Reports

Machine Learning and Inference Laboratory

Attributional Ruletrees:
A New Representation for AQ Learning

Ryszard S. Michalski

MLI 02-1
P 02-5
October, 2002
(slightly edited, May, 2004)

School of Computational Sciences

George Mason University

ATTRIBUTIONAL RULETREES: A NEW REPRESENTATION FOR AQ LEARNING

Ryszard S. Michalski [*]*

Machine Learning and Inference Laboratory, George Mason University,
Fairfax, VA 22030-4444

michalski@gmu.edu
http://www.mli.gmu.edu

Abstract

Attributional ruletrees are proposed as an extension of the current ruleset representation used by AQ type learning. The ruletrees split a multiclass classification problem into separate subproblems using a *class splitting attribute*. The resulting representation can be graphically represented as a tree whose root is assigned the class splitting attribute, branches stemming from the root are values (or sets of values) determining subsets of classes, and leaves are assigned ruleset families for classifying events to classes in these subsets. The values on the branches from the root thus define preconditions for applying ruleset families. Ruletrees are easy to interpret and understand, and can be generated by a relatively simple modification of the AQ algorithm presented below..

Acknowledgments

The author thanks Ken Kaufman of the Machine Learning and Inference Laboratory for valuable suggestions and comments that helped to significantly improve an earlier version of the paper. This research was conducted in the Machine Learning and Inference Laboratory of George Mason University. The Laboratory's research activities are supported in part by the National Science Foundation Grants No. IIS 9906858 and IIS 0097476, and in part by the UMBC/LUCITE #32 grant.

.

[*] Also with GMU Department of Systems Engineering and Operations Research, Department of Computer Science, and Institute of Computer Science at the Polish Academy of Sciences

1 INTRODUCTION

In the classification problems with many classes (related concepts), an AQ-type learning program has to run independently for each class (concept). This makes the process inefficient if one seeks a decision structure to classify events to classes from a fixed set. In this case, decision tree may be a more effective representation, though it may difficult to interpret and understand (when the tree has many levels), and more complex (because trees are has lower expressive power than attributional rules).

To address this problem, an *attributional ruletree* is introduced. An attributional ruletree is a tree-like structure, in which the root is assigned a *class splitting attribute*, which can be one of two types. If it divides the family of decision classes into mutually disjoint sets of classes, then it is called *p-type* (partition generating); if it divides the family into overlapping decision classes, it is called *o-type* (overlap generating). The *o-type* attributes can be ranked based on some attribute quality criterion (e.g., Baim, 1982; Clark and Niblett, 1989).

Given a family of training sets, each set characterizing one decision class (or concept), an attribute is assumed to be of p-type, if it splits the family into subsets of training sets, such that events from these sets share values from disjoint subsets of the splitting attribute domain; otherwise, an attribute is of o-type.

In the following, we will first concentrate on the method involving only a p-type splitting attribute. In Section 3, we will consider also the method involving an o-type splitting attribute. Once a splitting attribute is determined, the original problem is split into *first-level subproblems*. One can then apply the AQ-type attributional rule learning program to each subproblem, or seek a splitting attribute for each subproblem, and then create *second-level subproblems*. Going beyond two levels is not recommended, because the obtained rule structure would be difficult to understand and interpret by people, and thus would defeat the idea of natural induction, which is a central objective of AQ learning ((Michalski, 1972 ; Michalski et al. 1986, Wnek et al., 1995; Michalski, to appear).

