

# Learning Attributional Ruletrees

Jaroslav Pietrzykowski and Janusz Wojtusiak

Machine Learning and Inference Laboratory, George Mason University, USA

## Abstract

Attributional ruletrees are shallow decision-tree-like structures whose leaves are sets of attributional rules or groups of classes. By using attributional ruletrees one is able in some cases to improve computational efficiency of inductive learning and understandability of generated hypotheses. A comparison of results of creating attributional ruletrees, attributional rules, and decision trees for a well-known classification problem indicates advantages of this approach.

**Keywords:** Attributional Ruletrees, Attributional Calculus, Natural Induction, Comprehensibility, AQ Learning

## 1 Introduction

An important direction in machine learning is the development of novel knowledge representations that present high understandability of produced results without sacrificing the efficiency of the learning algorithms. The concept of attributional ruletrees, proposed by Michalski (2002), aims at improving the efficiency of rule-based learning in situations where one wants to classify examples into many classes of a fixed set. In such cases, knowledge representations, even though having lower expressive power, are more appealing because they enhance the speed of learning. The attributional ruletree method, presented in this paper, addresses this problem by dividing the original learning problem into a set of simpler sub-problems. Instead of learning descriptions of given classes (related concepts) based on a training dataset, the dataset is divided into disjoint parts, in which each part is related to a subset of the set of all classes to be learned. To achieve this, the method uses a partitioning attribute that maps the event space into such partitions. Such a partitioning attribute can be either from the original representation space or can be constructed from existing attributes. The advantage of using ruletrees can be two-fold, namely it can improve execution time efficiency, and it can improve understandability of learned knowledge. Solving each sub-problem is less computationally expensive because there are fewer examples that have to be dealt with than in the original problem. Also, the number of attributes used in learning is decreased because any attributes from which the partitioning attribute was generated are excluded. Moreover, the resulting knowledge representation can be simpler because it uses fewer attributes and attributional rules are learned on smaller datasets. It also gives a “natural” grouping of classes into larger categories. For example in medical domain, it makes sense to use different sets of

attributes when analyzing various groups of patients, defined by their age and sex. In case where some classes can be described solely by the values of the partitioning attribute(s), their description can be much simpler.

On the other hand, the advantage of using this method is balanced by the overhead needed for searching for the partitioning attribute, which can be a time-consuming process. The ART (Attributional Rule Tree) method creates a tree with the partitioning attribute assigned to the root, its values are assigned to the branches, and its classes or groups of classes form the leaves. A path leading from the root of the tree created by the ART method to one of the leaves forms a precondition for each ruleset assigned to this leaf. If more than one partitioning attribute is discovered, each of them can be used to create additional level of such tree. No more than three-level trees are constructed to maintain their high level of interpretability. Significant increase in the AQ learning efficiency due to the use of the presented approach can be shown by looking at the computational complexity of the learning process, which can be roughly approximated by a linear function of the number of negative training events (a.k.a. examples). For instance, in a classification problem with 3 classes C1, C2 and C3, with  $n_1$ ,  $n_2$  and  $n_3$  events belonging to each class, the computational complexity (denoted  $\text{comp}(N)$ , where  $N$  is the number of negative examples) of the AQ learning is approximately  $\text{comp}(n_2 + n_3)$ ,  $\text{comp}(n_1 + n_3)$  and  $\text{comp}(n_1 + n_2)$  for each class respectively. Therefore the total complexity is the sum of the above:  $\text{comp}(n_2 + n_3) + \text{comp}(n_1 + n_3) + \text{comp}(n_1 + n_2)$ . If ART discovers a partitioning attribute forming a tree with 2 branches, one linked to C1 and the other linked to C2 and C3, the overall complexity of subsequently applying the AQ learning would be just  $\text{comp}(n_3) + \text{comp}(n_2)$ , since no learning for the class C1 would be needed. This benefit is magnified for larger numbers of classes with many events. Further reduction in complexity is achieved by excluding partitioning attribute from the representation space used in the subsequent learning process.

