To appear in the Proceedings of the 2nd International Workshop on Multistrategy Learning, Harpers Ferry, VW, May, 1993

Multistrategy Constructive Induction: AQ17-MCI

E. Bloedorn, R.S. Michalski and J. Wnek

Artificial Intelligence Center
George Mason University
Fairfax, VA 22030
{bloedorn, michalski, wnek}@aic.gmu.edu

Abstract

This paper presents a method for multistrategy constructive induction that integrates two inferential learning strategies—empirical induction and deduction, and two computational methods—data-driven and hypothesis-driven. The method generates inductive hypotheses in an iteratively modified representation space. The operators modifying the representation space are classified into "constructors," which expand the space (by generating additional attributes) and "destructors" which contract the space (by removing low relevance attributes or abstracting attribute values). Constructors generate new dimensions (attributes) by analyzing original or transformed examples (data-driven) and by analyzing the rules obtained in the previous iteration (hypothesis-driven). Destructors detect the irrelevant components of the representation space by rule-based inference or statistical analysis. The method has been implemented in the AQ17-MCI program. The preliminary results from applying it to a problem with noisy training data and large number of irrelevant attributes demonstrated a superiority of the method over other constructive induction methods both in terms of the predictive accuracy, as well as the overall simplicity of the generated descriptions.

Key words: multistrategy learning, inductive inference, constructive induction, representation space, concept learning.

1. Introduction

Conventional concept learning techniques generate hypotheses in the same representation space in which original training examples are presented. In many learning problems, however, the original representation space is inadequate for formulating the correct hypothesis. This inadequacy can be evidenced by a high degree of irregularity in the distribution of instances of the same class in the original representation space.

In a situation, there exists a mismatch between the complexity of concept boundaries in the space and the capabilities of the descriptive constructs of the representation language to describe the boundaries. Consequently, if the boundaries are highly irregular, typical constructs used in learning systems will likely be inadequate for representing them. Such typical constructs include nested axis-parallel hyper-rectangles (decision trees), arbitrary axis-parallel hyper- rectangles (conjunctive rules with internal disjunction, as used in VL1), hyperplanes or higher degree surfaces (neural nets), compositions of elementary structures (grammars), etc.

To address such problems, the idea of constructive induction has been introduced (Michalski, 1978; Watanabe and Elio, 1987,

Matheus and Rendell, 1989; Rendell and Seshu, 1990; Wnek and Michalski, 1991). Constructive induction can be viewed as a "double-search" process, that searches both for a hypothesis and for an adequate representation space in which to express this hypothesis.

Most constructive induction methods use a specific technique within one basic computational method. Basic methods are classified to data-driven, hypothesis-driven and knowledge-driven (Wnek and Michalski, 1991, 1993). Recently, there has been a trend toward "multistrategy" constructive induction approaches that integrate several techniques and methods.

This paper presents early results on the development of a multistrategy constructive induction system, AQ17-MCI, that aims at integrating a wide range of constructive induction techniques and methods. The basic ideas and the architecture of the system are based on the Inferential Theory of Learning (ITL), proposed by (Michalski, 1992). In ITL, learning is viewed as a "goal-directed process of modifying the learner's knowledge by exploring the learner's experience."

As mentioned above, a constructive induction learner performs two types of searches—a search for an inductive hypothesis and a search for an adequate representation space in which the hypothesis is represented. These two types of searches require different types of search operators.

The search for a hypothesis applies operators provided by the given inductive learning method. For example, the AQ17-MCI method (briefly, MCI) uses operators employed in the AQ-type learning systems, such as "dropping conditions," "extension against," "adding an alternative," "closing interval," and "climbing a generalization tree."

The representation space search operators modify the representation space. The AQ17-MCI method uses both "constructors," that expand the space by adding new dimensions (attributes), "destructors" that contract the space by removing less relevant attributes and/or abstracting values of some attributes.

To perform a representation space search, meta-operators are introduced that allow the system to suggest different representation space search operators and methods ("constructive induction strategies"). Using the ITL framework, the selection of constructive induction strategies is done by applying the operator selection rules based on the evaluation of hypotheses generated in consecutive iterations i.e. by "exploring the learner's experience".

This paper describes several techniques and methods for representation space search, their integration in AQ17-MCI system, and the results from testing the system and comparing it with several other systems. The hypothesis space search is assumed to be done by the standard AQ-type algorithm.

2. Related Research

The MCI method is relevant to both the research in constructive induction and multistrategy learning. Related work includes the system LAIR (Watanabe and Elio, 1987), and "Principled Constructive Induction" (Mehra, Rendell and Benjamin, 1989). LAIR uses domain-specific background knowledge to construct new attributes. Principled

constructive induction uses geometric interpretations of various constructors to guide their selection. Neither of these approaches, however, possesses the wide range of constructors and destructors available in MCI.

Other related systems are STAGGER that integrates techniques for Boolean, numerical and weight learning (Schlimmer, 1987). GABIL, for adaptive strategy selection, based on classification performance (Spears and Gordon, 1991), and MBAC, which uses parabolic models of strategy performance for strategy selection (Holder, 1991). The strategy selection in MCI also draws inspiration from research on the development of large scale inference systems, especially INLEN (Kaufman, Michalski and Kerschberg, 1991). In INLEN, knowledge acquisition or discovery is based on *learning* rules from expert-supplied examples. The automated acquisition of rules is most suitable to areas where expertise is difficult to quantify, or where rules may need to be modified often, such as in the case of strategy selection for constructive induction.

