The Fourth Workshop on Intelligent Information Systems (WIS '95),
Augustow, Poland, June 5-10, 1995.

Learning Hybrid Concept Descriptions

Ryszard S. Michalski* and Janusz Wnek

Center for Machine Learning and Inference
George Mason University
Fairfax, VA 22030, USA

*Also with the Institute of Computer Science,
Polish Academy of Sciences, Warsaw, Poland

Abstract

Most symbolic learning methods are concerned with learning concept descriptions in the form of a decision tree or a set of rules expressed in terms of the originally given attributes. For some practical problems, these methods are inadequate because they cannot learn conditions that require counting of some object properties. Such problems occur, for example, in engineering, economy, medicine and software engineering. This paper describes a method for learning *hybrid descriptions* that combine logic-type and arithmetic-type properties. The presented method builds hybrid descriptions in the form of *conditional counting rules*, which are logic-type (DNF) expressions with *counting conditions* (expressing a relationship involving a count of some object properties). The method employs a *constructive induction* approach in which the learning system performs two intertwined searches: one—for the most appropriate knowledge representation space, and second—for the "best" hypothesis in the space. The first search is done by determining maximum symmetry classes of binary attributes in the initial DNF-type hypotheses, and extending the initial representation space by *counting attributes* that correspond to these symmetry classes. The search for the "best" hypothesis in so extended representation space is done by a standard AQ inductive rule learning program. It our experiments, the proposed method learned simple and accurate concept descriptions when conventional learning methods failed.

1 INTRODUCTION

Most inductive learning methods search for the concept description in the same representation space in which concept examples are expressed. The underlying assumption is that the originally given representation space—as defined by the attributes or terms used in the examples—is adequate for the problem at hand. Therefore, the learning process in these methods concentrates on determining the "best" hypothesis in the originally given space.

In many practical problems, however, the above assumption does not hold, as the original attributes or descriptive terms may be insufficiently relevant for the given learning problem. To address this problem, constructive induction learning methods split the learning process into two intertwined searches: one–for the most appropriate representation space for the given learning problem, and second—for the "best" inductive hypothesis in the newly created representation space (Michalski and Wnek, 1993; Wnek and Michalski, 1994b).

The first search is thus concerned with the representation space design or improvement. This process is performed by applying *constructive induction* operators that modify the representation space to maximize some measure of the representation space quality. This modification can involve a combination of three types of representation space transformations: generating new, more relevant dimensions (by inventing new descriptive concepts), removing less relevant dimensions (representation space reduction or "feature selection") or changing the quantization level of dimensions (dimension abstraction). Because there are many ways in which such operators can be applied (especially of the first type), the search through the possible representation space transformations can be potentially unmanageable. Therefore, the central research issue in this approach is the invention of meta-rules and heuristics for applying constructive induction operators.

This paper uses ideas of constructive induction to develop a method for a class of learning problems that cannot be satisfactorily solved by conventional symbolic methods that construct DNF-type descriptions (decision trees or decision rules). Specifically, we address problems in which relevant descriptive concepts involve counting the presence of some object properties. Problems of this class require DNF-type descriptions that are prohibitively long. For example, a decision tree correctly representing the so-called MONK2 problem requires 434 nodes, while a solution involving a counting property requires only one condition.

The presented method builds a form of *hybrid* descriptions that combine logical-type (DNF) and arithmetic-type conditions. Specifically, the method learns *conditional counting rules*, which are DNF descriptions with “counting conditions”. A simple example of a conditional counting rule is the M-of-N concept. Here, the counting condition is “at least M out of N properties of some kind must present in an object,” and the logic-type condition is null. Problems of this type occur in many real-world problems, for example, in medicine (Spackman, 1988), planning (Callan & Utgoff, 1991), game playing (Fawcett & Utgoff, 1991), biology (Baffes & Mooney, 1993) and biochemistry (Towell & Shavlik, 1994).

The proposed method can learn DNF descriptions (decision rules), one or more counting conditions (e.g., M-of-N concepts), or any combination of the logic-type DNF descriptions with counting conditions. Thus, it extends the class of learning problems to which conventional symbolic learning methods apply. The key idea of the method is to detect the need for and then construct a "counting attribute" that is added to the representation space. The idea stems from the early work on the *counting attribute generalization rule* (Michalski, 1983).

To detect the need for a counting attribute, the method searches for maximum symmetry relations among attributes in the DNF concept descriptions constructed in each iteration of the method. For each maximum symmetry group, one new counting attribute is constructed. Counting attributes are added to the representation space and a new iteration begins. The process ends where a satisfactory concept description (according to dome description quality criterion) has been found or the allocated computational resources have been exhausted. The introduction of "counting attributes" to the concept representation can be viewed as a conceptual change from logic-style to arithmetic-style descriptions. The next section explains briefly the origins of this method and characterizes its relation to other related work.

2. RELATION TO THE PAST WORK

This research was inspired by the failure of well-known symbolic learning systems to solve the MONK2 problem used in an international competition of learning programs (Thrun et al, 1991; Wnek & Michalski, 1994a). The MONK2 problem was to learn from a set of examples the following target concept: "Exactly two of six given attributes have their first value for each concept instance." All conventional symbolic learning methods (which learn a DNF-type description, such as decision tree or a set of decision rules), performed poorly on this problem.