## 2 Methods

For completeness of this section, we first briefly describe attributional rules and AQ learning, and then attributional ruletrees and method for learning them.

### 2.1 Creating Attributional Rules by AQ Learning

Natural induction is an approach to machine learning that puts an equal importance to knowledge accuracy and understandability (Michalski et al., 2006). Attributional calculus (AC) is a formal language that combines predicate, propositional, and multi-valued logics, and whose constructs resemble simple natural language statements Michalski (2004). Because of that AC seems to be a perfect language for natural induction. The main form of knowledge in AC is an attributional rule. Two important forms of attributional rules are (1) and (2).

$$\text{CONSEQUENT} \Leftarrow \text{PREMISE} \lfloor \text{EXCEPTION} \quad (1)$$

$$\text{CONSEQUENT} \Leftarrow \text{PREMISE} \lceil \text{PRECONDITION} \quad (2)$$

In some situations another form can also be used (3).

$$[ \text{PRECONDITION CONSEQUENT} \Leftarrow \text{PREMISE} \quad (3)$$

Here, CONSEQUENT, PREMISE, and PRECONDITION are conjunctions of attributional conditions, and EXCEPTION is either a conjunction of attributional conditions or an explicit list of examples that constitute exceptions to the rule. A simple form of attributional conditions is:

$$[L \text{ rel } R] \quad (4)$$

where  $L$  is an attribute;  $R$  is a value, a disjunction of values, or a conjunction of values if  $L$  is a compound attribute; and  $rel$  is a relation that applies to  $L$  and  $R$ . Other forms of attributional conditions may involve count attributes, simple arithmetical expressions, conjunctions and disjunctions of attributes, comparison or attributes, etc. (Wojtusiak et al., 2006).

In this study, to learn attributional rules we used the AQ21 system, which has been developed in our laboratory (Wojtusiak et al., 2006). Given input data, problem definition, and optional background knowledge, AQ21 induces rules in the forms (1) and (2), describing one or more class in the data. A set of rules constituting a description of a given class is called a *ruleset*. By repeating learning for all classes defined by values of an output attribute, AQ21 generates a *classifier*.

In order to learn rules for a given class AQ21 starts with one example belonging to this class. It generates a set of maximally general rules that cover the selected example and do not cover any examples from other classes. This is done by repeating an operation that generalizes the seed against examples not belonging to the concept being learned. Results of applying the operation are intersected and the best rules are selected according to user-defined criteria. If selected rules do not cover all examples belonging to the class being learned, another uncovered example is selected and additional rules are learned. The process is repeated until all examples of the class are covered by the learned rules. AQ21 implements several modifications to the above basic algorithm as described, for example, by Wojtusiak et al. (2006).

## 2.2 Determining Partitioning Attributes

Finding partitioning attributes is performed with a stand-alone ART program, written in a C++ programming language. The program takes as the only input a text file with the data describing the problem, where the data points are separated by commas. Each row of data forms an example describing a class. Each column contains values of one attribute, with the distinction of the first column that has the names of the classes. At present, the program does not handle any meta-values (missing, not applicable or irrelevant values).

During the input processing, data structures representing the domains of the attributes and the classes are created in the form of maps of bitstrings. Thus, for each attribute an association between its values and the classes takes form of a table, with columns representing attribute's values, and rows representing classes.

A value of “1” at the intersection of a given column and a given row indicates that in the data corresponding value of that attribute occurred for that class.

After the input processing phase, the main algorithm loops over all attributes and for each attribute it compares pairwise the bitstrings corresponding to classes. This is done using bitwise logical operations AND, OR and NOT, and it allows to determine if classes share values. If that is a case, then such classes are grouped together. This procedure has a complexity that can be expressed as  $O(noa * noc^2)$  where *noa* and *noc* mean number of attributes and classes correspondingly.

The last phase of the algorithm checks for each attribute if there are any groups of classes created. If such groups exist, that means that a partitioning was found and the mappings between attribute’s values and class names are printed.

