94-1

SYMBOLIC LEARNING OF M-OF-N CONCEPTS

Janusz Wnek and Ryszard S. Michalski

MLI 94-1

April 1994

Symbolic Learning of M-of-N Concepts

Abstract

This paper addresses a class of learning problems that require a construction of descriptions that combine both M-of-N rules and traditional Disjunctive Normal form (DNF) rules. The presented method learns such descriptions, which we call *conditional M-of-N rules*, using the hypothesis-driven constructive induction approach. In this approach, the representation space is modified according to patterns discovered in the iteratively generated hypotheses. The need for the M-of-N rules is detected by observing "exclusive-or" or "equivalence" patterns in the hypotheses. These patterns indicate symmetry relations among pairs of attributes. Symmetrical attributes are combined into maximal symmetry classes. For each symmetry class, the method constructs a "counting attribute" that adds a new dimension to the representation space. The search for hypothesis in iteratively modified representation spaces is done by the standard AQ inductive rule learning algorithm. It is shown that the proposed method is capable of solving problems that would be very difficult to tackle by any of the traditional symbolic learning methods.

Key words: Constructive induction, decision rules, conditional M-of-N rules, XOR patterns

Acknowledgements

This research was conducted in the Center for Artificial Intelligence at George Mason University. The Center's research is supported in part by the Advanced Research Projects Agency under Grant No. N00014-91-J-1854, administered by the Office of Naval Research, and the Grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, in part by the Office of Naval Research under Grant No. N00014-91-J-1351, and in part by the National Science Foundation under Grants No. IRI-9020266, CDA-9309725, and DMI-9496192.

1 Introduction

Constructive induction (CI) can be viewed as a process of conducting two intertwined searches: one–for the most appropriate representation space, the second—for the "best" inductive hypothesis in the space. Search for the representation space involves applying constructive induction operators ("CI operators") that modify the representation space by generating new dimensions, removing dimensions, or changing their quantization level. Since there is no limit on what operators can be applied, the search is very large. Therefore, a central research issue in constructive induction is the development of rules and heuristics for applying CI operators.

This paper concerns a class of learning problems that cannot be satisfactorily solved by symbolic methods that construct DNF-type descriptions using original attributes, because such descriptions would be prohibitively long. Specifically, such problems require a construction of descriptions that involve "counting properties" (e.g., that M properties out of N possible properties are present in an object), which may be additionally combined with logical conditions (e.g., in the DNF form). Problems of this type occur in many real-world problems (e.g., Spackman, 1988; Towell & Shavlik, 1994). The proposed solution is based on the application of a new type of constructive induction rule, "counting attribute generation rule," which explores an attribute symmetry in generated hypotheses. Such a symmetry is indicated by the presence of the "exclusive-or" or "equivalence" patterns in the hypothesis.

It is shown that *M-of-N* concepts are easy to detect in descriptions generated by the AQ learning system, because they form exclusive-or patterns *(XOR-patterns)*. The XOR-pattern characterizes relationship between two attributes, in a way that only one of the two binary attributes can be present. This translates to *exactly 1-of-2*, which is a specific form of M-of-N concept. The presence of an *XOR (or non-equivalence symmetry)* relationship among two or more binary attributes signals that these attributes represent independent object properties and, therefore counting them (i.e., how many of them are true) is likely to produce a new relevant descriptor. On the other hand, the presence of the *equivalence symmetry (EQ)* among two or more attributes signals that they are interdependent and can be replaced by only one attribute.

By observing the relationships among attributes, the proposed method combines ideas of logic and arithmetic. The introduced "counting attribute" captures a conceptual transition from logic to arithmetic.

