Reports

Machine Learning and Inference Laboratory

Reasoning with Meta-values in AQ Learning

Ryszard S. Michalski
Janusz Wojtusiak

MLI 05-1
P 05-2
June, 2005

School of Computational Sciences

George Mason University

REASONING WITH META-VALUES IN AQ LEARNING

Ryszard S. Michalski*
Janusz Wojtusiak
Machine Learning and Inference Laboratory
George Mason University

Fairfax, VA 22030-4444, USA
{michalski, jwojt}@mli.gmu.edu
http://www.mli.gmu.edu

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

Abstract

This paper describes methods for reasoning with *missing*, *irrelevant* and *not applicable* meta-values in the AQ attributional rule learning. The methods address issues of handling these values in datasets both for rule learning and rule testing. In rule learning, the presence of these values affects the *extension-against* generalization operator in star generation, and the rule matching operator. In rule testing, these values affect the execution of the rule matching operator. The presented methods have been implemented in the AQ21 learning program and tested on four datasets.

Keywords: Machine learning, Concept learning, AQ learning, Meta-values, Missing values, Irrelevant values, Not applicable values.

Acknowledgments

The authors thank Dr. Kenneth Kaufman for his useful comments on the earlier version of this paper, and for valuable suggestions regarding examples used to illustrate the methodology. Jarek Pietrzykowski helped to prepare data for experiments involving the *Computer Users* and *ROBOTS* datasets.

The presented research was conducted in the Machine Learning and Inference Laboratory of George Mason University, whose 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. The findings and opinions expressed here are those of the authors, and do not necessarily reflect those of the above sponsoring organizations.

1 INTRODUCTION

In the general problem of learning concepts or classifications, one needs to consider cases when training or testing datasets may not have values specified for all the attributes and for all entities (concept examples). Some attributes may have missing values. Some attributes may apply to some but not to all entities (e.g., the number of pages applies to a book, but not to a chair in the library). Also, certain attributes may be known to have no relevance to a given learning problem, thus, removing them from the data is desirable (e.g., the value of a stock tomorrow will unlikely depend on the last name of the broker's barber).

To represent such cases, Attributional Calculus employed in AQ learning assumes that the domain of every attribute includes three *meta-values* in addition to its regular values (Michalski, 2004). These meta-values correspond to three possible answers to a question requesting an attribute value in situations in which a regular value is not provided. Specifically, these are *"missing," "not applicable" and "irrelevant,*" meta-values. Their semantics is defined as follows:

- *Missing* (a.k.a. *"Don't know" or "Unknown"*), denoted by a "?", is given to an attribute whose value for a given entity is not available for some reason. For example, the attribute has not been measured for this entity, or was measured but was not recorded in the database. In such situations, the meta-value "?" in inserted in the training and/or testing datasets for this attribute in the event (example, instance, record, or datapoint) characterizing the given entity.

- *Not-applicable*, denoted by an "NA," is given to an attribute that is not applicable to a given entity.

- *Irrelevant,* denoted by an "*", indicates that this attribute is considered irrelevant for the learning problem, or for the concept (class) to be learned, or in the particular event.. Consequently, three types of irrelevant attributes are distinguished, *task-irrelevant*, *class-irrelevant*, and *event-irrelevant*.
  - An attribute is *task-irrelevant* if it is irrelevant for the entire learning problem. For example, a student's hair color can be declared as irrelevant for learning rules for classifying students into groups representing their academic performance.
  - An attribute is *class-irrelevant* if it is irrelevant for a given class (value of the output attribute), but may relevant for other classes. For example, the patient's PSA (prostatic specific antigen) level is relevant for diagnosing prostate diseases, but is irrelevant for diagnosing eye diseases.
  - An attribute is *event-irrelevant* if it is irrelevant only for a particular event in the class to be learned. For example, the attribute "stock price" is relevant to any event in the class "stocks_to_acquire," but in a particular instance when it is the stock of company you work for and is given free to employees, it may be considered irrelevant.

  The task-irrelevance is handled by simply removing the attribute in question from the training and testing datasets. The class-irrelevance is handled by removing the attribute from training dataset for the given class, but it remains in dataset when learning classes for which it is relevant. Therefore, only the problem of handling event-irrelevant attributes needs to be considered.

The presence of missing values may be unavoidable in some problem domains. As to the irrelevance or not-applicability of an attribute, such decisions are made by an expert setting a learning problem. These decisions can be viewed as prior knowledge communicated to the learning program. This knowledge is provided by entering appropriate meta-values into the training and testing datasets.

In addition to methods that fill-in the missing values before the learning process starts (wrapper methods), we describe also methods for handling missing values during the process of AQ learning. These methods are applied during the execution of two basic operators:

- The "extension against" generalization operator in the training phase

- The matching examples against rules operator, in the training and testing phases.

The first operator is employed during star generation of the AQ learning method and second is employed both in star generation and in determining the degree of match between an event and a rule (e.g., Michalski, 1975; 2004; both papers are downloadable from address www://www.mli.gmu.edu/papers). For completeness, before describing these methods, we briefly review the extension against and matching operators.

2 THE EXTENSION AGAINST AND RULE MATCHING OPERATORS

To explain the extension against and rule matching operators, we assume that the reader is familiar with the basic terminology of Attributional Calculus (Michalski, 2004; downloadable from address www.mli.gmu.edu/papers). We start by explaining the *extension against* operator, which is the basic generalization operator in AQ learning.

Given two attributional events e1 and e2 (vectors of attribute-values), the *extension of* e1 *against* e2, denoted as e1 —| e2, is equivalent to the *extension of* e1 *in* the negation of e2, denoted as e1 |— ~e2, that is:

$$e1 \text{ —| } e2 \quad = \quad e1 \text{ |— } \sim e2 \tag{1}$$

To explain the function of the *extension-in* operator, let us first assume that ~e2 is a single complex (attributional conjunction), denoted by L. The *extension of* e1 *in* L is defined:

$$e1 \text{ |— } L \quad = \quad L, \text{ if } e1 \in L, \text{ otherwise } \varnothing \tag{2}$$

By $e1 \in L$ is meant that e1 strictly satisfies (or is covered by) the complex L, that is, satisfies all conditions in L. The negation of an event, e.g., ~e2 in (1), is equivalent to a disjunction of complexes consisting of a single condition. Extension-in is distributive over disjunction, therefore, if L in (2) is replaced by a disjunction of complexes, $L1 \vee L2 \vee ,\ldots \vee Lk$, the result of the |— operator is a disjunction of the results obtained from applying (2) to individuals complexes L1, L2, .., Lk:

$$e1 \text{ |— } (L1 \vee L2 \vee \ldots \vee Lk) \quad = \quad (e1 \text{ |— } L1) \vee (e1 \text{ |— } L2) \vee .. \vee (e1 \text{ |— } Lk) \tag{3}$$

Let us now generalize the extension against operator to the case in which e1 is an arbitrary complex. Suppose that L^+ and L^- are two complexes characterizing positive and negative training examples, respectively. Suppose further that L^+ can be represented in the form:

$$L+ = [xi = A] \ \& \ CTX1 \tag{4}$$

and L^- in the form:

$$L- = [xi = B] \ \& \ CTX2 \tag{5}$$

where x_i is an attribute, A and B are subsets of the domain of x_i (represented in Attributional Calculus by linking their elements by internal disjunction), and CTX1 and CTX2 are "context" complexes that do not contain attribute x_i, or are null expressions.