### 3 Experimental Results

In this section we present a sample result from application to the well-known soybean disease identification problem available from the UCI Machine Learning Repository<sup>1</sup>. Below is the output of the ART program. The execution time in this case was approximately 1 sec, where the machine that was used had Intel(R) 1.60GHz T2050 cpu with the cache size 2048 KB and 1GB of memory.

Attribute *leaf mildew*:

absent → alternarialeaf-spot, anthracnose, bacterial-blight, bacterial-pustule, brown-spot, brown-stem-rot, charcoal-rot, diaporthe-stem-canker, frog-eye-leaf-spot, phyllosticta-leaf-spot, phytophthora-rot, purple-seed-stain, rhizoctonia-root-rot

upper-surf → powdery-mildew

upper-surf → downy-mildew

Attribute *int-discoloration*:

none → alternarialeaf-spot, anthracnose, bacterial-blight, bacterial-pustule, brown-spot, diaporthe-stem-canker, downy-mildew, frog-eye-leaf-spot, phyllosticta-leaf-spot, phytophthora-rot, powdery-mildew, purple-seed-stain, rhizoctonia-root-rot

brown → brown-stem-rot

black → charcoal-rot

Attribute *sclerotia*:

absent → alternarialeaf-spot, anthracnose, bacterial-blight, bacterial-pustule, brown-spot, brown-stem-rot, diaporthe-stem-canker, downy-mildew, frog-eye-leaf-spot, phyllosticta-leaf-spot, phytophthora-rot, powdery-mildew, purple-seed-stain, rhizoctonia-root-rot

present → charcoal-rot

Attribute *fruit pods*:

---

<sup>1</sup>[http://archive.ics.uci.edu/ml/datasets/Soybean+\(Large\)](http://archive.ics.uci.edu/ml/datasets/Soybean+(Large))

normal, diseased → alternarialeaf-spot, anthracnose, bacterial-blight, bacterial-pustule, brown-spot, brown-stem-rot, charcoal-rot, diaporthe-stem-canker, downy-mildew, frog-eye-leaf-spot, phyllosticta-leaf-spot, powdery-mildew, purple-seed-stain

NA → phytophthora-rot, rhizoctonia-root-rot

In the last case the program found not so useful fact, that certain attribute is not applicable when describing some of the diseases. In order to compare with the traditional AQ learning, the AQ21 program was run for the problem in two modes. In the first mode the data was not modified. In the second mode, AQ21's parameters were set to values allowing to ignore the attributes found by ART, and also to use only a subset of the data with the diseases not determined by the values of the partitioning attributes found (see (Wojtusiak, 2004) for details). As an expected result, it turned out that the execution time in the latter case was approximately twice as long. Although it is not clear what is the reason for that behaviour, mostly likely it should be attributed to the data subsetting function of the AQ21 program. This issue requires further investigation in the future.

A graphical illustration of the discovered partitioning is presented in Figure 1. In this problem 19 classes represent soybean diseases, and each event is described by 35 attributes. The 266 events with no missing values were used. The program found 4 partitioning attributes, 2 of which were combined to form the ruletree. The attributes are leaf-mildew and int-discolor (internal discoloration) and they are shown in the root node. Branches of the tree correspond to combinations of attributes' values, 4 of them identify individual classes *powdery mildew*, *downy mildew*, *charcoal-rot* and *brown-stem-rot*, whereas the last branch links root with the subclassifier for the other 11 classes. The classifier created by AQ21 only is presented on the right side of the picture. The ruleset for each class in the (sub)classifier is symbolized by a square with a box and characterized by number of rules and conditions that compose it.