2 ILLUSTRATION OF A P-TYPE ATTRIBUTE

To illustrate a p-type splitting attribute, assume that T1, T2, T3 ,..,Tm are sets of training events for classes (concepts), C1, C2, C3,…,Cm, respectively, and an attribute, A, has the domain: {a,b,c,d, e}. The attribute A is a p-type class splitting attribute, if it splits the training sets, for example, to three sets, defined by cases:

Case 1: If [A = a v c] => {C1, C4, C5}

Case 2: If [A = b v d] => {C2, C3, C6}

Case 3: If [A = e] => {C7, C8,.., Cm}

In this example, sets of values of A in the three cases, {a,c}, {b,d}, and {e}, are mutually disjoint sets, and the corresponding subsets of classes, {C1,C4, C5}, {C2,C3, C6},{C7, C8,.., Cm} are also mutually disjoint. If one chooses A as the root a tree, and creates three branches that area

assigned subsets {a,c}, {b,d} and {e}, then the original classification problem with m classes, is now transformed to three classification problems with 3,3, m-3, classification problems, respectively.

3 ADVANTAGE OF USING RULETREES

Let us try to roughly estimate the advantage of using a ruletree rather over rulesets as a function of the number of classes, m, and other relevant factors. We will start by considering the simple example above.

Suppose that the number of classes m = 9, and the training sets for each class contain p=100 examples. Thus, in each of three cases above, the combined training has 300 examples. Let's assume that the complexity of an AQ-type learning process can be roughly approximated by a linear function of the number of the negative training events. Let com(N) denote the complexity of running AQ to learn a concept when the size of the negative training set is N. Since there are 3 groups of classes, three classes in each group, and for each class the number of negative examples is 200, the complexity of learning concepts in one group is comp(600); and the total complexity of learning a ruletree is roughly comp(1800).

If AQ runs in a regular mode (without a splitting attribute), the complexity of learning each concept is approximately comp(800), and for all 9 concepts, comp(7200), that is, four times larger than in the case of learning a ruletree. If the complexity of determining a splitting attribute is less than (comp(7200) – comp(1800)), using an splitting attribute leads a reduction of computational complexity. There will be an additional reduction of complexity due to learning with training examples in which the number of attributes is reduced by 1 (the splitting attribute).

Let us now consider a more general case. If there are m classes, and the training set for each class has p examples, the complexity of running AQ for m classes is approximately:

$$\text{comp}(m * (m - 1) * p) \tag{1}$$

If a splitting attribute partitions m classes to r groups, then there are m/r classes in each group, and each group has p * m/r examples. The number of negative examples in learning rules for each class is ((m/r) -1)* p, and the complexity of learning concepts in each group is then m/r * comp((m/r -1) * p). The total complexity of learning rules for r groups is then:

$$\text{comp}(m * ((m/r) - 1) * p) + m * (\text{comp-s} / \text{comp-a}) \tag{2}$$

where *m* * comp-s stands for the complexity of determining a splitting attribute in the case of *m* classes, and comp-a estimates a reduction of AQ complexity in learning rules for one class due to the reduction of the number of attributes in examples by 1. The advantage due to introducing a splitting attribute, AS, can then be estimated as:

$$AS = \text{comp}(m * (m - 1) * p) / (\text{comp}(m * ((m/r) - 1) * p) + m * (\text{comp-s}/\text{comp-a}) \tag{3}$$

AS estimates how many times the complexity of running AQ is reduced by introducing a splitting attribute when the number of classes is m.

If *m* is large, and the expression for AS can be approximated by:

$$r * \text{comp}(m^2 * p) / (m * (\text{comp-s} / \text{comp-a}) \tag{4}$$

which can be simplified further to:

$$r * comp(m * p) * f \quad (5)$$

where f = comp-a / comp-s.

Expression (5) states that the advantage of using a splitting attribute grows with r, m , p and f. The f factor represents the ratio of the decrease of AQ complexity due to the reduction of attributes in examples by 1 to the complexity of searching for a splitting attribute per class. Thus, the more classes and more examples are involved in a learning problem, the more advantageous is to learn a rule-tree than just rules, and to use a splitting attribute with a large domain.

Summarizing, the advantage of a ruletree when the number of classes is large is due to the fact that in each AQ run, only a small number of classes are used as contrasting classes, and the number of negative examples is correspondingly reduced. There is also an additional advantage in using fewer attributes in examples.