Let us assume first that references A and B are disjoint, i.e., $A \cap B = \varnothing$. The *extension of* L^+ *against* L^- *along dimension* x_i, denoted

$$L^+ \text{ —| } L^- / x_i \tag{6}$$

is equivalent to the *extension of* L^+ *in* negation of L^- *along* x_i, denoted $L^+ \text{ |— } \sim L^- / x_i$, and produces

$$L = [xi \neq B \cup \varepsilon] \quad (7)$$

where ε is a *generalization margin*, which is a set disjoint from A and B, ranging between the set $D(x_i) - (A \cup B)$ and ∅, where $D(x_i)$ is the domain of attribute x_i.

If $\varepsilon = D(x_i) - (A \cup B)$, then L is $[x_i = A]$, that is, a complex created by repetitively applying the *dropping condition* generalization operator to remove CTX1 from L^+ (Michalski, 1983). If ε = ∅, then L is the maximal possible consistent generalization of L^+, that is, the maximally general complex that covers L^+ and does not intersect with L^-. (Note that L includes neither CTX1 nor CTX2.)

If the contrast complex, L^-, is a conjunction of several selectors in the form $[x_i = A_i]$, i = 1,2,3.., the extension-against is performed for all attributes (dimensions), x_i.

Let us now consider a more general case when $A \cap B \neq \emptyset$. This case can be treated in three different ways, as is done in AQ learning with regard to ambiguous events:

1. ***Include_in_Pos***: Assume that $L^+ = [x_i = A]$ & CTX1, and $L^- = [x_i = B \setminus A]$ & CTX2, and proceed as in the case above, i.e., when $A \cap B$ was empty. This assumption means that events satisfying $[x_i = A \cap B]$ are treated as positive examples, but not as negative.

2. ***Include_in_Neg:*** Assume that $L^+ = [x_i = A \setminus B]$ & CTX1, and $L^- = [x_i = B]$ & CTX2, and proceed as in the case when $A \cap B$ was empty. This assumption means that events satisfying $[x_i = A \cap B]$ are treated as negative examples, but not as positive.

3. ***Ignore***: Assume that $L^+ = [x_i = A \setminus B]$ & CTX1, and $L^- = [x_i = B \setminus A]$ & CTX2, and proceed as in the case when $A \cap B$ was empty. This assumption means that events satisfying $[xi = A \cap B]$ are treated as neither positive nor negative examples.

If $B \setminus A = \emptyset$, i.e., when $B \subseteq A$, then L = ∅ (null expression), and this step of application of the extension-against operator is skipped.

Let us now consider the most general case in which conditions $[x_i = A]$ and $[x_i = B]$ in (4) and (5) are arbitrary logically disjoint attributional conditions, say, S1 and S2, respectively. They may be, e.g., extended conditions in which A and B are attributes, rather than subsets of the domain of x_i (Michalski, 2004). We have:

$$L^+ = S1 \ \& \ CTX1 \quad (8)$$

$$L^- = S2 \ \& \ CTX2 \quad (9)$$

In this case, the extension of L^+ against L^- produces:

$$L = \sim S2 \ \& \ \varepsilon, \text{ if } \sim S2 \text{ logically intersects with } L^+; \text{ otherwise, } \emptyset \quad (10)$$

where ε is a *generalization margin*, which is a complex, CPX, that ranges between a value defined by the expression ~S2 & CPX = S1 & CTX1 (in which case L is not generalized) and the value "True" (in which case L is the maximal consistent generalization of (8), that is, maximal generalization that does not cover any part of L^-).

The above assumed that A, B, S1 and S2 all include no meta-values. One problem considered in this paper is how to execute the extension-against operator when complexes L^+ and/or L^- include such values. Another problem is how to compute the degree to which an event matches a complex (rule) in cases in which the complex and/or event includes such values. Both problems occur in rule learning, and the second problem occurs in rule testing, that is, in applying rules to classify new events.

The rule matching operator is not a single operator, but rather a combination of constituent operators used for interpreting an attributional ruleset. Thee basic constituent operators are defined in a ruleset interpretation schema, namely, an operator for matching an event with an attributional condition (selector), an operator for matching an event with a conjunction of selectors (a complex), and operator for matching an event with a disjunction of complexes (Michalski, 2004).

Before applying a star generation operator, it is desirable to sort positive and negative examples in descending order of the number of meta-values in them. This way, the operations involving events with meta-values are delayed until the later stages of the process. This may lead to better results, because operations with such events narrow down the range of possible generalizations.

3 REASONING WITH *MISSING* META-VALUES

3.1 Wrapper Methods

Wrapper (or preprocessing) methods for handling missing values are applied to datasets before starting the learning process. After they have been applied, there is no need for modifying the regular extension-against and matching operators to handle missing meta-values.

Method P1: Ignore events with attributes in the training set that have a "?" value in the training dataset for the purpose of rule learning. Note that if the original dataset is transformed into a target set through an attribute selection operator, some attributes will be removed. Therefore, events with a "?" value for the removed attributes may not have any more missing values. This method is recommended when the training dataset is large.

When rules are tested, or applied to new examples for the purpose of classification, events with missing values are kept in the testing set. Because classification rules do not require knowledge of values of all attributes, it may happen that the rules can be evaluated without knowing the missing value/s. If in the testing/application phase the missing values are required to evaluate a rule, then Method P2 or P3 is applied.

Method P2: Replace "?" by the average value (for numerical attributes) or the most frequent value (for nominal attributes) in the *s* most similar training and/or testing events, where *s* is a program parameter. If the training dataset is large, finding the *s* events most similar to a given event can be a time-consuming operation. For default, use $s = 1$, and select a single, the most similar example.

Method P3: Learn rulesets for determining the values of the attributes with missing values in the training dataset, and then use these rulesets to predict the missing values when learning rules for other output attributes. Learn first rules for determining values of those attributes with the largest number of missing values in the dataset. Use method P2 for handling "?" values in

attributes other than the one serving as output attribute. After a ruleset for an attribute was learned, apply it to training and testing events that have a "?" value for this attribute, and replace the "?" value by the rule-predicted value. Continue such a process until all missing values are replaced by regular values.

A disadvantage of methods P2 and P3 is that values inserted in place of missing values may be incorrect, and in such a case the performance of the learning or testing processes may be negatively affected.

3.2 Executing the Extension-against Operator in Learning

Method L1: When applying the extension-against operator to negative events with "?" values, ignore (skip) the extension-against operation for attributes with missing values in such events.