This work was inspired by the failure of well-known symbolic learning systems in solving the MONK2 problem used in an international competition of learning programs (Thrun et al, 1991). The MONK2 problem was to learn the concept, "exactly two of the six attributes have their first value," which is a special case of the M-of-N concept. There has been several efforts concerned with learning M-of-N concepts. For example, the system CRLS learns *M-of-N* rules by employing non-equivalence symmetry bias and criteria tables (Spackman, 1988), ID-2-of-3 incorporates *M-of-N* tests in decision trees (Murphy & Pazzani, 1991), AQ17-DCI (Bloedorn & Michalski, 1991) employs a variety of operators to construct new attributes, NEITHER-MofN (Baffes & Mooney, 1993) is able to refine M-of-N rules by increasing or decreasing either of M or N. The idea of "counting attributes" and the proposed method to derive such attributes is related to research on detecting symmetry in Boolean functions of many variables (Michalski, 1969), and implementation of SYM programs (Jensen, 1975).

2 Class-patterns

Our earlier work on AQ-HCI method involved search for three types of patterns, value-patterns, condition-patterns, and rule-patterns. Value-patterns aggregate subsets of attribute values that often co-occur in a given description. Condition-patterns represent a conjunction of two or more elementary conditions that frequently occur in a ruleset of a given concept. A rule-pattern is a rule or a subset of rules. Detecting such patterns and using them for expanding the representation space has shown to be effective in improving performance accuracy in DNF-type problems (Wnek & Michalski, 1994b).

Following the scheme of different types of patterns, it was conjectured that there may exist *class-patterns*. Such patterns would represent relations that are common for subsets of learned classes (concepts). The XOR-patterns satisfy this definition. They apply both to examples of a M-of-N concept as well as its negation. This would imply that new descriptor is needed to discriminate between classes that have such property.

Following the notation introduced by Michalski (1983), we will formulate a rule for generating "counting attribute." To this end, we need to introduce some notation. We use the general term "symmetry" for two relations: "exclusive-or" or "equivalence." Since one is the negation of the other, from now on, we will discuss primarily the "exclusive-or" relation.

Relational conditions

Let Ci be relational conditions on values of single attributes (selectors), such as xi = 5, xi > 3, xi=2..5. Relational conditions evaluate to *true* (1) or *false* (0). Such conditions are building blocks in the rules generated by the rule learning system AQ15 employed in the proposed method (Michalski et al., 1986).

Binary symmetry class

Let R1 and R2 be two conjunctive rules in a ruleset representing a hypothesis. Suppose further that R1 can be represented as C_i & ~C_j & CTX_1 and R2 as ~C_i & C_j & CTX_2, where CTX_1 and CTX_2 are "context" conditions, expressed in the form of a conjunction of zero or more relational conditions. It is said that C_i and C_j represent a "binary symmetry class," if CTX_1 and CTX_2 are in a *subsumption* relation, that is $CTX_1 = CTX_2$ & CTX_3 or $CTX_2 = CTX_1$ & CTX_3, where CTX_3 is a context condition. In other words, two attributes or relational conditions constitute a symmetry class if an XOR relationship binding them constitutes a pattern.

Maximum symmetry class

Given two binary symmetry classes, BC_1 = {C_{i1}, C_{j1}} and BC_2 = {C_{i2}, C_{j2}), they can be combined into ternary symmetry class if they have non empty intersection. The resulting class consists of three relational conditions, the one that is common in both classes, and the two that do not appear in the other class.

The maximum symmetry class is the maximum number of relational conditions that can be combined together.

Counting attribute generation rule

Given a k-ary symmetry class, generate a *counting attribute* CA that stands for the expression [C1+C2+...+Ck] that sums up evidence given by relational conditions. The domain of the attribute is an integer interval from 0 to k. Values of the counting attribute represent the number of relational conditions (attributes) that hold for the given concept example.

The counting attribute represents an arithmetic sum of two or more relational conditions. (Each relational condition evaluates to 1 or 0.) When the relational conditions Ci use binary attributes then the "counting attribute" agglomerates evidence described by those attributes. If the conditions use multivalued attributes then the evidence agglomeration is carried over whole relational conditions.