In order to solve this problem, a learning system needs to be able to form what we call a *counting condition*.. There have been several early efforts in this research direction. For example, the CRLS system learns *M-of-N* rules (a special case of a counting condition) by employing non-equivalence symmetry bias and criteria tables (Spackman, 1988). The ID-2-of-3 system incorporates *M-of-N* tests in decision trees (Murphy & Pazzani, 1991). The AQ17-DCI program employs a variety of data-driven operators to construct new attributes (Bloedorn & Michalski, 1991). Fawcett and Utgoff (1991) used feature representation similar to the Michalski's (1983) counting arguments rule to expand a representation space by summing up the number of distinct values of the set of variables that can satisfy a set of conditions.

Such a modified representation allows the expression of the "number of ways" in which conditions can be satisfied.

Callan and Utgoff (1991) used a restricted form of the counting arguments rules to create a numeric function from a Boolean expression that begins with a universal quantifier. The function calculates the percentage of permutations of variable bindings satisfying the Boolean expression. Such a function is useful because it indicates the degree to which a subgoal is satisfied in a given state. More recently, the NEITHER-M-of-N system refined M-of-N rules by increasing or decreasing either M or N (Baffes & Mooney, 1993).

The proposed method is based on the idea of "counting arguments" generalization rule (Michalski, 1983), the algorithm for detecting symmetry in Boolean functions of many variables (Michalski, 1969), and its subsequent implementation in the SYM program (Jensen, 1975). This work is also related to the work by Seshu (1989), who studied the problem of learning concept descriptions that are logically equivalent to a combination of DNF and M-of-N rules but in the context of decision tree learning. He calls such learning problems *unsplittability or parity problems*. Seshu's solution involves determining "best" XOR combinations in a randomly selected subset of the original attributes. Seshu has shown that such a method improves the prediction accuracy of learned descriptions. Disadvantages of the method are that it creates attributes without clear meaning and, due to the randomness of attribute selection, it may miss important relationships.

3 THE AQ-HCI METHOD

An important property the proposed method is that it is hypothesis-driven, which means that the change of the concept representation space is based on the analysis of hypotheses generated in that space (in contrast to data-driven methods that propose changes in the representation space based on the analysis of examples).

Another property is the way in which the method constructs changes of the representation space. In each iteration, the method searches for *attribute symmetry patterns* in the hypotheses generated in this iteration, and then uses these patterns to modify the representation space for the next iteration.

The proposed method is a new version of the AQ-HCI approach that combines an inductive rule learning algorithm A^q (Michalski, 1969b) implemented in the AQ15c system (Michalski et al., 1986; Wnek et al., 1995) with a procedure for an iterative hypothesis-driven constructive induction (HCI) for transforming the representation space. The AQ-HCI approach is based on detecting *patterns* in the generated hypotheses (concept descriptions).

By a pattern we mean a component of a concept description (in this paper, a set of decision rules) that covers a significant number of positive training examples and only a small number of negative examples. The meaning of "significant" and "small" is defined by a user-specified parameter. Our earlier work in this area involved the search for three types of patterns: value-patterns, condition-patterns, and rule-patterns. Value-patterns aggregate subsets of co-occurring attribute values in a description into single, more abstract values. Condition-patterns represent a conjunction of two or more elementary conditions that frequently occur in a ruleset of a given concept. A rule-pattern consists of a set of rules. Detecting such patterns and using them for expanding the representation space have proven to be highly effective in improving the performance accuracy in DNF-type learning problems (Wnek & Michalski, 1994b).

Following the scheme of different types of patterns, it was conjectured that there may also exist *class-patterns* (Wnek, 1993). Such patterns would represent relations that are common for subsets of learned classes (concepts). This led to the introduction of XOR-patterns that represent a special case of class patterns. Such patterns facilitate the determination of M-of-N concepts, as well as their negation.

Concept learning from examples can be viewed as learning an incompletely specified discrete function (Michalski, 1969; Michalski, Rosenfeld and Aloimonos, 1994). The function is incompletely specified because the set of examples usually represents only a (small) subset of instances of that function. In *single* concept learning, the function takes value 1 for any instance that belongs to the concept and value 0, otherwise. In *multiple-valued* concept learning, the function to be learned takes outcomes from a discrete set of values representing a degree to which an instance satisfies the concepts. In *multi-concept* learning, the function takes a vector of values (each value corresponds to one concept).

It has been observed that functions that are symmetrical with regard to some of its variables may have a very complex DNF expression (Michalski, 1969a). A function is symmetrical with regard to a set of attributes, if replacing any attribute by any other from the set and vice versa does not change the function value for any assignment of the input attributes. A set of attributes for which a function is symmetric is called an *attribute symmetry group* for this function. Section 3.1 provides more details on this topic.

Although a concept may not be symmetrical with regard to a set of original attributes, its projection on a subspace of the representation space may be symmetrical. This leads to concept descriptions in which one part identifies such a subspace and the other part represents a symmetrical function within that subspace. Learning such conditional symmetrical descriptions is an objective of this research.

3.1 Symmetrical Functions And Concepts

The concept of symmetry of a function has been studied already a long time ago (e.g., Michalski, 1969). For completeness, we will briefly review it here and adapt to the problems of concept learning.

A function $f(x_1, x_2, ..., x_n)$ is *symmetrical with respect to variables* $\{x_i, x_j\}$ if

$$f(..., x_i, ..., x_j, ...) = f(..., x_j, ..., x_i, ...) \qquad (1)$$

where, $n \geq 2$, $0 < i \leq n$, $0 < j \leq n$, $i \neq j$.