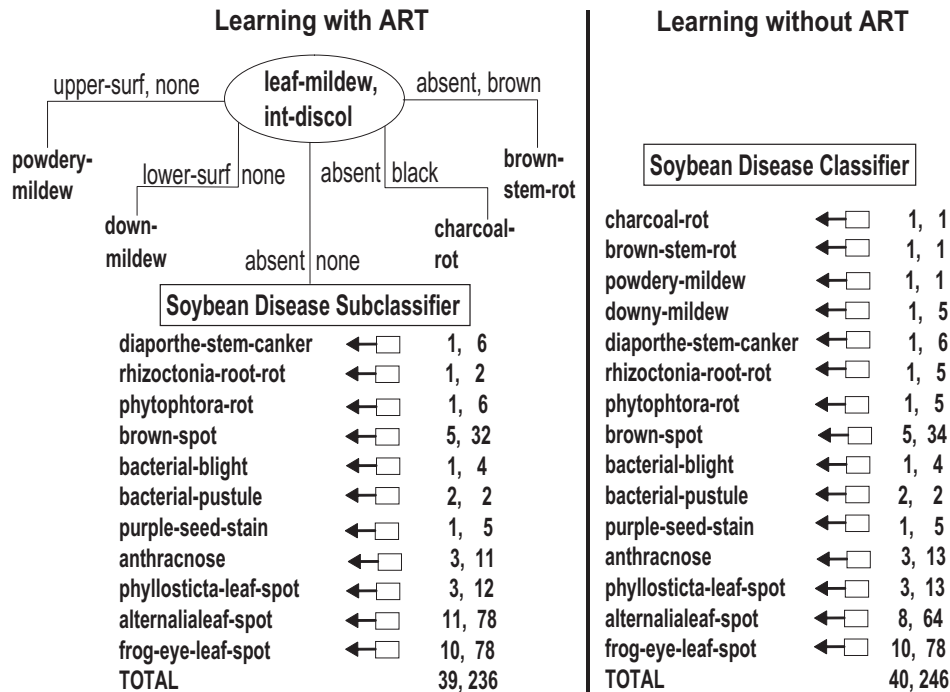
As one can see, the tree representation created by the ART method is fairly easy to understand and interpret, and the rulesets in the subclassifier are in most of the cases less complex than the ones in the classifier. In fact, there is only one case, concerning the class *alternarialeaf-spot*, where traditional approach produced simpler ruleset.

For comparison the traditional ruleset and corresponding preconditioned ruleset are shown below, where the class being described is *rhizoctonia-root-rot*.

Class description created using AQ21 only:

```
[disease = rhizoctonia-root-rot]
<=  [precip > norm] AND
      [temp < norm] AND
      [leaves = norm] AND
      [mold-growth = absent] AND
      [seed-discolor = absent]
```

Class description created using both the ART and AQ21 (the whole ruleset is preceded by the precondition):



In the pairs of numbers above, the first is the number of rules, and the second is the total number of conditions.

FIGURE 1: A comparison of a 2-level ruletree vs. a traditional ruleset

```
[
[leaf-mildew = absent] AND
[int-discolor = none]

[disease = rhizoctonia-root-rot]
<= [plant-growth = abnorm] AND
    [leaves = norm]
```

The description can be read as:

*Provided that there is no leaf mildew, nor internal discoloration, the soybean disease is rhizoctonia-root-rot if the plant's growth is abnormal and its leaves are normal.*

One may observe that in some cases, AQ21 was able to find very simple rules describing the classes discovered by ART, actually as three out of four of them. Nevertheless, rule induction did not discover the fourth one. At this point it is difficult to assess in which cases that happens, but the intuition suggests that this depends on the complexity of the problem. The more complex data, the more complex is the search process utilized by the AQ learning algorithms and which affects how the attributes are evaluated.

It is interesting to compare what kind of description was produced by a decision tree learning method. In this case, we used the implementation of the C4.5 algorithm (Quinlan, 1993), from the WEKA (Witten and Frank, 2005) software package available from The University of Waikato<sup>2</sup>.

When the program was applied to not modified dataset used for AQ21 and ART experiments it produced result corresponding to the following description of the class *rhizoctonia-root-rot*:

```
disease = rhizoctonia-root-rot
IF
int_discolor = none AND
leaf_mildew = absent AND
leafspot_size = NA AND
fruit_pods = NA AND
leaves = norm
```

Because the program cannot handle non-applicable values well, in another experiment these values were denoted as missing. This time created description was as following:

```
disease = rhizoctonia-root-rot
IF
leaf_mildew = absent AND
int_discolor = none AND
leafspots_halo = absent AND
stem_cankers = below_soil AND
leaves = norm
```

Although both decision-tree-based approach and the method combining AQ21 and ART created description covering all positive examples of the class, the result produced by the latter seems simpler and more understandable.