Note: Systems that use different representational formalisms for data and hypotheses may not be able to detect class-patterns in hypotheses. In such cases, data could be a source for finding these patterns. In the case of the AQ learning method that uses the VL1 representational formalism (a form of propositional calculus) for both data and hypothesis representation, class-patterns are passed from data to hypotheses. Initial examination of descriptions generated by FOIL (Quinlan, 1990) indicates that they also contain XOR-patterns (see Appendix Figs A1 & A2). The class-patterns are different from intra-construction and inter-construction operators used in Duce and CIGOL (Muggleton, 1987; Muggleton & Buntine, 1988).

3 The Relationship Between Logic and Arithmetic

For several decades logic circuits were used in digital computers. It is well known that such circuits not only facilitated logical operations but also arithmetic ones. Actually, all possible arithmetic operations, starting with addition through multiplication, ending with the most complex functions, are in fact represented by logic circuits. The basic link between logic and arithmetic is the logical XOR operator. Fig. 1 shows the relationship between "XOR" and "ADD" operators.

x	y	Carry/AND	Sum/XOR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

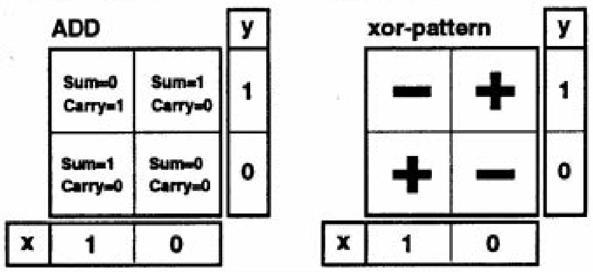

Fig. 1. (a) Truth table for addition of two binary digits, (b) visualization of ADD operator in a binary domain, (c) visualization of an XOR-pattern

Carry can be represented as logical AND, and *Sum* can be represented as logical XOR. Given the "ADD" operator we can implement multiplication, and other arithmetic operators.

In the case of concept learning, the presence of an *XOR* relationship among two or more binary attributes signals that these attributes represent independent object properties, therefore counting them (i.e., how many of them are true) is likely to produce a relevant new attribute.

Grouping attributes representing pairwise such patterns into *XOR-classes* is based on the property of the XOR relationship. Fig. 2 gives an outline of the algorithm.

The determination of the DNF concept representation is done using AQ algorithm. By examining learned concept descriptions XOR-patterns are detected. XOR-related attributes are grouped into XOR-classes. For example, if following patterns were detected, x1 XOR x2, x2 XOR x5, x5 XOR x7, x2 XOR x4, the following XOR-class is created {x1, x2, x4, x5, x7}. XOR relationship holds the entire class because it holds for all attribute pairs in the set. Next, for each XOR-class a "counting" attribute is created. The name of such an attribute reflects names of attributes from the XOR-class. Values of the counting attribute represent the number of properties that hold for the learned concept. They are established based on direct counting of values of attributes from the XOR-class. For the above example, the attribute CA <:: x1+x2+x4+x5+x7 is created. Its domain is an integer interval from 0 to 5.

Counting attributes used within DNF expressions allow for representing various forms of M-of-N concepts. Fig. 3 shows examples of such concepts. Note that if the target concept includes an M-of-N rule as a part of its description, the above method will generate a conditional M-of-N rules. Such a rule will contain a M-of-N rule with additional conditions represented as a rule set (DNF expression). This is a consequence of using AQ.

```
1. Determine a DNF concept representation. If the expression is sufficiently simple, STOP.
2. Detect XOR-patterns in the learned concept description.
3. If XOR-patterns do not exist, then STOP. Otherwise:
       Build maximum XOR-classes
       For each XOR-class, generate a "counting attribute"
       Project data to the new representation space
       Go to 1.
```

Fig. 2. Algorithm for Changing the Representation Space Based on XOR-patterns

Parity 5 expressed using 5 binary attributes: x1-x5.
[CA = 0, 2, 4], where CA <:: x1+x2+x3+x4+x5