Several systems have been developed that exhibit constructive induction capabilities. Some of the earliest were INDUCE (Michalski, 1980) and LEX (Mitchell, Utgoff and Banerji, 1983). Many systems are based either on an analysis of the training data, i.e., data-driven systems (e.g., Schlimmer, 1987; Bloedorn and Michalski, 1991), or an analysis of hypotheses, i.e., hypothesis-driven (Matheus and Rendell, 1989), (Pagallo and Haussler, 1990), (Wnek and Michalski, 1991, 1993).

These techniques are not very useful in situations requiring different types of knowledge representation space change. For example, learning from complex and noisy sensory data (e.g., learning to recognize textures or shapes), seems to require a number of different techniques for the representation space change.

Given several such techniques, a problem arises of choosing the one that is most fit for a given situation. This problem is somewhat analogous to the problem of choosing an inductive learning method to fit the given problem at hand. Aha (1992) has proposed to solve the latter problem by using meta-rules that link the properties of training datasets with various empirical inductive learning methods.

To choose among many representation space modification operators, the MCI method uses meta-rules that link the properties of the training datasets and properties of the hypotheses generated from these datasets with the appropriate representation space modification operators.

3. The MCI Method

3.1 An Overview

The MCI method integrates a large number of different representation space modification techniques that are used to determine an adequate representation space for concept learning. The process of concept learning itself if done by an AQ-type inductive learning method.

A general flow diagram for the MCI method is shown in Figure 1. The input data are initially a user-provided training dataset plus a characterization of the initial representation space, which includes a description of attributes, their types and their domains. The training dataset is split into a primary and a secondary dataset. The primary training set is inputted to the Decision Rule Generation

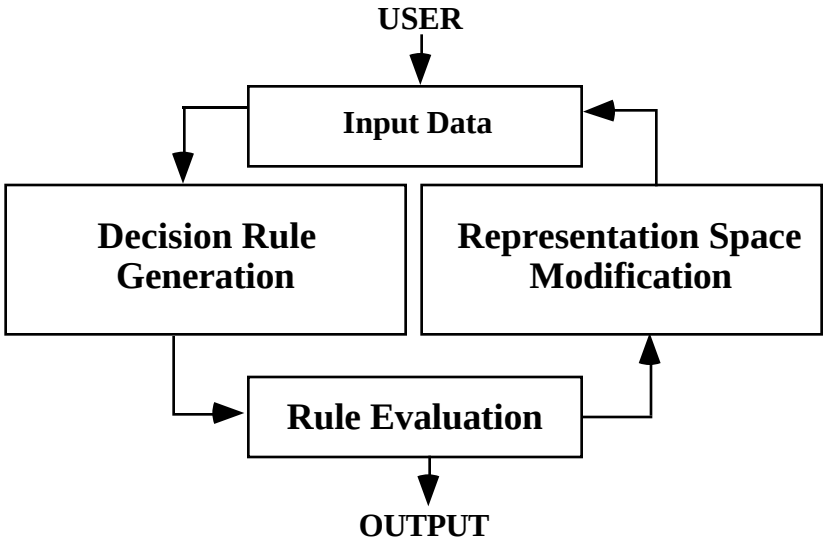

Figure 1. A functional diagram of the MCI method.

module, which uses an empirical inductive learning program (AQ14) to generate general concept descriptions (rulesets). The obtained rulesets are evaluated in terms of their complexity and their performance on the secondary training set. Based on the results of this evaluation, the system decides either to stop the learning process (the obtained rules are outputted as the solution), or to move to the Representation Space Modification module. This decision is based on special control meta-rules (Section 3.2.). The final decision rules are evaluated on the testing examples to determine their performance. Figure 2 shows the partitioning of the input examples into different classes (primary and secondary training examples, and testing examples), and explains how they are used.

The representation space modification is done by an application of various constructive induction operators, acting as constructors or destructors. Once a new representation space has been determined, both the primary and secondary training dataset is reformulated into this space, and the process is repeated. The next sections describe in greater detail various aspects of the above process.

3.2 Determining When to Modify the Representation Space

The representation space needs to be modified if there exists a mismatch between the distribution of examples in the space and the capability of the representation language to adequately describe this distribution. This mismatch can be removed either by developing a learning algorithm capable of generating more complex discrimination surfaces in the given representation space, or by changing the representation so that simple discrimination surfaces will do the job. For some problems, the first approach is infeasible.

The constructive induction approach is to modify the representation space to remove the mismatch. An illustrative example of such a mismatch is the "bit parity detection" problem. A description of binary strings with this property in terms of the bit positions in the string is very complicated and long. If, however, one generates an additional attribute