Method L2: Treat "?" as a regular value in the events, but do not use events with a "?" for seeds. When extending a seed against a missing value, create a selector: $[x_i \neq ?]$, regardless of the value of attribute x_i in the seed. This means that selector $[x_i \neq ?]$ is assumed to cover any seed event (the probability of this grows with the number of attributes). When logically multiplying a regular selector, $[x_i = R]$, by the selector $[x_i \neq ?]$, follow the rule:

$$[x_i = R] \,\&\, [x_i \neq ?] = [x_i = R, \sim ?] \tag{11}$$

where the reference, R, does not include any meta-value. When multiplying two selectors with the same attribute and a "?" in the reference, create two different "?" symbols:

$$[x_i = R1, \sim ?] \,\&\, [x_i = R2, \sim ?] = [x_i = R1, \sim ?1, \sim ?2] \tag{12}$$

When selecting complexes from intermediate or final stars, select only those that do not have a "?" value in any selector. If such a complex does not exist in the final star, do not select any complex but generate a star from another seed, or apply method L1.

3.3 Matching Rules with Events Containing Missing Values

We consider here a typical case when a "?" does not occur in rules but only in events. When an event includes one or more "?" values, it may still be possible to determine if the event matches the rule or not, because the rule may not refer to this attribute. If this is not possible, one of the following methods is applied:

Method M1: Determine *k* most similar events in the training data to the given event, and estimate the probability of matching the rule on the basis of the distribution of attribute values in this group.

Method M2*:* Suppose a rule, COND ➔ CONS, in which the condition, COND, is a complex, and the consequent, CONS, is a selector, is to be matched with an event, *e*, in which attributes from the set A have "?" values. Create a product of two rules:

$$\text{COND-A} \rightarrow \text{CONS} \quad \& \quad \text{COND-B} \rightarrow \text{CONS} \tag{13}$$

where COND–A is the part of COND with attributes from A, and COND-B is the part that has the remaining attributes. Determine the degree of match, DM, between *e* and COND-B, and create a rule:

$$\text{COND-A} \,\&\, \text{DM} \rightarrow \text{CONS} \tag{14}$$

When using a strict interpretation schema *e* either matches or does not match COND-B, so DM is either 0 or 1. In the first case, the result is that the event does not match the rule. In the second case, we have

$$\text{COND-A} \rightarrow \text{CONS} \qquad (15)$$

The rule (15) indicates which attributes need to be measured in order to assign a definitive decision to the event *e.* When using a flexible interpretation schema (Michalski, 2004), the rule (14) in which DM has some a smaller than 1, but greater than some acceptance threshold, is returned as the output of the matching procedure,

To approximate the positive coverage, *p*, and negative support, *n*, of a learned rule (the number of positive and negative events covered by the rule, respectively), the *Coverage Range* method is used.

For all events that cannot be matched with a rule with a specific degree of match because of the presence of a missing attribute value in the event, determine two values of *p*, $p_{\min}$ and $p_{\max}$, and two values of *n*, $n_{\min}$ and $n_{\max}$, respectively. $p_{\min}$ and $n_{\min}$ are computed by assuming that the event does not match the rule, and $p_{\max}$ and $n_{\max}$ are computed by assuming that it does match. The positive rule coverage is characterized by the range ($p_{\min}$ .. $p_{\max}$), and the negative coverage by the range ($n_{\min}$ .. $n_{\max}$).

4 REASONING WITH *IRRELEVANT* META-VALUES

As mentioned in the Introduction, one can distinguish between three kinds of irrelevant attributes, task-irrelevant, class-irrelevant, and event-irrelevant. A classification of an attribute into any of these classes is done by a domain-expert. The task-irrelevant attributes are handled by removing them from the training and testing datasets. The class-irrelevant attribute are handled by removing them from the training data for the classes for which they are irrelevant.

The event-irrelevant attributes are irrelevant only for specified events. The following sections describe methods for handling them in rule learning and rule testing phases, respectively.

4.1 Rule Learning: Executing the *Extension-against* Operator

Suppose one or more training events for a given class have an irrelevant ("*") value for one or more attributes. These irrelevant values were presumably introduced by an expert in a given problem domain. An irrelevant value is equivalent to, and thus can be replaced by, a disjunction of all the values from the domain of the attribute in question (assuming that the domain is finite).

A training event with such a value can thus be transformed to a disjunction of events, each having a different value from the attribute domain. In general, which includes also the case of attributes with an infinite domain, such an event is equivalent to a complex that does not have a selector with the attribute with an "*" value in the event.

This idea leads to the following method for executing the extension-against operator with events that have an irrelevant value for some attributes.

Method IR: If an attribute is indicated as irrelevant in one or more events of a given class, but indicated as relevant for other events; that is, is irrelevant for one or more combinations of values

of other attributes, but not for all combinations, then in executing the extension against operator, ignore the attribute with the value "*" in events with that value, but do not ignore it in other events.

A proof of the correctness of this method is straightforward. Consider first the case of extending an event against another event in which an attribute has an "*" value. Recall that

$$e1 \text{ —| } e2 \quad = \quad e1 \text{ |— } \sim e2 \tag{16}$$

Suppose, without reducing the generality, that e2 = (x1=a1 & x2= a2 & x3= *), and values a1 and a2 do not appear in e1. Thus,

$$e1 \text{ |— } \sim e2 = [x_1 \neq a1] \vee [x_2 \neq a2] \vee [x_3 \neq *] \tag{17}$$

Based on the definition of the irrelevant value, "*", [x3 ≠ *] is equivalent to:

$$[x_3 \neq a_{31} \text{ v } a_{32} \text{ v } \ldots \text{v } a_{3k}], \text{ which, } \varnothing \tag{18}$$

where a_{3i}, i = 1, 2,3,..., span all values in the domain of x3. This proves the procedure.

A proof for the case in which e1 has an irrelevant value or the general case in which events e1 and e2 are complexes is straightforward, because L & $[x_i = *]$ = L for every L.

4.2 Rule Matching: Determining Coverage of Events with *Irrelevant* Values

If an event with some attributes indicated as irrelevant is matched against an attributional rule, this attribute is removed from the event. This is equivalent to asserting that the irrelevant value always matches a selector with this attribute.

5 REASONING WITH *NOT APPLICABLE* META-VALUES

5.1 Rule Learning: Executing the *Extension-against* Operator

If a dataset has "Not applicable" values ("NA") for some attributes, the attributes are removed from all events with that value when executing the extension-against operator, regardless of whether they are positive or negative events. This operation is justified by the "NA" semantics, according to which, asking for a value of the attribute of an entity for which an attribute is not applicable is meaningless.

5.2 Rule Matching: Determining Coverage of Events with "NA" Values

If a training event has a "Not applicable" value for some attribute, the attribute is removed from the event when determining the rule coverage during the learning process. Therefore, the event does not match the rule if the rule references the NA attribute.