3-of-6
[CA >= 3], where CA <:: x1+x2+x3+x4+x5+x6

exactly 3-of-6
[CA = 3], where CA <:: x1+x2+x3+x4+x5+x6

MONK2 problem "exactly two of six attributes have their first value"
[CA = 2], where CA <:: c1+c2+c3+c4+c5+c6

XOR (exactly 1-of-2)
[CA = 1], where CA <:: x1+x2

Fig. 3. Concept representation using counting attributes in DNF

4 Illustrative Example: MONK2 problem

The concept to be learned is the MONK2 problem (Thrun et al., 1991; Wnek & Michalski, 1994a). Fig. 8a shows a diagram visualizing the problem. The total number of possible instances in the representation space is 432. In the diagram, the target concept is represented by 142 instances (shaded area). The remaining 290 instances represents the negation of the concept. The training set, an input to the learning system is represented by 64 positive (+) and 105 negative (-) examples. The data contains no noise.

4.1 Learning in the Original Representation Space

MONK2 problem is hard for symbolic learning systems. In fact, none of the 18 symbolic learners taking part in the international competition learned the MONK2 concept (Thrun et al., 1991). This problem is also hard for the AQ15 program. The descriptions generated may slightly vary with different parameter settings but all have the following characteristics: they are inaccurate (about

75%), they consist of many rules, rules use many conditions. The example[1] of program's output is presented in Fig 4. As many as 16 rules generalize only 64 positive examples. Almost all rules involve all six attributes in describing the concept. It means that all attributes are equally important in the concept description, and moreover, the logical operators (and, or) are not capable to capture meaningful relationships.

4.2 Representation Space Transformation—Iteration #1

In this description however, there are certain patterns that are easy to detect. Many conditions involving the same attribute tend to use the same set of attribute values and form value-patterns (Wnek and Michalski, 1994). For example, conditions involving attribute HS use the following groupings of values {s,o} nine times, {r} four times, {r,o} and {o} once. Taking into consideration that HS attribute has three values {r,s,o}, this value-pattern is suggestive that the division of this set into two subsets, {r} and {s,o} should be meaningful. Similarly, values of other attributes could also be grouped. The AQ-HCI method changes the representation space according to the following new attribute definitions (Fig. 5). (To contrast the new representation with the original one we also substitute new attributes for SM and TI. Since these attributes are binary this is only a change in names.)

```
1   [HS=r]     & [BS=s,o] & [SM=y] & [HO=f,b] & [JC=y,g,b] & [TI=n]   (t:9, u:9)
2   [HS=s,o]   & [BS=s,o] & [SM=y] & [HO=f,b] & [JC=y,g,b] & [TI=y]   (t:9, u:9)
3   [HS=s,o]   & [BS=s,o] & [SM=n] & [HO=f,b] & [JC=r]     & [TI=y]   (t:7, u:7)
4   [HS=r]     & [BS=s,o] & [SM=n] & [HO=f,b] & [JC=y,g,b] & [TI=y]   (t:5, u:5)
5   [HS=r,o]   & [BS=r]   &          [HO=s,f] & [JC=g,b]   & [TI=n]   (t:5, u:4)
6   [HS=s,o]   & [BS=r,o] & [SM=y] & [HO=s,b] & [JC=g]     & [TI=n]   (t:4, u:4)
7   [HS=s,o]   & [BS=s,o] & [SM=n] & [HO=s]   & [JC=y,g,b] & [TI=y]   (t:4, u:4)
8   [HS=s,o]   & [BS=r]   & [SM=y] & [HO=f,b] & [JC=y]     & [TI=n]   (t:4, u:4)
9                [BS=r,o] & [SM=n] & [HO=s]   & [JC=y,g]   & [TI=n]   (t:4, u:3)
10  [HS=s,o]   & [BS=r,o] & [SM=n] & [HO=s,b] & [JC=r]     & [TI=n]   (t:3, u:3)
11  [HS=s,o]   & [BS=r]   & [SM=n] & [HO=f,b] & [JC=y]     & [TI=y]   (t:3, u:3)
12  [HS=s,o]   & [BS=s,o] & [SM=y] & [HO=f,b] & [JC=r]     & [TI=n]   (t:2, u:2)
13  [HS=r]     & [BS=s,o] & [SM=n] & [HO=f,b] & [JC=r]     & [TI=n]   (t:2, u:2)
14  [HS=s,o]   & [BS=s]   & [SM=y] & [HO=s]   & [JC=y,b]   & [TI=n]   (t:2, u:2)
15  [HS=o]     & [BS=s]   & [SM=y] & [HO=s]   & [JC=g]     & [TI=n]   (t:1, u:1)
16  [HS=r]     & [BS=r]   & [SM=n] & [HO=b]   & [JC=y]     & [TI=n]   (t:1, u:1)
```