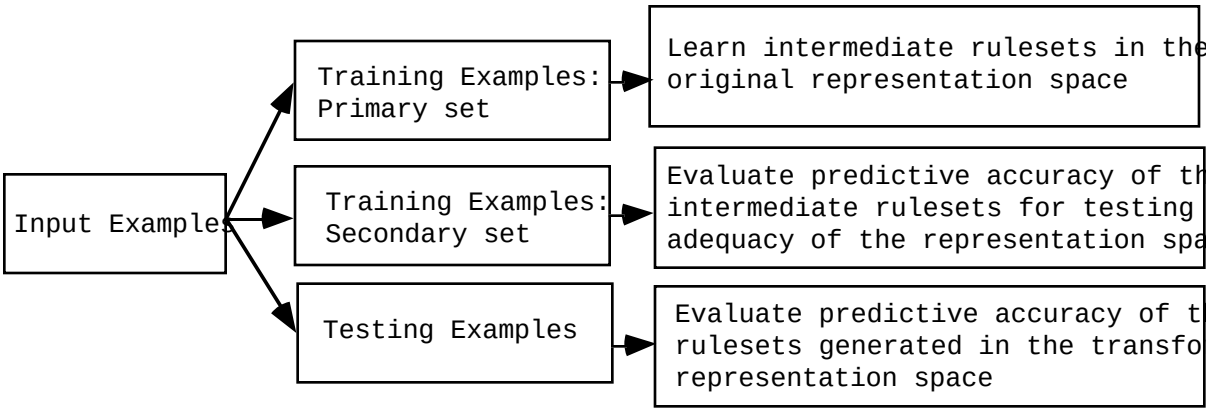

Figure 2. Subsets of the examples and their role.

(a dimension in the representation space) "mod2" of the sum of the bits in the string, the problem becomes trivial. The MCI approach is to apply a wide range of such operators for representation space change in order to determine a description space in which it would be easy to find the correct or approximately correct decision rules.

The problem arises of how to detect the need for representation space change. The MCI method solves this problem on the basis of the "quality" of descriptions (rulesets) generated by the Decision Rule Generation Module. The "quality" of the obtained ruleset is evaluated in terms of its predictive accuracy on the secondary training set and its complexity. If the quality is "satisfactory", according to the user or some heuristic criterion, then the process stops. A description of the dataset of examples in terms of certain meta-attributes is stored in the system's knowledge base to serve as a "meta-training example."

Meta-examples are used to represent datasets that both require some kind or representation space modification and those that do not. These meta-examples are used to develop meta-rules guiding the decisions about the need for the representation space change. If the rule quality is unsatisfactory, the method enters the Representation Space Modification module.

3.3 Determining How to Modify the Representation Space

3.3.1 Meta-attributes and Meta-rules

The representation space is modified by applying a variety of operators. These operators include both constructors that expand the space and destructors that contract the space (see Section 3.3.3). The choice of the operators is guided by the meta-rules that relate the properties of the example dataset and the rule evaluation results on the secondary training set to the most appropriate operators. These rules are initially provided by the user, and later improved through learning from the meta-examples mentioned below.

The meta-examples are described in terms of meta-attributes. These meta-attributes are organized into four classes: those characterizing types of the original attributes (numeric, multivalued nominal, Boolean, etc.), those characterizing the attribute quality, such as the attribute *utility* (Imam and Michalski, 1993) or the entropy measure (Quinlan, 1983), those characterizing the expected level of quality of

Meta-attribute category	Meta-attribute	Values	Explanation
Meta-attributes detecting the presence of various types of attributes	Numeric_attributes_present	Yes, No	Yes, if data contains two more numeric attributes; No, otherwise
	Nominal_attributes_present	Yes, No	True, if data contains two or more multi-valued nominal attributes; False, otherwise
	Boolean_attributes_present	Yes, No	True, if data contains two or more Boolean attributes; False, otherwise
Meta-attributes characterizing the attribute quality	Irrelevant_attributes_present	Yes, No	Yes, if data contains any irrelevant attributes; No, otherwise
	Attribute_group_quality	Sufficient, Insufficient	Sufficient, if the minimum quality of the set of attributes is above an assumed threshold; Insufficient, otherwise
Meta-attributes estimating the quality of examples	Overprecision	Yes, No	Yes, if an attribute in the given set is measured with an excessive precision. No, otherwise
	Attribute_value_noise_level	Error rate in percentage (1..100%)	Teacher-estimated error rate in the measurement of the attribute values in the examples
	Classification_noise_level	Error rate in percentage (1..100%)	Teacher-estimated error rate in the assignment of examples to classes by the teacher ("mislabeling")
Meta-attributes estimating ruleset performance	Performance_estimation	Accuracy in percentage (1..100%)	Predictive accuracy of the last ruleset generated from the primary training example set and tested on the secondary testing set.
	Performance_change	Strongly Up, Up, No change, Down, Strongly Down	Measures the difference in performance between the n^{th} ruleset learned and the $n\text{-}1^{st}$ ruleset learned

Table 1. Meta-attributes for characterizing datasets.

the examples, and those characterizing the changes in the performance of the generated rules on the secondary training dataset. Table 1 presents a list of meta-attributes. With the exception of Irrelevant_attributes_present, and Attribute_group_quality, which can be automatically calculated in a manner described below, the values of these meta-attributes are provided by the user.

a) Attribute Type

The applicability of the representation space modification operators (for short, RSM operators) depends on the type of the attributes. For example, arithmetic operators apply to numeric attributes, logical operators apply to Boolean and multi-valued nominal attributes, etc. The type of attributes for which different RSM operators are available are currently numeric, multi-valued nominal and Boolean.

b) Attribute Quality

Attribute quality measures the ability of a single attribute to discriminate among given classes of examples. An attribute may contribute individually, or as part of an attribute group. Individual attribute quality can be measured statistically by calculating the ability of an attribute to partition the example set appropriately. One such measure is the information gain used in ID3 (Quinlan, 1983).