During the testing process, when matching an event against a rule, it is important to correctly interpret the meaning of the condition referencing an attribute that is not applicable to an entity. Consider the following example involving robot-like objects used in the iAQ program (downloadable from *http://www.mli.gmu.edu/msoftware.html*). Suppose a testing event:

$$e = (\text{Has\_jacket} = \text{no}, \text{jacket\_color} = \text{NA}, x_3 = a_3) \tag{19}$$

is matched against the rule:

$$[\text{robot} = \text{friendly}] \leftarrow [\text{jacket\_color} = \text{red}] \quad (20)$$

The jacket_color is NA in this event because the robot does not wear a jacket. The rule is interpreted as not applicable to this event, and thus ignored. The result would be the same if one treated the value "NA" as a regular value of the attribute.

Suppose now that the event (18) is matched against the rule:

$$[\text{robot} = \text{friendly}] \leftarrow [\text{jacket\_color} \neq \text{red}] \quad (21)$$

If one would consider "NA" as a regular value of the attribute "jacket_color," then event (19) would match this rule, which would be incorrect. In this case, the "NA" value has to be interpreted according to its semantics. Because the "jacket_color" is not applicable, matching the event against rule (21) should produce a no-match answer. In other words, the condition [jacket_color ≠ red] should be interpreted as asking for the color of the jacket only if the robot wears a jacket.

6 IMPLEMENTATION OF META-VALUES IN AQ21 LEARNING PROGRAM

6.1 Bitstring Representation of Discrete Attributes

This section describes an implementation of the presented methods for reasoning with meta-values in the AQ21 program for learning and testing of attributional rules (Wojtusiak, 2004). Because discrete and continuous attributes are represented differently, these two types of attributes are handled in different ways. Discrete attributes are represented by bitstrings and continuous attributes are represented by ranges of values (Michalski and Wojtusiak, 2005). The method described here concerns only basic selectors, in which the reference is a single value or an internal disjunction of attribute values, but can not be another attribute, as is in compound selectors (Michalski, 2004).

In the bitstring representation, both events and complexes are represented by equal-length binary strings. Each such bitstring is a concatenation of the characteristic vectors of the selector references. The length of a bitstring is thus:

$$\#D(x_1) + \ldots + \#D(x_n) + n \quad (22)$$

where $D(x_i)$ is domain of the attribute x_i, and #D denotes the cardinality of D. The value n in (22) is added to account for the representation of missing meta-values. The next section describes this representation in detail.

6.2 Handling Meta-values for Discrete Attributes

As indicated above, events comprising values of discrete attributes, as well as complexes describing sets of events are represented in AQ21 by equal-length bitstrings. In this representation, each bit indicates the presence (denoted by "1") or absence (denoted "0") of the attribute value corresponding to the bit's position in the string. For example, if the domain of x, D(x), is {0,1,2,3,4}, then value x = 3 is represented by a string <00010>. Thus, in a representation of an event only one bit is set to "1"for each regular attribute value (not a meta-value) in the event.

In representing a selector with a discrete attribute, all the bits representing attribute values in the selector reference are set to "1", and the remaining bits are set to "0". For example, if the domain of attribute "color" is {red, green, blue}, the selector [color = red ∨ blue] is represented by the bitstring <101>. An attributional complex is represented by a concatenation of bitstrings representing constituent selectors.

The meta-value "Missing" is represented by an additional bit at the end of the bitstring, a *meta-bit*, whose value is set to 1 when "missing" is assigned to the attribute. For example, the event e1 = (color = green)(size = ?) is represented by the bitstring <(0100)(0001)>, assuming that the domain D(size) is {small, medium, large}.

The meta-value Irrelevant ("*") is represented by setting all value bits to "1", and the meta-bit to "0". Thus, an event e3 = (color = *) is represented by <1110>.

The meta-value "Not applicable" is represented by setting all bits to "0." Thus, the event e4 = (color = NA) is represented by the bitstring <0000>.

Using this representation, the extension against operator checks for the presence of meta-values in both positive and negative events. If there are none, the program performs a standard extension-against operation as described in Section 2. If a meta-value is detected, the program performs the extension against operation for attributes with known values, and uses the methods described in Sections 3.2, 4.1 and 5.1 for attributes with the meta-values.

If there is no "?" value in an event, the matching operation between an event and a complex is straightforward. It is simply done by logically multiplying the corresponding bitstrings. The meta-bit is treated as any other bit. If the logical multiplication produces a string in which at least one bit is 1, then the match is strict, otherwise it is not. For instance, matching event (color = *) against rule R = [color = red ∨ blue] involves a logical multiplication of bitstrings <1110> and <1010>, which produces <1010>. The presence of "1s" in the result indicates that event e3 strictly matches rule R.

Suppose now that the event e4 = (color=NA) is matched against rule R from the previous example. e4 is represented by the bitstring <0000> and R is represented by the bitstring <1010>. A logical multiplication of the two bitstrings produces <0000>, which indicates no strict match. If an event does not match every selector in a complex, the whole complex is not strictly matched. (In this paper we do not consider partial, or flexible, matches of complexes. For a discussion of such methods, see (Michalski, 2004)).

To implement methods M1 and M2 for reasoning with the unknowns requires computation of probabilities, as described in Section 3.3. For this purpose, all selectors of a given rule need to be evaluated separately. When a selector cannot be matched because of the "missing" value, method M1 estimates the probability of matching it, and method M2 displays a message informing user about this fact. To increase the program's efficiency, the matching condition operation is applied only to events marked as having missing values.

6.3 Handling Meta-values for Continuous Attributes

Selectors with continuous attributes are represented in AQ21 by ranges (pairs of real values), in which the first number is the lower bound, and the second number is the upper bound on the

values of a given attribute. Both events and complexes are represented this way. For example, suppose "distance" (in meters) is a continuous attribute, whose domain ranges from 0 to 1000. An event e1 = (distance = 37.25) would be represented by the pair (37.25, 37.25), in which the lower bound and the upper bound are the same. If an attributional condition in a rule is [distance = 25.3..32.1], the program would represent it by the pair (25.3,32.1) associated with attribute "distance."

The "Missing" meta-value is represented by the (+∞, +∞), where the lower and upper bounds are set to infinity, which in the computer representation is the largest positive value representable on the given computer. The meta-value "Irrelevant" is represented by the range (-∞, +∞), which spans the entire range of real numbers representable on the given computer.

It should be noted that actual attribute domain does not have to span the entire range of real numbers. For instance, the domain may just be numbers in the range from 0 to 100, but the selector [x = *] would still be represented by the pair (-∞, +∞). The meta-value "Not applicable" is represented by a pair (+∞, -∞), where the lower bound is set to plus infinity and the upper bound is set to minus infinity, that is, the opposite of the representation of irrelevant values.

The "infinity" values are used in this representation, because they are assumed to never appear in data. AQ21 uses the convention that infinity is encoded as the largest possible number in double precision. The number is represented by the constant DBL_MAX[1] that, according to the IEEE standard, equals approximately $1.8*10^{308}$.