It is assumed that the domains (value sets) of the attributes x and y are the same. For simplicity, we will assume henceforth that the symmetrical variables are placed at the beginning of the list of arguments of the function, and the remaining variables are denoted by R ("Rest"). Since concept learning can be viewed as learning a function with a binary output and discrete (or continuous input), we introduce the following definition. A concept C(x, y, R) is called *symmetrical (potentially symmetrical) with respect to attributes x and y*, if for all instances (*all known instances*) of that concept:

$$C(x, y, R) = C(y, x, R) \qquad (2)$$

The idea of “potentially symmetrical” is very important for concept learning, because here the set of training example (positive and negative) usually represents a small subset of possible concept instances, and the whole problem of learning is to extrapolate the information provided by the training set. Definition (2) applies for single and multiple-valued concept learning. The symmetry of the concept C(x, y, R) with respect to attributes x and y implies existence of a symmetrical relation between instances of that concept. Below we discuss two basic classes of such symmetry.

(1) Equivalence symmetry — EQ_{xy}: (x & y or ~x & ~y)

Figure 1 illustrates the equivalence symmetry of a concept with regard to attributes by using a “generalized logic diagram” for visualizing discrete functions. The representation space is defined by four binary attributes: x0, x1, x2 and x3. Each cell of the diagram corresponds to a unique combination of attribute values (a

concept example). Positive examples are marked by +, and negative examples by –
For example, the negative instances, e1 and e2, in Figure 1a can be described by the following VL1 expressions (VL1 stands for variable-valued logic 1—a form of multiple-valued logic calculus; Michalski, 1975):

e1 <:: (x0=1) & (x2=1) & (x1=1) & (x3=0)

e2 <:: (x0=0) & (x2=0) & (x1=1) & (x3=0)

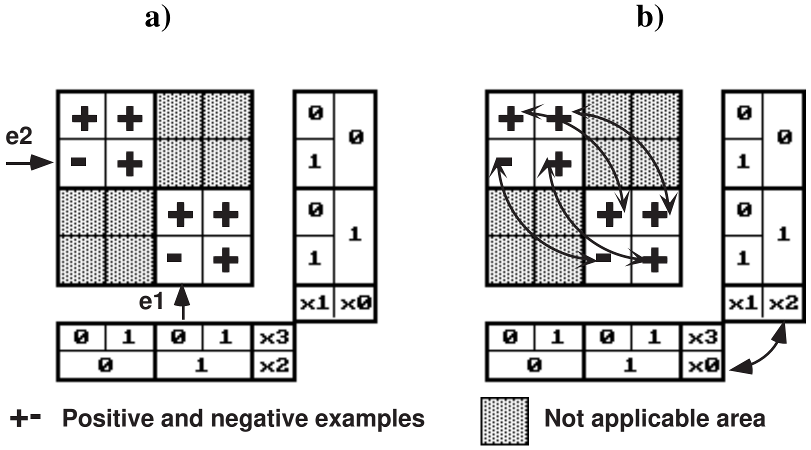


Figure 1. An equivalence symmetry of a concept with respect to x0 and x2.

The concept, as defined by the shown examples, is symmetrical with regard to x0 and x2 in the subspace described by (x0=0) & (x2=0) v (x0=1) & (x2=1). The gray area denotes the part of the concept (or its negation) that is may be or may not symmetrical with regard to x0 and x2. Figure 1b has a different arrangement of attributes (attributes x0 and x2 were switched), but the projection of the function on the non-gray area is the same.

(2) Exclusive-or symmetry — XOR_{xy}: (x & ~y or ~x & y)

Figure 2 illustrates the exclusive-or symmetry. It shows a projection of three concepts (C1, C2, C3) into the subarea described by (x0 & ~x2) or (~x0 & x2). The gray area in the diagram marks the parts of the concepts that may or may not be symmetrical with regard to x0 and x2.

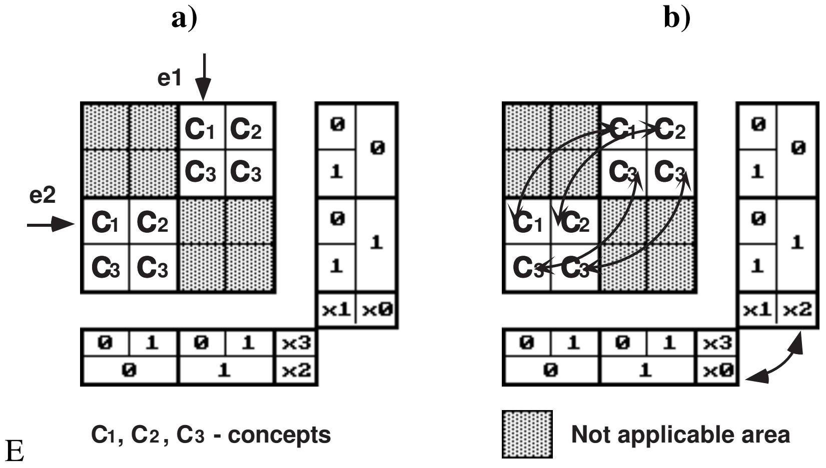


Figure 2. An exclusive-or symmetry of concepts C1, C2, and C3 with respect to attributes x0 and x2.

An important property of a symmetrical relation between two attributes of a concept (either EQ or XOR symmetry) is that it is transitive. The following theorem defines this property precisely.