The value of the meta-attribute "Attribute_group_quality" is "True" if each attribute in the given group of attributes has gain greater than a user-defined minimum. This meta-attribute is useful for detecting situations in which each original attribute has some relevance, but not very high, which may be suggestive of the need for some multi-argument representation space operator (e.g., a mod_x of the sum of the values of the attributes).

The contribution of an individual attribute in the context of a set of attributes can be measured by analyzing rules generated from examples described in terms of these attributes (Wnek and Michalski, 1993). The meta-attribute "Irrelevant_attributes_present" views an attribute as irrelevant if this attribute is not present in the rules, or is present only in the "light" rules (rules associated with low values of t-weight parameter the coverage of training examples by a rule).

An alternative measure of the individual attribute quality is the *attribute utility* (Imam and Michalski, 1993). An attribute utility is the sum of the class utilities of an attribute. The class utility of an attribute is the number of classes whose attribute value set has no common values with the value set occurring in the given class. An attribute is considered irrelevant if its attribute utility is low. Thus, Irrelevant_attributes_present is true if, for any attribute present in the data, the utility of that attribute is below threshold.

c) Example Quality

The quality of training examples is characterized in terms of three meta-attributes. The first one, "Overprecision," tests if a given attribute is measured with an excessive precision. In such a situation, the valueset of the attribute is reduced, and the values of this attribute in the examples are substituted by more abstract values. The second meta-attribute, "Attribute_value_noise_level" expresses a teacher-estimated error rate in the measurement of the attribute values in the examples. The third meta-attribute "Classification_noise_level" expresses a

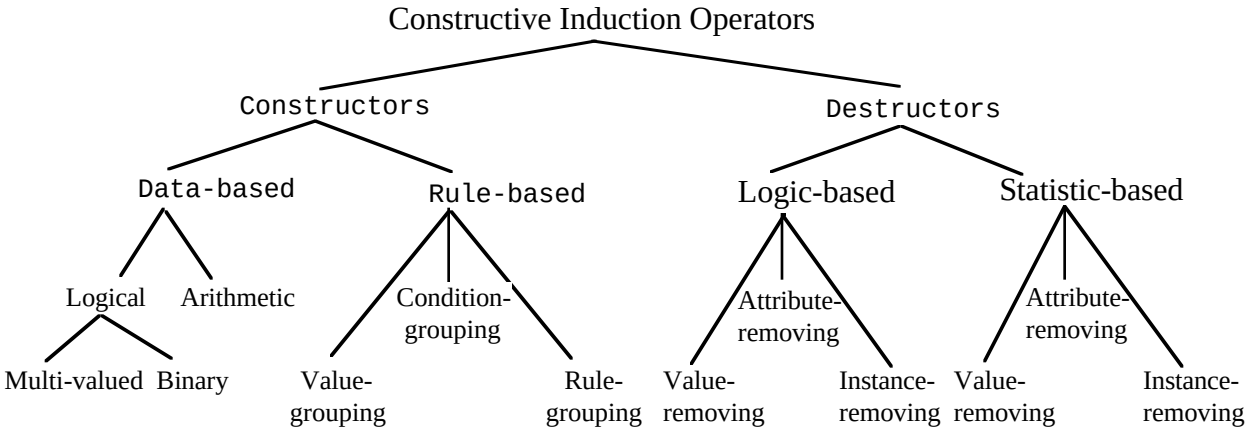

Figure 4. A hierarchy of constructive induction operators

teacher-estimated error rate in the assignment of classes to examples to classes by the teacher ("mislabeling").

Overprecision is reduced by proper quantization of the attributes (e.g., Kerber, 1992). Noise in the data is reduced by filtering training data through "heavy" rules (with high-weight) in the induced descriptions (Pachowicz, Bala and Zhang, 1992).

d) Rule Performance

There are two meta-attributes in this category: "Performance_level" that measure the performance accuracy of rules on secondary training examples, and "performance change" that expresses the change in performance from one rule generation iteration to the next. These meta-attributes help guide the selection of representation space modifiers by detecting when successive iterations are making significant positive increases in rule quality, or when the change in quality has declined or ceased.

3.3.2 Applying Meta-rules for Operator Selection

Each example dataset is characterized by a vector of the previously listed meta-attributes and their values. Operator selection is a deductive process of applying previously learned representation space modification operator rules to these meta-attribute vectors. This matching procedure calculates a degree of match between the meta-example and the RSM rules using ATEST (Reinke, 1984). Representation space modifiers are then ranked in decreasing order of match. If no single RSM rule is the top rule, then the user is asked to select. This selection may be based on the user's preference for different types of modifications such as arithmetic constructions over logical constructions.

It may occur that the same RSM operator is repeatedly selected. In other words the search stagnates on a local maximum. MCI attempts to prevent this by updating the database characterization after each ruleset evaluation. Since the meta-attributes are updated continuously, the selection stage picks the operator that best matches the *current* database characterization. If all available operators fail to match the description (i.e., the degree of match is below a threshold) then selection stops and MCI evaluates the current ruleset on the testing examples. At minimum the best performance of MCI will be that which is achieved when no modifications are made to the representation

space. In this case the performance of MCI will be equal to just selective induction.