Both the "extension-against" and "matching" operators require special treatment of meta-values by checking each case separately. To illustrate this problem, let x be a continuous attribute with the domain (0, 100), rule r = [x = 10..20], and event e1 = (x = *). The event e is encoded by a pair (-∞, +∞), and rule r is represented by the pair (10, 20). In this representation, event e1 is not included in rule r, but according to the definition of irrelevant values, it should be. A similar situation involves the "missing" value. To illustrate this, suppose that event e2= (x = ?), represented by the pair (+∞, +∞), is matched against the rule r = [x = 10..20]. In this representation the event does not match the rule. This is correct when computing value of p_{min} described in Section 3.3, but incorrect for computing value of p_{max}. Thus, matching events with meta-value "?" against rules is done according to a special procedure that corrects the indicated problem.

To increase efficiency, AQ21 marks all events containing a meta-value. When an event with a meta-value is detected, the program calls an appropriate procedure for handling it.

6.4 Implementing Wrapper Methods for Handling Missing Values

The implementation of the P1 method described in Section 3.1 is straightforward. Before AQ21 is run, all events with a "?" value for some attribute are removed from the data. This is a very fast operation requiring only one pass through the data. As mentioned before, this method is inappropriate for small datasets with many missing values, because too many events may have to be ignored.

[1] In IEEE standard infinity is not encoded as the largest representable number and the presented method is used only in AQ21.

Method P2 requires the computation of statistics on the data. The values are computed according to the following algorithm.

```
For each event e with one or more missing values
Select the s events most similar to e in the same class
   For each attribute with value "?"
     If the attribute is numeric, compute the average value
     Else compute the most frequent value in the s events
     Replace "?" with the computed value.
```

This algorithm assumes that missing values are infrequent, so that the algorithm will be efficient, and that within the *s* selected events there is at least one regular value. The latter is most likely true when *s* is sufficiently large.

P3 is the most advanced wrapper method for dealing with missing values. Using the provided training data, the method learns rules to predict missing values of attributes. The following pseudocode describes this algorithm.

```
For each class C
  Order attributes into list L in ascending order of the number of
    events in the training dataset missing their value.
  For each attribute x from L, in the order defined by L:
    Learn rules for all the values of the attribute
      from L using examples from C.
    Using the learned rules predict "missing" values
      of the attribute in the events of that class.
```

To apply this method, two problems have to be taken care of. First, the program must deal with missing values present in the training events for learning value-predicting rules. The simplest method is to ignore attributes with "missing" values. If many attributes have missing values, then predictive rules can be learned using method L1 or L2, as described in Section 3.2.

A more complex problem is when an event in which a value is predicted has more than one "missing" value, and it happens that another "missing" value is instrumental in the value-predicting ruleset. One of two methods can be applied when learning the value-predicting rules:

- When learning value-predicting rules, ignore all attributes that have missing values in events in which values are being predicted. This may not be possible when a large number of missing values is present in the dataset, because all attributes would have to be ignored.
- Use method M1 to compute probabilities of match, and choose the match with the highest probability to predict the value.

To use these methods, the value-predicting rulesets must be logically disjoint so that the rule will predict only one value. This is achieved by setting the AQ21 parameter that controls the type of rulesets to be learned to "disjoint covers." In cases where learned rulesets are not disjoint (when "intersecting covers" were learned) one may choose the value that is suggested, for example, by the rule with the highest support.

7 EXPERIMENTAL RESULTS

7.1 Testing Methods for Handling Missing Values

The methods described above have been implemented in the AQ21 learning program and tested on three datasets: *Volcanoes*, *World Factbook 2004*, and *Computer Users*.

The *Volcanoes* dataset, provided by the Smithsonian Institution, contains information about a large number of volcanoes from around the world. The dataset that was used in the study contained 13,787 training and 5,858 testing events for predicting whether or not fatalities would occur due to volcanic eruptions. Each eruption is described by 45 multitype attributes (Kaufman and Michalski, 2005).

The dataset has 79,829 missing values in the training dataset, out of 12,787 x 45 = 575,415 total values that is, about 14 %, and 33,843 missing values out of 263,610 total values in the testing dataset, that is, about 13%. The main reason for the amount of missing values is that much of the data come from records of eruptions from centuries ago, in which these values were not measured.

The *World Factbook* dataset contains information about 266 countries of the world. Each country is described in terms of 36 multitype attributes, such as Gross Domestic Product (GDP), Unemployment level, Fertility, Mortality, Population, etc. The dataset was prepared by the CIA and is downloadable from their website: *http://www.cia.gov/cia/publications/factbook.* In this dataset, 2552 values are missing, that is, about 27% of the data.

The *Computer Users* dataset contains datastreams from process tables recorded during the interaction of 10 users with their computers. The datastreams were used to learn models ("user signatures") of the users' interactions with the computer for the purpose of detecting illegitimate user activities (Michalski et al., 2005). For each of the 10 users, the dataset contains 10 training and 5 testing sessions (datastreams from login until logout).

Summary of Results

AQ21 learned rulesets from the *Volcanoes* dataset for the output attribute "Fatalities" whose values are "present" and "absent." Four methods for handling missing values were applied: L1 (ignore attribute in the extension-against operation), P1 (remove events), P2 (estimate values), and P3 (infer missing values). Table 1 presents the accuracies of classifying the testing data by the rules learned using these four methods.

	Method			
	L1	**P1**	**P2**	**P3**
Accuracy	98.51%	96.53%	98.48%	98.05%
Learning Time	13 min	2.6 min	13 min	48 min

Table 1: Results from comparing methods for handling missing values in the Volcano dataset.

As shown in Table 1, rules learned using methods L1, P2 and P3 gave very similar and relatively high degrees of accuracy on classifying the testing data. The P1 method gave slightly lower

accuracy, but was by far the fastest. Overall, if one considers accuracy to be the primary factor and the learning time as the secondary factor, L1 performed the best.

Table 2 presents results from applying the same four methods to the problem of learning rules from the World Factbook dataset for the output attribute "Birth Rate" with two classes (its values): "≤ 20," and ">20". The best results in terms of accuracy and learning time were again obtained by L1. The second best was P3 which gave relatively good results, but the learning time was significantly longer than that of the other methods. The P1 method performed poorly for this problem in terms of classification accuracy because too many events were removed from the training dataset.

	Method			
	L1	**P1**	**P2**	**P3**
Accuracy	94.29%	54.29%	40.00%	87.14%
Learning Time	0.3 sec	0.01 sec	0.2 sec	107 sec

Table 2: Results from comparing methods for handling missing values in the World Factbook dataset.

Table 3 presents results from applying the same four methods to the problem of learning rules for the *Computer Users* dataset. Here, the output attribute was "User" that has 10 values identifying each of ten computer users.

Again, L1 gave the best classification accuracy on the testing dataset, while its learning time was comparable to that of other methods. The P1 method had the shortest learning time, as before, but its accuracy was lower on the testing data than that of L1. The P3 method was worst in terms of accuracy, as well as the learning time.