Fig. 4. The MONK2 concept learned in the original representation space.

```
(c1 = 1) <:: [HS=r]         (c2=1) <:: [BS=r]          (c3=1) <:: [SM=y]
(c1 = 0) <:: [HS=s,o]       (c2=0) <:: [BS=s,o]        (c3=0) <:: [SM=n]

(c4 = 1) <:: [HO=s]         (c5=1) <:: [JC=r]          (c6=1) <:: [TI=y]
(c4 = 0) <:: [HO=f,b]       (c5=0) <:: [JC=y,g,b]      (c6=0) <:: [TI=n]
```

Fig. 5. Attributes constructed from value-patterns.

The transformed learning task is visualized in Fig. 8b. The new representation space has become significantly smaller, there are only 64 instances in the new space vs. 432 instances in the original representation space. The number of attributes is the same but all of them have fewer values. The number of instances representing the target concept is 15, therefore in the worst case, the number of rules required to describe the concept is 15. This is a reduction in description complexity in

[1] The following parameters were used: disjoint covers mode, so the generated positive and negative descriptions are disjoint over areas not represented by training examples; the most specific rules are generated; during rule generation 10 alternative solutions are considered (beam width); among those rules, one rule is selected that best satisfies the default criteria, i.e. maximizes the number of newly covered (not covered by previous rules) examples.

comparison to the original representation space. Each instance in the new space represents from 1 to 24 instances that were mapped from the original space. The transformation does not cause ambiguity in the new representation space, i.e., each new instance represents instances of the same class, either positive or negative. For more details see (Wnek, 1993).

In this representation space, all possible positive examples are present, and only 13 negative examples are missing (in the original space only 64 positive examples out of 142 are present). It seems that learning should give better results. However, the AQ15 learning program still generates a long and inaccurate description of the concept (Fig. 6). Errors are caused by overly general rule #1. This rule covers not only two positive examples but also covers two negative instances.

1	[c1=0]	&			[c3=0]	&			[c5=1]	&	[c6=0]	(t:2, u:2)
2	[c1=1]	&	[c2=1]	&	[c3=0]	&	[c4=0]	&	[c5=0]	&	[c6=0]	(t:1, u:1)
3	[c1=1]	&	[c2=0]	&	[c3=1]	&	[c4=0]	&	[c5=0]	&	[c6=0]	(t:1, u:1)
4	[c1=1]	&	[c2=0]	&	[c3=0]	&	[c4=1]	&	[c5=0]	&	[c6=0]	(t:1, u:1)
5	[c1=1]	&	[c2=0]	&	[c3=0]	&	[c4=0]	&	[c5=0]	&	[c6=1]	(t:1, u:1)
6	[c1=0]	&	[c2=1]	&	[c3=1]	&	[c4=0]	&	[c5=0]	&	[c6=0]	(t:1, u:1)
7	[c1=0]	&	[c2=1]	&	[c3=0]	&	[c4=1]	&	[c5=0]	&	[c6=0]	(t:1, u:1)
8	[c1=0]	&	[c2=1]	&	[c3=0]	&	[c4=0]	&	[c5=0]	&	[c6=1]	(t:1, u:1)
9	[c1=0]	&	[c2=0]	&	[c3=1]	&	[c4=1]	&	[c5=0]	&	[c6=0]	(t:1, u:1)
10	[c1=0]	&	[c2=0]	&	[c3=1]	&	[c4=0]	&	[c5=1]	&	[c6=0]	(t:1, u:1)
11	[c1=0]	&	[c2=0]	&	[c3=1]	&	[c4=0]	&	[c5=0]	&	[c6=1]	(t:1, u:1)
12	[c1=0]	&	[c2=0]	&	[c3=0]	&	[c4=1]	&	[c5=0]	&	[c6=1]	(t:1, u:1)
13	[c1=0]	&	[c2=0]	&	[c3=0]	&	[c4=0]	&	[c5=1]	&	[c6=1]	(t:1, u:1)
14	[c1=1]	&	[c2=0]	&	[c3=0]	&	[c4=0]	&	[c5=1]	&	[c6=0]	(t:1, u:1)