The set of constructive induction operators can be organized hierarchically as shown in Figure 4. This hierarchical organization captures the relationships between CI operators and allows selection rules to provide better guidance when confronted with new domains. The current MCI system has capabilities for both types of constructors, logical attribute and logical instance destructors, and statistical attribute-value removal.

The system was bootstrapped by providing meta-examples describing datasets for which appropriate representation space modifications were already determined. This was done to confirm if the resulting meta-rules agree with experience. Descriptions for seven domains were provided including: two monk's problems (Thrun, et al., 1991). Congressional voting records from 1984 (Bloedorn and Michalski, 1991), texture data (Pachowicz, et al., 1992), artificially generated DNF4 functions and multiplexer 11 (Wnek, 1993) and finally wind-bracing data from a civil engineering domain (Arciszewski et al., 1992).

The appropriate RSM operator for each domain was found experimentally. These meta-examples were given to AQ14 classified by CI method so that strategy selection rules could be learned. Table 2 shows the learned representation space modification operator selection rules. Default rules are used in the case of RSM operators that do not yet have meta-examples in the knowledge base.

The degree of match between an example and a rule is calculated using the method of ATEST (Reinke, 1984). The degree of match for all meta-rules matching to the dataset characterization greater than threshold are displayed to the user.

```
dci_numeric ⇐
        [Numeric_attributes_present = Yes]&
        [Attribute_value_noise_level = 0%]

dci_boolean  ⇐
        [Numeric_attributes_present = No]&
        [Nominal_attributes_present = No]&
        [Irrelevant_attributes_present = No]

dci_nominal ⇐
        [Nominal_attributes_present = Yes] &
        [Attribute_value_noise_level = 0%]

hci_rule_grouping⇐
        [Attribute_value_noise_level = 0%]
        [Irrelevant_attributes_present = Yes]

rule_based_instance_removal ⇐
        [Overprecision = Yes] &
        [Attribute_value_noise_level =5%]

stat_based_attribute_value_removal ⇐
        [Overprecision = Yes] &
        [Attribute_value_noise_level = 5%]
```

Table 2. Examples of learned meta-rules for representation space modification

3.3.3 Example Reformulation

After the representation space modification has been selected, the training data are reformulated in this space. The generation module has a number of fundamental CI operators with which it can modify the primary and secondary training set. These operators include those used by a number of previous systems (Bloedorn and Michalski, 1991), (Pachowicz, et al., 1992), (Wnek and Michalski, 1993). Some of these fundamental operators have been reported

by others, notably Rendell and Seshu (1990). The following MCI operators are equivalent to the terms used in Rendell: attribute removal (projection), attribute-value removal (puncturing), and hypothesis-driven constructive induction (superpositioning).

a. Attribute Removal

Attribute removal makes a selection of a set X' of attributes from the original attribute set X. In MCI, a logic-based attribute removal is performed based on the quality of an attribute (as described by the meta-attribute "Irrelevant_attributes_present"). The irrelevancy of an attribute is calculated by analyzing rules generated by the Decision Rule Generation module. For each attribute, a sum is calculated of the total number of examples covered by a discriminant rule which includes that attribute. Attributes that are irrelevant will be useful only to explain instances that are distant from the majority of examples in the distribution. Thus, these attributes will have low total-weight sums. Logic-based attribute removal is performed in MCI by AQ17-HCI.

b. Attribute-value Modification

Attribute-value modification can be either the addition, (concretion) of values to an existing attribute domain, or the deletion (abstraction) of attribute values. Currently MCI implements only abstraction, based on the chi-square correlation between an attribute-value interval and the class. Using chi-square to quantize data was first proposed by Kerber (Kerber, 1992). Attribute value modification (AVM) selects a set $V' \subset V$ (where V is the domain of A) of allowable values for attribute A. AVM can be used to reduce multi-valued nominal domains, or real-valued continuous data into useful discrete, values. Discretization is especially important for empirical induction methods that allow only small number of discrete attribute values such as ID3 (Quinlan, 1983) and AQ (Michalski, 1983a). In MCI, statistic-based attribute-value removal is performed by a chi-square based method.

c. Hypothesis-driven CI

Hypothesis-driven CI (HCI) is a method for constructing new attributes based on an analysis of inductive hypotheses. Useful concepts in the rules can be extracted and used to define new attributes. These new attributes are useful because explicitly express hidden relationships in the data. This method of hypothesis analysis as a means of constructing new attributes is detailed in a number of places including (Wnek, 1993; Wnek and Michalski, to appear 1993). Wnek and Michalski define a hierarchy of hypothesis patterns from the simplest (value-groupings) to the most complex (rule-groupings). which is implemented in AQ17-HCI. AQ17-HCI is used in MCI to perform rule-based constructions of attributes based on value-groupings, condition groupings and rule-groupings, and attribute removal (see section a).

d. Data-driven CI

Data-driven (DCI) methods build new attributes based on an analysis of the training data. One such method is AQ17-DCI (Bloedorn and Michalski, 1991). In AQ17-DCI new attributes are constructed based on a generate and test method using generic domain-independent arithmetic and boolean operators. In addition to simple binary application of arithmetic operators including +, -, *, and integer division, there are multi-argument functions such as maximum value, minimum value, average value, most-common value, least-

common value, and #VarEQ(x) (a cardinality function which counts the number of attributes in an instance that take the value x). Another multi-argument operator is the boolean counting operator. This operator takes a vector of m boolean-valued attributes (m>=2) and counts the number of true values for a particular instance. This approach is able to capture m-of-n type concepts. Data-based logical construction in MCI is performed by AQ17-DCI using the multi-argument functions of #VarEQ(x), most-common, least-common, boolean counting, and binary boolean operators. Data-based arithmetic construction is performed by AQ17-DCI through maximum, minimum, average, and +, -, * and integer division.