	Method			
	L1	**P1**	**P2**	**P3**
Accuracy	70.21%	68.09%	65.96%	63.83%
Learning Time	20 min	17 min	18 min	34 min

Table 3: Results from comparing four methods of handling missing values on the *Computer Users* dataset.

The best performance of L1 in the experiments can be explained by the fact that the extension against operation ignores only the missing values in the event, but takes into consideration other values (see Section 3.2). Thus, it uses more information than other methods. The P1 method removes not only the missing values but also entire events, thus uses less information for learning. The P2 and P3 methods draw inferences about the training dataset that may or may not be correct. Because the AQ21 learning program working in *Theory Formation* mode (as in our experiments) learns descriptions that are complete and consistent with regard to the entire training dataset, any incorrectly inferred values will negatively affect its performance.

The above experiments tested methods for handling missing values by comparing accuracies and learning times obtained by these methods on three different real-world datasets. The next set of experiments tested the methods by determining their accuracies on datasets in which we changed a certain percentage of values into missing values. Thus, in these experiments we compared the

performance of the methods when applied to datasets with different amount of values missing, starting with the original data in which no values were missing.

In first step, we removed from the Computer Users dataset all events that contained any missing values. The resulting dataset had 8579 training and 3929 testing events (with no missing values). In the next six experiments, we randomly changed into missing values 5%, 10%, 15%, 20%, 25% and 30% of values in the dataset, respectively. Each of these six training sets was then used as input to the AQ21 learning program. The learned rules were then tested on the complete testing set (with all the values present). The classification accuracies (based on the "correct match" evaluation, (Wojtusiak, 2005)) obtained from these experiments are presented in Table 4.

	Method			
	L1	**P1**	**P2**	**P3**
Original data	60.87%	60.87%	60.87%	60.87%
5% missing	58.70%	50.00%	65.22%	60.87%
10% missing	54.35%	45.65%	56.52%	58.70%
15% missing	41.30%	21.74%	60.87%	36.96%
20% missing	28.26%	13.04%	60.87%	30.43%
25% missing	19.57%	15.22%	43.48%	15.22%
30% missing	13.04%	0.00%	19.57%	15.22%

Table 4: Classification accuracies obtained by four methods of handling missing values applied to training sets with different percentages of missing values.

For up to 10% of missing values, L1, P2 and P3 methods all performed similarly. For above 10% of missing values only P2 preformed well. A particularly surprising result is that rules learned using P2 gave better performance accuracy when the training dataset had 5% missing values than when it had no missing values, which is counterintuitive. It was also surprising that the rules learned using P2 gave the same accuracy when the dataset had 20% missing values as when it had no missing values. As expected, all method gave progressively worse results with the increasing percentage of missing values. The strongest such effect was for P1, as it was learning from an increasingly smaller amount of data.

The next set of experiments investigated the performance of rule matching methods on data with different percentages of missing values in the testing set. To this end, we applied the learned rulesets from the complete training dataset (i.e., with no missing values) to the testing datasets with 5%, 10%, 15%, 20%, 25% and 30% values missing.

Results of the experiments, presented in Table 5, show that for the *Computer Users* dataset, the presence of up to 10% missing values in the testing dataset did not affect the overall classification accuracy when using a strict matching method. When the *Selectors Ratio* flexible matching method was used (Michalski, 2004), the classification accuracy was much higher, but at the expense of classification *Precision*, and it decreased much slower with the increasing percentage of missing values in the testing dataset.

Testing dataset	Strict Matching		Flexible Matching (Selectors Ratio)	
	Accuracy	Precision	Accuracy	Precision
Original data	60.87%	87.18%	89.13%	24.63%
5% missing	60.87%	89.11%	84.78%	24.63%
10% missing	60.87%	93.20%	82.61%	24.88%
15% missing	47.83%	97.64%	78.26%	25.14%
20% missing	34.78%	100.00%	69.57%	27.32%
25% missing	26.09%	95.37%	73.91%	27.03%
30% missing	21.74%	95.37%	65.22%	26.47%

Table 5: A performance of the strict and flexible matching methods on testing datasets with increasing percentages of missing values.

It should be mentioned that the results from all the methods are relatively poor in this case, because the *Computer Users* dataset presents a particularly difficult classification problem due to a low relevance of the data to the problem (Michalski et al, 2005).

7.2 Testing the Method for Handling *Irrelevant* Values

To test methods for handling *Not-Applicable* and *Irrelevant* meta-values, we used an example from the ROBOTS problem used in the iAQ program for demonstrating natural induction (downloadable from http://www.mli.gmu.edu/msoftware.html). In this experiment, the dataset is a collection of imaginary robots that are classified as "Friendly" (positive examples) or "Unfriendly" (negative examples). Each robot is described in terms of attributes defined in Table 6. The "Robot class" is the output attribute (with two values, "Friendly" and "Unfriendly)," and the rest are input attributes. In addition, iAQ generates various derived attributes (Michalski and Pietrzykowski, 2005).

Attribute Name	Attribute Type	Attribute Domain
Robot class	Nominal	Friendly, Unfriendly
Head shape	Nominal	Round, square, triangle
Body shape	Nominal	Round, square, triangle
Smiling	Nominal	Yes, no
Holding	Structured	Sword, balloon, flag, Canadian flag, US_flag, Polish_flag
Height	Linear	short, medium, tall
Antenna's color	Nominal	Red, yellow, blue, green, black, white
Jacket's color	Nominal	Red, yellow, blue, green, black, white
Has tie	Nominal	Yes, no

Table 6: Original attributes used for describing examples in the ROBOTS domain.

Training examples for a ROBOTS problem chosen for our experiments are presented in Table 7. The first column represents the output attribute "Robot class", and the other columns represent the input attributes shown in Table 6.

Robot class	Examples of robots							
	Head shape	**Body shape**	**Smiling**	**Holding**	**Height**	**Antenna's color**	**Jacket's color**	**Has tie**
Friendly	round	square	Yes	Polish flag	Tall	green	blue	yes
Friendly	round	triangle	Yes	balloon	Medium	green	yellow	no
Friendly	square	square	Yes	balloon	Short	red	yellow	no
Friendly	round	triangle	Yes	Polish flag	Medium	green	yellow	no
Unfriendly	triangle	square	No	US flag	Medium	green	yellow	yes
Unfriendly	round	square	Yes	sword	Medium	green	blue	yes
Unfriendly	square	square	No	balloon	Medium	red	green	yes
Unfriendly	square	triangle	Yes	sword	Short	green	yellow	no
Unfriendly	round	triangle	No	Polish flag	Short	green	black	yes
Unfriendly	square	square	Yes	sword	Tall	red	red	yes

Table 7: Training events used for learning the concept of *Friendly Robots*.

The learning dataset presented in Table 7 consists of events with no meta-values. Given this training dataset, AQ21 generated the following rule:

[robot is friendly] ← [it is smiling: 4,3] &
[it is not holding a sword: 4,2] : p=4, n=0

The rule covers all four positive examples and no negative examples (p=4, n=0). Its premise is a conjunction of two conditions: [robot is smiling], which covers four positive and three negative events, and [robot is not holding a sword], which covers four positive and two negative events.