The disadvantage is the cost of determining a class splitting attribute. When the number of classed is sufficiently large, the advantages of building a ruletree should easily overcome this cost. Experimental studies are needed to determine conditions under which learning ruletrees is advantageous, and to which degree.

Given a family of training sets, the set of original attributes defining training events may not include a p-type splitting attribute. It may be possible, however, to construct a function of a subset of these attributes (a derived attribute) that constitutes a p-type splitting attribute. This will, of course, add to the overhead of searching for a class splitting attribute.

An alternative way is to seek an o-type attribute that minimizes the intersection of groups of classes. The above ideas have been incorporated into the algorithm AQrt described next.

4 ALGORITHM AQRT

Let us first consider the case of using a p-type splitting attribute. Let us assume that are m training example sets: E1, E2, E3,…,Em, defined by values n attributes of specified type and domains. Given is also a rule selection criterion, LEF that defines a ranking of learned rules. For the second part of the algorithm, assume that given is also a method for constructing new attributes (knowledge-driven or data-driven).

Step 1. Search for a p-type class splitting attribute among nominal, and then linear attributes in the original attribute set specified in the training data. In the case of linear attributes, seek range splitting points that define a class partitioning.

This step may end after finding the first class splitting attribute, or can continue to find a set of alternative attributes. In the latter case, the algorithm can build a ruletree for each alternative attribute, and then select the best ruletree according to some criterion (e.g., the simplest ruletree, and/or the ruletree with the highest performance accuracy on the testing set).

If the above search does not produce a class splitting attribute, apply an attribute construction method to determine a derived class splitting attribute. If a class splitting attribute has been found, proceed to Step 2.

If this process continues without success, and a termination condition is reached, e.g., the process exceeds the allocated computational resources, terminate. In this case, AQ is run its normal mode, that is, it is applied to all the classes, one by one.

Step 2. Assign the class splitting attribute to the root of the ruletree, and assign subsets of its values that split the classes to branches stemming from the root. If the class splitting attribute was selected from the original set of attributes, remove it from the training data. If the class splitting attribute was constructed, the algorithm may continue in one of two ways controlled by parameter *exlude* that is set by the user.

If *exlude = yes*, all attributes that are arguments of the function defining the constructed attribute are removed from the original attribute set. If *exlude = no*, the original attribute set remains without change. The *exclude* parameter is recommended to be set to *yes*, if the set of original attributes is relatively large and the allocated time for the algorithm execution is short; otherwise, to *no.*

Step 3. Run AQ separately for classes defined in each group of classes in the partition (in the example above, for cases 1,2,3).

The result of learning is a set of families of rulesets, each family describing classes in the group defined by the class splitting attribute. This procedure is similar to the way AQ handles structured output attributes (Kaufman and Michalski, 1996).