e. Instance Removal

Instance removal (IR) methods detect and filter noisy, or misclassified training examples. The method used in MCI is a logic-based approach implemented in AQ-NT (Pachowicz, et al., 1992). The IR operator removes instances from the training data if they are covered by 'light' disjuncts. Light disjuncts are those disjuncts in the rule which cover only a small fraction of the total number of instances in the class. Thus if the ratio of covered instances to total instances in a class is below some threshold the covered instances are removed from consideration by the training data. The relationship between the weight of learned rules and the plausible prototypicality of examples was first described in the AQ15-TRUNC method (Michalski, 1983b). Other work, based on calculating the statistical significance of individual instances is done in (Holte, Acker and Porter, 1989)

3.4 Rule Evaluation

Once a CI operator has been selected and applied to the data, or as a part of the initial detection step, the resulting classification rules must be evaluated. (Figure 1). Control is returned to either the representation space modification module, or the process stops dependent upon rule quality. Rule evaluation is based on a number of criteria. As described in (Bergadano, et al., 1988) the quality of a concept description may be judged by three criteria: accuracy, simplicity and cost. In their approach, as in MCI, the user selects the relative importance of each of these criteria.

The *predictive accuracy* of a rule set is a measure of the ability of the rule set to correctly classify examples that were previously unseen. In MCI predictive accuracy is tested using a secondary training set. The secondary set is selected from the data the learner has not yet seen. Both primary and secondary data are not used for testing. Rules learned from the primary training set, but which perform well on the secondary set, are also less likely to be overfitted to the original data. Predictive accuracy is measured as the percentage of secondary training examples correctly classified.

Complexity of a ruleset is evaluated by counting the number of rules in the ruleset and the total number of conditions.

Cost is a measure of the price of evaluating the values of variables used in the description. Each variable has an associated cost provided by the user. A parameter within the rule-learning program, AQ, can be used to control the use of attributes in a description based on cost. For this reason cost is not included in the quality calculation presented here.

The final quality of the rule is evaluated lexiographically. Rulesets are evaluated first according to the accuracy criterion. If the

accuracy is within a user defined threshold of the goal accuracy, the ruleset is then further evaluated according to the complexity criterion. If, the ruleset does not meet the minimum standard for accuracy it is rejected and no further processing is done. The lexiographic evaluation permits the user to set a constraint on the minimum allowable accuracy.

3.5 Storing Experience of Operator Selection: Meta-examples

Each time a strategy is selected, and evaluated against the secondary training examples data, the results of the modification must be stored. If the application resulted in an improvement in rule quality, the meta-example characterizing the dataset is inserted into the knowledge base under the class representing RSM operator which made the useful modification of the representation space. If the quality remained constant or declined, the user determines if the meta-example should be stored. The problem of learning meta-rules which not only link a dataset to an operator, but also make the selection in the context of previous selections is discussed in section 6.

Given set of classified meta-examples, new meta-rules can be learned or improved. Meta-rules are generated by AQ14. The new meta-rules generalize the previous meta-examples Meta-rules will now be capable of classifying unseen databases according their suitability to representation space modification. Examples of learned meta-rules are presented in Table 2.

4. Experiments

The MCI method was tested in an artificial problem, an extension of the difficult second monk's problem in which both misclassification noise and irrelevant attributes are added. This problem, "Noisy and Irrelevant Monk2" (NIM2), extends the difficulty of the second Monk's problem, by including 5% random misclassification noise (9 training examples) and 7 irrelevant, randomly generated attributes to the original set of 6. The goal concept of the NIM2 problem, like the original monk 2, is: "exactly two of the 6 attributes take their first value". In the monk 2 problem, dci-nominal attribute construction modified the training data by adding a new attribute which represented the number of values which take their first value. This modification allowed AQ

Problem	**Method**	**Accuracy** (Exact match)	**Complexity**	
			#Rules	#Conds
Noisy Monk2 (5% noise) #Classes=2 #Attributes=13 AVD Size=3	AQ14 (No data modifications)	47.2 %	37	327
	AQ14 (with stat. attrib. value removal)	46.8 %	43	236
	AQ17-HCI (Rule-based attribute construction and removal)	42.1 %	13	55
	AQ-NT (Rule-based instance removal)	43.1 %	19	125
	AQ17-DCI (Data-driven attribute construction)	81.5 %	17	122
	MCI	90.2 %	8	23

AVD - "attribute value domain"

Table 3. A performance comparison of the MCI method with several single strategy methods.

to find the goal concept resulting in rules which perfectly stated the goal concept. AQ17-HCI also solved this problem by constructing new attributes based on "xor-rule-patterns" (Wnek, 1993).

The NIM2 problem, however, is more difficult. For this problem, dci-nominal construction builds the same attribute, but the goal concept has been disrupted by misclassified examples. This results in fairly accurate, but complex rules (81% predictive accuracy, 17 rules).