Suppose now that a new positive event e1 = (friendly, round, triangle, *, US flag, medium, green, yellow, no) is added to the training dataset by the teacher. In this event, the "Smiling" attribute is indicated as *irrelevant* (it may be relevant for other events in the class "Friendly").

To handle the additional event, e1, AQ21 needs to either learn a different rule that will describe all positive events, or to learn an additional rule to cover the additional example. In our experiment, AQ21 learns two rules, one that is identical to the previously learned rule, and the second describing the added example:

[robot is friendly] ← [it is smiling: 4,3] &
[it is not holding a sword: 4,2] : p=4, n=0

[robot is friendly] ← [it is not holding a sword: 4,2]
[it has no tie: 4,1] : p=4, n=0

Let us suppose that event e2 = (unfriendly, round, square, *, sword, medium, yellow, red, yes) is now added as a negative example to the dataset for the class "Friendly". The original ruleset is still a complete and consistent description of the class of friendly robots. Although attribute "smiling" is irrelevant for the negative event e2, it is holding a sword that eliminates it, and the rules are consistent.

To illustrate matching events with meta-value "*" against rules, suppose that the rule

[robot is friendly] ← [it is smiling] & [it is not holding a sword]

has been previously learned. The event e1: (friendly, round, triangle, *, US_flag, medium, green, yellow, no) matches this rule because it is not holding a sword and it has been declared that the feature "is smiling" is irrelevant. Similarly the event e2: (unfriendly, round, square, *, sword, medium, yellow, red, yes) does not match the rule, because the robot is holding a sword.

7.3 Testing the Method for Handling *Not Applicable* Values

To test the method for handling "Not applicable" meta-values, suppose we added to the dataset for the ROBOT problem a positive example

e3 = (friendly, triangle, square, yes, NA, medium, green, blue, yes),

whose value of attribute "holding" is NA because the robot under consideration has no hands. In this case, the originally learned rule does not cover the event e3. AQ21 learns an additional rule, and produces the ruleset:

[robot is friendly] ← [it is smiling: 5,3] & [it is not holding a sword: 4,2] : p=4, n=0

[robot is friendly] ← [its head is square or triangle: 2,4]
[its body is square: 3,4]
[it is smiling: 5,3]
[its height is short or medium: 4,5] : p=2, n=0

The second rule has four selectors needed to eliminate all negative events, while still covering event e3. Such a treatment of NA meta-values is consistent with the method described in Section 5.

A similar example can be developed for the case when NA appears in a negative example.

To illustrate matching events with the NA meta-value against rules, suppose that the rule:

[robot is friendly] ← [it is smiling] & [it is not holding a sword]

has been learned. The event

e4 = (friendly, triangle, square, NA, balloon, medium, green, blue, yes)

is matched against the rule. The event e4 does not match it because its attribute "Smiling" is not applicable.

8 RELATED RESEARCH

The problem of handling meta-values described in this paper has not been adequately addressed in literature on machine learning and data mining. Most authors concentrate solely on handling *missing* values, or treat all three meta-values in the same way. Even if they distinguish between different meta-values, as, for example, in (Kononenko, 1992; Bruha, 2004), they do not address the distinctions between them. The methods presented in this paper are original and different from those described in the literature. They also are applied in the context of more expressive representation language, namely attributional calculus.

To give the reader a sense of the differences between our methods and other methods, below is a brief review of some of the research in this area, which is mainly concerns handling of *missing* values.

In C4.5 program (Quinlan, 1993), the learning phase assigns probabilities to the missing values in order to evaluate an attribute. Events with missing values are assigned probabilities of belonging to partitioned sets. The probabilities are used when computing Gain Ratio, a measure for evaluating attributes when building a decision tree. This probabilistic approach is also used when evaluating new examples. In such cases, the program explores all possibilities in evaluating the decision tree, and assigns the class with the highest probability.

CN4 described in (Brucha and Krokowa, 1994) is an extension of the *CN2* program (Clark and Niblett, 1988). It employs six routines for processing missing values (Brucha and Franek 1996; Brucha, 2004): ignore missing values; add "missing" value to an attribute domain; use the most common value; create weighted copies of the original examples having different values replacing the missing one; randomly select values; and match any value in learning and classification.

In (Brucha, 2004), the author proposes a multistrategy approach based on the six methods listed above to handle missing attribute values. Two of the presented methods are similar to our methods P1 (ignore events with missing values) and P2 (replace missing values with average/most common value), but applied with a different learning program.

Ragel and Cremilleux (1999) presented an approach similar in spirit to P3, but in the context of using association rules to fill-up the missing values. The proposed MVP method learns rules using the Robust Association Rules Algorithm (RAR). Rules with a high support are used to complete missing values in data. Another approach similar to P3 is discussed in (Lakshminarayan et al., 1996), in which the authors use the Autoclass Bayesian clustering program and C4.5 decision tree learner for filling-in the missing values.

Wu and Barbara (2002) describe a method for handling missing values for numerical attributes. The method assumes the availability of constraints on attribute values, such as data summaries contained in data warehouses. Three types of problems are considered: well-constrained, in which the available summary is sufficient for inferring missing values; under-constrained, in which summaries are accurate but insufficient; and over-constrained, in which summaries are inconsistent. The authors propose three methods for filling-in missing values: by solving linear equations in order to find exact missing values, by maximizing entropy, and by minimizing cross-entropy. Experimental results show that the accuracy of the presented methods increases with the number of constraints (summaries).

Wang (2004) proposes a fuzzy set-based method to handle missing values in learning Hopfield neural nets. Each training example with missing values is replaced by a set of "fuzzy examples" without missing values and whose weights/probabilities are computed according to fuzzy set theory. Such examples are used to learn the neural networks. Similarly for classification problems, the author proposes using fuzzy copies of testing example for evaluation; each copy of the testing example is given a value estimated based on the fuzzy sets theory and having an assigned probability (weight). The weight is taken into consideration to compute the final degree of match.

A number of papers describe statistical approach to handling *missing* meta-values. Holt and Benfer (2000) propose an iterative regression approach named MISDAT. The program iteratively improves estimates of missing values until a test based on squared multiple correlation stabilizes. In practice, fewer than ten iterations are usually sufficient. An overview of a number statistical methods for dealing with missing values is presented by Little and Rubin (2002).

A number of researchers have investigated the handling and imputation of missing values to particular datasets. Heikki et al. (2004) discuss several methods, such as simulation of missing data, interpolation, regression analysis, nearest-neighbor etc. as applied to air quality data. Engels and Diehr (2003) discuss statistical methods for imputing missing values in longitudinal data. Sartori et al. (2004) use statistical methods for multiple imputation of missing values in cancer mortality data.