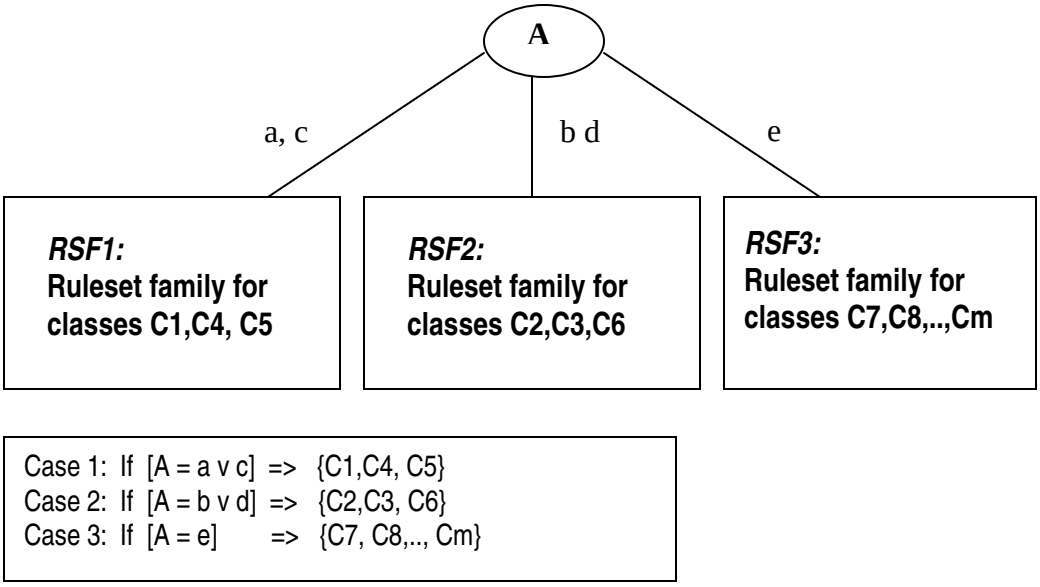

Figure 1. A ruletree with the class splitting attribute A.

To illustrate the algorithm, Figure 1 presents an attributional ruletree that would be generated for the example presented above. In this figure, the original multiclass problem has been split to subproblems, each involving learning a ruleset family for a smaller number of classes. As discussed in Section 3, running AQ for these subproblems represents a simpler problem than running AQ for all classes. Ruleset families, RSFi, i=1,2,3, inside of rectangular blocks may be

listed as attributional rules, or presented graphically as concept association graphs (Michalski and Kaufman, 2000; Kaufman and Michalski, 2000).

In the figure, RSFi, i=1,2,3 are ruleset families in which each rule is in the form

$$Class = j] \ <= \ Premise\text{-}j \tag{6}$$

where *Premise-j* is a product of attributional conditions.

The interpretation of a ruletree is very simple. Suppose there is a rule

$$Class = C4] \ <= \ Premise\text{-}C4 \tag{7}$$

in the block RSF1, then it would be interpreted as:

> *"If A is a or c in the event being classified, then assign the event to class C4, if Premise-C4 holds."*

If there is a censored rule in the RSFi (a rule with an exception clause), that is, for example, a rule:

$$[Class = C4] \ <= \ Premise\text{-}C4 \ \backslash\_ \ Exception\text{-}C4 \tag{8}$$

> *"If A is a or c in the event being classified, then assign the event to class C4, if Premise-C holds, except when Exception-C4 holds."*

5 USING O-TYPE SPLITTING ATTRIBUTE

In practical problems a p-type splitting attribute may not exist among original attributes, and constructing a derived attribute of this type may be computationally costly. In such situations, one may seek an o-type splitting attribute that maximizes some criterion of the quality of split. One can use for this purpose any of the existing methods for determining attribute quality, for example, gain ratio, gini index, promise (Baim, 1982), etc. Alternatively, one may define a criterion of class group disjointness, which expresses the degree to which an o-type attribute approximates a p-type attribute.

After determining the attribute that scores the highest on the chosen split quality measure, the attribute is assigned to the root, and the algorithm proceeds as in the case with p-type splitting attribute. The procedure is identical to determining the root and the first level of an ordinary decision tree. The difference from the case with a p-type splitting attribute is that a ruletree created with o-type splitting attribute will have some decision classes included in more than one group of classes defined the splitting attribute.

An advantage of using o-type splitting attributes is that every attribute could potentially be used as a class splitting attribute. A disadvantage is that the classes are not partitioned to disjoined sets, and the interpretation of such rule-trees is more complex.

6 CONCLUSION

The idea of attributional rule-trees represents a slight by potentially important modification of the current representation of ruleset families used for representing a decision structure in AQ type of

learning. The class splitting attribute can be viewed as an attribute that defines preconditions for applying ruleset families.

To implement such representation, a relatively simple modification of the current AQ algorithm is required. The main addition to AQ is the process of determining a class splitting attribute. This process can be computationally implemented in different ways, but is conceptually straightforward, if a p-type splitting attribute exists among original attributes, or one uses an o-type splitting attribute. If a derived class splitting attribute is to be constructed, the modification of the AQ algorithm is more involved, but still relatively simple. An initial implementation of the AQrt algorithm may not include this step.