Theorem. Let SYM: $C_k(x, y, R)$ stand for an expression stating that concept C_k is symmetrical with regard to attributes x and y (R stands for other concept attributes). Then we have:

$$\forall\ x, y, z\ \ \text{SYM: } C_k(x, y, R)\ \ \&\ \ \text{SYM: } C_k(y, z, R)\ \ \Rightarrow\ \ \text{SYM: } C_k(x, z, R) \qquad (3)$$

Proof.

By definition, the SYM: $C_k(y, z, R)$ is logically equivalent to SYM: $C_k(z, y, R)$. Therefore, in any description of C_k, every occurrence of attribute y may be replaced by attribute z, and vice versa. If a replacement of y by z is done in $C_k(x, y, R)$, then SYM: $C_k(x, y, R)$ becomes SYM: $C_k(x, z, R)$. Q.E.D.

3.3 Maximum Symmetry Classes

A decision rule is meant here as an expression:

DECISION <:: CONDITION

where DECISION is an assignment of a specific value to a decision variable (or variables, <:: is a decision assignment operator, and CONDITION is a conjunction of unary relational expressions, each stating a condition on values of a single attribute. Examples of unary relational expressions are: xi = 5, xi > 3 and xi = 2..5 (in the latter, xi takes a value between 2 to 5, inclusively). A general form of a unary relational expression is:

$$[xi \;\; REL \; Values] \qquad (4)$$

where REL is one of {=, <>, <, >, >= or <=}, and Values stand for one or more values of the attribute linked by the internal disjunction or the range operator ("..").

A relational expression defined by (4), called for short RE, evaluates to 1 or 0, depending on whether the value of the attribute in an input example satisfies or does not satisfy, respectively, the expression. If the attribute in a relational expression is binary, then there are only two relational expressions possible: xi=1, written briefly as xi, and xi=0, abbreviated as ~xi. Unary relational expressions were originally defined in the variable-valued logic system VL1 (where they are called selectors; Michalski, 1975). They are building blocks of decision rules generated by the AQ-type rule learning system employed in the method described here.

Let C1 and C2 be condition parts of two decision rules in a ruleset representing a concept description. Suppose C1 is in the form REi & ~REj & CTX1 and C2 is in the form ~REi & REj & CTX2, where REi and REj are relational expressions, and CTX1 and CTX2 are "context" conditions that are conjunctions of zero or more relational expressions. It is said that REi and REj represent a *binary XOR-symmetry class* for the concept description, if CTX1 and CTX2 are in a *subsumption* relation, that is, CTX1 = CTX2 & CTX3 or CTX2 = CTX1 & CTX3, where CTX3 is a context condition. If C1 is in the form REi & REj & CTX1 and C2 in the form ~REi & ~REj & CTX2, then REi and REj represent a binary EQ-symmetry class for the concept description. If REi and REj are in a XOR- or EQ- symmetry relation, then we say generally that they are in a symmetry relation.

Due to the transitivity of the symmetry relation, a set of k binary symmetry classes of REs can be combined into one k-ary symmetry class (SC), if they can be ordered into a chain in which every two neighboring relational expressions have a non-empty intersection. For example, if SC1 = {RE1, RE3} and SC2 = {RE1, RE2} then they can be combined into a larger symmetry class SC3 = {RE1, RE2, RE3}. A symmetry class to which no additional expressions can be added is called *a maximal symmetry class*.

3.4 Generating Counting Attributes

Suppose a k-ary maximum symmetry class {RE1, RE2, ...REk} has been formulated for a set of decision rules constituting a concept description. The *counting attribute generation rule* is a constructive induction operator that creates a *counting attribute*, defined by the arithmetic sum {v(RE1)+ v(RE2) +...+ v(REk)}, where v(REi) is the value of REi for the object to which the attribute is applied; thus, it can be 1 or 0. The counting attribute represents the number of relational expressions that hold for the given concept example and its domain (value set) is the set of integers from 0 to k. The counting attribute thus sums up the "evidence" contributed by each relational expression in a given symmetry class. The so created counting attribute is added as a new dimension to the representation space. The following general notation is introduced for a counting attribute:

$$\#\text{AttrIn}\{\text{Attribute Set: IREL VAL}\} \qquad (5)$$

where, *Attribute Set* is a list of attributes, *IREL* specifies an *internal relation*: EQ, NEQ, GT, LT, GE or LE, and *VAL* is a value from the domain of the attributes on the list. The attributes in the set may be (binary) relational expressions, as discussed above, or they can be multi-valued. The counting attribute has the following meaning: "The number of attributes in the *Attribute Set* that are in relation IREL with VAL." If attributes are binary (as in the case of relational expressions), then the notation for the counting attribute is simplified to: #AttrIn{Attribute Set}. A counting attribute is treated as any other discrete attribute, and thus can be used to create relational expressions in the form defined by (4). For example

$$\#AttrIn\{x1, x4, x5, x6\}: > 4\} = 2 \quad (6)$$

denotes a relational expression stating that there are 2 attributes from the set {x1, x4, x5, x6} that have the value greater than 4 in an example or examples of a concept). As one can see, relational expressions with counting attributes can express complex relations, but nevertheless are easy to comprehend.