The MCI method was also applied to the problem. The detection step was performed with AQ14 generating 37 rules with a predictive accuracy of 47%. When presented with NIM2, MCI first invoked rule-based instance removal. Using AQ-NT 5% of the training examples were removed, and new rules were learned. There were 19 new rules with an accuracy of 43%. MCI next invoked dci-nominal. dci-nominal constructed a new attributed representing the number of attributes which take their first value. With this new attribute, AQ was able to generate 12 rules with an accuracy of 86%. When the operator selection module was invoked again, the meta-rule for dci-nominal construction continued to have the greatest match to the meta-example describing the dataset. When dci-nominal was invoked again, no new attributes were constructed-no representation space modification was made-so the next best method, HCI was selected by the user.

In this third representation modification, HCI added two new attributes and deleted seven features x2, x3, x6, x7, x8, x10 and x13. When AQ was invoked on the transformed database 8 rules with only 23 conditions were generated with an accuracy of 90%. MCI selection ceased when dci-nominal was selected again, and no new attributes were constructed. The combination of rule-based instance removal (AQ-NT), Data-driven CI (dci-nominal) and Hypothesis driven CI and attribute removal, produced a ruleset which has significantly fewer total rules (8 vs. 17), significantly shorter rules (23 vs. 122 total conditions) and which are better performing (90% vs. 81%) than the next best single strategy constructive induction method of AQ17-DCI. In table 4, MCI is compared to AQ14 which has not methods for data modification, the results of AQ14 after processing the data with a chi-square based attribute value removal method, AQ17-HCI, AQ-NT and AQ17-DCI.

The problem of determining the context of operator selection decisions is a matter of future work. It is interesting to note that when different meta-rules are used (characteristic vs. discriminant), the MCI method selects only dci-nominal construction and then HCI. The resulting ruleset is still superior, in predictive accuracy to any single strategy method, but is more complex (88.2% accuracy, 17 rules, 57 conditions).

5. Summary

This paper presented a methodology of multistrategy constructive induction that integrates two inferential learning strategies -empirical induction and deduction, and two computational methods--data-driven and hypothesis-driven. Empirical induction was performed in the Rule Generation module, and in the search for appropriate Representation Space Modifications (the double search of constructive induction). Deduction was used in the application of learned meta-rules to the characterization of incoming datasets in order to

select an appropriate representation space modification. MCI includes "constructor" and "destructor" modifiers. Modifier selection is based on meta-rules learned from the results of past applications of modifiers.

The MCI approach was tested on a problem, the NIM2, characterized by misclassification noise and irrelevant attributes. The MCI method produced rules which surpassed not only traditional selective induction learning (no representational modifications), but also single strategy methods in terms of the quality of rules produced. The quality of the resulting ruleset was superior both in terms of predictive accuracy on the testing examples, and complexity.

6. Future Work

One important area of improvement of the current method is the determination of a good criterion when to stop applying representation space modification (RSM) operators. In general, rule quality changes when representation space modifications are made. Currently, RSM operator selection, application and evaluation is repeated until the user is satisfied with the current ruleset quality. But if the user is not satisfied, and the change in the rule quality has been negative, the question arises as to whether the system should not recommend to the user some new ways of continuing the search process.

Such a decision should be based on a new type of meta-knowledge that keeps track of which RSM operators have been tried so far, and which have not. The meta-attribute set must capture this knowledge, and the matching algorithm must support, a more sophisticated concept of context and the sequence of RSM operators before this process can be completely automated.

Constructive induction is a knowledge intensive learning process. Further research should provide even more advanced capabilites for introducing and employing domain knowledge to guide constructive induction (Ragavan and Rendell, 1991). For example, there should be a facility for a user to indicate different preferences for various types of constructive induction operators. Also, it should be easy to the user to give advice as to the use of some new type of operators.

This raises a general issue of how to include within a constructive induction system sophisticated knowledge representation capabilties. Consequently, there is a need for developing a general method for what type of knowledge should be represented that might be useful for creating a more adequate knowledge representation, and how it should be used.

AQ17-MCI uses rule-based knowledge representation system. An interesting issue is to investigate how various ideas and operators implemented in AQ17-MCI could be employed in learning systems using different knowledge representation, e.g., decision tress, semantic network, neural nets, etc. To employ any type of modification operator withing another representation language will need to deal with the problems already addressed here, such as detection and reduction of the overprecision of data, noise in the training data, or low quality data (e.g., many irrelevant attributes). It is believed that the same cues used to select transformations relevant to a rule-based representation will be useful for other representation languages.

The above raises a general issue of developing a constructive induction learning system that employes multi-type representation language. This would allow the system to represent different types of knowledge in the form that is most suitable to them.

Acknowledgments

The authors wish to thank Mike Hieb and Ken Kaufman for their helpful comments and critical review of this paper.

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

References

Aha, D.W., "Generalizing from Case Studies: A Case Study," *Ninth International Workshop on Machine Learning*, pp. 1-10, Edinburgh, Scotland, 1992.

Arciszewski, T., Bloedorn, E., Michalski, R.S., Mustafa M. and Wnek J., "Constructive Induction in Structural Design," *ASCE Journal of Computing in Civil Engineering (submitted)*, Vol. pp. 1992.

Bergadano, F., Matwin, S., Michalski, R.S., Zhang, J., "A General Criterion for Measuring Quality of Concept Descriptions," Center for AI, George Mason University, MLI 88-31, 1988.