REFERENCES

Baim, P., "The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learning Systems," *Reports of the Intelligent Systems Group*, ISG 82-1, UIUCDCS-F-82-898, Department of Computer Science, University of Illinois, Urbana, September, 1982.

Bloedorn E. and Michalski, R.S., "Data-Driven Constructive Induction," *IEEE Intelligent Systems, Special Issue on Feature Transformation and Subset Selection*, pp. 30-37, March/April, 1998.

Clark, P. and Niblett, T., "The CN2 Induction Algorithm," *Machine Learning* 3, pp. 261-283, 1989.

Craven, M. W. & Shavlik, J.W., "Rule Extraction: Where Do We Go from Here?". Department of Computer Sciences, University of Wisconsin, Machine Learning Research Group Working Paper 99-1, 1999.

Hong, J., Mozetic, I., Michalski, R.S., "AQ15: Incremental Learning of Attribute-Based Descriptions from Examples, the Method and User's Guide," *Reports of the Intelligent Systems Group*, University of Illinois at Urbana-Champaign, ISG 86-5, May, 1986.

IJCAI'95, *Materials of the Workshop on Machine Learning and Comprehensibility*, W22, Montreal, Canada, August 19, 1995.

Kaufman, K.A. and Michalski, R.S., "Learning in an Inconsistent World: Rule Selection in AQ19," *Reports of the Machine Learning and Inference Laboratory*, MLI 99-2, George Mason University, Fairfax, VA, 1999.

Kaufman K. and Michalski R.S., "A Knowledge Scout for Discovering Medical Patterns: Methodology and System SCAMP," *Proceedings of the Fourth International Conference on Flexible Query Answering Systems, FQAS'2000*, Warsaw, Poland, pp. 485-496, October 25-28, 2000.

Michalski, R. S., "A Variable-Valued Logic System as Applied to Picture Description and Recognition," in Nake, F. and Rosenfeld, A. (eds.), *Graphic Languages*, North-Holland Publishing Co., 1972.

Michalski, R.S., "Variable-Valued Logic and Its Applications to Pattern Recognition and Machine Learning," in Rine, D.C. (ed.), *Computer Science and Multiple-Valued Logic: Theory and Applications*, North-Holland Publishing Co., pp. 506-534, 1977.

Michalski, R.S, "A Theory and Methodology of Machine Learning, in Michalski, R.S, Carbonell, J.G. and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, pp. 83-134, 1983.

Michalski, R.S., “ATTRIBUTIONAL CALCULUS: A Logic and Representation for Natural Induction,” *Reports of the Machine Learning and Inference Laboratory*, George Mason University, Fairfax, VA (to appear 2004).

Michalski R.S. and Kaufman K., "Building Knowledge Scouts Using KGL Metalanguage," *Fundamenta Informaticae* , 40, pp 433-447, 2000.

Michalski, R.S. and Larson, J., “AQVAL/1 (AQ7) User's Guide and Program Description,” Report No. 731, Department of Computer Science, University of Illinois, Urbana, June 1975.

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N., “The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains,” *Proceedings of AAAI-86*, pp. 1041-1045, Philadelphia, PA, 1986.

Wnek, J. and Michalski, R.S., “Experimental Comparison of Symbolic and Subsymbolic Learning,” *HEURISTICS, The Journal of Knowledge Engineering*, Special Issue on Knowledge Acquisition and Machine Learning, Vol. 5, No. 4, pp. 1-21, 1992.

Wnek, J., Kaufman, K., Bloedorn, E. and Michalski, R.S., “Inductive Learning System AQ15c: The Method and User’s Guide,” *Reports of the Machine Learning and Inference Laboratory*, MLI 95-4, George Mason University, Fairfax, VA, 1995.