Using a counting attribute, the well-known M-of-N concept ("At least M out of N properties from {P1, P2, ...PN} hold") can be represented: #AttrIn{P1, P2, ...PN} ≥ M. To express the condition "Between 2 and 4 (non-binary) attributes in the set {A2, A3, A5, A7, A9, A12} have value greater than 5, one can write: #AttrIn{A2, A3, A5, A7, A9, A12: GT 5} = 2.. 4. To illustrate the usefulness of a counting attribute, let us describe the concept represented in Fig. 3E. The <u>simplest</u> DNF description of this concept is:

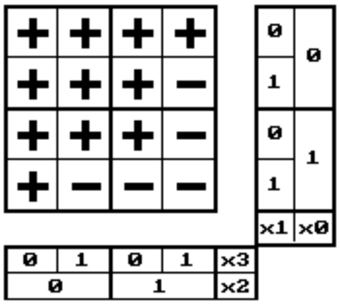
$$\begin{array}{rcrcrcrl}
\sim x0 & \& & \sim x1 & \& & \sim x2 & \& & x3 & \text{or} \\
\sim x0 & \& & \sim x1 & \& & x2 & \& & \sim x3 & \text{or} \\
\sim x0 & \& & x1 & \& & \sim x2 & \& & \sim x3 & \text{or} \\
x0 & \& & \sim x1 & \& & \sim x2 & \& & \sim x3 & \text{or} \\
\sim x0 & \& & x1 & \& & x2 & \& & x3 & \text{or} \\
x0 & \& & \sim x1 & \& & x2 & \& & x3 & \text{or} \\
x0 & \& & x1 & \& & \sim x2 & \& & x3 & \text{or} \\
x0 & \& & x1 & \& & x2 & \& & \sim x3 &
\end{array} \quad (7)$$

Using the counting attribute, we can write a logically equivalent expression in the form:

$$[\ \#AttrIn\{x0,x1,x2,x3\} = 1 \text{ v } 3] \quad (8)$$

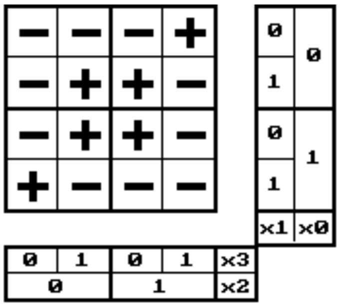
which is both shorter and easier to understand (it reads: "Either one or three attributes from among x0, x1, x2, x3 take value 1").

The above example show that the counting attribute is powerful descriptive concept that allows one to express concisely a wide range of relations for which an equivalent DNF expression would be very long. If the counting attribute involves N binary attributes, it allows to represent all combinations of counts of N properties. There are 2^{N+1} such combinations (2^N subsets multiplied by two attribute values for each subset). Figure 3 illustrates six out of 32 possible concepts that can be expressed by a unary relational expression involving the counting attribute #AttrIn{x0,x1,x2,x3}.



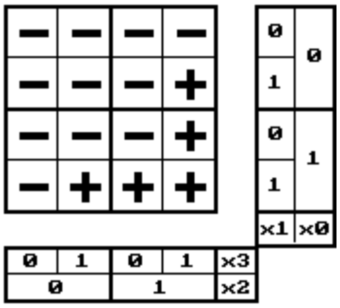
A. At most 2-of-4

[#AttrIn{x0,x1,x2,x3} <= 2]



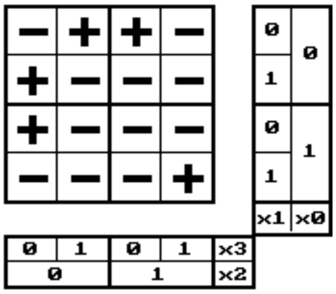
B. Exactly 2-of-4

[#AttrIn{x0,x1,x2,x3} = 2]



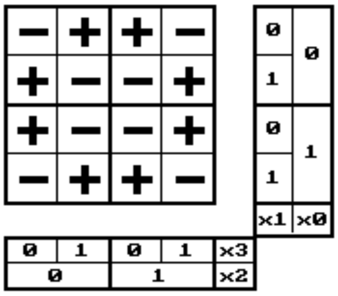
C. At least 3-of-4

[#AttrIn{x0,x1,x2,x3} >= 3]
(standard M-out-of-N concept)



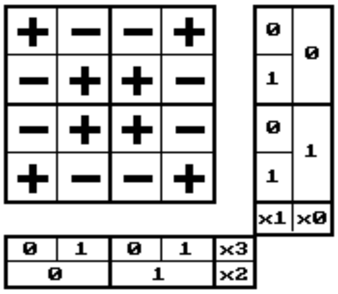
D. 1 or 4 of-4

[#AttrIn{x0,x1,x2,x3} = 1,4]



E. Odd number of-4

[#AttrIn{x0,x1,x2,x3} = 1,3]



F. Even number of-4

[#AttrIn{x0,x1,x2,x3}=0,2,4]

Figure 3. Examples of relations that can be represented by using the counting attribute #AttrIn{x0,x1,x2,x3}

In order to establish the value of a concept for a given instance (a cell in the diagram), the number of occurrences of value "1" in the vector of attribute values that characterizes this instance is counted. If this number satisfies the concept description, then the instance belongs to the concept and is marked by "+" in the diagram; otherwise it does not belong to the concept, and is marked by "-".

Decision rules that combine conventional logic-type relational expressions with arithmetic-type expressions are called generally *hybrid decision rules.* The presented method builds hybrid rules that involve only counting attributes (if it is important to distinguish such rules from more general hybrid rules, we will call them the *counting hybrid rules)* .

Fig. 4 shows examples of concept descriptions in the form hybrid decision rules.

Conditional Parity 5 defined on 10 binary attributes: x0-x9.

[#AttrIn {x1, x2, x3, x4, x5} = 0 v 2 v 4] & [x7 = 0]

The parity-5 concept involves 5 binary attributes. [x7 = 0] specifies condition under which the parity is satisfied.