Fig. 6. Concept learned in the representation space developed in iteration #1.

4.3 Representation Space Transformation—Iteration #2

The description obtained after the first transformation of the representation space is more accurate but still very complex (Figs. 6, 8b). Therefore, search for a better representation is continued, and the XOR-patterns are found. Fig. 7 lists examples of pairs of rules with XOR-patterns. Due to the properties of the XOR relation, this is a sufficient evidence that all six attributes are in such relation. A new counting attribute is constructed. It is defined as CA <:: c1+c2+c3+c4+c5+c6. Its domain is an integer interval from 0 to 6. Summing up values in the XOR-patterns gives exact value 2. The final concept description is [CA = 2], i.e., exactly two of the six attributes are present. Fig. 8c visualizes the final representation space and the concept learned.

Rule No						
3	**[c1=1]**	**[c2=0]**	[c3=1]	[c4=0]	[c5=0]	[c6=0]
6	**[c1=0]**	**[c2=1]**	[c3=1]	[c4=0]	[c5=0]	[c6=0]
3	[c1=1]	[c2=0]	**[c3=1]**	[c4=0]	[c5=0]	**[c6=0]**
5	[c1=1]	[c2=0]	**[c3=0]**	[c4=0]	[c5=0]	**[c6=1]**
2	**[c1=1]**	[c2=1]	[c3=0]	**[c4=0]**	[c5=0]	[c6=0]
7	**[c1=0]**	[c2=1]	[c3=0]	**[c4=1]**	[c5=0]	[c6=0]
6	[c1=0]	**[c2=1]**	[c3=1]	[c4=0]	**[c5=0]**	[c6=0]
10	[c1=0]	**[c2=0]**	[c3=1]	[c4=0]	**[c5=1]**	[c6=0]
10	**[c1=0]**	[c2=0]	**[c3=1]**	[c4=0]	[c5=1]	[c6=0]
14	**[c1=1]**	[c2=0]	**[c3=0]**	[c4=0]	[c5=1]	[c6=0]

Fig. 7. Example of a necessary set of XOR-patterns to create CA <:: c1+c2+c3+c4+c5+c6 attribute

