accuracy of learned models.

The new argument-based rule learning algorithm (ABCN2e), introduced in this paper, learns all rules for one class in one pass. While there are several existing rule-learning algorithms implementing such non-reductive strategy [5], none of the techniques could be effectively applied in our case. ABCN2e is an extension of the algorithm for *single* rule construction in CN2 [1], which learns a rule in a top-down fashion using beam search strategy. Our algorithm

```
Function 2e(E, T):
    Input: A set of (argumented) examples $E$ and a target class $T$.
    Output: A set of classification rules $\mathcal{R}^*$ explaining $E$.
1   Let $\mathcal{P}$ be the set of rules created from all positive arguments.
2   $\mathcal{R} \leftarrow \mathcal{P} \cup \{\texttt{if } true \texttt{ then } T\}$
3   $\forall e \in E[T]; \mathbf{b}[e] \leftarrow \varnothing$        // $E[T]$ are examples from $T$
4   while $\mathcal{R} \neq \emptyset$ do
5       foreach $r \in \mathcal{R}$ do
6           $\forall e \in E[T]; \mathbf{b}[e] \leftarrow r$ if $\texttt{ABcovers}(r, e) \wedge \texttt{Q}(r) > \texttt{Q}(\mathbf{b}[e]) \wedge \texttt{Sig}(r)$
7       end
8       $\mathcal{R} \leftarrow \mathcal{R} \setminus \{r \| r \in \mathcal{R} \wedge \texttt{Stop}(r)\}$
9       $\mathcal{R} \leftarrow \texttt{FilterRules}(\mathcal{R}, E[T])$
10      $\mathcal{R} \leftarrow \texttt{RefineRules}(\mathcal{R})$
11  end
12  $\mathcal{R}^* \leftarrow \{\mathbf{b}[e] \| e \in E[T]\}$
13  return $\mathcal{R}^*$
```

Algorithm 1: The argument-based rule learning algorithm for learning a set of classification rules for class T from argumented examples E.

is implemented as an add-on[1] for the open data mining toolkit Orange [3].

The pseudo code is presented in Algorithm 1. The algorithm starts by setting the star $\mathcal{R}$, which at any point of the algorithm contains the set of candidate rules. Initially, $\mathcal{R}$ contains the default rule and all rules converted from arguments, where reasons of positive arguments were simply used as conditions of rules.

The main difference between the ABCN2e and the CN2 algorithm is how the best rule is stored. While in CN2 we only need to remember one best rule, in ABCN2e we have to remember all relevant rules, since the complete rule set is to be learned. We therefore keep the best rule for each learning example in

[1]The latest sources can be found at `https://ailab.si/abml`.

```
Function FilterRules (R, E):
    Input: Set $\mathcal{R}$ is the set of current rules, $E$ are learning examples.
    Output: Returns $K$ most promising rules from $\mathcal{R}$.
1   Let $\forall e \in E; \mathbf{s}[e] \subset \mathcal{R}$ be a descending sorted list (according to
      evaluation Q) of rules AB-covering $e$.
2   Let $rank(r, e)$ be rank of rule $r$ in $\mathbf{s}[e]$.
3   $\forall r \in \mathcal{R}; rank(r) \leftarrow \texttt{min}(rank(r, e); \forall e \in E)$
4   $\forall r \in \mathcal{R}; freq(r) \leftarrow \#(rank(r, e) = rank(r); \forall e \in E)$
5   $\mathcal{R} \leftarrow \texttt{Sort}(\mathcal{R})$ //use the following sorting criteria:
    // $\forall r_i, r_j \in \mathcal{R}, r_i < r_j \leftrightarrow (rank[r_i] < rank[r_j]) \vee$
    // $\vee (rank[r_i] = rank[r_j] \wedge freq[r_i] > freq[r_j])$
6   return $\mathcal{R}[1..K]$;
```

Algorithm 2: Function selects K best candidate rules from $\mathcal{R}$ for further specialization. The method selects rules with high quality estimated with `Q` and cover as many examples.

b. At the beginning (line 3), the list of best rules is empty.