MONK2 problem "exactly two of six attributes have their first value"

[#AttrIn{First(x1), First(x2), ..., First(x6)} = 2]

Conditional XOR defined on 10 binary attributes: x0-x9.

[#AttrIn {x1, x2} = 1] & [x0 = 1] & [x8 = 0]

CR10 a conditional M-of-N rule defined on 10 binary attributes

[#AttrIn{x0,x2,x4,x6,x8} >= 3] & [#AttrIn{x1,x3,x5,x7} >= 2] & [x9 = 1]

Figure 4. Concept descriptions in the form of counting hybrid rules (only the condition part of the rules is shown).

3.5 LEARNING COUNTING HYBRID RULES: AQ17-HCI

This section presents an algorithm for learning counting hybrid descriptions (Fig. 5) that was implemented in the AQ17-HCI program. It is based on detecting symmetry patterns in the DNF concept description (a set of decision rules) created by an AQ-type learning program (the current version of the program detects only XOR-symmetry patterns).

1. Determine a DNF concept description from training examples projected into the current representation space. If the expression is "sufficiently" simple (according to a given criterion), then STOP.

2. Detect symmetry patterns in the learned concept description.

3. If there are no such patterns, then STOP. Otherwise:

   Build the maximum symmetry class (MSCs) for each pattern. For each MSC-class, introduce a "counting attribute," and add the attribute to the representation space. Project the training data into the new representation space.

   Go to step 1.

Figure 5. Algorithm for Learning Counting Hybrid Descriptions

Suppose, for example, that the following XOR-symmetry patterns were detected: x1 XOR x3, x1 XOR x5, and x1 XOR x7. The following MSC-class is created {x1, x3, x5, x7}. For each MSC-class a counting attribute is created and added to the representation space. In the above example, the attribute #AttrIn{x1,x3,x5,x7} is created. Its domain is an integer interval from 0 to 4. Training examples are mapped into the new representation space, and a new rule learning iteration is performed.

The algorithm described above is independent of the learning program used. However, systems that use different representational formalisms for examples and concept descriptions (unlike the AQ-type learning programs) may have difficulty detecting symmetry patterns in generated concept descriptions. In such cases, symmetry patterns can be detected by a data-driven approach, that is be examining training examples. In AQ learning systems that use the VL1 description language both for data and rules, patterns can be easily detected either by a data-driven or a hypothesis-driven approach. An examination of descriptions generated by FOIL (Quinlan, 1990) for the MONK2 problem and other problems indicates that these descriptions can be used for detecting symmetry-patterns. The symmetry patterns are a form of class-patterns. They are different from the intra-construction and inter-construction operators used in Duce and CIGOL systems (Muggleton, 1987; Muggleton & Buntine, 1988), which are forms of rule-patterns in Horn-clauses (Wnek & Michalski, 1994b).

4 AN ILLUSTRATIVE EXAMPLE: THE MONK2 PROBLEM

The concept to be learned is the MONK2 problem (Thrun et al., 1991; Wnek & Michalski, 1994a). Figure 6A shows a diagram visualizing the representation space with indicated target concept and training examples. The total number of possible instances in the space is 432. In the diagram, the target concept is represented by 142 instances (shaded area). The remaining 290 instances represent the negation of the concept. The training set is represented by 64 positive (+) and 105 negative (-) examples. The data contain no noise.

4.1 Learning In The Original Representation Space

The MONK2 problem poses a difficult problem for conventional symbolic learning systems. In fact, none of the 18 conventional symbolic learning systems (decision tree or rule learning programs that did not use constructive induction) that took part in the international competition was able to learn the MONK2 concept (Thrun et al., 1991). The descriptions generated by these programs all have the following characteristics: a relatively low prediction accuracy (about 75%) and high complexity (many decision rules or nodes in the decision tree; rules involve many conditions). As many as 16 decision rules were needed to characterize 64 positive examples. Almost all rules involved all six attributes in describing the concept, which means that all attributes are considered important. This result means that representation space or the representation language is inadequate for learning this concept.