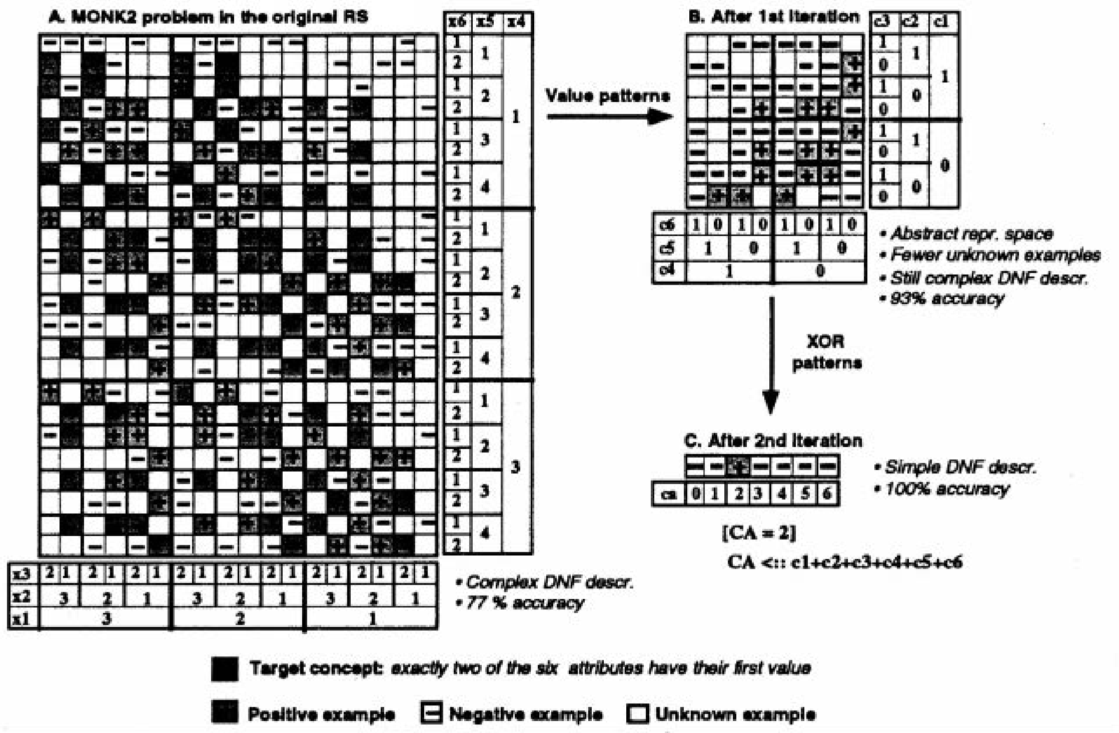

Fig. 8. Learning the MONK2 problem in two iterations of the AQ-HCI method

5 Conclusions

This paper demonstrated that M-of-N relations are easy to detect and learn from descriptions generated by the AQ inductive learning system, because they form XOR-patterns. The existence of such patterns in learned descriptions speeds their detection but this is not necessary for the application of the method with other learning systems. If a learning system does not form XOR-patterns in descriptions, because of a different syntax in different representational formalisms, then the input data should be examined for XOR-patterns.

The *counting attribute generation rule* was introduced and applied using algorithm for changing representation space based on XOR-patterns. By observing the relationships among attributes, the proposed method combines ideas of logic and arithmetic. The introduced "counting attribute" captures a conceptual transition from logic to arithmetic.

It is shown that the proposed method is capable of learning *conditional M-of-N rules,* using the hypothesis-driven constructive induction approach. As an illustrative example, the MONK2 learning problem was solved with 100% accuracy, and "minimal" complexity of the description. The method gives a good promise for its applicability in real-world problems carrying M-of-N relationships. The study confirming such applicability is in progress.

References

Baffes, P. T. and Mooney, R. J., "Symbolic Revision of Theories with M-of-N Rules," *Proceedings of the Second International Workshop on Multistrategy Learning*, Michalski, R.S. and G. Tecuci, Harpers Ferry, WV, pp. 69-75, 1993.

Bloedorn, E. and Michalski, R.S., "Data Driven Constructive Induction in AQ17-PRE: A Method and Experiments," *Proceedings of the Third International Conference on Tools for AI.* San Jose, CA, 1991.

Jensen, G.M., "Determination of Symmetric VL1 Formulas: Algorithm and Program SYM4," *M.S. Thesis*, Report No. UIUCDCS-R-75-774, Department of Computer Science, University of Illinois, Urbana-Champaign, December 1975.

Michalski, R.S., "Recognition of Total or Partial Symmetry in a Completely or Incompletely Specified Switching Function," *Proceedings of the IV Congress of the International Federation on Automatic Control (IFAC)*, Vol. 27, pp. 109-129, Warsaw, June 16-21, 1969.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), TIOGA Publishing, Palo Alto CA, 1983.

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, Morgan Kaufmann, San Mateo, CA, 1986.