## 4 Conclusion and Future Work

The results presented above show very promising potential of the ART approach. Combing decision-tree-like structure with rule-based classifier seems to have quite high cognitive appeal when compared to using rules and trees separately. Also fast execution time of the ART algorithm is very encouraging, therefore it should not undermine the efficiency of the knowledge discovery process. There is a number of issues that require closer examination, among them: significantly increased AQ21 execution time for the modified data, analysis of the conditions influencing the complexity of the inferred rules after application of ART to the problem, applicability of the presented approach to different types of problems.

Finding p-type (partition generating) attributes presented above in the original dataset can be difficult in many real world applications, therefore an important topic for research is to equip the ART program with the ability to find derived p-type attributes based on combinations of several attributes. Another research

---

<sup>2</sup><http://www.cs.waikato.ac.nz/ml/weka/>

opportunity emerges when one considers discovering o-type (overlap generating) attributes, that divide a classifier into overlapping subclassifiers and are evaluated based on a certain partition quality criterion, indicating how well an o-type attribute approximates a p-type attribute.

An interesting research direction is the application of the ART operator jointly with the multihead rule learning method. Multihead rules, are the rules with their consequent composed of more than one attributional condition (Glowinski and Michalski, 2002). An example of a class description resulting from such combination, where the consequent denotes occurrence of various diseases, is presented below.

```
[
[sex = male] AND
[age = 50..63]

[high-blood-pressure = present] AND
[arteriosclerosis = present] AND
[hypertension = present]
<= [cholesterol-level = high] AND
    [exercise < once-a-week] AND
    [diet = high-carb]
```

The rule can be read as:

*Provided that the patient is a male between 50 and 63 years old, patient's diagnosis is high blood pressure, arteriosclerosis and hypertension if his cholesterol level is high, he exercises less than once a week and is on high-carb diet .*

## References

- Cezary GLOWINSKI and Ryszard S. MICHALSKI (2001), Discovering Multi-head Attributional Rules in Large Databases, in *Proceedings of the International Symposium on Intelligent Information Systems X*, pp. 31–41.
- Ryszard S. MICHALSKI (2002), Attributional Ruletrees: A New Representation for AQ Learning, *Reports of the Machine Learning and Inference Laboratory*, MLI 02-1, George Mason University, Fairfax, VA, October, 2002.
- Ryszard S. MICHALSKI (2004), ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction, *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA, April, 2004.
- Ryszard S. MICHALSKI, Kenneth A. KAUFMAN, Jaroslaw PIETRZYKOWSKI, Janusz WOJTUSIAK, Scott MITCHELL and William D. SEEMAN (2006), Natural Induction and Conceptual Clustering: A Review of Applications, *Reports of the Machine Learning and Inference Laboratory*, MLI 06-3, George Mason University, Fairfax, VA, August, 2006.
- Ross QUINLAN (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA.
- Ian H. WITTEN and Eibe FRANK(2005), *Data Mining: Practical machine learning tools and techniques, 2nd Edition*, Morgan Kaufmann, San Francisco, 2005.

Janusz WOJTUSIAK (2004), AQ21 User's Guide, *Reports of the Machine Learning and Inference Laboratory*, MLI 04-3, George Mason University, Fairfax, VA, September, 2004.

Janusz WOJTUSIAK, Ryszard S. MICHALSKI, Kenneth A. KAUFMAN, Jaroslaw PIETRZYKOWSKI (2006), Multitype Pattern Discovery Via AQ21: A Brief Description of the Method and Its Novel Features, *Reports of the Machine Learning and Inference Laboratory*, MLI 06-2, George Mason University, Fairfax, VA, June, 2006.