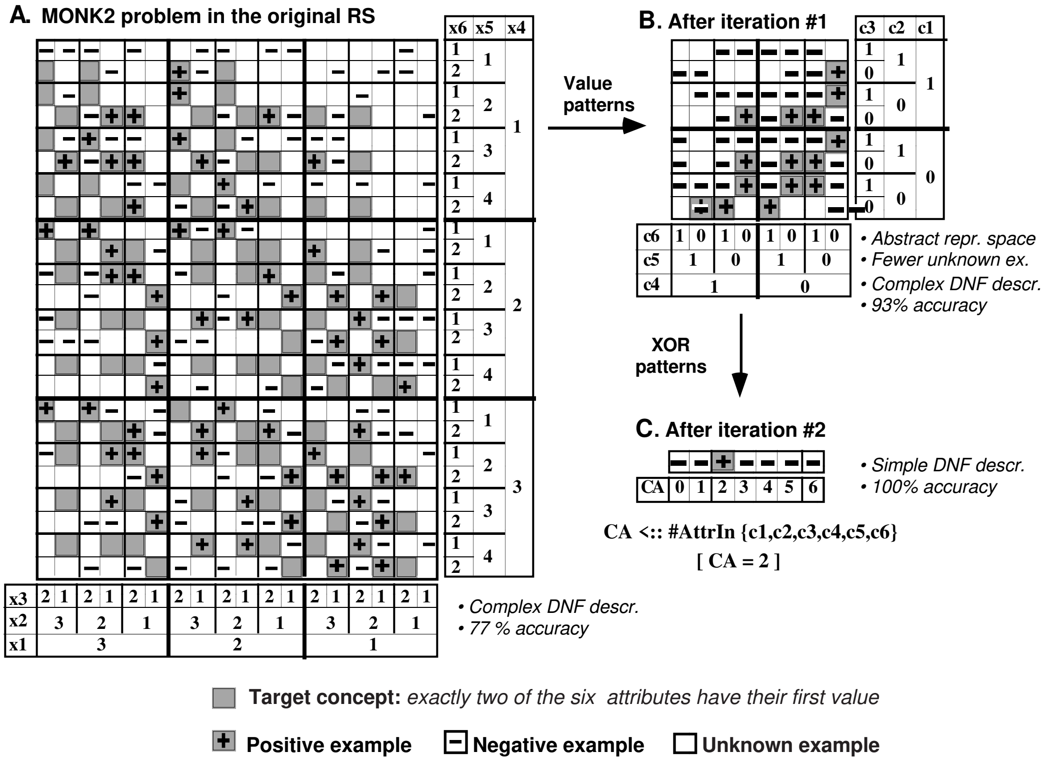


Figure 6. Learning a description for the MONK2 problem in two iterations of the AQ17-HCI method.

4.2 Representation Space Transformation: Iteration #1

Program AQ15 was used to learn the initial concept description from the given training examples. This description was then inspected for the presence of XOR-symmetry patterns. In many conditions involving the same attribute the same set of attribute values was present. Such sets form value-patterns (Wnek and Michalski, 1994a), which were used to transform the initial representation space (Fig. 7). The learning task in the new space is visualized in Fig. 6B. The new representation space has become significantly smaller; there are only 64 instances in the new space versus 432 instances in the original representation space. The number of attributes is the same, but all of them are binary.

(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]

Figure 7. Attributes Constructed From Value-patterns.

The number of instances representing the target concept is now 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 comparison to the original representation space. Each instance in the new space represents between 1 and 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. In the new representation space, all possible positive examples are now present, and only 13 of the possible negative examples are missing (the original space had only 64 positive examples out of a possible 142). Although the representation space was simplified, AQ15 still generated a long and inaccurate description of the concept. Errors were caused by overly general decision rules. These rules covered not only two positive examples but also two negative examples. For more details see (Wnek, 1993).

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 (Fig. 6B). Therefore, the search for a better representation is continued, and the XOR-patterns are found (Wnek and Michalski, 1994c). All six attributes form a MSC class. From this class a new counting attribute is constructed. It is defined as #AttrIn{c1,c2,c3,c4,c5,c6}. Its domain is an integer interval from 0 to 6. Summing up values in the XOR-patterns always gives the exact value 2. The final concept description was [#AttrIn{c1,c2,c3,c4,c5,c6}=2], i.e., exactly two of the six attributes are present. Fig. 6C visualizes the final representation space and the final concept learned.

5 EXPERIMENTAL RESULTS

We have conducted a set of experiments with the proposed method to learn concepts from examples of different complexity (involving 6, 9, 10, and 16 binary attributes). Figure 8 summarizes the results. They demonstrate that the AQ-HCI method is capable of learning counting hybrid descriptions very effectively. The MONK2 learning problem (Thrun et al., 1991) was solved with 100% accuracy and produced descriptions exactly equivalent to the target description. When applied to more complex concepts the program produced significantly better descriptions in terms of both prediction accuracy and simplicity, as compared to the original AQ15 system.

Concept:	**"More than 3 and more than 2"**
	[#AttrIn{x0,x2,x4,x6,x8}>= 3] & [#AttrIn{x1,x3,x5,x7}>=2]
Domain:	**9 binary attributes (x0, x1, ..., x8)**
Training set	**50% of all examples**
AQ15:	**Prediction accuracy 78% 42 rules**
AQ17-HCI:	**Prediction accuracy 100% 1 rule**

Concept:	**"More than 3 and more than 2 when x9 holds"**	
	[#AttrIn{x0,x2,x4,x6,x8}>=3] & [#AttrIn{x1,x3,x5,x7} >= 2] & [x9= 1]	
Domain:	**10 binary attributes (x0, x1, ..., x9)**	
Training set	**20% of all examples**	
AQ15:	**Prediction accuracy: 96 %**	**Complexity: 22 rules**
AQ17-HCI:	**Prediction accuracy: 100%**	**Complexity: 1 rule**

Target Concept:	**"More than 3 and more than 2 when x9 holds, or Equal 2"** **[#AttrIn{x0,x2,x4,x6,x8} >= 3] & [#AttrIn{x1,x3,x5,x7}>= 2]** **& [x9= 1] or [#AttrIn{xC,xD,xE,xF} = 2]**
Domain:	**16 binary attributes (x0, x1, ..., x9, xA, ..., xF)**
Training set	**10% of all examples**
AQ15:	**Prediction accuracy 93% Complexity: 92 rules**
AQ17-HCI:	**Prediction accuracy 100% Complexity: 5 rules**

Figure 8. Summary of experiments on learning hybrid descriptions.