Muggleton, S., "Duce, an Oracle-Based Approach to Constructive Induction," *Proceedings of IJCAI-87*, pp. 287-292, Morgan Kaufmann, San Mateo, CA, 1987.

Muggleton, S. and Buntine, W., "Machine Invention of First Order Predicates by Inverting Resolution," *Proceedings of the 5th International Conference on Machine Learning*, pp. 339-352, Morgan Kaufmann, San Mateo, CA, 1988.

Murphy, P. M. and Pazzani, M. J., "ID2-of-3: Constructive Induction of M-of-N Concepts for Discriminators in Decision Trees," *Proceedings of the 8th International Workshop on Machine Learning*, Evanston, Ill., pp. 183-187, 1991.

Quinlan, J.R., "Learning Logical Definitions from Relations," *Machine Learning*, Vol. 5, No. 3, pp. 239-266, 1990.

Spackman, K.A., "Learning Categorical Decision Criteria in Biomedical Domains," *Proc. of the 5th International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, pp. 36-46, 1988.

Thrun, S.B., Bala, J., Bloedorn, E., Bratko, I., Cestnink, B., Cheng, J., DeJong, K.A., Dzeroski, S., Fahlman, S.E., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R.S., Mitchell, T., Pachowicz, P., Vafaie, H., Van de Velde, W., Wenzel, W., Wnek, J. and Zhang, J., "The MONK's Problems: A Performance Comparison of Different Learning Algorithms," *Technical Report*, Carnegie Mellon University, December 1991.

Towell, G. G. and Shavlik, J. W., "Refining Symbolic Knowledge Using Neural Networks," in *Machine Learning: A Multistrategy Approach* , Vol. IV, Michalski, R.S. and G. Tecuci, Morgan Kaufmann, San Mateo, CA, pp. 1994.

Wnek, J. *Hypothesis-driven Constructive Induction*. Ph.D. Dissertation, School of Information Technology and Engineering, George Mason Univ., Fairfax, VA, University Microfilms Int., Ann Arbor, MI, 1993.

Wnek, J. and Michalski, R.S., "Comparing Symbolic and Subsymbolic Learning: Three Studies," in *Machine Learning: A Multistrategy Approach*, Vol. 4., R.S. Michalski and G. Tecuci (Eds.), Morgan Kaufmann, San Mateo, CA, 1994a.

Wnek, J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments," *Machine Learning*, Vol. 14, No. 2, pp. 139-168, 1994b.

Appendix

```
FOIL 6.1   [February 1994]
--------

    Options:
        verbosity level 0
        minimum clause accuracy 50%

Relation is_MONK2

----------
is_MONK2:

***  Warning: the following definition
***  matches 9 tuples not in the relation
***  does not cover 1 tuple in the relation

1.is_pos(a0,B,c0,D,e1,f0).
2.is_pos(A,b1,C,D,E,F) :- C<>c1, E<>e1, F<>f1.
3.is_pos(A,b0,C,D,E,F) :- D<>d1, A<>a1, E<>e0.
4.is_pos(A,B,c1,d0,E,F) :- E<>e1, B<>b1.
5.is_pos(A,b0,C,D,E,F) :- E<>e1, C<>c1, F<>f0.
6.is_pos(A,b0,c0,D,E,f0) :- A<>a0.
7.is_pos(A,B,C,D,E,F) :- E<>e1, A<>a1, B<>b0, D<>d1.

Time 2.6 secs
```

Fig. A1. Output from FOIL (with added rule numbers) for MONK2 problem in an abstracted space

```
2.is_pos(A,b1,C,D,E,F) :- C<>c1, E<>e1, F<>f1.
5.is_pos(A,b0,C,D,E,F) :- C<>c1, E<>e1, F<>f0.

3.is_pos(A,b0,C,D,E,F) :- A<>a1, D<>d1, E<>e0.
7.is_pos(A,B,C,D,E,F) :- A<>a1, B<>b0, D<>d1, E<>e1.
```

Fig. A2. Examples of XOR-patterns found in MONK2 description generated by FOIL