Theoretical aspects of learning from examples with meta-values are discussed in (Schuurmans and Greiner, 1997; Greiner et al., 1997). The authors discuss missing and irrelevant meta-values in context of PAC learning.

9 CONCLUSION

Methods for reasoning with *missing*, *irrelevant*, and *not applicable* meta-values in data have been described for both training and application/testing phases of attributional rule learning using AQ learning method. The three meta-values have different semantics, and appear in the data for different reasons. Therefore, they have been considered in this paper as separate problems.

The *missing value* problem appears in many applications domains, as it is quite common that values of some attributes may not available in the data for some reason. The *irrelevant* and *not applicable* values represent problem background knowledge communicated to the program by an expert, and the problem is how to adequately utilize this knowledge. Semantics of the meta-values are used on level of extension-against, the most important operation in AQ learning, therefore are handled on the basic level of the learning algorithm. This fact significantly differentiates the presented methodology from methods known from literature that are based on filling-in missing values.

The presented methods have been implemented in the AQ21 learning program, and tested on datasets from three different real world domains, and one designed domain in order to evaluate their strengths and weaknesses. Four methods were investigated for handing missing values, L1 (that ignores these values during the extension-against operation), P2 and P3 (that fill-in data by estimated or hypothesized values), and P1 (that ignores events with such values). In our initial experiments, the best results were obtained from the L1 method. Methods for handling irrelevant and not applicable meta-values gave results fully consistent with the meaning of these meta-values. The initial results confirmed our hypothesis that handling semantics of meta-values on level of extension-against and basic matching operators, the lowest level operations in AQ learning, provide the best results.

The implementation of the developed methods in the AQ21 program makes it applicable to many real-world domains where such meta-values occur. Such domains include medicine, agriculture, bioinformatics, intrusion detection, classification of geological phenomena, and others.

REFERENCES

Brucha, I., “Meta-Learner for Unknown Attribute Values Processing: Dealing with Inconsistency of Meta-Databases,” *Journal of Intelligent Information Systems*, 22:1, pp. 71-87. 2004.

Brucha, I. and Franek, F., “Comparison of Various Routines for Unknown Attribute Value Processing: The Covering Paradigm,” *International Journal of Pattern Recognition and Artificial Intelligence*,. 10:8, pp. 939-955, 1996.

Clark, P. and Niblett, T., “The CN2 Induction Algorithm,” *Machine Learning*, 3:4, pp. 261-283, 1989.

Engels, J. M. and Diehr, P., "Imputation of Missing Longitudinal Data: A Comparison of Methods," *Journal of Clinical Epidemiology*, 56, pp. 968-976, 2003.

Greiner, R., Grove, A.J. and Kogan, A., “Knowing What Doesn’t Matter: Exploring the Omission of irrelevant data,” *Artificial Intelligence*, 97:1-2, pp. 345-380, December 1997.

Holt, B. and Benfer, R.A., Jr., "Estimating Missing Data: An Iterative Regression Approach," *Journal of Human Evolution*, 39, pp. 289-296, 2000.

Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J. and Kolehmainen, M., "Methods of Imputation of Missing Values in Air Quality Data Sets," *Atmospheric Environment*, 28, pp. 2895-2907, 2004.

Kaufman, K.A. and Michalski, R.S., “An Application of AQ Learning to the Analysis of Volcanic Activities,” *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 2005 (to appear).

Lakshminarayan, K., Harp, S. A., Goldman, R. and Samad, T., "Imputation of Missing Data using Machine Learning Techniques," *Proceedings of the Second International Conference on Knowledge Discovery & Data Mining*, Portland, OR, 1996.

Little, R.J.A. and Rubin, D.B., *Statistical Analysis with Missing Data*, Second Edition, John Wiley & Sons, 2002.

Michalski, R.S., "Synthesis of Optimal and Quasi-Optimal Variable-Valued Logic Formulas," *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, Bloomington, IN, pp. 76-87, 1975.

Michalski, R. S., "A Theory and Methodology of Inductive Learning," Chapter in the book, Machine Learning: An Artificial Intelligence Approach, R. S. Michalski, T. J. Carbonell and T. M. Mitchell (Eds.), pp. 83-134, TIOGA Publishing Co., Palo Alto, 1983.

Michalski, R.S., "ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction," *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA, April, 2004.

Michalski, R.S. Kaufman, K.A., Pietrzykowski, J., Sniezynski, B., and Wojtusiak, J. “Learning User Models for Computer Intrusion Detection: Results from a Preliminary Study Using Natural Induction Approach” *Reports of the Machine Learning and Inference Laboratory*, George Mason University, Fairfax, VA, 2005 (to appear).

Michalski, R.S and Pietrzykowski, J., "iAQ: A Natural Induction System for Education and Research in Machine Learning and Knowledge Mining," *Reports of the Machine Learning and Inference Laboratory,* George Mason University, Fairfax, VA, 2005 (to appear).

Michalski, R.S. and Wojtusiak, J., "Semantic and Syntactic Attribute Types in AQ Learning," *Reports of the Machine Learning and Inference Laboratory,* George Mason University, Fairfax, VA, 2005 (to appear).

Quinlan, J. R., "Unknown Attribute Values in Induction," *Proceedings of the 6th International Workshop on Machine Learning*, San Mateo, CA, 1989.

Quinlan, J. R., *C4.5: Systems for Machine Learning*, Morgan Kaufmann Publishers Inc., 1993.

Ragel, B. and Cremilleux, B.,"MVC - A Preprocessing Method to Deal with Missing Values," *Knowledge-Based Systems*, 12, pp. 285-289, 1999.

Satori, N., Salvan, A., and Thomaseth, K., "Multiple Imputation of Missing Values in Cancer Mortality Analysis with Estimated Exposure Dose," *Computational Statistics & Data Analysis*, 2005 (to appear).

Schuurmans, D., and Greiner, R., "Learning to Classify Incomplete Examples," in *Computational Learning Theory and Natural Learning Systems*, Vol. IV, MIT Press, 1997.

Wang, S., "Classification with Incomplete Survey Data: A Hopfield Neural Network Approach," *Computers and Operations Research*, Volume 32, Issue 10, 2005.

Wojtusiak, J., "AQ21 User's Guide," *Reports of the Machine Learning and Inference Laboratory*, MLI 04-3, George Mason University, Fairfax, VA, 2004.

Wu, X. and Barbara, D., "Learning Missing Values from Summary Constraints," *SIGKDD Explorations*, 4, 2002.

A publication of the *Machine Learning and Inference Laboratory*
School of Computational Sciences
George Mason University
Fairfax, VA 22030-4444 U.S.A.
http://www.mli.gmu.edu

Editor: R. S. Michalski
Assistant Editor: K. A. Kaufman

The *Machine Learning and Inference (MLI) Laboratory Reports* are an official publication of the Machine Learning and Inference Laboratory, which has been published continuously since 1971 by R.S. Michalski's research group (until 1987, while the group was at the University of Illinois, they were called ISG (Intelligent Systems Group) Reports, or were part of the Department of Computer Science Reports).