7 CONCLUSION

The presented method addresses a class of learning problems that require concept descriptions that combine logic-type descriptions with counting conditions. Conventional symbolic learning methods produce prohibitively long and inaccurate descriptions for such problems. The method, based on ideas of constructive induction, works iteratively. In each iteration, it searches for symmetry patterns in the descriptions generated by an AQ-type learning program. In the implemented method, detecting symmetry was based on detecting XOR-symmetry patterns. Detected patterns are used to create new dimensions, called counting attributes, in the knowledge representation space. Adding these dimensions represents a task-oriented adaptation of the representation space. Experiments demonstrated that the method was very effective in solving testing problems that required learning counting hybrid descriptions.

The method can be applied to a wide range of domains where logical conditions need to be combined with simple arithmetic relations (counting) to capture the essence of the target concept. Such domains include economy, medicine, computer vision, biochemistry and others.

Future research needs to address other classes of problems for which symbolic methods are not adequate, if applied directly, such as learning concepts requiring logical and complex arithmetic expressions or detecting geometrical object properties.

ACKNOWLEDGMENTS

The authors thank Eric Bloedorn and Ken Kaufman for useful comments and criticism. This research has been conducted in the Center for Machine Learning and Inference at George Mason University. The Center's research is supported in part by the National Science Foundation under grants No. CDA-9309725, IRI-9020266, and DMI-9496192, in part by the Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research, the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under grant No. N00014-91-J-1351.

REFERENCES

Baffes, P.T. & Mooney, R.J. (1993). Symbolic Revision of Theories with M-of-N Rules. In *Proceedings of the 2nd International Workshop on Multistrategy Learning* (pp. 69-75). Harpers Ferry, WV: Morgan Kaufmann.

Bloedorn, E. & Michalski, R.S. (1991). Data Driven Constructive Induction in AQ17-PRE: A Method and Experiments. In *Proceedings of the Third International Conference on Tools for AI* (pp. 25-35). San Jose, CA.

Callan, J.P. & Utgoff, P.E. (1991). A Transformational Approach to Constructive Induction. In *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 122-126). Evanston, IL: Morgan Kaufmann.

Fawcett, T.E. & Utgoff, P.E. (1991). A Hybrid Method for Feature Generation. In *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 137-141). Evanston, IL: Morgan Kaufmann.

Jensen, G.M. (1975). *Determination of Symmetric VL1 Formulas: Algorithm and Program SYM4*. Master's thesis. (Tech. Rep. No. UIUCDCS-R-75-774). Urbana-Champaign: University of Illinois, Department of Computer Science.

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

Michalski, R.S. (1969b). "On the Quasi-Minimal Solution of the General Covering Problem, *Proceedings of the V International Symposium on Information Processing (FCIP 69)*,Vol. A3 (Switching Circuits), (pp. 125-128), Yugoslavia, Bled, October 8-11.

Michalski, R.S. (1975). "Variable-valued Logic and Its Application to Pattern Recognition and Machine Learning." In D. Rine (Ed.) *Computer Science and Multiple-Valued Logic Theory and Applications*, North-Holland Publishing, 1974.

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

Michalski, R.S., Mozetic, I., Hong, J. & Lavrac, N. (1986). The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. In *Proceedings of AAAI-86* (pp. 1041-1045). San Mateo, CA: Morgan Kaufmann.

Michalski, R.S., Rosenfeld, A. & Aloimonos, Y. (1994). *Machine Vision and Learning: Research Issues and Directions* (Tech. Rep. No. MLI 94-6). Fairfax, VA: George Mason University, Center for Machine Learning and Inference. (Tech. Rep. No. CAR-TR-739, CS-TR-3358). College Park, MD: University of Maryland, Center for Automation Research.

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

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

Murphy, P. M. & Pazzani, M. J. (1991). ID2-of-3: Constructive Induction of M-of-N Concepts for Discriminators in Decision Trees. In *Proceedings of the 8th International Workshop on Machine Learning* (pp. 183-187). Evanston, IL: Morgan Kaufmann.

Quinlan, J.R. (1990). Learning Logical Definitions from Relations. *Machine Learning, 5*, 239-266.

Seshu, R. (1989). Solving the Parity Problem. In *Proceedings of EWSL-89* (pp. 263-271). Montpellier, France.

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

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. & Zhang, J. (1991). *The MONK's Problems: A Performance Comparison of Different Learning Algorithms*. (Tech. Rep. December). Pittsburgh, PA: Carnegie Mellon University, School of Computer Science.

Towell, G. G. & Shavlik, J. W. (1994). Refining Symbolic Knowledge Using Neural Networks. In R.S. Michalski & G. Tecuci (Eds.), *Machine Learning: A Multistrategy Approach, Vol. IV* (pp. 405-429). San Mateo, CA: Morgan Kaufmann.

Wnek, J. (1993). Hypothesis-driven Constructive Induction. (Doctoral Dissertation, George Mason University, School of Information Technology and Engineering, Fairfax, VA). Ann Arbor, MI: University Microfilms Int.

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

Wnek, J. & Michalski, R.S. (1994b). Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments. *Machine Learning, 14*, 139-168.

Wnek, J. & Michalski, R.S. (1994c). Discovering Representation Space Transformations for Learning Concept Descriptions Combining DNF and M-of-N Rules. In *Working Notes of the ML-COLT'94 Workshop on Constructive Induction and Change of Representation* (pp. 61-68). New Brunswick, NJ.

Wnek, J., Kaufman, K., Bloedorn, E. and Michalski, R.S. (1995). "Selective Induction Learning System AQ15c: The Method and User's Guide," *Reports of Machine Learning and Inference Laboratory*, MLI 95-4, George Mason University, Fairfax, VA.