Bloedorn, E., and Michalski, R.S., "Constructive Induction from Data in AQ17-DCI: Further Experiments," Reports of the Machine Learning and Inference Laboratory, MLI 91-12, Center for AI, GMU, 1991.

Holder, L., "Selection of Learning Methods Using an Adaptive Model of Knowledge Utility," *Proceedings of the First International Workshop on Multistrategy Learning*, pp. 247-254, Harper's Ferry, WV, 1991.

Holte, R.C., Acker, L.E., and Porter, B.W., "Concept Learning and the Problem of Small Disjuncts," *Proceedings of IJCAI-89*, pp. 813-818, Detroit, MI, 1989.

Imam, I.F., and Michalski, R.S., "Learning Decision Trees from Rules: A Comparative Study", *JIIS Journal of Intelligent Information Systems, L.*, Kerschberg,. Z., Ras, and M., Zemankova, (eds.), Kluwer Academic, 1993.

Kaufman, K., Michalski, R.S., and Kerschberg, L., "Knowledge Extraction from Databases: Design Principles of the INLEN System," *Sixth International Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, 1991.

Kerber, R., "ChiMerge: Discretization of Numeric Attributes," *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 123-128, San Jose, CA, 1992.

Matheus, C.J., "The Need for Constructive Induction," *Proceedings of the Eighth International Workshop (ML91)*, pp. 173-177, Evanston, IL, 1991.

Matheus, C.J., and Rendell, L., "Constructive Induction on Decision Trees," *Proceedings of IJCAI-89*, pp. 645-650, Detroit, MI, 1989.

Mehra, P., Rendell, L., and Benjamin, W., "Principled Constructive Induction," *Proceedings of IJCAI-89*, pp. 651-656, Detroit, MI, 1989.

R. S. Michalski, "Pattern Recognition as Knowledge-Guided Computer Induction," Department of Computer Science Reports, No. 927, University of Illinois, Urbana, June 1978.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," *Machine Learning: An Artificial Intelligence Approach, Vol. I*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell

(Eds.), Palo Alto, CA: Morgan Kaufmann, 1983.

Michalski, R. S., "Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-Tiered Representation" *Machine Learning: An Artificial Intelligence Approach vol. 3*, Y. Kodratoff and R.S. Michalski (eds.), pp. 63-102. San Mateo: Morgan Kaufmann. 1983.

Michalski, R.S., "Inferential Theory of Learning: Developing Foundations for Multistrategy Learning," in *Machine Learning: A Multistrategy Approach*, Vol. IV, R.S. Michalski and G. Tecuci (Eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Mitchell, T.M., Utgoff, P.E., and Banerji, R., "Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics", *Machine Learning: An Artificial Intelligence Approach, Vol. I*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Palo Alto, CA: Morgan Kaufmann, 1983.

Pachowicz, P.W., Bala, J., and Zhang, J., "Iterative Rule Simplification for Noise-Tolerant Inductive Learning," *Proceedings of the Fourth International Conference on Tools for Artificial Intelligence*, pp. 452-453, Arlington, VA, 1992.

Pagallo, G., and Haussler, D., "Boolean Feature Discovery in Empirical Learning," *Machine Learning*, 1990.

Quinlan, J.R., "Learning Efficient Classification Procedures and their Application to Chess End Games," *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski Carbonell, J.G., and Mitchell, T. (Eds.), Palo Alto, CA: Morgan Kaufmann, 1983.

Ragavan, H. and Rendell, L., "Relations, Knowledge and Empirical Learning," *Proceedings of the Eighth International Workshop on Machine Learning (ML91)*, pp. 188-192, Evanston, IL, 1991.

Reinke, R.E., *Knowledge Acquisition and Refinement Tools for the ADVISE Meta-Expert System*, University of Illinois at Urbana-Champaign, Master's Thesis, 1984.

Rendell, L. and Seshu, R., "Learning Hard Concepts Through Constructive Induction: Framework and Rationale," *Computer Intelligence*, Vol. 6 pp. 247-270. 1990.

Schlimmer, J.C., "Learning and Representation Change," *Proceedings of AAAI-87*, pp. 511-515, Seattle, WA, 1987.

Spears, W., and Gordon, D., "Adaptive Strategy Selection for Concept Learning," *Proceedings of the First International Workshop on Multistrategy Learning*, pp. 231-246, Harper's Ferry, WV, 1991.

Thrun, S.B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzerowski, 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., "The MONK'S Problems: A Performance Comparison of Different Learning Algorithms," *(revised version)*, Carnegie Mellon University, Pittsburgh, PA, CMU-CS-91-197, 1991.

Watanabe, L. and Elio, R., "Guiding Constructive Induction for Incremental Learning from Examples," *Proceedings of IJCAI-87*, pp. 293-296, Milan, Italy, 1987.

Wnek, J. and Michalski, R.S., "Hypothesis-Driven Constructive Induction in AQl7: A Method and Experiments," *Proceedings of the IJCAI-91 Workshop on Evaluating and Changing Representation in Machine Learning*, Sydney, Australia, August 1991.

Wnek, J., "Hypothesis-driven Constructive Induction," *Ph.D. dissertation*, School of Information Technology, *Reports of Machine Learning and Inference Laboratory*, MLI 93-2, Center for Artificial Intelligence, George Mason University, March 1993.

Wnek, J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments," *Machine Learning*, Special Issue on Representation, to appear 1993.