We defined a rule as relevant, if it is best for at least one learning example. A rule is best for a particular example, if it AB-covers it and has the highest quality (estimated with a user-defined heuristics `Q`) among all generated rules that also AB-cover the same example. Lines 6-8 in Alg. 1 implement this behavior: best rule for example $\mathbf{b}[e]$ is updated if a rule r is better than the current best rule and rule is accepted by function `Sig`. The `Sig` function validates whether a rule is good enough to be present in the final set of rules, where statistical significance is often used (hence the name `Sig`). By default, we use the EV-corrected relative frequency measure [8] for `Q` and a simple function that always returns true for `Sig`.

The next step of the algorithm (line 8) removes all rules that are deemed unpromising by the rule stopping criteria function `Stop`. The default implementation of this function removes rules with more than 5 conditions and removes rules where the last condition did not remove any of the covered examples.

Line 9 of Alg. 1 removes rules from the beam if its size exceeds a certain threshold K. In all our experiments, K was arbitrarily set to 100; changing this value did not significantly influence the results. The selection of K rules in our algorithm differs from standard algorithms for single rule construction, where it is common to simply take K rules with the highest value estimated by a rule evaluation heuristic Q. In our case, we need to select a set of rules that a) have high quality and b) cover the complete space of learning examples. The algorithm for selecting K most promising rules is given in Alg. 2. The algorithm will order rules by their rank, where $rank(r)$ is defined as the best rank of rule r over all learning examples. For example, if a rule $r_i \in \mathcal{R}$ has $rank(r_i) = 1$, that means that there exists a learning example that is AB-covered by this rule and no other rule AB-covering this example is better than r_i. Similarly, if a rule r_j has rank $rank(r_j) = 3$, there is an example where this rule is third best, and there are no learning examples where rule r_j is first or second.

In line 10 of Alg. 1, all candidate rules in star $\mathcal{R}$ get refined: all rules in $\mathcal{R}$ are extended with all possible conditions (these were denoted selectors in the original CN2 paper) in the domain. The while loop continues until there are candidate rules in $\mathcal{R}$. At the end, the set of all relevant rules $\mathcal{R}^*$ is constructed from **b** and returned.

References

[1] 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.

[2] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning Journal*, 4(3):261–283, 1989.

[3] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka

Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[4] Pedro Domingos. Toward knowledge-rich data mining. *Journal of Data Mining and Knowledge Discovery*, 15:21–28, 2007.

[5] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrač. *Foundations of Rule Learning*. Springer-Verlag Berlin Heidenberg, 2012.

[6] Vida Groznik, Matej Guid, Aleksander Sadikov, Martin Možina, Dejan Georgiev, Veronika Kragelj, Samo Ribarič, Zvezdan Pirtošek, and Ivan Bratko. Elicitation of neurological knowledge with argument-based machine learning. *Artificial intelligence in medicine*, 57(2):133–144, 2013.

[7] Matej Guid, Martin Možina, Vida Groznik, Aleksander Sadikov, Dejan Georgijev, Zvezdan Pirtošek, and Ivan Bratko. Abml knowledge refinement loop: A case study. In *Proceedings of the 2012 IEEE 20th International Symposium (ISMIS 2012)*, pages 41–50, 2012.

[8] Martin Možina, Janez Demšar, Jure Žabkar, and Ivan Bratko. Why is rule learning optimistic and how to correct it. In Johannes Fuernkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proceedings of 17th European Conference on Machine Learning (ECML 2006)*, pages 330–340, Berlin, 2006. Springer-Verlag.

[9] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument-based machine learning. *Artificial Intelligence*, 171(10/15):922–937, 2007.

[10] Martin Možina, Matej Guid, Jana Krivec, and Aleksander Sadikov. Learning to explain with abml. In *Proceedings of the 5th International Workshop on Explanation-aware Computing (Exact 2010)*, pages 37–48, Lisbon, Portugal, 2010.